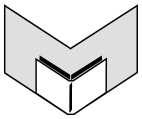
Active Learning Challenge
Challenges in Machine Learning, Volume 6

Active Learning Challenge
Challenges in Machine Learning, Volume 6

Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire and Alexander Statnikov, editors

Gideon Dror and Nicola L. C. Talbot, production editors



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

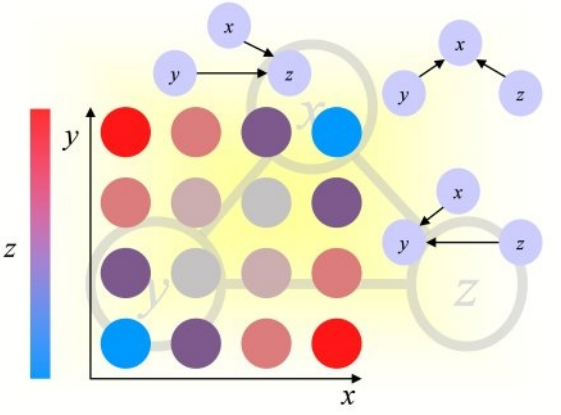
Active Learning Challenge
Challenges in Machine Learning, Volume 6

Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire and Alexander Statnikov, editors

Gideon Dror and Nicola L. C. Talbot, production editors

ISBN-13: 978-0-9719777-6-1

Causality Workbench
⟨http://clopinet.com/causality⟩



Foreword

I believe we are at the beginning of an age of Active learning, where Active learning is considered a good choice for solving most supervised learning problems. Two observations lead me to this conclusion.

The most obvious, is that the mechanisms for accomplishing active learning have become radically more apparent. For example, Amazon's Mechanical Turk has made just about any specifiable human-accomplishable information task easily doable, with the only limitation being budget. In other areas, automation is making experimentation increasingly routine. In both cases, the obvious metric to optimize is the number of individual labeled examples required to accomplish a goal, and doing this in an adaptive "active" fashion is potentially exponentially more efficient than in a nonadaptive fashion.

We have also developed a much better understanding of how to accomplish active learning under weak assumptions comparable to the foundations standard nonadaptive supervised learning. Typical starter theories in active learning assume that the world behaves according to some known parametric form, and that label noise is either nonexistent or according to some known distribution. We now understand in principle and in practice how to use active learning to compete with any set of predictors, regardless of the distribution in the world generating examples, even if label noise is chosen by an adversary. This process of weakening the assumptions is critical: algorithms that work under weak assumptions are widely and reliably applicable. My connection to active learning is here, where as part of a coalition-of-the-able (including at least 3 phd theses!), we have a transformed understanding of where active learning can be done and how to do it. The combination of these results imply that active learning can now be safely used virtually anywhere supervised learning is used.

Two observations about active learning make a well run challenge in this area substantially more compelling than for a typical supervised learning challenge.

Experimental active learning is difficult because researchers often do not have reliable access to a labeling oracle. Consequently, a common procedure is take an existing labeled dataset and transform it into an active learning dataset by pretending the labels are revealed only when the learning algorithm asks for them. When the labeled set is the universe of all instances, this is a reasonable approach. However, that's rarely the case in practice, leading to easily misinterpreted experimental results. An active learning algorithm which chooses to label only amongst a random subset of examples enjoys the same consistency guarantee as supervised learning on a labeled set of the same size. Critically, an active learning algorithm performing well in this setting may be inconsistent in an open domain setting, where the choice of which example to label is free to range over all unlabeled examples.

Actual active learning algorithms commonly have at least one free parameter controlling which examples they choose to label. Since an active learning algorithm controls it's own source of labeled examples, small changes in this parameter can radically change the quality of the learned predictor on the test set. This is far more severe than the typical overfitting problem because choosing the best of two parameter settings based on the test set can radically alter perceived performance.

This challenge addresses the second of these evaluation problems, since tuning the active learning parameters for good active learning performance had to be done within the process of active learning. Consequently, the winning algorithms deserve a much more serious consideration than those of a typical paper on active learning. I was also personally enthused to discover that the winning entries sampled labels where there was predictive uncertainty, which broadly agrees with the sound theoretical basis we have come to understand for active learning in a manner which provably addresses the first evaluation issue.

John Langford
Yahoo! research

Preface

Much of machine learning and data mining has been so far concentrating on analyzing data already collected, rather than collecting data. While experimental design is a well developed discipline of statistics, data collection practitioners often neglect to apply its principled methods. As a result, data collected and made available to data analysts, in charge of explaining them and building predictive models, are not always of good quality and are plagued by experimental artifacts. In reaction to this situation, some researchers in machine learning and data mining have started to become interested in experimental design to close the gap between data acquisition or experimentation and model building. This has given rise of the discipline of active learning. In parallel, researchers in causal studies have started raising the awareness of the differences between passive observations, active sampling, and interventions. In this domain, only interventions qualify as true experiments capable of unraveling cause-effect relationships.

To stimulate research in this area, we organized a competition in Active Learning in 2010, which featured six binary classification problems from various application domains (chemo-informatics, handwriting recognition, text processing, marketing, ecology, and embryology). This challenge addressed machine learning problems in which labeling data is expensive, but large amounts of unlabeled data are available at low cost. Such problems might be tackled from different angles: learning from unlabeled data or active learning. In the former case, the algorithms must satisfy themselves with the limited amount of labeled data and capitalize on the unlabeled data with semi-supervised learning methods. In the latter case, the algorithms may place a limited number of queries to get labels. The goal in that case is to optimize the queries to label data and the problem is referred to as active learning.

The competition was followed by a workshop on Active Learning and Experimental Design held in conjunction with AISTATS 2010, which gathered about fifty academic and industry researchers, belonging to the various communities of Artificial Intelligence, Machine Learning, Statistics and Data Mining. The results of the Active Learning competition were presented and new areas of active learning and experimental design were discussed in an attempt to bridge the gap between data acquisition or experimentation and model building. This volume gathers the best contributions of that workshop, which first appeared on-line in JMLR W&CP, volume 16.

The first chapter is of tutorial a nature. Burr Settles, a leading figure in active learning, wrote an overview of the literature. Particularly interesting in this chapter is the exploration of cases, which violate the basic assumptions of earlier foundational work.

The second part of the book comprised eight chapters, dedicated to the results of the Active Learning challenge. The first chapter describes the setup of the challenge

and summarizes the results as well as the methods used by the competitors. The following chapter by Gavin Cawley offers a systematic study that reproduces the results of the challenge. In a nutshell: it is not trivial to find a strategy to query an oracle for labels, which is better than random sampling. Random sampling is both a hard baseline to beat, and a component of most successful active learning strategies. The key to success in active learning seems to be a proper balance of "exploration" and "exploitation". Simple-minded strategies like "uncertainty sampling" (requesting labels for the samples, which the learning machine classifies with least confidence) give sometimes good results (in well chosen examples published in the literature) but may lead to dramatically bad results, as illustrated in the challenge. The next chapter describes the contribution of the overall winners, the Intel team of Alexander Borisov, Eugene Tuv, and George Runger. They tackled the problem with a heuristic method incorporating some random sampling in a "query by committee" algorithm based on ensembles of tree classifiers. In the next chapter, the NTU team of Chia-Hua Ho, Ming-Hen Tsai and Chih-Jen Lin, who ranked second overall, used a similar tactic by elaborating on the idea of uncertainty sampling applied to SVMs: they sample at various distances of the decision boundary using a stronger prior close to the boundary. It is notable that these two teams have been winning many challenges over the past few years and that they did consistently well across all datasets in this challenge.

The challenge also offered an opportunity to test semi-supervised learning strategies, particularly useful at the beginning of the learning curve, when a lot of unlabeled data are available. Chia-Hua Ho, Ming-Hen Tsai and Chih-Jen Lin used SVMs for semi-supervised learning. Jianjun Xie and Tao Xiong, who ranked third overall, proposed a method blending clustering and supervised learning, which proved effective in the challenge wheras Liang Lan, Haidong Shi, Zhuang Wang, and Slobodan Vucetic, who ranked fifth overall, combined active learning and semi-supervised learning in a consistent strategy using the simple Parzen windows classifier and clustering. The idea of using clustering for semi- supervised learning is further explored in the chapter of Zalan Bodo, Zsolt Minier, and Lehel Csato, who propose a new method based on graph partitioning.

One of the difficult aspect of the challenge (and of many real world problems) is that the classes are very unbalanced: for half of the datasets of the challenge one of the two classes includes less than 5% of the examples. This difficulty was particularly addressed in the last chapter of this part, written by Yukun Chen and Subramani Mani.

The third part of the book comprises a number of case studies: Chris Lovell, Gareth Jones, Steve R. Gunn, and Klaus-Peter Zauner give an exciting report on a on-going effort to build an autonomous experimentation machine (an actual robot) to perform biology experiments. They target the characterization of the behaviors of networks of enzymes and select the experiments to be assayed using active learning. Matthieu Geist and Olivier Pietquin, tackle the dilemma between exploration and exploitation in reinforcement learning. They show how uncertainty information can be derived from a Kalman-based Temporal Differences and used for active learning in combination with exploratory strategies. Katrin Tomanek and Katharina Morik conduct an empirical and

theoretical analysis of the bias introduced by a learning machine acting conduct an analysis as "selector" during an active learning session. They investigate putative explanatory factors for empirical results of sample reusability and show that the selected sample might be suboptimal for training another learning machine called "consumer".

In the appendix, the reader can find a report on the datasets of the challenge featuring details that were not available to the challenge participants and a report describing the implementation of the challenge using the Virtual Lab of the Causality Workbench. The website of the challenge remains open for post-challenge submissions at http://clopinet.com/al. Together with the websites of the challenge and of the workshop, this book offers a complete teaching toolkit and a valuable resource for engineers and scientists.

April 2011

The Editorial Team:

Isabelle Guyon
Clopinet
isabelle@clopinet.com

Gavin Cawley
University of East Anglia
G.Cawley@uea.ac.uk

Gideon Dror
Academic College of Tel-Aviv-Yaffo
gideon@mta.ac.il

Vincent Lemaire
Orange Labs
vincent.lemaire@orange-ftgroup.com

Alexander Statnikov
New-York University
Alexander.Statnikov@med.nyu.edu

Table of Contents

Case Studies in Active Learning

Appendices

From Theories to Queries: Active Learning in Practice

Burr Settles BSETTLES@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh PA 15213 USA

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

This article surveys recent work in *active learning* aimed at making it more practical for real-world use. In general, active learning systems aim to make machine learning more economical, since they can participate in the acquisition of their own training data. An active learner might iteratively select informative *query* instances to be labeled by an *oracle*, for example. Work over the last two decades has shown that such approaches are effective at maintaining accuracy while reducing training set size in many machine learning applications. However, as we begin to deploy active learning in real ongoing learning systems and data annotation projects, we are encountering unexpected problems—due in part to practical realities that violate the basic assumptions of earlier foundational work. I review some of these issues, and discuss recent work being done to address the challenges.

Keywords: Active Learning, Applied Machine Learning, Human-Computer Interaction.

1. Introduction

It is fairly well established now that *active learning*—a family of machine learning methods which may *query* the data instances to be labeled for training by an *oracle* (e.g., a human annotator)—can achieve higher accuracy with fewer labeled examples than passive learning. Historically, most active learning research has focused on mechanisms for (and the benefit of) selecting queries from the learner's perspective. In essence, this body of work addresses the question, "can machines learn with fewer labeled training instances if they are allowed to ask questions?" By and large, the answer to this question is "yes," with encouraging results that have been demonstrated for a variety of problem settings and domains.

For example, query algorithms (sometimes called "utility measures") based on *uncertainty sampling* select query instances which have the least label certainty under the current trained model. This simple approach is no more computationally expensive than performing inference, and has been shown to work well in a variety of applications (e.g., Lewis and Gale, 1994; Tong and Chang, 2001; Tür et al., 2005; Settles and Craven, 2008). Similarly, algorithms based on *query-by-committee* aim to minimize the version space of the model, and satisfying theoretical bounds on label complexity have been es-

tablished for these and related methods (Freund et al., 1997; Dasgupta, 2004; Hanneke, 2009). For a more detailed overview of active learning algorithms—containing many example references—see Settles (2009). In addition to all these published accounts, consider that software companies and large-scale research projects such as CiteSeer, Google, IBM, Microsoft, and Siemens are increasingly using active learning in the applications they are building[1]. Published results and increased industry adoption seem to indicate that active learning methods have matured to the point of usefulness in many real situations.

However, there are still plenty of open problems when it comes to using active learning in practice. In a recent survey of annotation projects for natural language processing tasks (Tomanek and Olsson, 2009), only 20% of the respondents said they had ever decided to use active learning. The authors even suspect that this is an overestimate, since it was advertised as a survey on the use of active learning and thus biased towards those familiar with it. Of the large majority who chose *not* to use active learning, 21% were convinced that it would not work well, with some stating that "while they believed [it] would reduce the amount of instances to be annotated, it would probably not reduce the overall annotation time." Furthermore, recent empirical studies—some employing live active learning with real annotators "in the loop"—have found puzzlingly mixed or negative results (Schein and Ungar, 2007; Guo and Schuurmans, 2008; Settles et al., 2008a; Baldridge and Palmer, 2009). Consider also that implementing query selection methods for certain more sophisticated learning algorithms can require significant software engineering overhead. Given the disconnect between the prevailing message in the literature and these mixed results in practice, coupled with high development costs, it is not surprising that researchers are hesitant to use active learning in live and ongoing projects.

I conjecture that the wealth of positive results in the literature (and there are few negative results to go by due to the publication bias) can be accounted for by the many simplifying assumptions made in previous work. For example, we have often assumed that there is a single infallible annotator whose labels can be trusted, or that the cost of labeling each query is uniformly expensive. Most of these assumptions were made to facilitate controlled experiments, where researchers often use gold-standard labeled data but pretend they are unlabeled until queried and labeled by a simulated oracle. In many real-world situations, though, these and other common assumptions do not hold. As a result, the research question has shifted over the last few years to "can machines learn *more economically* if they are allowed to ask questions?" While this seems related, it is a fundamentally different question. This new way of thinking removes emphasis from the learner and merely reducing the size of its training set, and begins to incorporate all aspect of the problem: annotators, costs, label noise, etc. This is a centrally important direction in active learning research, and the focus of this article.

1. Based on personal communication with (respectively): C. Lee Giles, David Cohn, Prem Melville, Eric Horvitz, and Balaji Krishnapuram.

2. Six Practical Challenges

In this section, we will survey six main research directions which address problems for active learning in practice. Each of the subsections that follow describes a common assumption from the literature, and summarizes ongoing research (mostly from the last three or four years) aimed at solving the problems that arise when these assumptions are violated.

2.1. Querying in Batches

In most active learning experiments, queries are selected in *serial* (one at a time), as opposed to *batches* (several to be labeled at once). It is typically assumed that the learner may inspect a large pool of unlabeled data $\mathcal{U}$ and select the single most informative instance to be labeled by the oracle (e.g., a human annotator). This setting is called "pool-based" active learning. Once the query has been labeled and added to the training set $\mathcal{L}$, the learner re-trains using this newly-acquired knowledge and selects another single query, given all the previously labeled instances, and the process repeats.

However, this is not always a realistic setting. For many applications, the process of inducing a model from training data may slow or expensive, which is often the case with state-of-the-art methods like large ensemble algorithms, or the graphical models used for structured-prediction tasks. In such cases, it is an inefficient use of labeling resources (e.g., a human annotator's time) to wait for the model to re-train before querying for the next label. Consider also that for some applications, it may be more natural to acquire labels for many different instances at once. A good example of this is high-throughput biology, where scientists are willing to wait a long time between experiments, but need to acquire measurements (to be used as training labels) for hundreds or thousands of molecules in a single screen. It is also the case that distributed, parallel labeling environments are increasingly available (some examples and additional challenges are discussed in Section 2.2), allowing multiple annotators to work on different queries at different workstations simultaneously over a network.

(*a*) serial (*b*) batch-mode

Figure 1: Serial vs. batch-mode active learning. When querying in batches, the instances should be diverse (to avoid redundancy) as well as informative to the learner.

In such settings, we wish to select a set of queries $\mathcal{Q} \subseteq \mathcal{U}$ to be labeled concurrently with model re-training, or in parallel if that is supported by the experiment or annotation environment. A naïve approach to constructing this batch is to simply evaluate all

the potential query instances, and select the "Q-best" as ranked by some utility measure. Unfortunately this is a myopic strategy, and generally does not work well since it fails to consider the overlap in information content among all the "best" instances. In other words, the best two queries might be so highly ranked because they are virtually identical, in which case labeling both is probably wasted effort. Instead, the instances in $\mathcal{Q}$ need to be both informative *and* diverse to make the best use of labeling resources.

To accomplish this, a few batch-mode active learning algorithms have been proposed. These approaches fall into three main categories. The first is to explicitly incorporate a density measure, e.g., by ranking the most informative instances, and then clustering those that are most highly ranked (Brinker, 2003; Kim et al., 2006; Xu et al., 2007). Then the batch can be constructed by querying representative instances (e.g., the centroids) from each cluster. A second approach views the task as a set optimization problem, where the utility function for any batch $\mathcal{Q}$ is the expected joint reduction in uncertainty of the model using Bayesian experimental design techniques (Hoi et al., 2006a,b). While these approaches use greedy heuristics, Hoi et al. (2006b) exploit the properties of a *submodular* functions (Nemhauser et al., 1978) to find a batch that is guaranteed to be near-optimal. A third approach (Guo and Schuurmans, 2008) attempts to construct a batch by treating the pool of candidate instances $\mathcal{U}$ as a bit vector (with 1's corresponding to the elements included in $\mathcal{Q}$), and use gradient methods to approximate the best query-set vector by numerically optimizing a discriminative expected information gain measure.

For the most part, these *batch-mode* approaches have been shown to be more economical (in terms of accuracy vs. the number of labeled batches) than passively selecting instances for a batch, which in turn is generally better than a myopic "Q-best" method. However, on some data sets a passive (random) batch-construction approach can still outperform the active methods (Guo and Schuurmans, 2008). Thus, there is still work to be done in characterizing the cases in which batch-mode active learning is likely to help, and in making further improvements to the state of the art.

2.2. Noisy Oracles

Another strong assumption in most active learning research is that the quality of labeled data from the oracle is high. In theory, of course, an "oracle" is by definition an infallible authority or guide. However, if labels come from an empirical experiment (e.g., in biological, chemical, or clinical studies), then one can usually expect some noise to result from the instrumentation or experimental setting. Even if labels come from human experts, they may not always be reliable: (*i*) some instances are implicitly difficult for both people and machines, and (*ii*) people can become distracted or fatigued over time, which introduces variability in the quality of their annotations. The recent introduction of Internet-based "crowd-sourcing" tools, such as Mechanical Turk[2] and the clever use of online games[3], have enabled researchers to attempt to "average out" some of this noise by cheaply obtaining labels from multiple non-experts. Such approaches

2. http://www.mturk.com
3. http://www.gwap.com

have been used to produce gold-standard quality training sets (Snow et al., 2008) and to evaluate learning algorithms on tasks for which no gold-standard labels exist (Mintz et al., 2009; Carlson et al., 2010).

How to use non-experts (or even noisy experts) as oracles in active learning is still a matter of ongoing investigation. One way of thinking about the problem is *agnostic* active learning (Balcan et al., 2006; Dasgupta et al., 2008), a framework which relaxes the assumption that the oracle's labels are trustworthy, yet still has positive theoretical results. Other recent work assumes that a learner may repeat queries to be labeled by multiple annotators. This introduces another interesting research issues. When should the learner decide to query for the (potentially noisy) label of a *new* unlabeled instance, versus querying for repeated labels to de-noise an *existing* but suspicious training instance? How can the learner even decide that the quality of a label is suspect? Sheng et al. (2008) study this problem using several heuristics that take into account estimates of both oracle and model uncertainty, and show that data can be improved by selective repeated labeling. However, their analysis assumes that (*i*) annotation is a noisy process over some underlying true label (in other words, there must not be any inherently difficult or ambiguous instances from the oracle's perspective), and (*ii*) all annotators are equally and consistently noisy. To my knowledge, no one has addressed the first problem. However, Donmez et al. (2009) address the second issue by allowing annotators to have different levels of accuracy in their annotations, and show that both true instance labels and individual oracle qualities can be estimated, so long as they do not change over time. Donmez et al. (2010) further relax these assumptions to allow for time-varying noise levels among annotators, and adaptively query different labelers based on the current estimate of their labeling quality.

There are still many open research opportunities along these lines. In particular, how might the effect of payment influence annotation quality (i.e., if you pay a non-expert twice as much, are they sufficiently motivated to be more accurate)? What if some instances are inherently ambiguous regardless of which annotator is used, so repeated labeling is not likely to improve matters? In most crowd-sourcing environments, the users are not necessarily available "on demand," thus accurate estimates of annotator quality may be difficult to achieve in the first place, and might possibly never be applicable again since the model has no real choice over which to use. Finally, most work in this area has been based on theoretical results or experimental simulations, and it would be helpful to see verification of these claims in practice.

2.3. Variable Labeling Costs

For many applications, variance shows up not only in label quality from one instance to another, but also in the *cost* of obtaining these labels. If our goal in active learning is to minimize the overall cost of training an accurate model, then simply reducing the number of labeled instances does not necessarily guarantee a reduction in overall labeling cost.

One proposed approach for reducing annotation effort in active learning involves using the current trained model to assist in the annotation of queries by pre-labeling

them in structured learning tasks like parsing (Baldridge and Osborne, 2004) or information extraction (Culotta and McCallum, 2005). However, such methods do not actually represent or reason about labeling costs. Instead, they attempt to reduce cost indirectly by minimizing the number of annotation actions required for a query that has already been selected.

2.3.1. KNOWN LABELING COSTS

Alternatively, *cost-sensitive active learning* approaches explicitly account for varying labeling costs while selecting queries (usually under the assumption the costs are known). Kapoor et al. (2007) propose a decision-theoretic approach that takes into account both labeling costs and misclassification costs. In this setting, each candidate query is evaluated by summing its labeling cost with the future misclassification costs that are expected to be incurred if the instance were added to the training set. They make the somewhat reasonable assumption that the cost of labeling an instances is a linear function of its length (e.g., $0.01 per second for voicemail messages). The approach also requires labeling and misclassification costs to be mapped into the same currency (e.g., $10 per error), which may not be appropriate or straightforward for some applications. King et al. (2004) use a similar decision-theoretic approach to reduce real labeling costs. They describe a "robot scientist" which can execute a series of autonomous biological experiments to discover metabolic pathways in yeast, with the objective of minimizing the cost of materials used (i.e., the cost of an experiment plus the expected total cost of future experiments until the correct hypothesis is found). Here again, the cost of materials for each experiment is fixed and known to the learner at the time of the experiment selection.

2.3.2. UNKNOWN LABELING COSTS

In the settings above, and indeed in much of the cost-sensitive active learning literature (e.g., Margineantu, 2005; Tomanek et al., 2007), the cost of annotating an instance is still assumed to be fixed and known to the learner before querying. Settles et al. (2008a) propose a novel approach to cost-sensitive active learning in settings where annotation costs are variable and *not* known. For example, if cost is a function of annotation time and we do not know in advance how long the annotator or experiment will take. In this approach, one can learn a regression cost-model (alongside the active task-model) which tries to predict the real, unknown annotation cost based on a few simple "meta features" on the instances. An analysis of four data sets using real-world human annotation costs reveals the following (Settles et al., 2008a):

- As shown in Figure 2, annotation costs are not approximately constant across instances, and can vary considerably in some domains. This result is supported by the subsequent findings of others working on different learning tasks (Arora et al., 2009; Vijayanarasimhan and Grauman, 2009a).
- Consequently, active learning approaches which ignore cost may perform no better than random selection (i.e., passive learning).

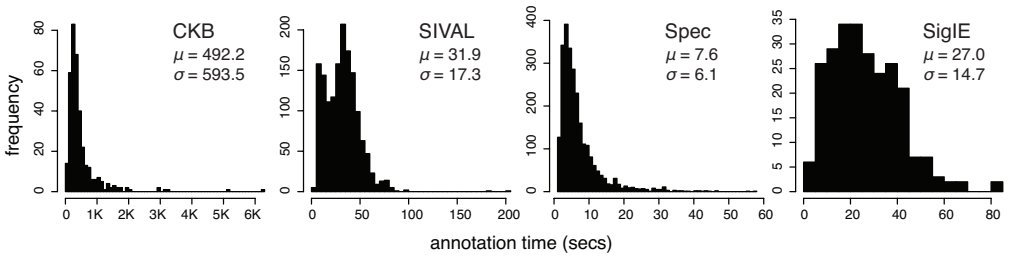


Figure 2: Histograms illustrating the distribution of annotation times for the data sets reported in Settles et al. (2008a). The mean annotation time μ and standard deviation σ for each data set is also reported.

- As shown in Figure 3, the cost of annotating an instance may not be intrinsic, but may instead vary based on the person doing the annotation. This result is also supported by the findings of Ringger et al. (2008) and Arora et al. (2009).
- The measured cost for an annotation may include stochastic components. In particular, there are at least two types of noise which affect annotation speed: *jitter* (minor variations due to annotator fatigue, latency, etc.) and *pause* (major variations due to interruptions, that should be shorter under normal circumstances).
- Unknown annotation costs can *sometimes* be accurately predicted, even after seeing only a few training instances. This result is also supported by the findings of Vijayanarasimhan and Grauman (2009a). Moreover, these learned cost-models are significantly better than simpler cost heuristics (e.g., a linear function of length).

While empirical experiments show that learned cost-models can be trained to predict annotation times fairly well, further work is warranted to determine how such approximate, predicted labeling costs can be utilized effectively by cost-sensitive active learning systems. Settles et al. experimented with a simple heuristic that divides the utility measure (e.g., entropy-based uncertainty sampling) by the predicted cost of the instances, but found that, despite fairly good cost predictions, this did not produce better learning curves in multiple natural language tasks when compared to random sampling (In fact, this was sometimes the case when *true* costs are known). Such degradation suggests that uncertainty and cost are correlated, but further investigation is needed. On the other hand, results from Haertel et al. (2008) suggest that this heuristic, which they call *return on investment* (ROI), can be effective for part-of-speech tagging, although they use a fixed heuristic cost model rather than a dynamic one trained in parallel with the task model. Vijayanarasimhan and Grauman (2009a) demonstrated potential cost savings with active learning using predicted annotation costs for computer vision. It is unclear whether these disparities are intrinsic, task-specific, or simply a result of differing experimental settings.

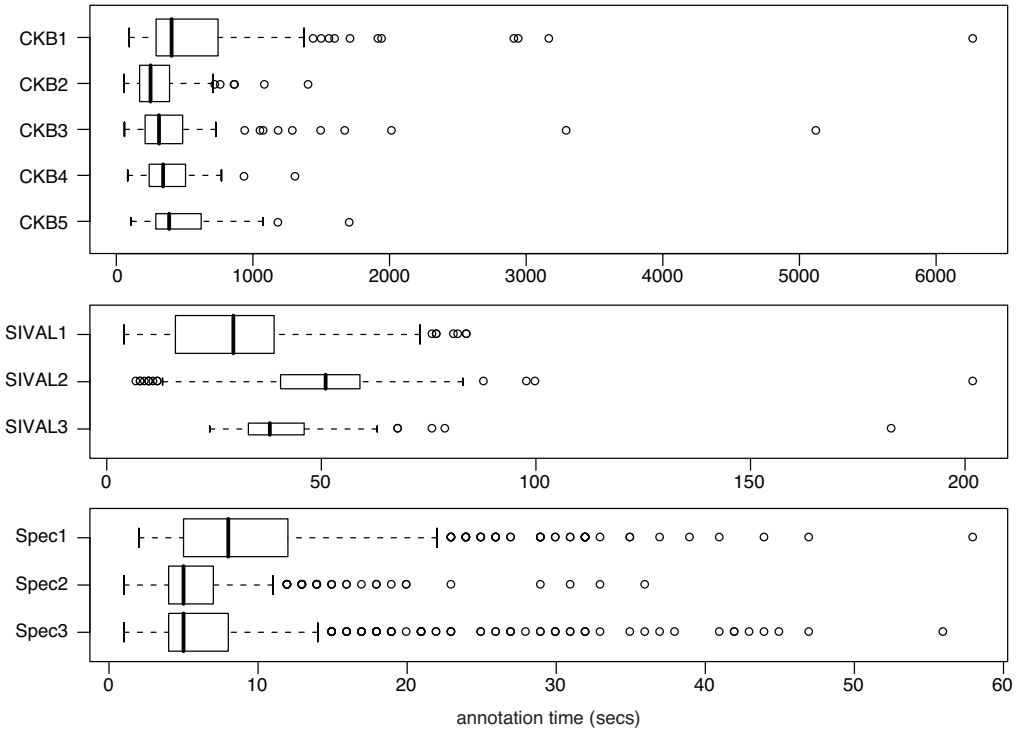


Figure 3: Box plots showing labeling time distributions for different human annotators on several data sets (Settles et al., 2008a). A box represents the middle 50% of annotation times, and the median is marked with a thick black line. Box heights are scaled in proportion to the number of instances labeled. Whiskers on either side span the first and last quartiles of each distribution; circles indicate possible outliers. Note that the range of the horizontal axis varies across data sets.

Even among methods that do not explicitly reason about annotation cost, several authors have found that alternative query types (such as labeling features rather than instances, see Section 2.4) can lead to reduced annotation costs for human oracles (Raghavan et al., 2006; Druck et al., 2009; Vijayanarasimhan and Grauman, 2009a). Interestingly, Baldridge and Palmer (2009) used active learning for morpheme annotation in a rare-language documentation study, using two live human oracles (one expert and one novice) interactively "in the loop." They found that the most cost-saving strategy differed between the two annotators, in terms of reducing both labeled corpus size and annotation time. The domain expert was a more efficient oracle with an uncertainty-based active learner, but semi-automated annotations—intended to assist in the labeling process—were of little help. The novice, however, was more efficient with a passive learner (selecting passages at random), but semi-automated annotations were in this case beneficial. There is also some preliminary evidence that for complex annotation

tasks, the design of the user interface can have as much or more impact on reducing costs as the active learning strategy (Druck et al., 2009). Continued work along these lines could also prove to be beneficial.

2.4. Alternative Query Types

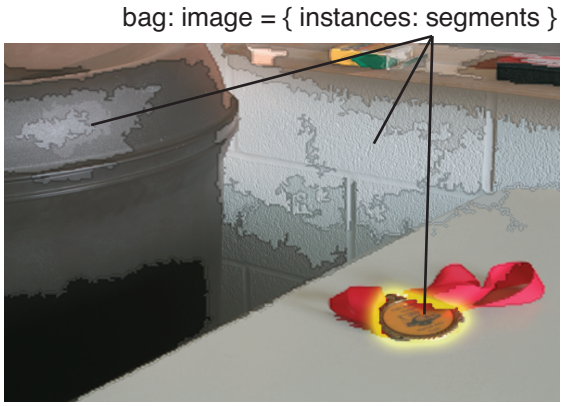
Most work in active learning assumes that a "query unit" is of the same type as the target concept to be learned. In other words, if the task is to assign class labels to text documents, the learner must query a document and the oracle provides its label. While Angluin (1988) outlines several potential query types in a theoretical analysis of active learning, only the commonly-used *membership query* has been deemed appropriate for most real-world applications. Nevertheless, recent advances have considered other types of queries for learning scenarios that can support them.

2.4.1. Multiple-Instance Active Learning

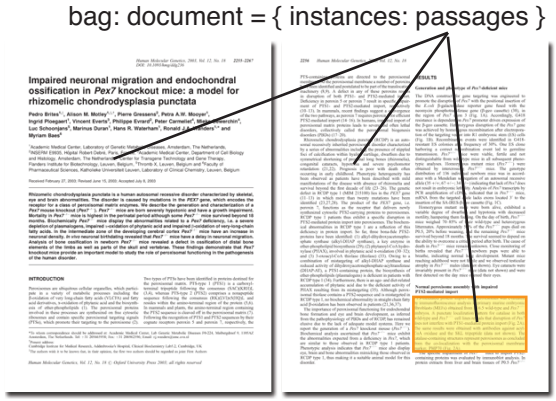
Settles et al. (2008b) introduce an alternative querying scenario called *multiple-instance active learning*, which allows the learner to query for labels at various levels of granularity. In the multiple-instance (MI) learning framework, instances are grouped into *bags* (i.e., multi-sets), and it is the bags—rather than instances—that are labeled for training. A bag is labeled negative if and only if all of its instances are negative. A bag is labeled positive, however, if at least one of its instances is positive (note that positive bags may also contain negative instances). A naïve approach to MI learning is to view it as supervised learning with one-sided noise (i.e., all negative instances are truly negative, but some positives are actually negative). However, special MI learning algorithms have been developed to learn from labeled bags despite this ambiguity. The MI setting was formalized by Dietterich et al. (1997) in the context of drug activity prediction, and has since been applied to a wide variety of tasks including content-based image retrieval (Maron and Lozano-Perez, 1998; Andrews et al., 2003; Rahmani and Goldman, 2006) and text classification (Andrews et al., 2003; Ray and Craven, 2005).

Figure 4 illustrates how the MI representation can be applied to (*a*) content-based image retrieval (CBIR) and to (*b*) text classification. For the CBIR task, images are represented as bags and instances correspond to segmented regions of the image. A bag representing a given image is labeled positive if the image contains some object of interest. The MI paradigm is well-suited to this task because only a few regions of an image may represent the object of interest, such as the gold medal in Figure 4(*a*). An advantage of the MI representation here is that it is significantly easier to label an entire image than it is to label each segment, or even a subset of the image segments. For the text classification task, documents can be represented as bags and instances correspond to short passages (e.g., paragraphs) that comprise each document. The MI representation is compelling for classification tasks for which document labels are freely available or cheaply obtained (e.g., from hyperlinks, indexes, or databases on the Internet), but the target concept is represented by only a few passages.

A traditional active learning approach for these tasks would be to query bags (i.e., images or documents) because that is the unit of classification. For MI learning tasks



(*a*) content-based image retrieval



(*b*) text classification

Figure 4: Multiple-instance active learning. In content-based image retrieval, images are represented as bags and instances correspond to segmented regions. An MI active learner may query which segments belong to the object of interest, such as the gold medal shown in this image. In text classification, documents are bags and the instances represent passages of text. In MI active learning, the learner may query specific passages to determine if they belong to the positive class.

such as these, however, it is possible to obtain labels both at the bag level and directly at the instance level. Fully labeling all instances is expensive; often the rationale for formulating the learning task as an MI problem is that it allows us to take advantage of coarse labelings that may be available at low cost (or even for free). Fortunately, in MI active learning the learner may selectively query for only the *informative* labels at a finer granularity, e.g., salient passages rather than entire documents, or segmented image regions rather than entire images. Settles et al. (2008b) focus on this type of mixed-granularity active learning with a multiple-instance generalization of logistic re-

gression, and show that it is helpful to incorporate the MI bias directly into the query selection strategy. Vijayanarasimhan and Grauman (2009a,b) have extended the idea to SVMs for the image retrieval task, and also explore an approach that interleaves queries at varying levels of granularity and cost.

2.4.2. Querying Features

Another alternative setting is to query on *features* rather than (or in addition to) instances. Raghavan et al. (2006) was the first to explore this idea with an approach called *tandem learning*, which incorporates feature feedback into traditional classification problems. In their work, a text classifier may interleave typical instance-label queries with feature-salience queries (e.g., "is the word *puck* a discriminative feature if hockey is one of the class labels?"). The values for salient features are then artificially amplified in instance feature vectors to reflect their relative importance. The authors reported that interleaving such queries is very effective for text classification, and also found that words (the features in this case) are often much easier for human annotators to label in empirical user studies, requiring a fifth of the time. Note, however, that answers to these feature queries only imply their discriminative value and do not tie features to class labels directly.

In recent years, several new methods have been developed for incorporating feature-based domain knowledge into supervised and semi-supervised learning (e.g., Haghighi and Klein, 2006; Druck et al., 2008; Melville et al., 2009). In this line of work, users may supply domain knowledge in the form of feature-label constraints, e.g., "the word *puck* indicates class label hockey." Mann and McCallum (2010) describe a semi-supervised method of combining such constraints with unlabeled data in exponential models, and Melville et al. (2009) combine this domain knowledge with labeled examples for naïve Bayes by pooling multinomials. When combined with labeled data instances, this is sometimes called *dual supervision*. Interestingly, Mann and McCallum determined that specifying many imprecisely-estimated constraints is generally more effective than using a few more precise ones, suggesting that human-specified feature labels (however noisy) are useful if there are enough of them. This begs the question of how to *actively* solicit such feature-based domain knowledge.

Druck et al. (2009) propose and evaluate a variety of active query strategies aimed at gathering useful feature-label constraints for two information extraction tasks. They show that active feature labeling is more effective than either "passive" feature labeling (using a variety of strong baselines) or instance-labeling (both passive and active) for two information extraction tasks. These results held true for both simulated and interactive human-annotator experiments. Liang et al. (2009) present a more principled approach to the problem grounded in Bayesian experimental design, however, they also resort to heuristics in practice due to intractability. Melville and Sindhwani (2009) have explored interleaving instance and feature label queries for sentiment classification in blogs using the pooling multinomials naïve Bayes approach, and Sindhwani et al. (2009) consider a similar query setting for a semi-supervised graph/kernel-based text classifier.

2.5. Multi-Task Active Learning

Most active learning settings assume that there is only one learner trying to solve a single task. In many real-world problems, however, the same data instances may be labeled in multiple ways for different subtasks. In such cases, it is probably more economical to label a single instance for all subtasks simultaneously. Therefore, *multi-task active learning* algorithms assume that a single query will be labeled for multiple tasks, and attempt to assess the "informativeness" of an instance with respect to all the learners involved.

Consider a database of film reviews, which might be used to build a system that (*i*) extracts the names of key actors and production crew, (*ii*) classifies the film by genre, and (*iii*) predicts a five-star rating based on the text. Such a system would probably employ three independent learners: a sequence model for entity extraction, a classifier for genres, and a regression model to predict ratings. Effectively selecting queries that benefit all three of these learners is still an open and promising direction in active learning.

Along these lines, Reichart et al. (2008) study a two-task active learning scenario for natural language parsing and named entity recognition (NER), a form of information extraction. They propose two methods for actively learning both tasks simultaneously. The first is *alternating selection*, which allows the parser to query sentences in one iteration, and then the NER system to query instances in the next. The second is *rank combination*, in which both learners rank the query candidates in the pool by expected utility, and the instances with the highest combined rank are selected for labeling. In both cases, uncertainty sampling is used as the base selection strategy for each learner, but other frameworks could be used as well. As one might expect, these methods outperform passive learning for both subtasks, while learning curves for each individual subtask are not as good as they would have been in a single-task active learning setting.

Qi et al. (2008) study a different multi-task active learning scenario, in which images may be labeled for several binary classification tasks in parallel. For example, an image might be labeled as containing a beach, sunset, mountain, field, etc., which are not all mutually exclusive; however, they are not entirely independent, either. The beach and sunset labels may be highly correlated in the data, for example, so a simple rank combination might over-estimate the informativeness of some instances. They propose and evaluate a new approach which takes into account the mutual information among labels.

2.6. Changing (or Unknown) Model Classes

An important side-effect of active learning that the resulting labeled training set $\mathcal{L}$ is not an i.i.d. sample of the data, but is rather a biased distribution which is implicitly tied to the model used in selecting the queries. Most work in active learning has assumed that the appropriate model class for the task is already known, so this is not generally a problem. However, it can become problematic if we wish to re-use the training data with a model of a different type—which is common when the state of the art advances—

or if we do not even know the appropriate model class (or feature set) for the task to begin with.

Fortunately, this change of or uncertainty about the model is not always an issue. Lewis and Catlett (1994) showed that decision tree classifiers can still benefit significantly from a training set constructed by an active naïve Bayes learner using uncertainty sampling. Tomanek et al. (2007) also showed that information extraction data gathered by a maximum entropy model using the query-by-committee algorithm can be effectively re-used to train more sophisticated conditional random fields (CRFs), maintaining cost savings compared with random sampling. Hwa (2001) successfully re-used natural language parsing data queried by one type of parser to train other types of parsers.

However, Baldridge and Osborne (2004) reported the exact opposite problem when re-using data queried by one parsing model to train a variety of other parsers. As an alternative, they perform active learning using a heterogeneous ensemble composed of different parser types, and also use semi-automated labeling to cut down on human annotation effort. This approach helped to reduce the number of training examples required for each parser type compared with passive learning. Similarly, Lu et al. (2010) employed active learning with a heterogeneous ensemble of neural networks and decision trees, when the more appropriate model class for the learning task was not known in advance. Their ensemble approach was able to simultaneously select informative instances for the overall ensemble, and bias the distribution of constituent weak learners toward the most appropriate model as more training data was gathered. Sugiyama and Rubens (2008) have experimented with an ensemble of linear regression models that used differing feature sets, to study cases in which the appropriate feature set is not yet decided upon.

This brings up a very important point for active learning in practice. If the appropriate model class and feature set happen to be known in advance—or if these are not likely to change much in the future—then active learning can probably be safely used. Otherwise, random sampling (at least for pilot studies, until the task can be better understood) may be more advisable than taking one's chances on active learning with the "wrong" learning algorithm. A viable alternative for active learning seems to be the use of heterogeneous ensembles in selecting queries, but there is still much work to be done in this direction.

3. Conclusion

This article surveys the main challenges currently facing the use of active learning in practice. While many of these issues are nontrivial and well beyond the current state of the art, I am optimistic that the research community will find pragmatic solutions that are of general use. After all, we have overcome comparable challenges in the past.

Over two decades ago, some exciting theoretical results for active learning (Sammut and Banerji, 1986; Angluin, 1988) led to a body of work applying these early ideas in neural networks. For the most part, these methods assumed that the learner may synthesize arbitrary query instances de novo, and applications studied only simple or

artificial learning tasks (e.g., geometrical shapes on a 2D plane). In an interesting early attempt at a real-world task, Lang and Baum (1992) employed active learning with a human oracle to train a classifier for handwritten characters and digits. They encountered an unexpected problem: many of the query images generated by the learner contained no recognizable symbols, only artificial hybrid characters that had no semantic meaning. This negative result did not discourage progress, however, but helped to motivate and justify the selective sampling and pool-based active learning scenarios commonly used today, since they guarantee that query instances are sensible because come from an underlying natural distribution.

I think we find ourselves in a similar situation today. While the past few decades have established active learning as a widely applicable tool for a variety of problem domains, these results are subject to assumptions which focus on the utility of a query to the learner, and not its cost to the teachers or other aspects of the problem as a whole. Rather than quell progress, though, I believe these practical challenges are leading to innovations which draw us closer to methods for effective interactive learning systems.

References

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 561–568. MIT Press, 2003.

D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

S. Arora, E. Nyberg, and C.P. Rosé. Estimating annotation cost for active learning in a multi-annotator environment. In *Proceedings of the NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 18–26. ACL Press, 2009.

M.F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 65–72. ACM Press, 2006.

J. Baldridge and M. Osborne. Active learning and the total cost of annotation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9–16. ACL Press, 2004.

J. Baldridge and A. Palmer. How well does active learning *actually* work? Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 296–305. ACL Press, 2009.

K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 59–66. AAAI Press, 2003.

A. Carlson, J. Betteridge, R. Wang, E.R. Hruschka Jr, and T. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*. ACM Press, 2010.

A. Culotta and A. McCallum. Reducing labeling effort for stuctured prediction tasks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 746–751. AAAI Press, 2005.

S. Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 337–344. MIT Press, 2004.

S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 353–360. MIT Press, 2008.

T. Dietterich, R. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

P. Donmez, J. Carbonell, and J. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 259–268. ACM Press, 2009.

P. Donmez, J. Carbonell, and J. Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, 2010.

G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602. ACM Press, 2008.

G. Druck, B. Settles, and A. McCallum. Active learning by labeling features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 81–90. ACL Press, 2009.

Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Selective samping using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *Advances in Neural Information Processing Systems (NIPS) 20*, pages 593–600, Cambridge, MA, 2008. MIT Press.

R. Haertel, K. Seppi, E. Ringger, and J. Carroll. Return on investment for active learning. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, 2008.

A. Haghighi and D. Klein. Prototype-driven learning for sequence models. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 320–327. ACL Press, 2006.

S. Hanneke. *Theoretical Foundations of Active Learning*. PhD thesis, Carnegie Mellon University, 2009.

S.C.H. Hoi, R. Jin, and M.R. Lyu. Large-scale text categorization by batch mode active learning. In *Proceedings of the International Conference on the World Wide Web*, pages 633–642. ACM Press, 2006a.

S.C.H. Hoi, R. Jin, J. Zhu, and M.R. Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 417–424. ACM Press, 2006b.

R. Hwa. On minimizing training corpus for parser acquisition. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 1–6. ACL Press, 2001.

A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning,. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 877–882. AAAI Press, 2007.

S. Kim, Y. Song, K. Kim, J.W. Cha, and G.G. Lee. MMR-based active machine learning for bio named entity recognition. In *Proceedings of Human Language Technology and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 69–72. ACL Press, 2006.

R.D. King, K.E. Whelan, F.M. Jones, P.G. Reiser, C.H. Bryant, S.H. Muggleton, D.B. Kell, and S.G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–52, 2004.

K. Lang and E. Baum. Query learning can work poorly when a human oracle is used. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 335–340. IEEE Press, 1992.

D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 148–156. Morgan Kaufmann, 1994.

D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. ACM/Springer, 1994.

P. Liang, M.I. Jordan, and D. Klein. Learning from measurements in exponential families. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 641–648. ACM Press, 2009.

Z. Lu, X. Wu, and J. Bongard. Adaptive informative sampling for active learning. In *Proceedings of SIAM Conference on Data Mining (SDM)*, 2010.

G.S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11:955–984, 2010.

D. Margineantu. Active cost-sensitive learning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1622–1623. AAAI Press, 2005.

O. Maron and T. Lozano-Perez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, pages 570–576. MIT Press, 1998.

P. Melville and V. Sindhwani. Active dual supervision: Reducing the cost of annotating examples and features. In *Proceedings of the NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 49–57. ACL Press, 2009.

P. Melville, W. Gryc, and R.D. Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1275–1284. ACM Press, 2009.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2009.

G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

G.J. Qi, X.S. Hua, Y. Rui, J. Tang, and H.J. Zhang. Two-dimensional active learning for image classification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

H. Raghavan, O. Madani, and R. Jones. Active learning with feedback on both features and instances. *Journal of Machine Learning Research*, 7:1655–1686, 2006.

R. Rahmani and S.A. Goldman. MISSL: Multiple-instance semi-supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 705–712. ACM Press, 2006.

S. Ray and M. Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 697–704. ACM Press, 2005.

R. Reichart, K. Tomanek, U. Hahn, and A Rappoport. Multi-task active learning for linguistic annotations. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 861–869. ACL Press, 2008.

E. Ringger, M. Carmen, R. Haertel, K. Seppi, D. Lonsdale, P. McClanahan, J. Carroll, and N. Ellison. Assessing the costs of machine-assisted corpus annotation through a user study. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association, 2008.

C. Sammut and R. Banerji. Learning concepts by asking questions. In *Machine Learning: An Artificial Intelligence Approach*, volume 2. Morgan Kaufmann, 1986.

A.I. Schein and L.H. Ungar. Active learning for logistic regression: An evaluation. *Machine Learning*, 68(3):235–265, 2007.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1069–1078. ACL Press, 2008.

B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1–10, 2008a.

B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 1289–1296. MIT Press, 2008b.

V.S. Sheng, F. Provost, and P.G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM Press, 2008.

V. Sindhwani, P. Melville, and R.D. Lawrence. Uncertainty sampling and transductive experimental design for active dual supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 953–960. ACM Press, 2009.

R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast—but is it good? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263. ACM Press, 2008.

M. Sugiyama and N. Rubens. Active learning with model selection in linear regression. In *Proceedings of the SIAM International Conference on Data Mining*, pages 518–529. SIAM, 2008.

K. Tomanek and F. Olsson. A web survey on the use of active learning to support annotation of text data. In *Proceedings of the NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 45–48. ACL Press, 2009.

K. Tomanek, J. Wermter, and U. Hahn. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 486–495. ACL Press, 2007.

S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ACM International Conference on Multimedia*, pages 107–118. ACM Press, 2001.

G. Tür, D. Hakkani-Tür, and R.E. Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186, 2005.

S. Vijayanarasimhan and K. Grauman. What's it going to cost you? Predicting effort vs. informativeness for multi-label image annotations. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Press, 2009a.

S. Vijayanarasimhan and K. Grauman. Multi-level active prediction of useful image annotations for recognition. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 1705–1712. MIT Press, 2009b.

Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the European Conference on IR Research (ECIR)*, pages 246–257. Springer-Verlag, 2007.

Results of the Active Learning Challenge

Isabelle Guyon ISABELLE@CLOPINET.COM
Clopinet, California

Gavin Cawley GCC@CMP.UEA.AC.UK
University of East Anglia, UK

Gideon Dror GIDEON@MTA.AC.IL
Academic College of Tel-Aviv-Yaffo, Israel

Vincent Lemaire VINCENT.LEMAIRE@ORANGE-FTGROUP.COM
Orange Labs, France

Editor: Neil Lawrence

Abstract

We organized a machine learning challenge on "active learning", addressing problems where labeling data is expensive, but large amounts of unlabeled data are available at low cost. Examples include handwriting and speech recognition, document classification, vision tasks, drug design using recombinant molecules and protein engineering. The algorithms may place a limited number of queries to get new sample labels. The design of the challenge and its results are summarized in this paper and the best contributions made by the participants are included in these proceedings. The website of the challenge remains open as a resource for students and researchers (http://clopinet.com/al).

1. Background

The accumulation of massive amounts of unlabeled data and the cost of labeling have triggered a resurgence of interest in active learning. However, the newly proposed methods have never been evaluated in a fair and open contest. The challenge we organized has stimulated research in the field and provides a comparative study free of "inventor bias".

Modeling can have a number of objectives, including understanding or explaining the data, developing scientific theories, and making predictions. We focus in this challenge on predictive modeling, in a setup known in machine learning as "supervised learning". The goal is to predict an outcome y given a number of predictor variables $\mathbf{x} = [x_1, x_2, ...x_n]$, also called features, attributes, or factors. During training, the model (also called the learning machine) is provided with example pairs $\{\mathbf{x}, y\}$ (the training examples) with which to adjust its parameters. After training, the model is evaluated with new example pairs (the test examples) to estimate its generalization performance. In our framework, example pairs can only be obtained at a cost; optimal data acquisition must compromise between selecting many informative example pairs and incurring a

large expense for data collection. Typically, either a fixed budget is available and the generalization performance must be maximized or the data collection expenses must be minimized to reach or exceed a given generalization performance. Data pairs $\{\mathbf{x}, y\}$ are drawn identically and independently from an unknown distribution $P(\mathbf{x}, y)$. In the regular machine learning setting (passive learning), a batch of training pairs is made available from the outset. In the active learning setting, the labels y are withheld and can be purchased from an oracle. The learning machine must select the examples, which look most promising in improving the predictive performance of theodel. There exist several variants of active learning:

- **Pool-based active learning:** A large pool of examples of $\mathbf{x}$ is made available from the outset of training.
- **Stream-based active learning:** Examples are made available continuously.
- **De novo query synthesis:** The learner can select arbitrary values of x, i.e. use examples not drawn from $P(\mathbf{x})$.

Other scenarios, not considered here, include cases in which data are not *i.i.d.*. Such situations occur in time series prediction, speech processing, unsegmented image analysis and document analysis.

Of the variants of active learning considered, pooled-based active learning is of considerable importance in current applications of machine learning and data mining, because of the availability of large amounts of unlabeled data in many domains, including pattern recognition (handwriting, speech, airborne or satellite images, etc.), text processing (internet documents, archives), chemo-informatics (untested molecules from combinatorial chemistry), and marketing (large customer databases). These are typical examples of the scenarios we intended to study via the organization of this challenge. Stream-based active learning is also important when sensor data is continuously available and data cannot be easily stored. However, it is more difficult to evaluate in the context of a challenge, so we focus instead purely on pool-based active learning. Several of the techniques thus developed may also be applicable to stream-based active learning. The last type of active learning, "de-novo" query synthesis, will be addressed in upcoming experimental design challenges in which we will allow participants to intervene on $\mathbf{x}$. In this challenge, however we limit the actions of the participants to sampling from a set of fixed points in input space and query for y, we do not allow interventions on $\mathbf{x}$, such as setting certain values x_i.

A number of query strategies with various criteria of optimality have been devised. Perhaps the simplest and most commonly used query strategy is uncertainty sampling (Lewis and Gale, 1994). In this framework, an active learner queries the instances that it can label with least confidence. This of course requires the use of a model that is capable of assessing prediction uncertainty, such as a logistic model for binary classification problems. Another general active learning framework queries the labels of the instances that would impart the greatest change in the current model (expected model change), if we knew the labels. Since discriminative probabilistic models are

usually trained with gradient-based optimization, the "change" imparted can be measured by the magnitude of the gradient (Settles et al., 2008a). A more theoretically motivated query strategy is query-by-committee (QBC) (Seung et al., 1992). The QBC approach involves maintaining a committee of models, which are all trained on the current set of labeled samples, but represent competing hypotheses. Each committee member votes on the labels of query candidates and the query considered most informative is the one on which they disagree most. It can be shown that this is the query that potentially gives the largest reduction in the space of hypotheses (models) consistent with the current training dataset (version space). A related approach is Bayesian active learning. In the Bayesian setting, a prior over the space of hypotheses is revised to give the posterior after seeing the data. Bayesian active learning algorithms (Tong and Koller, 2000, for example) maximize the expected Kullback-Leibler divergence between the revised posterior distribution (after learning with the new queried example) and the current posterior distribution given the data already seen. Hence this can be seen both as an extension of the expected model change framework for a Bayesian committee and a probabilistic reduction of hypothesis space. A more direct criterion of optimality seeks queries that are expected to produce the greatest reduction in generalization error, i.e. the error on data not used for training drawn from $P(\mathbf{x}, y)$ (expected error reduction). Cohn and collaborators (Cohn et al., 1996) proposed the first statistical analysis of active learning, demonstrating how to synthesize queries that minimize the learner's future error by minimizing its variance. However, their approach applies only to regression tasks and synthesizes queries de novo. Another more direct, but very computationally expensive approach is to tentatively add to the training set all possible candidate queries with one of the opposite labels and estimate how much generalization error reduction would result by adding it to the training set (Roy and McCallum, 2001). It has been suggested that uncertainty sampling and QBC strategies are prone to querying outliers and therefore are not robust. The information density framework (Settles et al., 2008b) addresses that problem by considering instances that are not only uncertain, but representative of the input distribution, to be the most informative. This last approach addresses the problem of monitoring the trade-off between exploration and exploitation. Methods such as "uncertainty sampling" often yield mediocre results because they stress only "exploitation" while "random sampling" performs only "exploration". For a more comprehensive survey, see (Settles, 2009).

2. Datasets and evaluation method

The challenge was comprised of two phases: a development phase (Dec. 1, 2009 - Jan. 31, 2010) during which the participants could develop and tune their algorithms, using six development datasets and a final test phase (Feb. 3, 2010 - Mar. 10, 2010). Six new datasets were provided for the final test phase. One of the exciting aspects of the organization of this challenge has been the abundance of data, which clearly signals that this problem is ripe for study, and solving it will have immediate impact. Several practitioners in need of good active learning solutions offered to donate data from their study domain. The data statistics are summarized in Table 1 for

Table 1: **Development datasets.** ALEX is a toy dataset given for illustrative purpose. The other datasets match the final datasets by application domain (see text).

Dataset	Feat. type	Feat. num.	Sparsity (%)	Missing (%)	Pos. lbls (%)	Tr & Te num.
ALEX	binary	11	0	0	72.98	5000
HIVA	binary	1617	90.88	0	3.52	21339
IBN SINA	mixed	92	80.67	0	37.84	10361
NOVA	binary	16969	99.67	0	28.45	9733
ORANGE	mixed	230	9.57	65.46	1.78	25000
SYLVA	mixed	216	77.88	0	6.15	72626
ZEBRA	continuous	154	0.04	0.0038	4.58	30744

Table 2: **Final test datasets.** The fraction of positive labels was not available to the participants.

Dataset	Feat. type	Feat. num.	Sparsity (%)	Missing (%)	Pos. lbls (%)	Tr & Te num.
A	mixed	92	79.02	0	13.35	17535
B	mixed	250	46.89	25.76	9.14	25000
C	mixed	851	8.6	0	8.1	25720
D	binary	12000	99.67	0	25.52	10000
E	continuous	154	0.04	0.0004	9.04	32252
F	mixed	12	1.02	0	7.58	67628

the development datasets and Table 2 for the final test sets. The data may be downloaded from: http://www.causality.inf.ethz.ch/activelearning.php?page=datasets#cont . All datasets are large (between 20000 and 140000 examples). We selected six different application domains, illustrative of the fields in which active learning is applicable: Chemo-informatics, embryology, marketing, text ranking, handwriting recognition, and ecology. The problems chosen offer a wide range of difficulty levels, including heterogeneous noisy data (numerical and categorical variables), missing values, sparse feature representation, and unbalanced class distributions. All problems are two-class classification problems.

The datasets from the final phase were matched by application domain to those of the development phase. During the challenge, the final test datasets were named alphabetically, so as not to make that matching explicit. However, the final datasets have mnemonic nicknames (unknown to the participants during the competition) such that the correspondences are more easily remembered:

- **A** is for AVICENNA, the Latin name of IBN SINA. This is a handwriting recognition dataset consisting of Arabic manuscripts by the 11^{th} century Persian author Ibn Sina.
- **B** is for BANANA, a fruit like ORANGE. This is a marketing dataset donated by Orange Labs.
- **C** is for CHEMO, this is a chemo-informatics dataset for the problem of identifying molecules that bind to pyruvate kinase. It is matched to HIVA, another chemo-informatics dataset for identifying molecules active against the HIV virus.
- **D** is for DOCS. This is a document analysis dataset, matched with NOVA.
- **E** is for EMBRYO, an embryology dataset, matched with ZEBRA.
- **F** is for FOREST, an ecology dataset. The problem is to find forest cover types like for SYLVA.

A report describing the datasets is available (Guyon et al., 2010).

The protocol of the challenge was simple. The participants were given unlabeled data and could purchase labels on-line for some amount of virtual cash. In addition, the index of a single positive example was given to bootstrap the active learning process. Participants were free to purchase batches of labels of any size, by providing the sample numbers of the labels they requested. To allow the organizers to draw learning curves, the participants were asked to provide prediction values for all the examples every time they made a purchase of new labels. Half of each dataset could not be queried and was considered a test set.

The prediction performance was evaluated according to the Area under the Learning Curve (ALC). A learning curve plots the Area Under the ROC curve (AUC) computed on all the samples with unknown labels, as a function of the number of labels queried.

To obtain our ranking score, we normalized the ALC as follows:

$$globalscore = (ALC - Arand)/(Amax - Arand)$$

where *Amax* is the area under the best achievable learning curve and *Arand* is the area under the average learning curve obtained by making random predictions. See http://www.causality.inf.ethz.ch/activelearning.php?page=evaluation#cont for details.

An obvious way of "cheating" would have been to use an "associate" or register under an assumed name to gain knowledge of all the labels, then submit results under one's real name. Preventing this kind of cheating is very difficult. We resorted to the following scheme, which gives us confidence that the participants respected the rules of the challenge:

- The participants had to register as mutually exclusive teams for the final phase. The membership of teams were manually verified.
- The team leaders had to electronically sign an agreement that none of his team member would attempt to exchange information about the labels with other teams.
- We announced that we would perform some verification steps to deter those participants who would otherwise be tempted to cheat.
- For one of the datasets (dataset A), we provided a different set of target labels to each participant, without letting them know. In this way, if two teams exchanged labels, their resulting poor performance should be suspicious. This would alert us and require us to proceed with further checks, such as asking the participants to provide their code.

Our analysis of the performances on dataset A did not give us any reason to suspect that anyone had cheated (see Appendix A for details). During the verification phase we asked the participants to repeat their experiments on dataset A, this time providing the same labels to everyone. Those are the results provided in the result tables.

3. Baseline results

We uploaded baseline results to the website for the development datasets under the name "Reference". The majority of these submissions used linear kernel ridge regression as the base classifier, where the regularisation parameter was tuned by minimising the virtual leave-one-out cross-validation estimate of the sum-of-squared errors, i.e. Allen's PRESS statistic (Allen, 1974; Saadi et al., 2007). The best results, shown in Table 3, were generally obtained using either passive learning (all labels queried at once) or random active learning, where samples are chosen at random for labeling by the oracle. The one exception was the HIVA dataset, where a naïve Bayes classifier was found to work well, with a Bayesian active learning strategy, where samples were submitted to be labelled in decreasing order of the variance of the posterior prediction

Table 3: Best benchmark results for the development and final datasets. The mean rank on test datasets is 3.833.

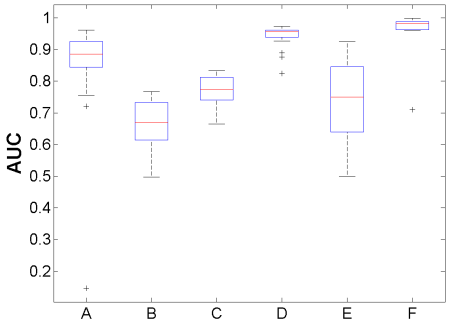
Dataset	Experiment	Classifier	Strategy	AUC	ALC	Rank
HIVA	gcchiva4	Naïve Bayes	Bayesian	0.805504	0.328535	—
IBN_SINA	gccibnsina1	Linear KRR	Random	0.978585	0.813690	—
NOVA	gccnova1	Linear KRR	Random	0.991841	0.715582	—
ORANGE	gccorange1	Linear KRR	Random	0.814340	0.283319	—
SYLVA	gccsylva1	Linear KRR	Random	0.996240	0.921228	—
ZEBRA	gcczebra1	Linear KRR	Random	0.785913	0.416948	—
Avicena	gccA004v	Linear KRR	Random	0.883768	0.586001	3
Banana	gccb1	Linear KRR	Passive	0.720291	0.370762	3
Chemo	gccc4	Linear KRR	Random	0.814450	0.301776	5
Docs	gccd2	Linear KRR	Random	0.962951	0.651222	6
Embryo	gcce1	Linear KRR	Passive	0.773262	0.496610	5
Forest	gccf2	Linear KRR	Random	0.954557	0.821711	1

of the probability of class membership. For a full description of the baseline methods, see Cawley (2011).

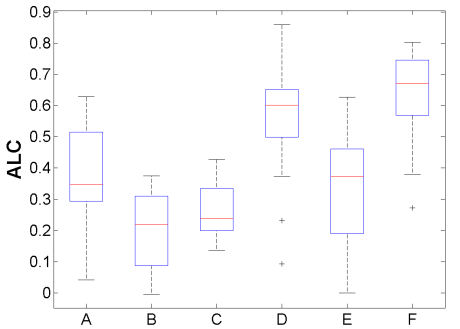
It is interesting to note that a simple linear classifier, with passive learning, or random active learning strategies performs so well (as indicated by the rankings, shown in Table 3). The overall rank of 3.833 for those submissions is subject to a strong selection bias resulting from choosing the best of the four baseline submissions for each benchmark. A more realistic overall ranking is obtained by looking at the results for linear KRR with random active learning (gccA002v, gccb2, gccc2, gccd2, gcce2 and gccf2), which gives an overall rank of 4.667, which would have been sufficient to achieve runner-up status in the challenge. This shows that active learning is an area where further research and evaluation may be necessary to reliably improve on such basic strategies.

4. Challenge results

The challenge attracted a large number of participants. Over 300 people registered to gain access to the data and participate in the development phase. For the final test phase 30 teams were formed, each comprised of between 1 and 20 participants. This level of participation is remarkable for a challenge that requires a deep level of commitment for participation because of the specialized nature of the problem and the iterative submission protocol (participants must query for labels and make predictions by interacting with the website).



(*a*)



(*b*)

Figure 1: **Distribution of results**. *We show box-whiskers plots for the various datasets. The red line represents the median, the blue boxes represent the quartiles, and the whiskers represent the range, excluding some outliers plotted individually as crosses. (a) Area under the ROC curve for the last point on the learning curve. (b) Area under the learning curve.*

It is difficult to make a fair assessment of the results on development data sets because the participants were allowed to perform multiple experiments on the same dataset, the knowledge of the labels obtained in previous experiments may have implicitly or explicitly been used in later experiments. Hence we report only the results on the final test sets for which the participants were only allowed to perform one single experiment. The distribution of performance with respect to AUC and ALC are shown in Figure 1. The results of the top ranking teams for each final dataset are found in Table 4. We also plotted the learning curves of the top ranking participants overlaid on top of all the other learning curves (Figures 2 to 7).

We encouraged the participants to enter results on multiple datasets by exponential scaling of the prizes with the number of wins. However, no team ended up winning on more than one dataset. The remaining prize money has been used to provide travel grants to encourage the winners to attend the workshop. For those participants who entered results on all 6 datasets, we performed a global ranking according to their average rank on the individual datasets. The overall winner by average rank (average rank 4.2) is the Intel team (Alexander Borisov and Eugene Tuv), who already ranked among the top entrants in several past challenges. The runner up by average rank (average rank 4.8) is the ROFU team of National Taiwan University (Ming-Hen Tsai and Chia-Hua Hu). Other members of this research group headed by Chin en Lin have also won several machine learning challenges. The next best ranking teams are IDE (average rank 5.7) and Brainsignals (average rank 6.7). The team TEST (Zhili Wu) made entries on only 5 datasets, but did also very well (average rank 6.4).

We briefly comment on the methods used by these top entrants:

- The Intel team used a probabilistic version of the query-by-committee algorithm (Freund et al., 1997) with boosted Random Forest classifiers as committee members (Borisov et al., 2006). The batch size was exponentially increasing, disregarding the estimated model error. Some randomness in the selection of the samples was introduced by randomly sampling examples from a set of top candidates. No use was made of unlabeled data. The technique used generated very smooth learning curves and reached high levels of accuracy for large numbers of training samples. The total run time on all development datasets on one machine is approximately 6-8 hours depending on model optimization settings. The method does not require any pre-processing, and naturally deals with categorical variables and missing values. The weakness of the method is at the beginning of the learning curve. Other methods making use of unlabeled data perform better in this domain.

- The ROFU team used Support Vector Machines (SVMs) (Boser et al., 1992) as a base classifier and a combination of uncertainty sampling and query-by-committee as active learning strategy (Tong and Koller, 2002). They made use of the unlabeled data (V. Sindhwani, 2005; Sindhwani and Keerthi, 2006) and they avoided sampling points near points already labeled. No active learning was performed on dataset B and E (inferring from the development dataset results

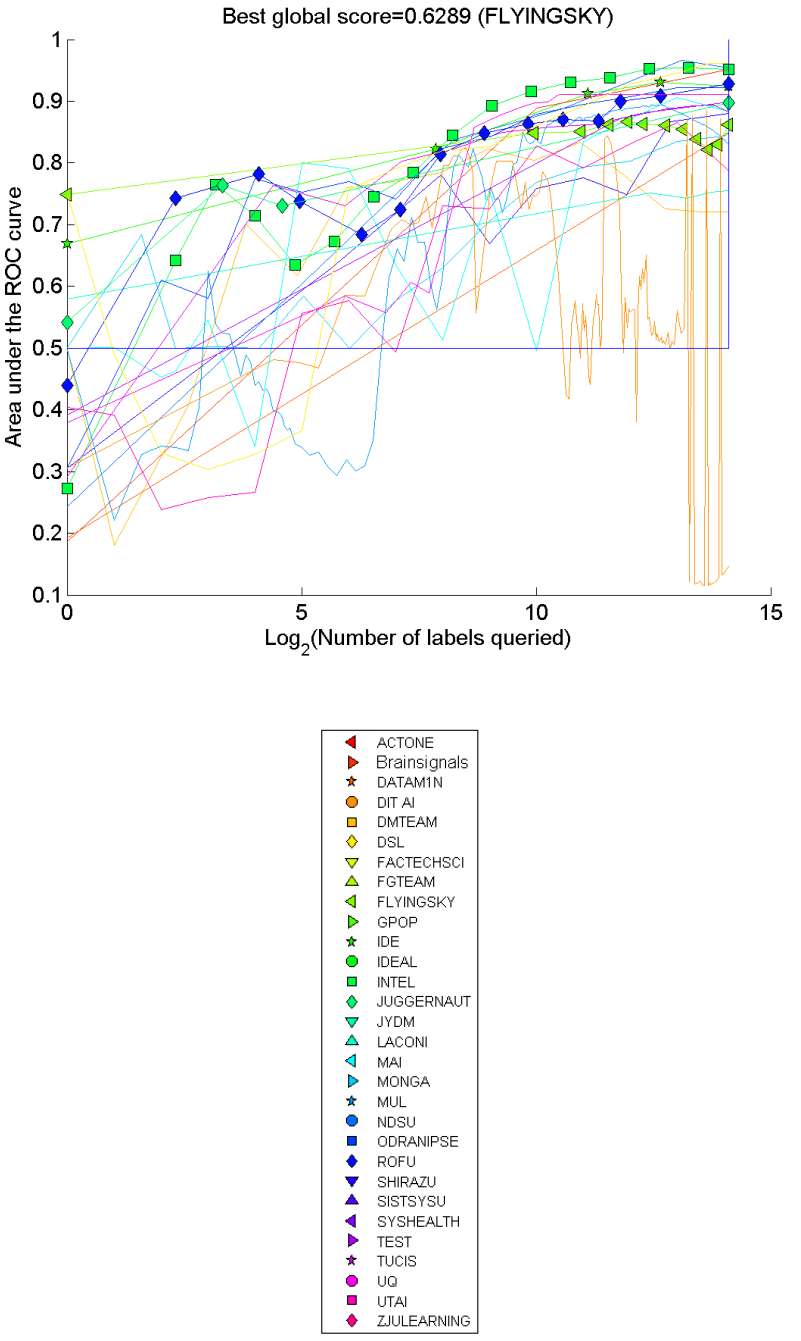


Figure 2: *Learning curves for dataset A.*

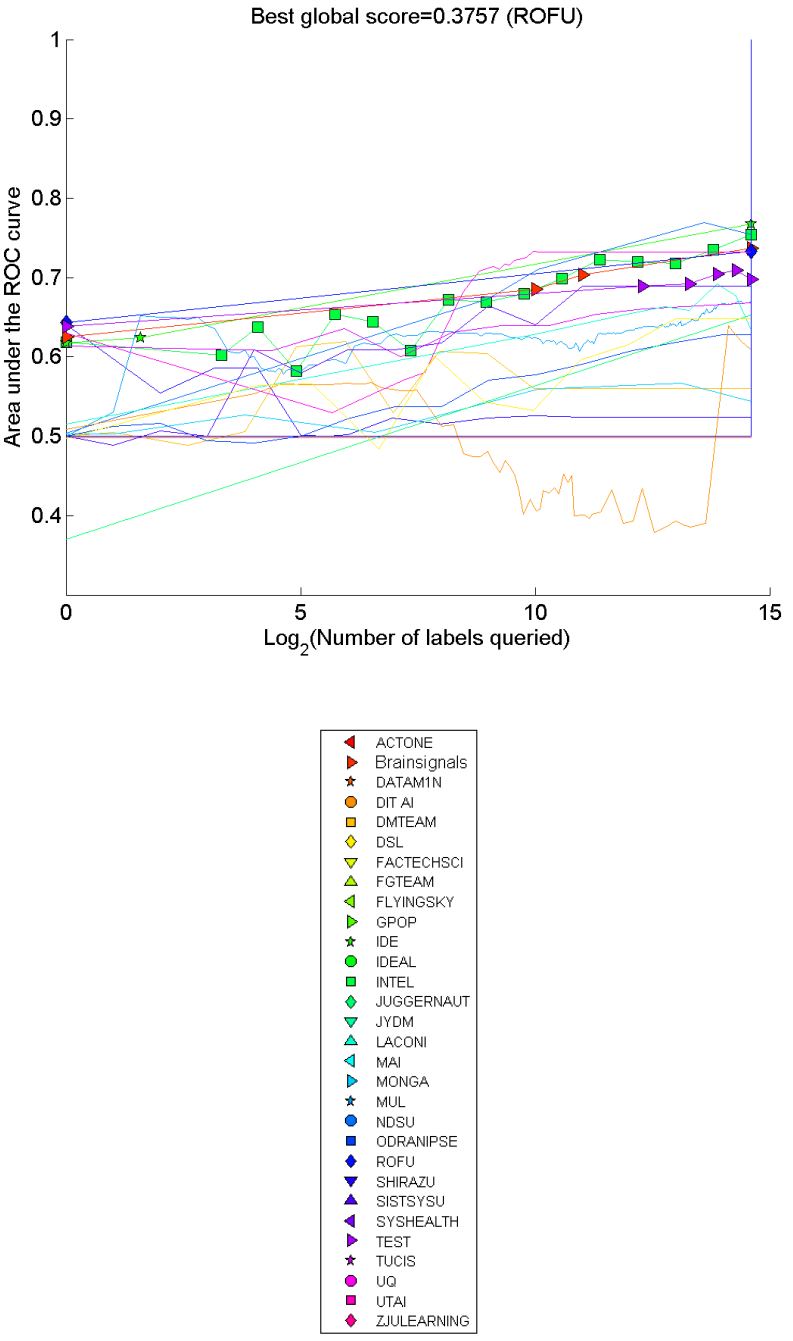


Figure 3: *Learning curves for dataset B.*

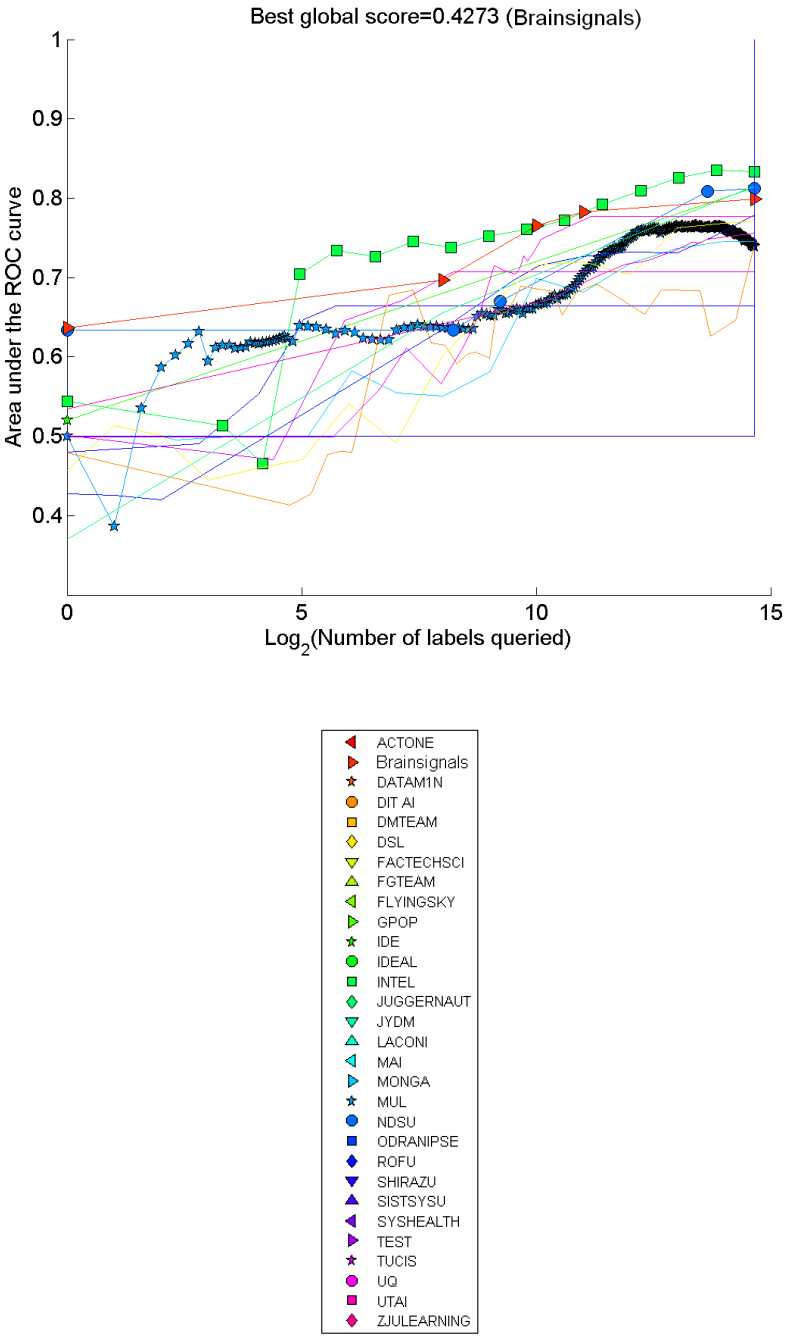


Figure 4: *Learning curves for dataset C.*

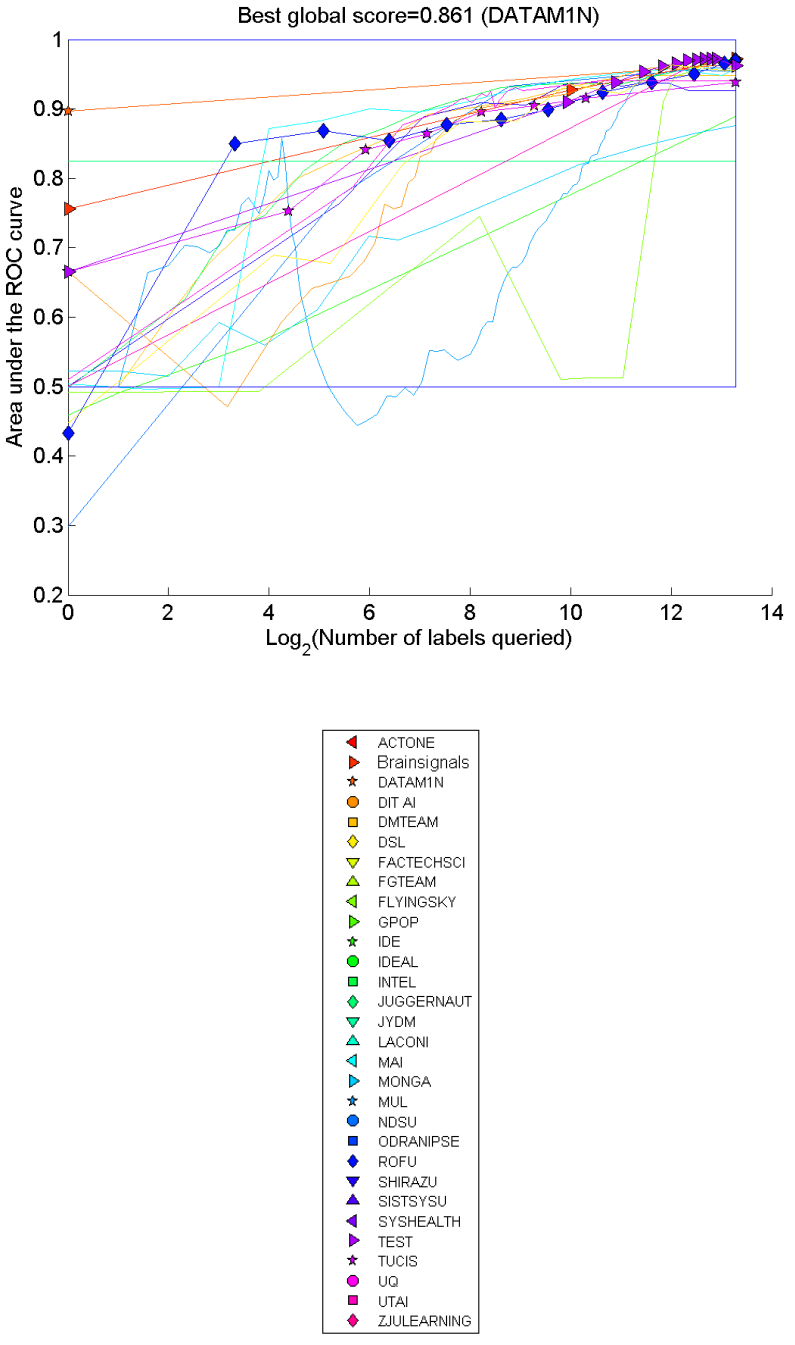


Figure 5: *Learning curves for dataset D.*

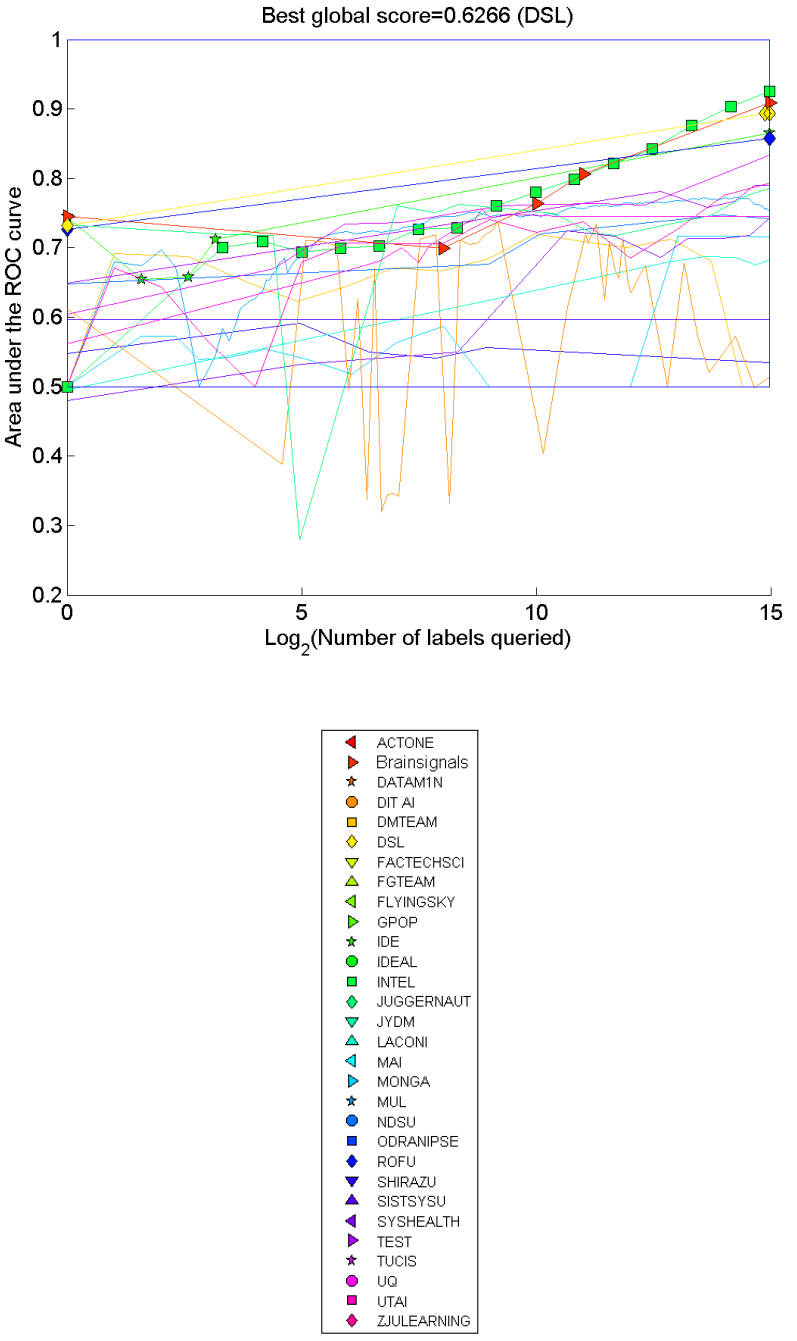


Figure 6: *Learning curves for dataset E.*

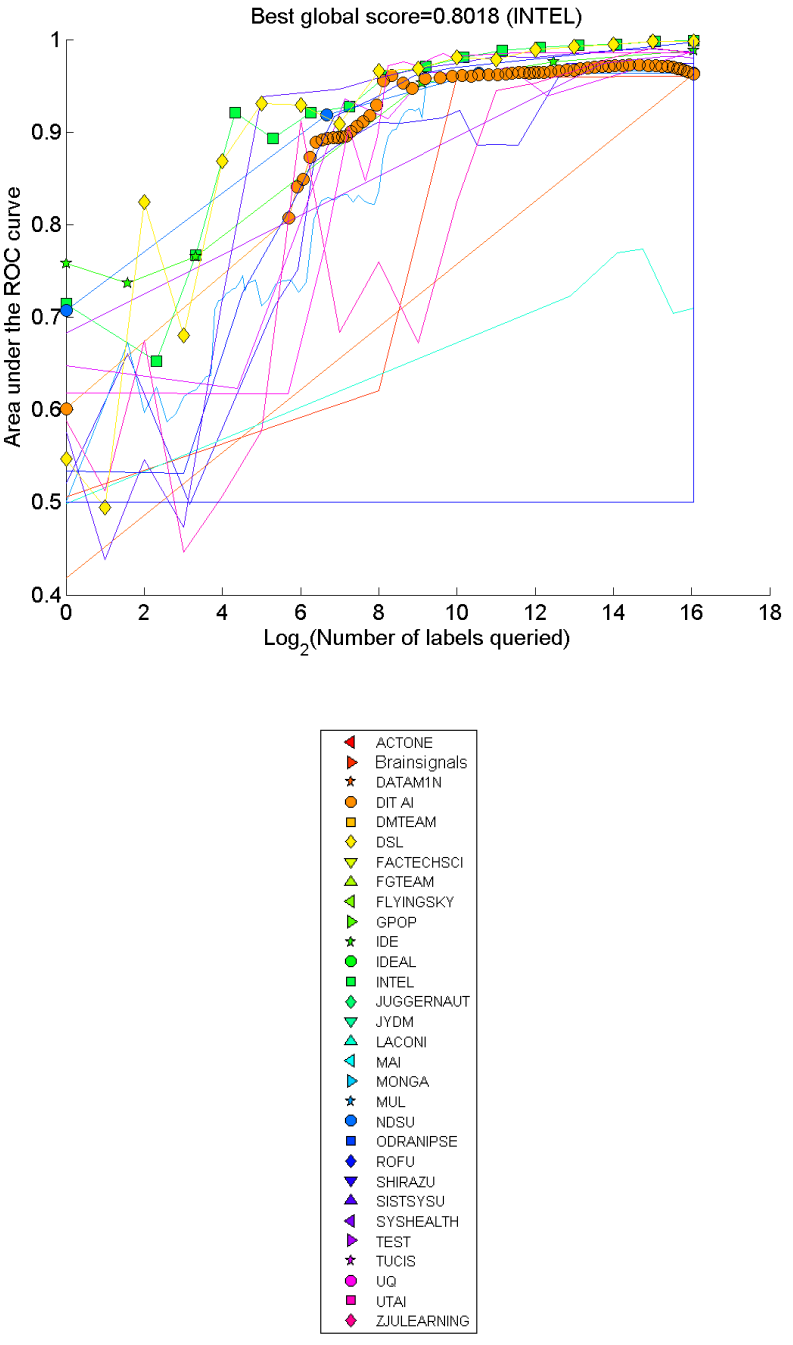


Figure 7: *Learning curves for dataset F.*

that active learning would not be beneficial for such data). The method employed for learning from unlabeled data must not have been very effective because the results at the beginning of the learning curves are quite bad on some datasets (dataset C and F), but the performances for a large number of labeled examples are good. The authors report that using SVMs is fast so they could optimize the hyper-parameters by cross-validation.

- The IDE team used hybrid approaches. In the first few queries, they used semi-supervised learning (*i.e.,* make use of both labeled and unlabeled data with cluster-and-label and self-training strategies), then switch to supervised learning. For active learning, they combined uncertainty sampling and random sampling. Logistic regression and k-means clustering were used when the number of labeled examples is very small (≤ 100). Boosted decision trees were used when the amount of labeled examples is large (≥ 500). The authors think that getting the representative positive examples in the first few queries was key to their success. Indeed, the authors had very few points on their learning curve and their performance for small number of examples on several datasets determined their good rank in the challenge on most datasets.

- The TEST team did not make any attempt to use unlabeled data and queried a large number of labels at once ($\simeq 2000$ examples). Hence its good performance in the challenge are essentially based on the second part of the learning curve. This is a conservative strategy that takes no risk in the first part of the learning curve, which from our point of view was the most interesting. The classifier used is logistic regression and the active learning strategy is uncertainty sampling. The learning curves are smooth. Hence, the use of uncertainty sampling with an sufficiently large initial pool of example seems to be a viable strategy.

- The Brainsignals team did not perform active learning per se. The strength of the entries made and their good ranking in the challenge stem from a good first point in the learning curve obtained with a semi-supervised learning method based on spectral clustering (Zhu, 2005). Then very few points are made on the learning curve at 256, 1024, and all samples. Random sampling was used and classical model selection techniques with cross-validation to select among ensemble of decision trees, linear classifiers, and kernel-based classifiers.

Several participants found that uncertainty sampling and query-by-committee, without introducing any randomness in the selection process, may perform worse than random sampling (see also Section 3 on baseline methods). To illustrate how things can go wrong when strictly using uncertainty sampling, we show in Figure 8 the learning curve of one team on dataset D who used such strategy for active learning. There is a catastrophic decrease in performance in the middle of the learning curve. Query by committee performs better than uncertainty sampling both in randomized and non-randomized settings. Techniques for pro-actively sampling in regions with low densities of labels were reported not to yield significant improvements. Ensemble methods

Table 4: Result tables for the top ranking teams.

Dataset A		
Team	AUC (Ebar)	ALC
Flyingsky	0.8622 (0.0049)	0.6289
IDE	0.9250 (0.0044)	0.6040
ROFU	0.9281 (0.0040)	0.5533
JUGGERNAUT	0.8977 (0.0036)	0.5410
Intel	0.9520 (0.0045)	0.5273

Dataset B		
Team	AUC (Ebar)	ALC
ROFU	0.7327 (0.0034)	0.3757
IDE	0.7670 (0.0038)	0.3754
Brainsignals	0.7367 (0.0043)	0.3481
TEST	0.6980 (0.0044)	0.3383
Intel	0.7544 (0.0044)	0.3173

Dataset C		
Team	AUC (Ebar)	ALC
Brainsignals	0.7994 (0.0053)	0.4273
Intel	0.8333 (0.0050)	0.3806
NDSU	0.8124 (0.0050)	0.3583
IDE	0.8137 (0.0051)	0.3341
MUL	0.7387 (0.0053)	0.2840

Dataset D		
Team	AUC (Ebar)	ALC
DATAM1N	0.9641 (0.0033)	0.8610
Brainsignals	0.9717 (0.0033)	0.7373
ROFU	0.9701 (0.0032)	0.6618
TEST	0.9623 (0.0033)	0.6576
TUCIS	0.9385 (0.0037)	0.6519

Dataset E		
Team	AUC (Ebar)	ALC
DSL	0.8939 (0.0039)	0.6266
ROFU	0.8573 (0.0043)	0.5838
IDE	0.8650 (0.0042)	0.5329
Brainsignals	0.9090 (0.0039)	0.5267
Intel	0.9253 (0.0037)	0.4731

Dataset F		
Team	AUC (Ebar)	ALC
Intel	0.9990 (0.0009)	0.8018
NDSU	0.9634 (0.0018)	0.7912
DSL	0.9976 (0.0009)	0.7853
IDE	0.9883 (0.0013)	0.7714
DIT AI	0.9627 (0.0017)	0.7216

combined with query-by-committee active learning strategies yielded smooth learning curves. Good performances for very small number of examples (≤ 100) were achieved only by teams using semi-supervised learning strategies.

4.1. Methods employed

In what follows, for each category of methods (active learning, pre-processing, feature selection, etc.) we report the fraction of participants having used each method. Note that the sums of these numbers do not necessarily add up to 100%, because the methods are not mutually exclusive and some participants did not use any of the methods.

We analyzed the information provided by the participants in the fact sheets:

- **Active Learning and use of Unlabeled Data:** In Figure 9, we show a histogram of the type of active learning methods employed. **Most of the participants used "uncertainty sampling"** as part of their strategy (81%) or random sampling (47%). Query-by-committee was also very popular (38%). Interestingly no participant used Bayesian active learning, **20% of the participants used no**

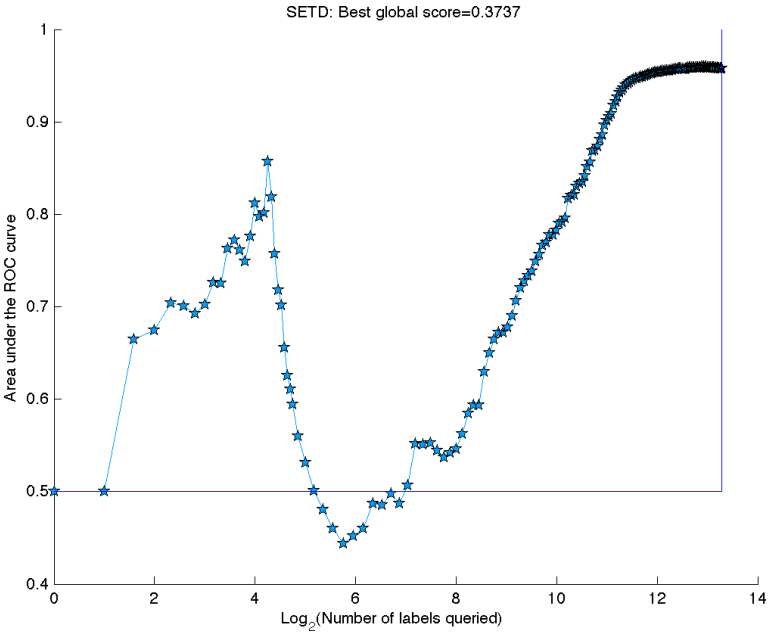


Figure 8: *Example of learning curves for dataset D using the uncertainty sampling strategy.*

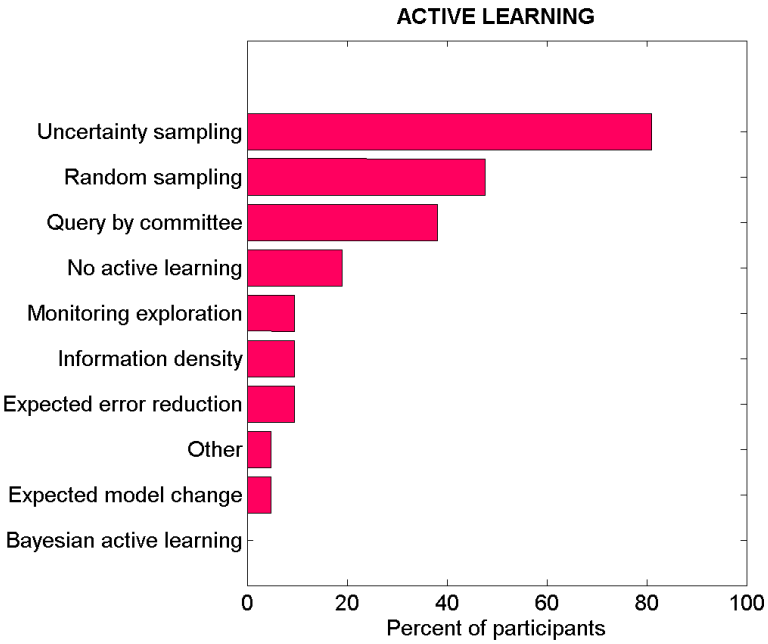


Figure 9: *Active Learning Methods Employed.* Also worth noting: 57% of the participants used unlabeled data (with or without active learning).

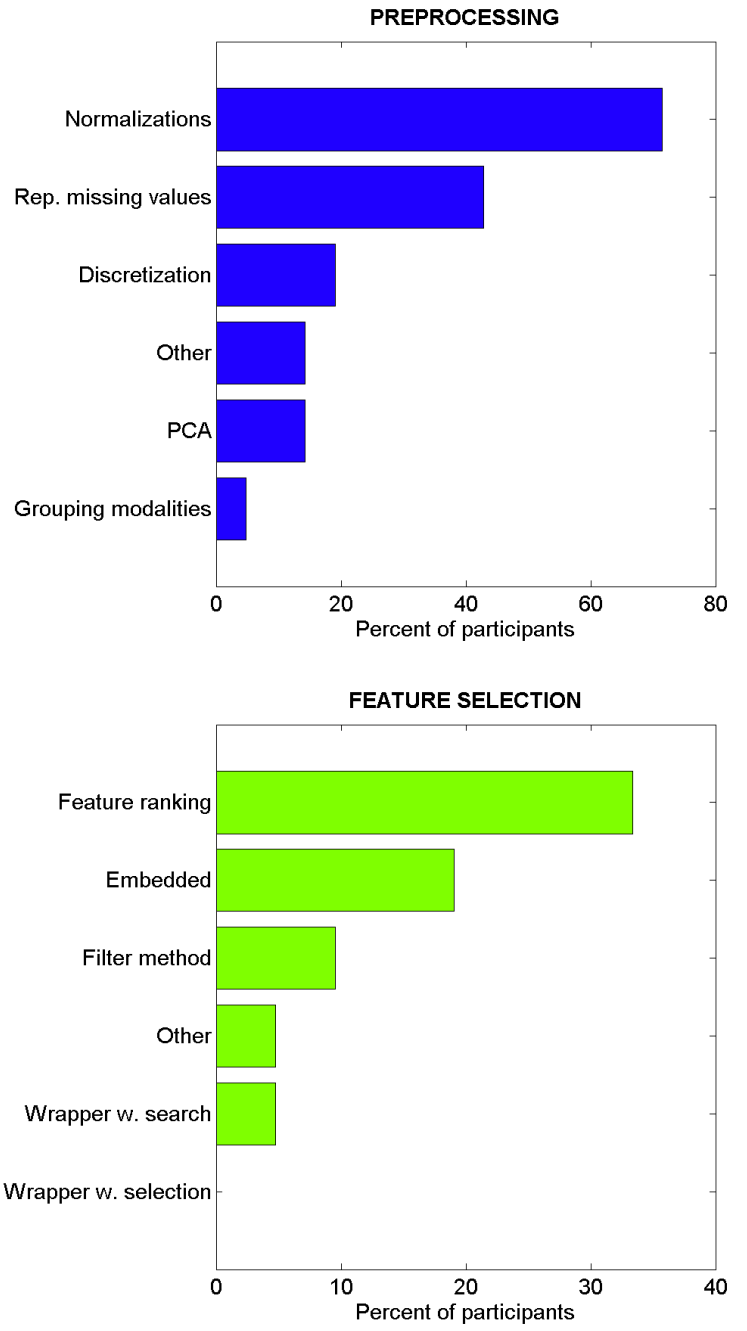


Figure 10: *Preprocessing and feature selection.*

active learning at all but **57% made use of unlabeled data** – not shown on the figure.

- **Pre-processing and Feature Selection:** In Figure 10 we show histograms of the algorithms employed for pre-processing, feature selection. Few participants did not use any pre-processing (14%) and most participants performed **data normalizations** (71%). A large fraction of the participants used **replacement of missing values** by the mean or the median or a fixed value (43%). Principal Component Analysis (PCA) was seldom used and reported not to bring performance improvements. Discretization and other types of pre-processing were not used very much in this challenge. About half of the participants (52%) used some form of **feature selection**. **Feature ranking methods** were the most widely used feature selection methods (33%) and other filter methods were also used (9%). Compared to previous challenges, many participants used embedded methods (19%), but consistent with previous challenges, wrapper methods with extensive search were unpopular.

- **Classification algorithm and model selection:** In Figure 11 we show the classification algorithms used. Most participants either used as part of their design a **linear classifier** (62%) or a **non-linear kernel method** (43%). **Decision trees** and **Naïve Bayes** are also quite popular (each have a 33% usage), while all the other methods, including neural networks, are much less popular (less than 20% usage). The statistics on loss function usage reveal the increasing popularity of the **logistic loss** (38%). The **hinge loss** used for SVMs remains popular (29%). Other loss functions, including the exponential loss (boosting) and the square loss (ridge regression and LS-SVMs) each have less than 20% reported usage. We also collected statistics not shown on those figures about regularizers, ensemble methods and model selection. Consistent with the popularity of linear and kernel methods in this challenge, a large fraction of the participants used regularization, with 43% usage of **2-norm regularizers** and 19% usage of 1-norm regularizers. Most participants made use of some ensemble method (about 80%) including 33% usage of bagging and 29% usage of boosting. The wide use of ensemble methods may explain the relatively low use of model selection methods (62%); cross-validation methods such as K-fold and leave-one-out remain the most popular (43%).

We also analyzed the fact sheets with respect to the software and hardware implementation:

- **Hardware:** Many participants made use of parallel implementations (67% used multiple processor computers and 24% ran experiments in parallel). Memory usage was relatively modest (38% used less than 2 GB and 33% less than 8 GB).

- **Software:** Few participants are Mac users; most use either Windows of Linux (about half and half). In terms of programming languages (Figure 12), higher

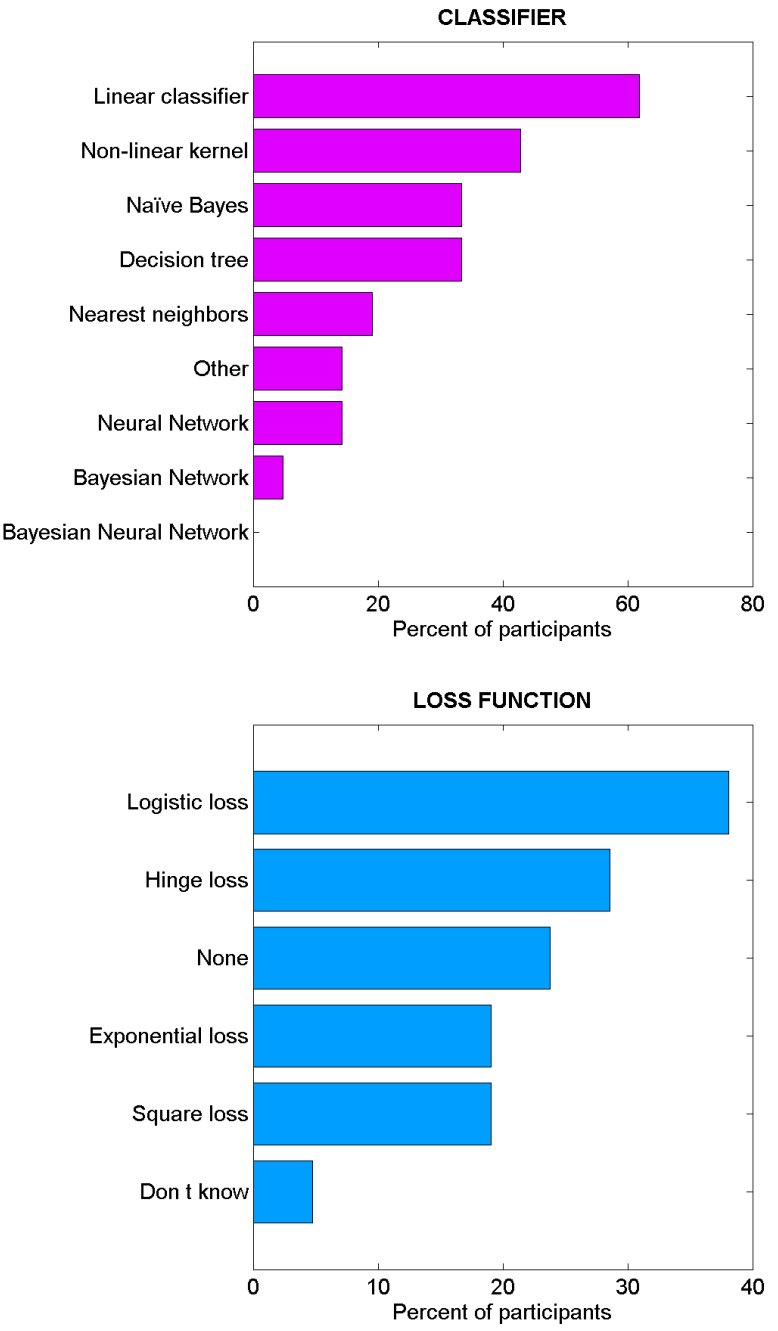


Figure 11: *Classification algorithms and loss functions.*

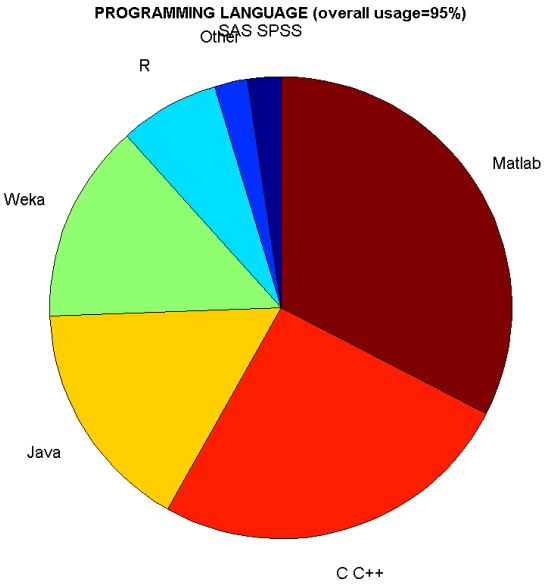


Figure 12: *Programming Languages.*

level languages (Matlab, R, and Weka) are most popular, Matlab coming first (67%), but C/C++ and Java are also used significantly. About 70% of the participants wrote their own code, 30% of which are making it freely available.

The amount of human effort involved in adapting the code to the problems of the challenge varied but was rather significant because about half of the participants reported spending less than two weeks of programming while half reported more than 2 weeks. The amount of time spent experimenting (computer effort) was distributed similarly. While the majority of participants reported having enough development time (67%) and enough test time (62%), a large fraction of participants ran out of time to do what they wanted.

5. Discussion

5.1. Statistical significance of the results

One of the aims of this competition is to try to assess the quality of active learning methods in an unbiased manner. To this end, it is important to examine whether the differences between the top ranking methods and the lowest ranking methods can be attributed to chance or are there are significant differences.

We performed a statistical test, specifically designed for settings in which multiple classifiers are being tested with multiple datasets: the Friedman test (Demšar, 2006).

The ranks of the algorithms on each dataset are used. A tabulated statistic is derived from the average ranks of the algorithms to test the null hypothesis that all algorithms are equivalent so their ranks should be equal. The above tests call for a full score matrix (teams vs. datasets) so we restricted our analysis for the set of 11 teams who submitted the results for all 6 datasets. We used the tests with a significance level of $\alpha = 0.05$. The Friedman test turned out highly significant (with a p-value of 0.0019), so there are significant differences in performances among the teams.

Since this first test was successful (i.e., the null hypothesis of equivalence between algorithms was rejected) we ran a post-hoc test as recommended by (Demšar, 2006): the Nemenyi test. That test looks for significant differences between the performances of given algorithms and the rest of them. Accordingly, only two teams to be significantly different: the first (INTEL group) and the last (DIT AI Group).

5.2. Post-challenge experiments

Lemaire and Salperwyck (2010) performed systematic post-challenge experiments to assess a number of classifiers used by the challenge participants in a controlled manner. The authors decoupled the influence of the active learning strategy from that of the strength of the classifier by simply using a random sampling strategy (passive learning). To reduce the variance on the results, each experiment was repeated ten times for various drawings of the training samples, but always starting from the same seed example provided to the participants. Learning curves were drawn by averaging the performances obtained for training set sizes growing exponentially $2^1, 2^2, 2^3, 2^4, ...$ The performances of the various classifiers were compared with the ALC and the AUC of the final classifier trained on all the examples. The authors compared several variants of Naive Bayes, logistic regression, several decision trees and ensembles of decision trees. The study allowed the authors to confirm trends observed in the analysis of the challenge and reported elsewhere in the literature:

- Tree classifiers perform poorly, particularly for small number of examples, but ensembles or trees are among the best performing methods.
- Generative models illustrated by naive Bayes methods perform better than discriminative models for small numbers of examples.

Interestingly, the challenge participants did not capitalize on the idea of switching classifier between when progressing through the learning curve and performed model selection globally using the ALC for a fixed classifier. The study reveals a ranking of classification methods similar to that of the challenge, using using only passive learning. This observation suggests that the choice of the classifier may have been a more determining factor in winning the challenge that the use of a good active learning strategy. The relative efficacy of active learning strategies for given classifiers remains to be systematically assessed and will be the object of further studies.

5.3. Comparison of the datasets

The datasets chosen presented a range of difficulties, as illustrated by the AUC results obtained ad the end of the learning curve (Figure 1-a). The median values are ranked as follows: $F > D > A > C > E > B$. The best AUC results are uniformly obtained by ensembles of decision trees for all datasets. For dataset F (Ecology), the best AUC result (0.999) is obtained by the Intel team using a combination of boosting and bagging of decision trees. This confirms results obtained with this dataset in previous challenges in which ensembles of tree classifiers won (Guyon et al., 2006). However, other methods including ensembles of mixed classifiers (NDSU) and SVMs (DSL and DIT AI) also work well. For dataset D (Text), the best AUC performance is also high (0.973) and it is obtained by the Intel team, but the profile of other top ranking methods is rather different: it includes mostly linear classifiers, which are close contenders. For dataset A (Handwriting Recognition), the best final AUC is also obtained by the Intel team with ensembles of decision trees (0.952), but other methods including heterogeneous ensembles and kernel methods line SVMs work well too. The best results on dataset C (Chemoinformatics) are also obtained by the Intel team with the same method (0.833) and are significantly better than the next best result obtained with an heterogeneous ensemble of classifiers. Dataset E (Embryology) exhibits the largest variance in the results, hence a good model selection is crucial for that dataset. The AUC result of the Intel team with ensembles of decision trees (0.925) is statistically significantly better than the next best result. The best AUC result (0.766) on the hardest dataset (dataset B, Marketing) was obtained by a different team (IDE) but also with an ensemble of tree classifiers (the Intel result comes close).

Another point of comparison between datasets is the shape of the learning curves and the success of active learning strategies. According to Figure 1-b, the ranking of dataset median performance with respect to ALC is similar to the ranking by AUC $F > D > E > A > C > B$ (only the position of E differs), but the variance is a lot higher. This shows that when it comes to active learning, it is easy to do things wrong! The analysis of Figures 2 to 7 provides some insight into the learnability of the various tasks from few examples. For dataset F, the learning curve of the best ranking participants climbs quickly and with as little as 4 examples a performance larger than AUC=0.9 is achieved. For dataset D, making a good initial semi-supervised entry using the seed example was critical. Then, the learning curves climb rather slowly. This can be explained by the relatively polarized separation chosen for the task: computer related topics *vs.* everything else. Learning from just one example came a long way. Rather similar learning curves are obtained for dataset E. The teams who did not perform semi-supervised learning got much worse results in the first part of the learning curve. For dataset A, there is a lot of variance in the first part of the learning curve until about 32 examples are given. This high variance was also observed in the post-challenge experiments. It may be due to the high heterogeneity of the classes. For dataset C, the Intel team who obtained the best final AUC and has a learning curve dominating all others starting at 32 examples did not do well for the small number of examples. Other teams including MUL used semi-supervised learning strategies and climbed the learn-

ing curve much faster. Dataset B presents the flatest learning curves. All top ranking learning curve start with an AUC between 0.61 and 0.65 and end up with and AUC between 0.68 and 0.72. Virtually all active learning strategies are found among the top ranking participants for that dataset but, according to our post-challenge experiments, random sampling does just as well.

5.4. Compliance with the rules of the challenge

We struggled to find a protocol that would prevent violations of the rules of the challenge. We needed a mechanism to ensure that the participants could not gain knowledge of the labels unless they had legitimately purchased them (for virtual cash) from the website. This implied that entries made to gain access to labels could not be corrected and that the participating teams could not exchange information about labels. Inevitably, in the course of the challenge, some participants made mistakes in their submissions that they could not correct. The verification process that we implemented (see Appendix A) was not uniformly well received because some participants felt that, had they known in advance that the verification round counted for ranking, they would have spent more effort on it. Hence, this generated some frustration. However, the validation process gave us confidence that the participants respected the rules, which is important in order to be able to draw valid conclusions from the analysis of the challenge.

5.5. Lessons learned about the challenge protocol

This challenge is one of the most sophisticated challenges that we have organized because it involved a complex website back-end to handle the queries made by the participants and manage the status of their on-going experiments. Some participants complained that the manual query submission/answer retrieval through web form was very inconvenient and time consuming. In future challenges we plan to automate that process by providing scripts to make submissions.

The nature of the challenge also made the use of part of the dataset for validation impossible during the development period. We had to resort to using different datasets during the development period and the final test period. However, due to limited resources, not all the final datasets were completely different from their matched development dataset. For dataset A, we added more samples and changed the targets, for dataset B, we changed the features and the targets but the samples were the same, for dataset C, the task was entirely new. For dataset D, we changed the features and the targets, but the samples were the same, for datasets E and F, we sub-sampled the data differently and provided different targets. For F, we also changed the features. For all datasets, the order of samples and features were randomized. In this way, the participants could not re-use samples or models from the development phase in the final phase. However, even through we did not explicitly match the datasets between the two phases, it was not difficult to figure out the match. Some participants seem to have made use of this information to select strategies of active learning (or decide not to perform active learning at all on certain datasets).

Table 5: The empirical means, (μ_0, μ_1) and standard deviations (σ_0 and σ) of the performances (AUC) on the first point (no label purchased) and last point (all available labels purchased) of the learning curve, respectively.

Dataset	μ_0	σ_0	μ_1	σ_1
A	0.416	0.164	0.879	0.068
B	0.572	0.063	0.658	0.086
C	0.509	0.080	0.782	0.034
D	0.535	0.122	0.953	0.021
E	0.595	0.106	0.733	0.142
F	0.586	0.101	0.981	0.015

To put most emphasis on learning from few labeled examples, our evaluation metric put higher weight on the beginning of the learning curve by choosing a logarithmic scaling of the x-axis. This was criticized because there is a lot of variance in the first part of the learning curve and can cause some participants to win by chance. This is a valid concern that, outside of the constraints of a challenge, may be addressed by averaging over multiple experiments performed on data sub samples. In the context of a challenge, samples for which labels have been purchases cannot be re-used, so averaging procedures are excluded. Rather, we have offered the possibility of working on several datasets. The participants who performed consistently well on all datasets are unlikely to have won by chance. In retrospect, giving prizes for individual datasets might have encouraged to use "gambling strategies": some participants provided only two points in the learning curve, the first and the last one. If by chance their first point was good on one of the 6 datasets, they could win one of the 6 prizes. In retrospect, we might have been better off imposing the condition that all the participants return results on all datasets and make a single ranking based on the average rank on all 6 tasks. We could also regularize the performance measure e.g. by a weighted average of performances obtained on different parts of the learning curve, such as to take into account the difference of their variances.

Table 5 details the empirical standard deviations of participants' performances on various datasets for the first and last points on the learning curve. As can be expected, for the majority of the datasets (A, C, D and F) the standard deviation on the first point and is considerably larger that that of the last point; On the remaining datasets (B and E) the differences are small. It may be a coincidence but the two datasets with largest standard deviations are exactly those for which the winning entries did not actually use active learning but used semi-supervised learning to optimize performance on the first point. Analyzing the spread of results by inter-quartile range as a measure for the spread of the performances, which is more resistant to outliers, produces the same picture.

6. Conclusion

The results of the Active Learning challenge exceeded our expectations in several ways. First we reached a very high level of participation despite the complexity of the challenge protocol and the relatively high level of expertise needed to enter the challenge. Second, the participants explored a wide variety of techniques. This allowed us to draw rather strong conclusions, which include that: (1) semi-supervised learning is needed to achieve good performance in the first part of the learning curve, and (2) some degree of randomization in the query process is necessary to achieve good results. These findings have been confirmed in large on-going Monte-Carlo experiments, the initial results of which are presented in Cawley (2011). This challenge proved the viability of using our Virtual Laboratory in challenges requiring users to interact with a data generating system. We intend to improve it to further automate the query submission process and use it in upcoming challenges on experimental design.

Acknowledgments

This project is an activity of the Causality Workbench supported by the Pascal network of excellence funded by the European Commission and by the U.S. National Science Foundation under Grant N0. ECCS-0725746. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Additional support was provided to fund prizes and travel awards by Microsoft and Orange FTP. We are very grateful to all the members of the causality workbench team for their contributions and in particular to our co-founders Constantin Aliferis, Greg Cooper, André Elisseeff, Jean-Philippe Pellet Peter Spirtes, and Alexander Statnikov, and to the advisors and beta-testers Olivier Chapelle, Amir Reza Saffari Azar and Alexander Statnikov. We also thank Christophe Salperwyck for his contributions to the post-challenge analyses. The website was implemented by MisterP.net who provided exceptional support. This project would not have been possible without generous donations of data. We are very grateful to the data donors: Chemoinformatics – Charles Bergeron, Kristin Bennett and Curt Breneman (Rensselaer Polytechnic Institute, New York) contributed a dataset, used for final testing. Embryology – Emmanuel Faure, Thierry Savy, Louise Duloquin, Miguel Luengo Oroz, Benoit Lombardot, Camilo Melani, Paul Bourgine, and Nadine Peyriéras (Institut des systèmes complexes, France) contributed the ZEBRA dataset. Handwriting recognition – Reza Farrahi Moghaddam, Mathias Adankon, Kostyantyn Filonenko, Robert Wisnovsky, and Mohamed Chériet (Ecole de technologie supérieure de Montréal, Quebec) contributed the IBN SINA dataset. Marketing – Vincent Lemaire, Marc Boullé, Fabrice Clérot, Raphael Féraud, Aurélie Le Cam, and Pascal Gouzien (Orange, France) contributed the ORANGE dataset, previously used in the KDD cup 2009. We also reused data made publicly available on the Internet. We are very grateful to the researchers who made these resources available: Chemoinformatics – The National Cancer Institute (USA) for the HIVA dataset. Ecology – Jock A. Blackard, Denis J. Dean, and Charles W. Anderson (US Forest Service, USA) for the

SYLVA dataset (Forest cover type). Text processing – Tom Mitchell (USA) and Ron Bekkerman (Israel) for the NOVA datset (derived from the Twenty Newsgroups dataset.

References

D. M. Allen. The relationship between variable selection and prediction. *Technometrics*, 16:125–127, 1974.

A. Borisov, V Eruhimov, and E. Tuv. Tree-based ensembles with dynamic soft feature selection. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction Foundations and Applications*, volume 207 of *Studies in Fuzziness and Soft Computing*. Springer, 2006.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992. ACM.

G. C. Cawley. Some baseline methods for the active learning challenge. *Journal of Machine Learning Research, Workshop and Conference Proceedings*, 15, 2011.

D. Cohn, Z. Ghahramani, and M.I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006. ISSN 1533-7928.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

I. Guyon, A. Saffari, G. Dror, and J. Buhmann. Performance prediction challenge. In *IEEE/INNS conference IJCNN 2006*, Vancouver, Canada, July 16-21 2006.

I. Guyon et al. Datasets of the active learning challenge. Technical Report, 2010. URL http://clopinet.com/al/Datasets.pdf.

Vincent Lemaire and Christophe Salperwyck. Post-hoc experiments for the active learning challenge. Technical report, Orange Labs, 2010.

D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. ACM/Springer, 1994.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning (ICML)*, pages 441–448. Morgan Kaufmann, 2001.

K. Saadi, G. C. Cawley, and N. L. C. Talbot. Optimally regularised kernel Fisher discriminant classification. *Neural Networks*, 20(7):832–841, September 2007.

B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 1289– 1296. MIT Press, 2008a.

B. Settles, M. Craven, and S. Ray. An analysis of active learning strategies for sequence labeling tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 20, pages 1069–1078. ACL Press, 2008b.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

H. S. Seung, M. Opper, and H. Sompolinskyy. Query by committee. In *ACM Workshop on Computational Learning Theory*, pages 287–294, 1992.

V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear SVMs. In *SIGIR*, 2006.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2002.

Simon Tong and Daphne Koller. Active learning for parameter estimation in bayesian networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 647–653, 2000.

S.S. Keerthi V. Sindhwani. *Newton Methods for Fast Solution of Semi-supervised Linear SVMs*. MIT Press, 2005.

Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Appendix A. Post-challenge verifications

The protocol of the challenge included several means of enforcing the rule that teams were not allowed to exchange information on the labels, including a manual verification of team membership and a method for detecting suspicious entries described in this appendix.

For one of the datasets (dataset A), we took advantage of the fact that the problem was a multi-class problem, with 15 classes. We assigned at random a different classification problem to each team. The teams were not informed that they were working on different problems.

To hide our scheme, we needed to provide the same seed example to all teams. To make this possible, we created 14 classifications problems for separating class1 or class i *vs.* all other classes, where i varies from 2 to 15. The seed was one example of class 1 (the same for all problems). We assigned the classification problems randomly to the teams. The intention was to detect eventual suspicious entries that could betray that the team tried to use label information acquired "illegally" from another team.

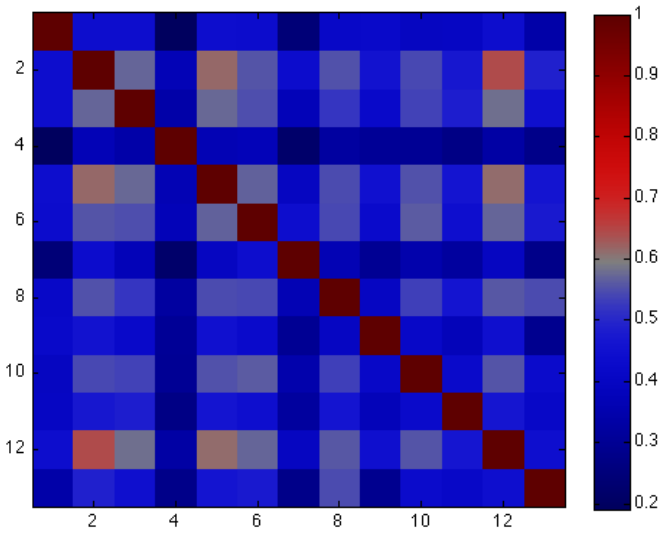


Figure 13: **Correlation matrix between target values for dataset A.** Targets 2 and 12 are particularly correlated.

For each team, we computed the learning curves and the corresponding score (normalized ALC, Area under the Learning Curve) for all possible target values, hence 14 scores. We investigated whether the teams scored better on a problem they were not assigned to, hence for which they could not legitimately purchase labels. Our scheme obviously relies on our ability to determine the significance of score difference. Theoretical confidence intervals for our score (normalized ALC) are not known. There is naturally some variance in the results making it possible that by chance a team would get a better result on one of the problems to which it was not assigned. Furthermore, as illustrated in Figure 13, the problems are correlated, which increases the chances of getting a better score on another problem.

Twenty one teams turned in results on dataset A, including all the competition winners. Only a few teams submitted results in the final phase on other datasets than dataset A, so they could not be checked. However, they scored poorly in the challenge and were not in the top five for any dataset. Hence, our scheme allowed us to verify all the top ranking teams. We proceeded in the following way:

- The teams whose score on their assigned problem was better then their score on all other problems (11 among 21) were declared beyond suspicion. This includes the overall winner (Intel) and 3 other winning teams (Brainsignals, ROFU, and DATAM1N).

- Because the problems were assigned at random, some were not assigned to any team (target values 10 and 11). Among the remaining 10 teams, those having their best score on problems 10 or 11 (5 among 10 remaining teams) were declared beyond suspicion, since none of the teams could purchase label for these tasks. This includes one other winning team (Flyingsky).

- For the remaining 5 teams, we waived suspicion to 4 of them who had a score lower than 0.3 for *any* problem (their score was so close to random guessing that it was easy to score higher on another problem by chance). Most of those teams also scored low on all the other datasets. Only one of those teams scored high on some other datasets (NDSU), but did not win on any dataset.

- There remained only one team (IDE). Their results are better on problem 2 than on their assigned problem, which happens to be problem 12. These results are not suspicious either because those two problems are very correlated.

- For the least conclusive cases, we performed a visual examination of the learning curves to detect eventual suspicious progressions.

In conclusion, none of the verification results raised any suspicion. Admittedly, these results are not very strong because of the imperfections of the test due to noise and label correlation. However, combined with the other measures we took to enforce the rules of the challenge, this test gave us confidence in the probity of all the teams and did not justify further verifications by asking the teams to deliver their code.

Baseline Methods for Active Learning

Gavin C. Cawley GCC@CMP.UEA.AC.UK
School of Computing Sciences
University of East Anglia
Norwich, Norfolk, NR4 7TJ, United Kingdom

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

In many potential applications of machine learning, unlabelled data are abundantly available at low cost, but there is a paucity of labelled data, and labeling unlabelled examples is expensive and/or time-consuming. This motivates the development of active learning methods, that seek to direct the collection of labelled examples such that the greatest performance gains can be achieved using the smallest quantity of labelled data. In this paper, we describe some simple pool-based active learning strategies, based on optimally regularised linear [kernel] ridge regression, providing a set of baseline submissions for the Active Learning Challenge. A simple random strategy, where unlabelled patterns are submitted to the oracle purely at random, is found to be surprisingly effective, being competitive with more complex approaches.

Keywords: pool based active learning, ridge regression

1. Introduction

The rapid development of digital storage devices has led to ever increasing rates of data capture in a variety of application domains, including text processing, remote-sensing, astronomy, chemoinformatics and marketing. In many cases the rate of data capture far exceeds the rate at which data can be manually labelled for the use of traditional supervised machine learning methods. As a result, large quantities of unlabelled data are often available at little or no cost, but obtaining more than a comparatively small amount of labelled data is prohibitively expensive or time consuming. Active learning aims to address this problem by constructing algorithms that are able to guide the labeling of a small amount of data, such that the generalisation ability of the classifier is maximized whilst minimising the use of the oracle. In pool-based active learning, a large number of unlabelled examples are provided from the outset, and training proceeds iteratively. At each step the active learning strategy chooses one or more unlabelled patterns to submit to the oracle, and the classifier updated using the newly acquired label(s). Pool-based active learning is appropriate in many applications, for instance drug design, where the aim is to predict the activity of a molecule against a virus, such as HIV, based on chemometric descriptors. A large number of small molecules have been subjected to chemometric analysis providing a large library of unlabelled data, however *in-vitro* testing is expensive. Active learning would therefore be useful in reducing the cost of

drug design by targeting the effort *in-vitro* testing only on those molecules likely to be effective. There is a significant overlap between active learning and unsupervised or semi-supervised learning as the need for labelled data may be minimised by a learning algorithm that is able to take advantage of the information contained in the unlabelled examples. For a more detailed overview of active learning, see Settles (2009).

This paper describes a set of simple baseline solutions for an open challenge in active learning, described in detail in Guyon et al. (2010). The remainder of the paper is structured as follows: Section 2 provides a brief technical description of the base classifier and active learning strategies employed. Section 3 presents the results obtained using the baseline methods for the development and test benchmark datasets. Finally the work is summarised and conclusions presented in Section 4.

2. Technical Description of Baseline Methods

This section describes the technical detail of the baseline submissions, based on optimally regularised ridge regression, with the pre-processing steps employed, and three very simple active learning strategies.

2.1. Optimally Regularised [Kernel] Ridge Regression

Linear ridge regression is used as the base classifier for those baseline methods for the active learning challenge described in this paper. While more complex non-linear methods could have been used, such as a decision tree (Quinlan, 1986), support vector machine (Boser et al., 1992; Cortes and Vapnik, 1995) or naïve Bayes (e.g. Webb, 2002) classifier , very little labelled data is available at the start of the active learning process, and so a more complex classifier would run a greater risk of over-fitting. In addition, these methods were intended to provide a reasonably competitive baseline representing a fairly basic approach to the problem, and so a simple linear classifier seemed most appropriate. Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{\ell}$ represent the training sample, where $\boldsymbol{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ is a vector of explanatory features for the i^{th} sample, and $y_i \in \{+1, -1\}$ is the corresponding response indicating whether the sample belongs to the positive or negative class respectively. Ridge regression provides a simple and effective classifier that is equivalent to a form of regularised linear discriminant analysis. The output of the ridge regression classifier, $\hat{y}_i$, and vector of model parameters, $\boldsymbol{\beta} \in \mathbb{R}^d$, are given by

$$\hat{y}_i = \boldsymbol{x}_i \cdot \boldsymbol{\beta} \qquad \text{and} \qquad \left[\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\right]\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y}, \tag{1}$$

where $\boldsymbol{X} = [\boldsymbol{x}_i]_{i=1}^{\ell}$ is the data matrix, $\boldsymbol{y} = (y_i)_{i=1}^{\ell}$ is the response vector and the ridge parameter, λ, controls the bias-variance trade-off (Geman et al., 1992). Note that classifiers used throughout this study included an unregularised bias parameter, which has been neglected here for notational convenience. Careful tuning of the ridge parameter allows the ridge regression classifier to be used even in situations with many more features than training patterns (i.e. $d \gg \ell$) without significant over-fitting (e.g. Cawley, 2006). Fortunately the ridge parameter can be optimised efficiently by minimising a

closed-form leave-one-out cross-validation estimate of the sum of squared errors, i.e. Allen's PRESS statistic (Allen, 1974),

$$P(\lambda) = \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\hat{y}_i^{(-i)} - y_i\right]^2 \qquad \text{where} \qquad \hat{y}_i^{(-i)} - y_i = \frac{\hat{y}_i - y_i}{1 - h_{ii}}, \tag{2}$$

$\hat{y}_i^{(-i)}$ represents the output of the classifier for the i^{th} training pattern in the i^{th} fold of the leave-one-out procedure and h_{ii} is an element of the principal diagonal of the hat matrix $\boldsymbol{H} = \boldsymbol{X}\left[\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right]^{-1}\boldsymbol{X}^T$. The ridge parameter can be optimised more efficiently in canonical form (Weisberg, 1985) via eigen-decomposition of the data covariance matrix $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}^T\boldsymbol{\Lambda}\boldsymbol{V}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues. The normal equations and hat matrix can then be written as

$$[\boldsymbol{\Lambda} + \lambda\boldsymbol{I}]\,\boldsymbol{\alpha} = \boldsymbol{V}^T\boldsymbol{X}^T\boldsymbol{y} \quad \text{where} \quad \boldsymbol{\alpha} = \boldsymbol{V}^T\boldsymbol{\beta} \qquad \text{and} \qquad \boldsymbol{H} = \boldsymbol{V}\,[\boldsymbol{\Lambda} + \lambda\boldsymbol{I}]^{-1}\,\boldsymbol{V}^T \tag{3}$$

As only a diagonal rather than a full matrix need now be inverted following a change in λ, the computational expense of optimising the ridge parameter is greatly reduced. For problems with more features than training patterns, $d > \ell$, the kernel ridge regression classifier (Saunders et al., 1998) with a linear kernel is more efficient and exactly equivalent. The ridge parameter for KRR can also be optimised efficiently via an eigen-decomposition of the kernel matrix (Saadi et al., 2007).

2.2. Pre-processing

The following pre-processing steps were used for all datasets: First all constant features are deleted, including features where all values are missing. Binary fields are coded using the values 0 and 1. Categorical and ordinal variables are encoded using a 1-of-n representation, where n is the number of discrete categories/values. Missing values are imputed using the arithmetic mean, and dummy variables are added to indicate the pattern of missing data for each feature. Lastly, continuous features are transformed to have a standard normal distribution, by evaluating the inverse standard normal cumulative distribution function for the normalised rank for each observation. It is hoped that this transformation prevents variables with highly skewed distributions from having a disproportionate effect on the classifier, whilst still allowing the extreme values to lie in identifiable ails of the distribution.

2.3. Pool Based Active Learning

A number of very basic strategies for pool based active learning, suitable for use as baseline submissions, are easily identified:

- **Passive Learning:** All patterns submitted to the oracle for labeling in the first step. This is not strictly speaking an active learning strategy, but it provides a useful baseline for comparison.

- **Random sampling:** At each iteration, one or more unlabelled samples are selected at random to be labelled by the oracle. This is perhaps the most basic algorithm for pool-based active learning, but is probably sub-optimal as it concentrates solely on exploration rather than exploitation.

- **Uncertainty sampling:** Unlabelled examples closest to the current decision boundary are selected for labeling by the oracle. This strategy aims to rapidly acquire labels for those examples that are classified with least confidence. Note that maximum margin classifiers and boosting algorithms also aim to concentrate on patterns close to the decision boundary, so it is perhaps not unreasonable to expect this strategy to perform well.

This gives three basic baselines, one with no active learning, one with a naïve active learning strategy, and one with a good active learning strategy.

3. Results

In this section, we present the results of experiments performed during the development phase of the challenge before moving on to describe the baseline submissions made on the final benchmark datasets.

3.1. Preliminary Experiments during the Development Phase

During the development phase of the challenge, a number of computationally intensive Monte-Carlo simulations were used to investigate the effectiveness of the three baseline active learning strategies. All of the labels made available for the training samples from each of the development datasets were downloaded. This allowed re-sampling to be used to estimate the variability in the performance of different active learning strategies due to the sample of data and due to any stochastic component of the learning procedure. For all experiments 100 replications were performed, each using a random partition of the available data to form training and test sets in the proportion of 3:1, and a positive example chosen at random from the training set as the "seed" pattern. The area under the receiver operating characteristic (ROC) curve (AUC) was recorded at approximately equal intervals on a logarithmic scale. The area under the resulting graph of AUC as a function of the number of labelled examples (on a logarithmic axis) then provides the test statistic, known as the area under the learning curve (ALC). Table 1 shows the ALC statistic for optimally regularised [kernel] ridge regression with passive, random sampling and uncertainty sampling active learning strategies. It can be seen that no active learning strategy is dominant, but more interestingly, random sampling is competitive with uncertainty sampling, even though it is a very naïve strategy.

The Friedman test, as recommended by Demšar (2006), reveals there is no significant difference in the average ranks of the three active learning strategies over the six development datasets. The lack of a significant difference is illustrated by the critical difference diagram, shown in Figure 1, which shows the average ranks of the three strategies, with the bar linking together cliques of statistically similar classifiers.

Table 1: Area under the learning curve for three simple active learning strategies for the development datasets. The results are given as the arithmetic mean, and their standard errors, calculated over 100 random replications of the experiment. The best results for each dataset are shown underlined, without implication of statistical significance.

Benchmark	Passive	Random	Uncertainty
HIVA	0.2997 ± 0.0018	0.2505 ± 0.0056	0.1536 ± 0.0077
NOVA	0.4899 ± 0.0001	0.6975 ± 0.0033	0.6999 ± 0.0064
IBN_SINA	0.4821 ± 0.0002	0.8017 ± 0.0045	0.7832 ± 0.0050
ORANGE	0.2920 ± 0.0017	0.1910 ± 0.0052	0.2227 ± 0.0057
SYLVA	0.4967 ± 0.0000	0.8612 ± 0.0037	0.8893 ± 0.0025
ZEBRA	0.2744 ± 0.0013	0.3564 ± 0.0095	0.2949 ± 0.0120

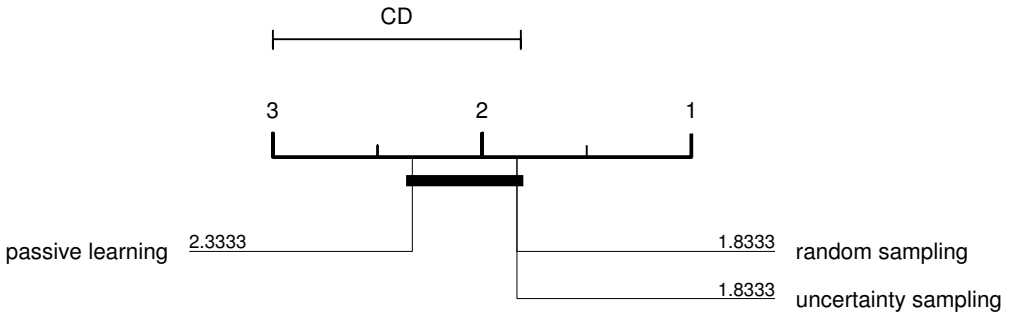


Figure 1: Critical difference diagram, showing the mean ranks of three basic active learning strategies over the final test benchmark datasets. The bar labelled "CD" shows the difference in mean rankings required for a statistically significant difference in performance to be detected.

Figure 2 shows the average learning curves for the three baseline active learning strategies over the development benchmark datasets. Clearly active rather than passive learning is more useful on some datasets (NOVA, IBN_SINA and SYLVA) than others, such as HIVA, ORANGE and ZEBRA, where relatively little can be usefully learned from a small number of training patterns, whether they are selected at random or according to uncertainty.

3.2. Why does Random Active Learning Work so Well?

Figure 3 shows quantiles of the distribution of learning curves for the `nova` and `zebra` benchmarks, for random and uncertainty sampling active learning methods. It can be seen that the uncertainty sampling strategy out-performs random active learning for the `nova` dataset with more than about 20 labelled examples (c.f. Figure 2b), while for smaller labelled datasets, however, uncertainty sampling performs poorly. The lower quantiles ($p_{.05}$ and $p_{.25}$) shown in Figure 3 suggest this is because of a large variability in the early part of the learning curves for the uncertainty sampling strategy. We conjecture that the downside of a principled strategy to active learning is that the selection of examples for labeling by the oracle depends on the current model, so if poor selections were made at an early stage, this adversely affects the quality of subsequent selections and hence learning proceeds slowly. This is less evident for random sampling, which gets locked into a poor hypothesis rather less frequently.

An effective active learning strategy must reach a near optimal trade-off between exploration and exploitation. The uncertainty sampling approach concentrates on exploiting the knowledge it has gained from the labels it has already acquired to further explore the decision boundary. The random sampling approach concentrates on exploration, and so is able to locate areas of the feature space where the classifier performs poorly. These results highlight the need for exploration as well as exploitation as the uncertainty sampling approach can become locked in a mistaken hypothesis of the location of the true decision boundary as it does not explore enough of the feature space that might suggest the current hypothesis is flawed.

3.3. Final Baseline Models

For the final test phase of the challenge, the baseline models were constructed according to the same protocol made available to the other participants (see Guyon et al., 2010, for details), and so Monte-Carlo simulations were not possible. A total of four baseline submissions were made using passive learning and random and uncertainty sampling based active learning. Two different initialization strategies were used: In the first, an initial classifier was constructed with the single positive seed pattern and the unlabelled patterns treated as if they belonged to the negative class. A second strategy was also used in conjunction with random sampling, where the prediction for unlabelled patterns was given by the Euclidean distance to the single positive pattern provided as a "seed" for the active learning procedure. This method would also have been used with the other active learning strategies had sufficient time been available, where the difference

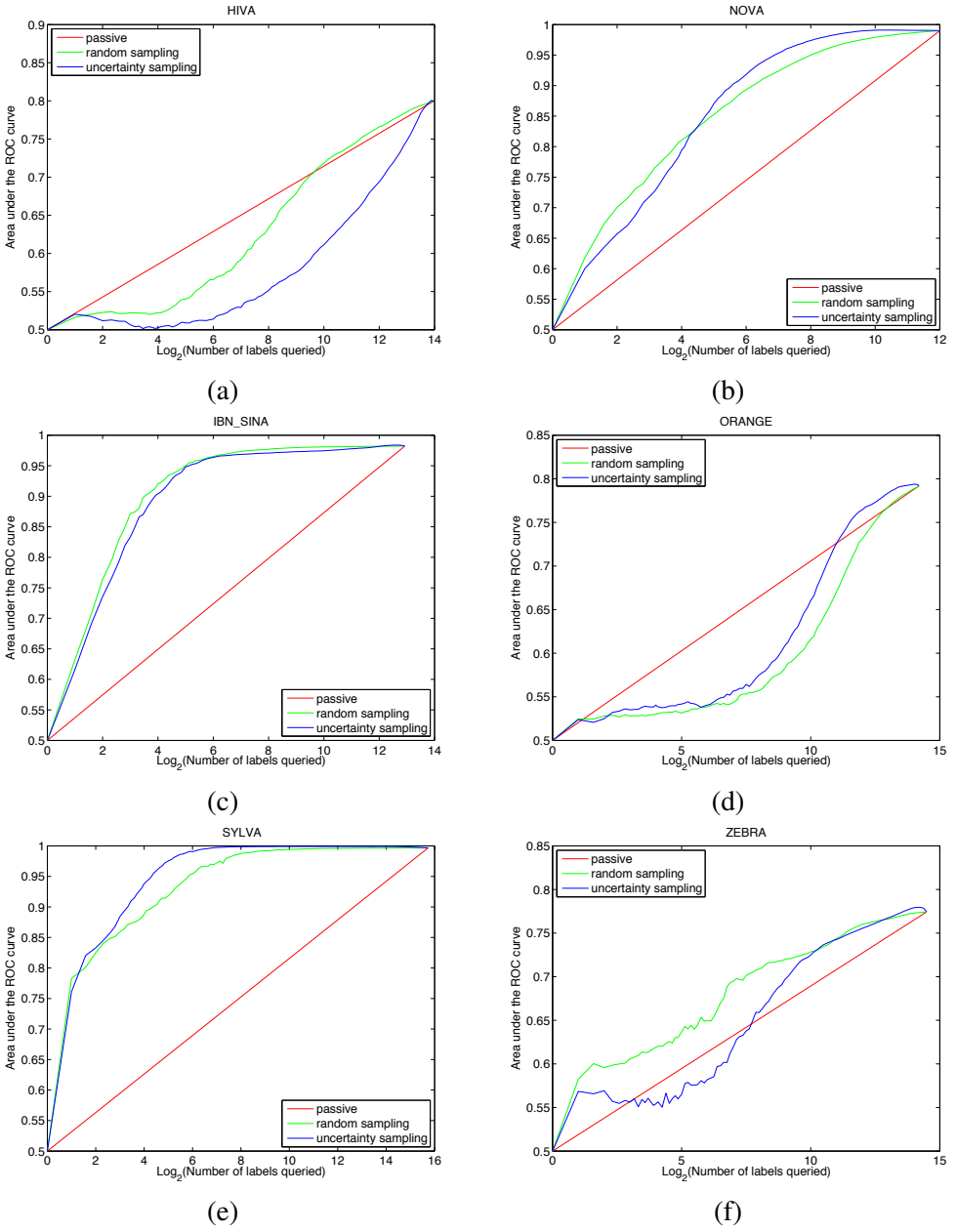


Figure 2: Average learning curves for active learning methods over 100 random realisations of the development benchmark datasets: (a) `hiva`, (b) `nova`, (c) `ibn_sina`, (d) `orange`, (e) `sylva` and (f) `zebra`.

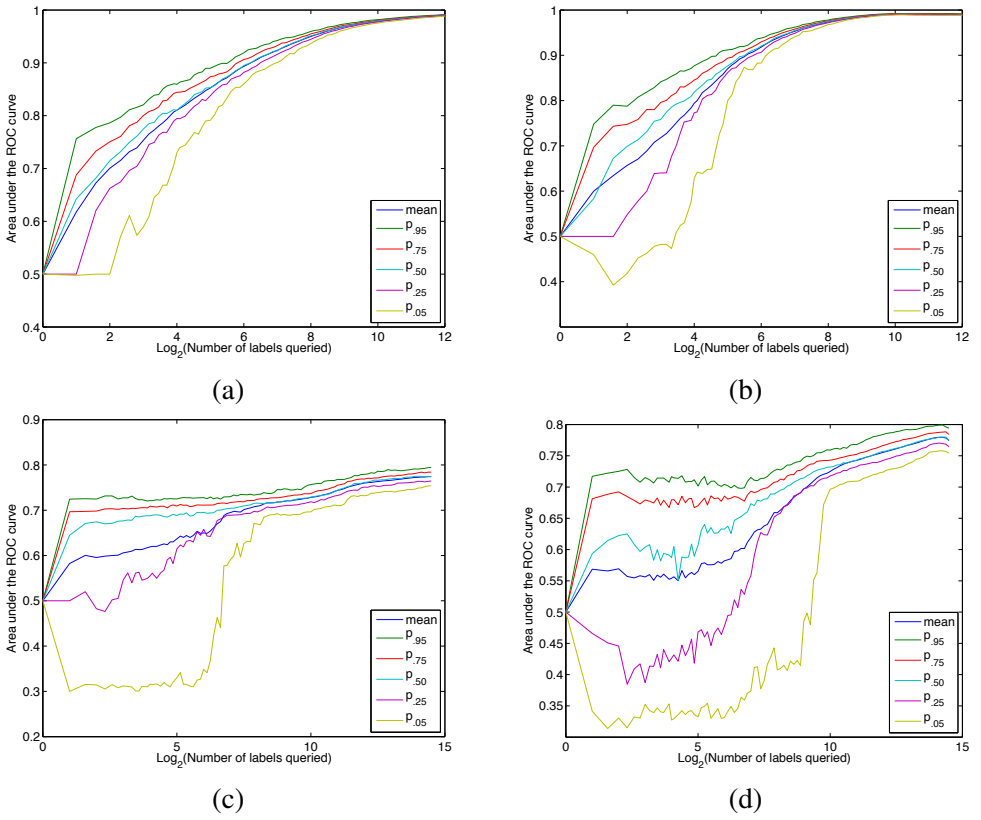


Figure 3: Quantiles of the distribution of learning curves for random (a) and (c) and least certain (b) and (d) active learning methods over 100 random realisations of the `nova` (a) and (b) and `zebra` (c) and (d) development benchmark datasets.

in initializations would have had a greater effect on the progress of the active learning procedure. The results obtained are shown in Table 2. The rankings of the baseline solutions show that a simple random sampling approach to active learning is effective and competitive with the results of some of the top submissions. The submission based on random sampling with linear initialization, for example, would have had an overall ranking of 4.667.

Table 2: Area under the Learning Curve (ALC) for the four baseline models and for the best entry for each of the final benchmark datasets. The best entries were as follows: A - `gcc4` (`reference`); B - `b` (`scan33scan33`); C - `C` (`chrisg`); D - `Dexp` (`datam1n`); E - `En` (`yukun`); F - `gccf2` (`reference`).

Method	**Global Score - ALC (rank)**					
	A	**B**	**C**	**D**	**E**	**F**
Passive	0.5455 (4)	0.3708 (3)	0.2663 (10)	0.4875 (21)	0.4966 (5)	0.7929 (5)
Random (linear)	0.5451 (5)	0.3084 (8)	0.2853 (6)	0.6512 (6)	0.4496 (8)	0.8217 (1)
Uncertainty sampling	0.4116 (15)	0.2689 (11)	0.2448 (11)	0.5748 (16)	0.3690 (16)	0.8074 (2)
Random (Euclidean)	0.6353 (1)	0.3195 (6)	0.3018 (5)	0.5996 (13)	0.4027 (12)	0.8048 (3)
Best	0.6353 (1)	0.3757 (1)	0.4273 (1)	0.8610 (1)	0.6266 (1)	0.8217 (1)

Again, the Friedman test was used to evaluate the statistical significance of any difference in the mean ranks of each approach, and again the differences were small, and not statistically significant. Figure 4 shows a critical difference diagram, illustrating the very similar rankings of the four baseline methods.

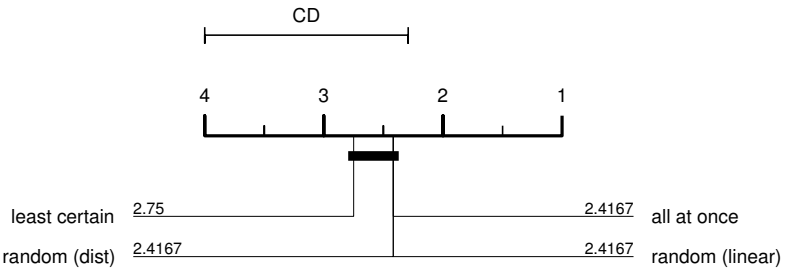


Figure 4: Critical difference diagram, showing the mean ranks of three basic active learning strategies over the final test benchmark datasets.

Figure 5 shows the learning curves obtained for the four baseline solutions for the six benchmark datasets used in the final phase of the challenge; the learning curve for

the best submission for each benchmark is also shown. It can be seen that the results obtained for small numbers of labelled patterns are highly variable for all active learning methods for all benchmark datasets.

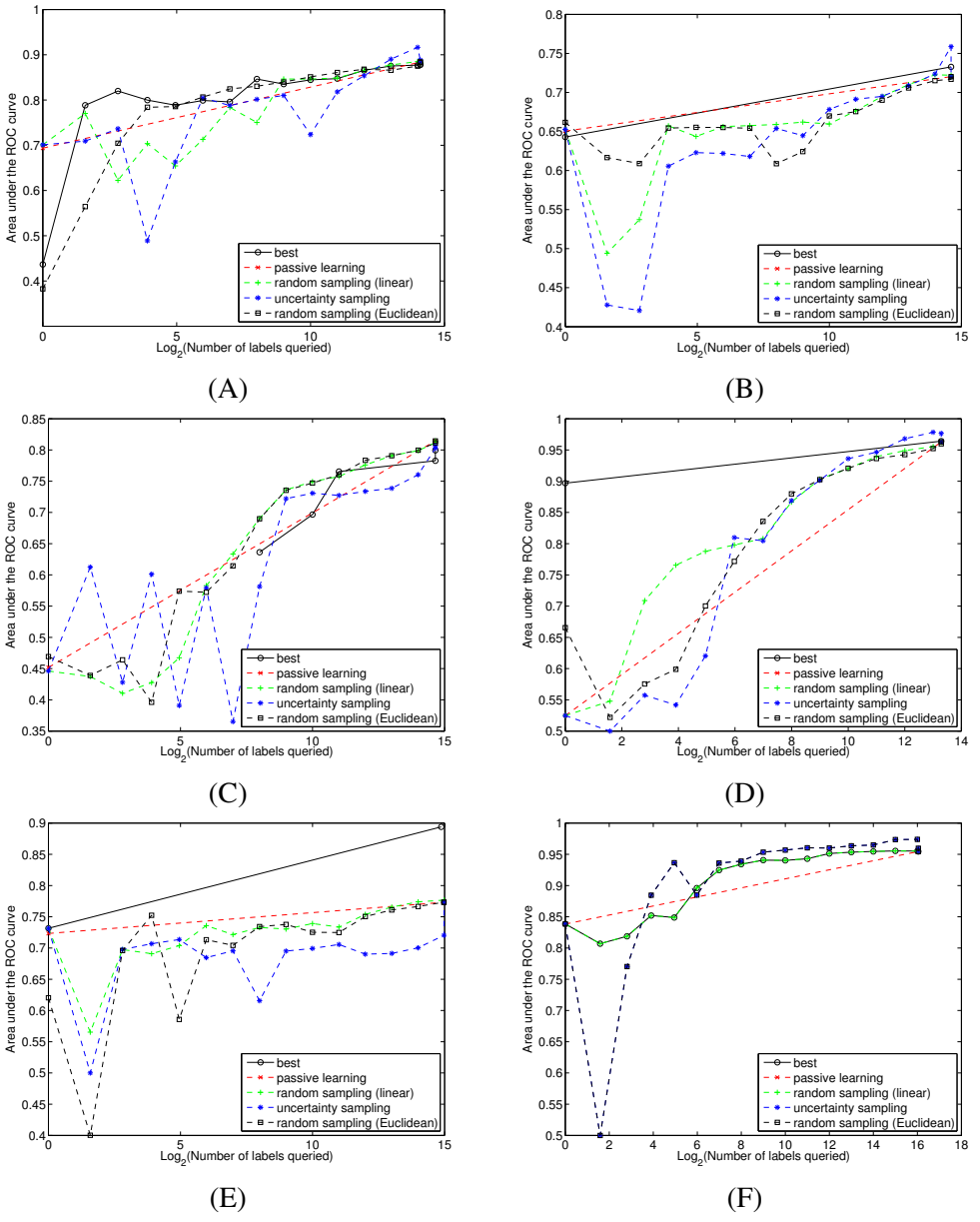


Figure 5: Learning curves for selected baseline models over the final benchmark datasets (A-F) of the active learning challenge.

4. Summary

In this paper, we have described some simple baseline methods for the active learning challenge, based on optimally regularised ridge regression. A very basic random sampling approach was found to be competitive with both a more advanced uncertainty sampling approach and with some of the better challenge submissions. The poor performance of the uncertainty sampling approach seems likely to be due to a lack of exploration of the feature space at the expense of exploitation of current knowledge of the likely decision boundary. It is probable that better performance might be obtained using semi-supervised or transductive learning methods to take greater advantage of the availability of unlabelled data.

Acknowledgments

I would like to thank the anonymous reviewers for their helpful and constructive comments and the co-organizers of the challenge for their efforts in staging a very interesting and (for myself, at least ;o) educational challenge.

References

D. M. Allen. The relationship between variable selection and prediction. *Technometrics*, 16:125–127, 1974.

B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992.

G. C. Cawley. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs. In *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks (IJCNN-06)*, pages 1661–1668, Vancouver, BC, Canada, July 16–21 2006. doi: 10.1109/IJCNN.2006.246634.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995. doi: 10.1007/BF00994018.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, January 1992. doi: 10.1162/neco.1992.4.1.1.

I. Guyon, G. Cawley, and G. Dror. Results of the active learning challenge. *Journal of Machine Learning Research, Workshop and Conference Proceedings*, 15, 2011.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986. doi: 0.1007/BF00116251.

K. Saadi, G. C. Cawley, and N. L. C. Talbot. Optimally regularised kernel Fisher discriminant classification. *Neural Networks*, 20(7):832–841, September 2007. doi: 10.1016/j.neunet.2007.05.005.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 515–521. Morgan Kaufmann, 1998.

B. Settles. Active learning literature survey. Technical Report 1648, School of Computer Sciences, University of Wisconsin-Maddison, 2009.

A. R. Webb. *Statistical pattern recognition*. Wiley, second edition, 2002.

S. Weisberg. *Applied linear regression*. Probability and Mathematical Statistics. John Wiley & Sons, second edition, 1985.

JMLR: Workshop and Conference Proceedings 16:59–69, 2011 Active Learning Challenge

Active Batch Learning with Stochastic Query-by-Forest (SQBF)

Alexander Borisov ALEXANDER.BORIOSV@INTEL.COM
Intel, Nizhniy Novgorod, Russia

Eugene Tuv EUGENE.TUV@INTEL.EDU
Intel, Chandler, AZ, USA

George Runger RUNGER@ASU.EDU
Arizona State University, Tempe, AZ, USA

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

In a conventional machine learning approach, one uses labeled data to train the model. However, often we have a data set with few labeled instances, and a large number of unlabeled ones. This is called a semi-supervised learning problem. It is well known that often unlabeled data could be used to improve a model. In real world scenarios, labeled data can usually be obtained dynamically. However, obtaining new labels in most cases requires human effort and/or is costly. An active learning (AL) paradigm tries to direct the queries in such way that a good model can be trained with a relatively small number of queries. In this work we focus on so-called pool-based active learning, i.e., learning when there is a fixed large pool of unlabeled data, and we can query the label for any instance from this pool at some cost. Existing methods are often based on strong assumptions for the joint input/output distribution (i.e., a mixture of Gaussians, linearly separable input space, etc.), or use a distance-based approach (such as Euclidean or Mahalanobis distances). That makes such methods very susceptible to noise in input space, and they often work poorly in high dimensions. Also, such methods assume numeric inputs only. In addition, for most methods relying on distance computations and/or linear models, computational complexity scales at least quadratically with respect to the number of unlabeled samples, rendering them useless on large datasets. In real world applications data is often large, noisy, contains irrelevant inputs, missing values, and mixed variable types. Often queries should be arranged in groups or batches (this is called batch AL). In batch querying one should consider both the 'usefulness' of individual queries within a batch, and the batch diversity. Batch AL, although being very practical by nature, is rarely addressed by existing AL approaches. Here we propose a new non-parametric approach to the AL problem called Stochastic Query by Forest (SQRF), that effectively addresses the challenges described above. Our algorithm is based on a QBC algorithm applied to an RF ensemble, and our main contribution is the batch diversification strategy. We describe two different strategies for batch selection, the first of which achieved the highest average score on the AISTATS 2010 active learning challenge and ranked top on one of the challenge datasets. Our work focuses on binary classification problems,

but our method can be directly applied to regression or multi-class problems with minor modifications.

Keywords: tree ensembles, query by committee, random forest

1. Introduction

The basic idea behind active learning (AL) is that a regression or classification algorithm can achieve better performance on limited data when it is allowed to choose the data for learning. In pool-based active learning, we are given a large fixed pool of unlabeled data, and are allowed to query the target value for each unlabeled instance at a given cost. Here we assume equal unit cost for all queries. The model is first built on all labeled instances, then we query an instance that is considered the most useful and update the model. The goal is to achieve a better learning curve (error vs. total query cost) for a model, compared to querying labels at random. The main challenge for an AL algorithm is computing the "utility" on unlabeled instances. The usual intuition behind the utility function is to select instances in dense regions of input distribution, or in regions of low sampled density, or where the is the most "uncertainty" in the model. For a comprehensive review of AL approaches see Settles (2009). Below we outline the most commonly used AL approaches.

Uncertainty sampling (see Lewis and Gale (1994) for example). Suppose we have a model that can report class probabilities p_i, $i = 1 \dots K$, where K is number of classes. Then all unlabeled instances are ranked according to the current model uncertainty measure (for a classification problem this is usually computed as $1 - \max(p_i)$). The next instance queried is the instance with the largest uncertainty, thus avoiding the instances that are predicted with high confidence.

Query-by-committee, or QBC (see Seung et al. (1992), Freund et al. (1997)). This approach first constructs an ensemble (committee) of diverse base learners, then ranks all unlabeled instances with respect to a committee disagreement measure. Disagreement can be computed as the entropy of predicted class probabilities over committee members, or in various other ways. Here one tries to select instances that represent regions of input space that are not covered by existing learners in the committee. QBC discourages querying instances from the same region of the input distribution where good prediction is impossible by nature (an inherent weakness of uncertainty sampling).

However, methods like uncertainty sampling and QBC do not take global properties of the input distribution into account, and can spend too much time querying outliers or sparsely populated regions. Density based methods try to overcome this issue by incorporating the input data density into the utility function. The resulting utility function for a data point x is computed as $U(x) \cdot D(x)^p$, where $U(x)$ is an expected utility, and $D(x)$ is an estimated input density. The parameter p controls the influence of the density factor. This encourages querying from more densely populated regions of input space. See, for example, Xu et al. (2007) for a density-based method in relevance feedback.

Another approach that uses global input distribution information directly minimizes the expected model generalization error (expected risk). This is similar in nature to a Bayesian experimental design Chaloner and Verdinelli (1996). However, the expected

risk can be computed in closed form only for a limited class of models, such as a mixture of Gaussians, or SVMs. Such methods often have high computational complexity because the model has to be rebuilt with each query, and the utility recalculated. For an example of an algorithm that solves those problems with a fast model update and utility recalculation strategy using Gaussian Random Fields (GRF) model see Zhu et al. (2003). GRF also naturally uses unlabeled data for learning (performs semi-supervised learning).

Expected variance reduction tries to reduce the expected variance of the model prediction. It is well known that model error can be decomposed into noise (term independent from model), bias (a term specific to the selected model class that estimates the difference between the best target function in the model class and the real underlying target), and variance. Given a model class (for example, linear functions or trees), noise and bias cannot be influenced by the model. Expected variable reduction selects instances that minimize the expected variance for the model. However, an estimate of the expected variance is also only available for a very limited class of models. Expected model change tries to select instances that maximize the model change with respect to the addition of a selected instance to the training set. This, however, does not guarantee model error improvement, just diversity of queries. Estimation of model change is also only possible for a limited number of model classes.

However an issue with most of the approaches described above is that they do not work with large and/or noisy data, or use very limiting assumptions on the model class. For example, methods relying on any distance metrics are susceptible to the curse of dimensionality (usually do not work well for more than 10-20 inputs), are sensitive to feature scaling and incur the additional complexity of calculating distances (although the later problem can be partially solved by clustering). Linear models or EM with a mixture of Gaussians rarely fit complex distributions in real world data. Any kernel-based method like SVM and GRF, also requires an approximately correct estimation of the kernel width parameter, and that is in itself a complex task for high-dimensional noisy data.

As stated earlier, querying more than one instance at time (batch learning) often can greatly reduce the labeling effort and computation time. For example, one does not need to rebuild the model for each queried instance, and parallel labeling is possible. In real problems labeling and/or querying is often done by human experts and processing unlabeled instances one-by-one is more costly than in groups (batches). However, batch learning introduces an additional challenge compared to single instance queries. In addition to optimizing individual queries, one must make sure that instances in the batch are diverse enough. That is the reason why a greedy selection of instances with the highest utility does not work. In addition, a batch learning algorithm should be fast enough, compared to querying instances one-by-one, to be useful. Several approaches to batch learning (Brinker (2003), Xu et al. (2007)) also use a greedy selection algorithm with a modified utility criteria. In the first work, for example, the authors use a linear combination of utility and diversity measures with a SVM model. Diversity for each sample measures how far it is from the other samples in the batch. The second

article Xu et al. (2007) introduces a "Relevance, Diversity and Density" batch learning framework. Relevance considers individual instance utility, density promotes sampling from more populated regions of the distribution, and diversity ensures that samples within the batch are not close to each other.

But, in practice, one often deals with very large datasets (with potentially hundreds to thousands of inputs and/or up to millions of instances), especially given the fact that AL deals with large amounts of unlabeled data. Also the data are usually very noisy and contain categorical variables, so that approaches based on linear models or Euclidean/Mahalanobis distance metrics are not computationally feasible. This also prohibits usage of "global" methods like empirical risk minimization because they usually rely on distance-based models like SVM or GRF. Neural Networks also rarely perform well with large and noisy data with an unknown distribution. Mixture models and clustering approaches fail when the data do not contain easily separable clusters. Many prospective AL approaches, for example GRF, are not well suited for batch learning, and each query involves quadratic complexity in the current number of labeled samples for the model update, resulting in a total of ($O(N^3)$ complexity for queries and model update with N instances in the initial unlabeled data pool (given that the initial number of labeled instance is very small compared to N). So most of the methods described above can only handle several thousand samples and tens of variables, severely limiting their practical application.

In this work we propose a nonparametric batch AL method using tree ensembles that works with huge data sets and overcomes most of the problems described above. We introduce decision tree ensembles in Section 2. Section 3 contains a detailed description of our algorithm. Then we describe successful application of our method to AISTATS 2010 active learning challenge problems that provide a very good representation of real life active learning tasks.

2. Decision tree models and tree ensembles

As stated above, to effectively deal with active learning problem one needs to impose some reasonable assumptions on the joint input/output data distribution. Those assumptions are represented by a particular data model. Among models used with huge, noisy and heterogeneous data, a very popular choice is the decision tree–because trees are fast to learn, resistant to outliers and noise and provide good predictive accuracy. Trees are usually induced in a recursive, greedy fashion. For each node the best split (split with greatest impurity reduction) is selected, then the process is repeated in the child nodes. For example, the CART algorithm described in Hastie et al. (2001) can be used. Commonly used node impurity measures are the variance of the target (for regression) or the Gini index (for classification). However, trees often suffer from instability, or low predictive power if the underlying data model is complex. Significant improvements over single tree models can be achieved with tree ensembles, sequential (Gradient Boosting Trees (GBT), Multi-class Logistic Regression Trees (MCLRT), Adaboost, see Hastie et al. (2001)) and parallel (Random Forest (RF), see Breiman (2001)). We briefly describe RF, because it is used extensively in our active learning algorithm, although we

use a GBT model as the final predictor (using samples queried by our AL algorithm). We refer to Hastie et al. (2001) for details on a GBT algorithm for the multi-class case and omit it here.

The idea behind RF is to combine many diverse trees into an ensemble. This allows for more stable model, reduces over-fitting, and improves predictive accuracy over a single tree. RF constructs a number of independent trees, each tree is built on a random portion (60% for example) of the training (labeled) samples. Additional diversity within the tree is added via split randomization. Instead of selecting the single best split among the best splits on each variable like CART does, a random, small subset of variables is selected at each tree node (a commonly used setting is $\sqrt{M}$, where M is the total number of variables). Then the best split is selected only within this subset. Prediction from an RF model is obtained with averaging for regression, and voting in classification (where the number of trees that predict each target class are counted and the most frequent class is selected as the predictor). RFs are usually applied to classification problems, because combining many weak trees via averaging does not always result in a better model in regression settings. RF is especially attractive for use with QBC, because it naturally introduces base learner diversity, and each tree has an intrinsic prediction probability estimate computed from the class proportions in the terminal node of the selected instance. RF can also be used to estimate various other properties of the joint data distribution, such as density, outlier scores, variable importance measures, or (supervised) distance metrics (Breiman (2001)).

3. Tree ensemble approach for active learning

Here we describe two our algorithms for batch AL. Denote N_0, N_1 as the counts of the target classes in the labeled data, $p_c = N_c/N,\ c = 1,2$ as the target class proportions in the currently labeled data, and the i-th tree in ensemble G as $T_i = T_i(G), i = 1 \ldots R$, where R is number of trees in the ensemble. Denote the terminal node of the i-th tree containing instance x as $T_i(x)$, and $p_{ic}(x)$ as the predicted class probabilities in the i-th tree for x, computed as the target class proportions in node $T_i(x)$. Denote the same probabilities weighted with class priors

$$p'_{ic}(x) = \frac{p_{ic}(x)/p_c}{\sum_c p_{ic}(x)/p_c},\ c = 1,2$$

Algorithm 1. Stochastic query by forest.

1. *Build an RF ensemble G (we used 700 shallow trees with depth = 2-6, depending on the current labeled data size).*

2. *For each sample, compute the committee disagreement $q(x) = sd(p'_{ic}(x))$ as the standard deviation of the weighted rare class probability among the ensemble of trees. Then sort all remaining unlabeled instances with respect to $q(x)$, so that $q(x_1) > q(x_2) > \ldots > q(x_{n_u})$, where n_u is the number of remaining unlabeled instances.*

3. *Sample the next batch randomly from $x_1, \ldots x_{\alpha n_u}$. Parameter α controls the discarded fraction of unlabeled instances, and indirectly introduces a tradeoff between randomness and high utility. We set $\alpha = 2/3$ in our experiments. Sampling probabilities are computed from utility scores as following. Denote the threshold $q_0 = q(x_{\alpha n_u})$, and $L(x) = (q(x) - q_0)/(q(x_1) - q_0)$. Then the sampling probability of instance x is computed as $p_{sel}(x) = L(x)/\sum_x L(x)$.*

4. *Sample the batch from the remaining unlabeled instances with the computed sampling probabilities. Rebuild model G and return to step 2 (until no unlabeled instances are left in the pool).*

This method addresses both the uncertainty score and the input density (as random sampling selects more instances from the dense regions of the input distribution). At the same time we enforce diversity within the batch through randomization. Instead of scoring uncertainty with a single class probability estimate, the tree ensemble allows an embedded uncertainty score to be calculated directly from the differences between the individual learners (as the standard deviation). This approach is distinct from other QBC approaches, and this provides one of our contributions. Also, the standard deviation is simple uncertainty score and alternatives (such as more robust measures) could be useful. We would not expect the results to change substantially with alternative measures.

Our perspective is that the tree ensemble is useful for this uncertainty score, but that in addition the tradeoff between utility and randomness is a key component of our strategy. The importance of a random element was illustrated by Cawley (2011) who concluded that a simple, random baseline method was competitive with more complex strategies. He conjectured that uncertainty sampling does not sufficiently explore the feature space and, instead, tends to expend samples to exploit the current knowledge of the likely decision boundary. Our tradeoff between uncertainty and randomness is easily managed by our alpha parameter, and in the challenge data we used a sizeable proportion (alpha = 2/3) of random sampling, but not entirely random. Also, our implementation of step 3 of the algorithm 1 uses rejection sampling to avoid a quadratic complexity in the number of unlabeled instances. When the number of rejected instances becomes too large, we just select the remaining instances with highest utility (although it occurs very rarely in practice).

Our second approach directly addresses diversity and density in a way similar to Brinker (2003). Suppose we present all labeled and unlabeled data through RF model G (resulting in each instance being assigned to a terminal node for each tree). Denote the labeled data count in a node T as $l(T)$, and the total count as $s(T)$. Then we can estimate the expected proportion $d(x)$ of the labeled instance density to the total density in the neighborhood of instance x as $d(x) = \sum_{i=1}^{R} l(T_i(x))/\sum_{i=1}^{R} s(T_i(x))$. The inverse proportion of labeled instances in the neighborhood can be used instead of local density as a multiplier for the utility measure, because it promotes both queries in dense regions, and in regions with few labeled points. Below is the detailed description of the algorithm that uses this modified utility function.

Algorithm 2. Local density based query-by-committee.

1. *Build an RF ensemble G.*
2. *Compute* $l(x) = \sum_{i=1}^{R} l(T_i(x))$ *and* $s(x) = \sum_{i=1}^{R} s(T_i(x))$ *for each unlabeled instance.*
3. *Compute a modified utility score* $q'(x) = q(x)/d(x) = q(x)s(x)/l(x)$ *for all unlabeled data. Then sort all remaining instances with respect to* $q'(x)$*, so that* $q'(x_1) > q'(x_2) > \ldots > q'(x_{n_u})$*. Initialize query count* $Q = 0$
4. *Select an instance with the highest value of* $q(x)/l(x)$ *from* $x_1, \ldots x_{n_0}$*, where* $n_0 \ll n_u$ *is a predefined number of lookup instances. We set* $n_0 = \min(1000, n_u)$*.*
5. *Mark it as labeled and propagate through all the trees in G (resulting in updated counts* $l(T_i(x)$ *in each tree). Increase Q.*
6. *If* $Q < Q_0$*, where* Q_0 *is predefined number of queries that can be completed without new sorting (say* $20 - 50$*), return to step 4. Else return to step 3 (and sort again).*
7. *Rebuild model G and return to step 2.*

Additional tricks in steps 4 and 5 are introduced to avoid sorting unlabeled instances with respect to the utility score after each query. Reasonable values for Q_0 and N_0 can prevent a large time complexity in the number of unlabeled instances, while selecting the top-scored instances with respect to utility. Computation of $q(x)/l(x)$ in step 4 has complexity $O(RD)$, where D is the maximum tree depth, as $q(x)$ are never updated. However, for a small tree depth (this is important for a more robust estimation of $q(x)$ and $d(x)$) this is not a major problem. Batch selection complexity is still negligible compared to RF and GBT model building complexities. We can use shallow trees because RF is used for AL only, and high predictive accuracy is not an issue. We use default settings for the RF count of attributes scored at a node (equal to the square root of the total number of attributes).

As a classifier we used GBT with embedded feature selection (see Borisov et al. (2006) for details), or RF when the number of labeled samples was small. Model selection and GBT parameter optimization (a simple grid search for tree depth and regularization over a predefined set of values) used two-fold CV error estimates. One could potentially use RF in all cases, but a GBT can improve predictions in some cases and we allowed this alternative in our strategy.

The robustness of tree-based ensembles allowed for a straight-forward approach. There was no preprocessing of the features, no feature generation, no data cleaning, and no preliminary data analysis. Missing values were handled with traditional tree-based approaches. Missing attribute values were ignored to score splits. To assign instances, surrogates (Breiman et al. (1984)) were used for GBT, while the majority child node was assigned for RF.

4. Experiments

We applied both our algorithms to the twelve AISTATS 2010 AL challenge datasets (six development datasets which were larger on average, and six test datasets). Below we briefly describe the challenge datasets and ranking measure. Data came from diverse real world domains, for example, marketing, ecology, and text processing. The largest datasets in the development group were (16969 x 9733), (216 x 72626), where the first number is the number of inputs, and the second is number of instances in the training data. The largest test datasets were (92 x 17535), (12000 x 10000) and (12 x 67628). Also, four of the development datasets had very unbalanced target distributions (1.8% - 6.15% as the proportions of the rare class).

The task was to achieve the best learning curve while querying data in arbitrary batches and updating the model after each query. The score was estimated as the area under the learning curve (model error versus number of labels queried), after all unlabeled instances are queried. The X-axis (number of labels) was $\log_2$ and this was scaled to favor good performance on a small number of labeled instances. Model error was calculated as the area under the ROC curve (AUC), to account for an unbalanced class distribution. The target was binary in all problems. The model error was estimated on separate test data, that had the same size as the training data, but with unknown labels. For detailed descriptions, see the challenge site http://www.causality.inf.ethz.ch/activelearning.php.

In small preliminary experiments with the test datasets both proposed AL approaches performed significantly better than random sampling, uncertainty sampling, or QBC . But because of small rare class proportions, the model error variation is very high, especially on a small number of labeled samples. There were small differences in performance between our algorithms 1 and 2 in these preliminary studies. Although the computational time for algorithm 2 with $n_0 = 1000$, $Q_0 = 20$ was not significantly higher than for algorithm 1, we chose to apply the first algorithm because of its simplicity, and it proved to be more robust for very unbalanced classes.

As mentioned previously, data preprocessing was not applied and the process to estimate the parameters was not complex. The predictive model, tree depth (over the range 2-6), and the GBT regularization parameter were selected via two-fold cross validation and alpha was fixed at 2/3 based on our preliminary experiments. For RF, the number of trees in an ensemble was fixed (at 700) and the number of the attributes scored at a node used the default (equal to the square root of the total number of attributes). Performance was not sensitive to either these serial or parallel ensemble parameters.

In our preliminary studies there were only minor differences between batches from 5-15 instances. Because it was necessary to submit and process each query manually, we limited ourselves to 15 queries per data set. The initial batch size was chosen as 5-10 depending on the number of inputs, then for each query the batch size was scaled exponentially with the exponent chosen in a way so that 15 queries covered all unlabeled data.

Our first algorithm (SQBF), had the top average rank on all six test datasets and had the first rank on one of the datasets. Figure 1 shows the ALC performance of SQBF (IdealAnalytics) and selected competitors over all the datasets where results were provided. Some competitors did not consider all the datasets. The selected competitors achieved a top-two result on at least one dataset. Table 1 supplements the ALC scores with additional results for the top-two competitors on each dataset. Further details of the challenge results were provided by Guyon. et al. (2011). SQBF provided consistent performance across these datasets. Figure 2 shows the ALC performance of SQBF (IdealAnalytics) and baseline methods over all the datasets. Details of the baselines methods were provided by Cawley (2011). SQBF (IdealAnalytics) was also a consistently strong performer compared to the baseline methods.

Our AL strategy is also very fast. It has the same asymptotic computational complexity as building an RF model, i.e $O(TN\log(N)\log(M))$, where T is number of trees, M is number of features, N is number of samples. The total run time (for either of the two algorithms) on all six development or test datasets on one machine was approximately 6-8 hours (Zeon workstation 3 GHz with 4 GB RAM, 2 processors with hyper-threading, Windows XP system) depending on the model optimization settings.

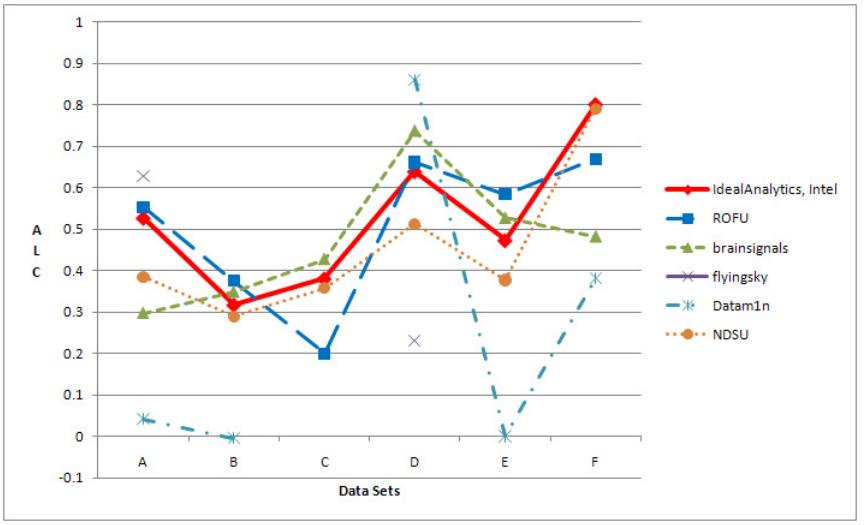


Figure 1: The ALC performance of SQBF (IdealAnalytics) and selected competitors over all the datasets where results were provided. The competitors selected achieved a top-two result on at least one dataset.

5. Acknowledgments

This research was partially supported by ONR grant N00014-09-1-0656. We thank the reviewers for helpful comments that improved this work.

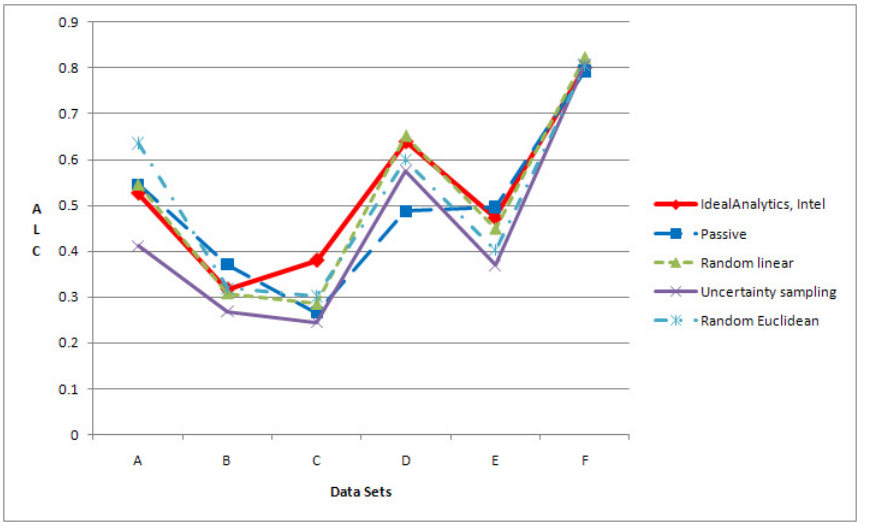


Figure 2: The ALC performance of SQBF (IdealAnalytics) and baseline methods (described by Cawley, 2011) over all the datasets.

6. Conclusions

We introduced a novel approach for pool-based, batch active learning using tree ensembles. We described two algorithms for batch selection that optimize both the query utility function and within batch diversity. Both algorithm are very fast, and can work with very large datasets. Both methods were successfully applied to real datasets from the AISTATS 2010 AL challenge. However, we are planning more experiments on artificial datasets where the underlying joint distribution is known, to investigate the relative strengths and weaknesses of the proposed approaches, and to compare them to other AL methods. We are also considering some form of semi-supervised learning (for example with auto-regressive trees or Gaussian Random Field models in tree terminal nodes). We do not currently use unlabeled data for learning in any way and results of some participants on AISTATS 2010 challenge show that on some datasets one can substantially benefit from semi-supervised learning.

References

A. Borisov, V. Eruhimov, and E. Tuv. Tree-based ensembles with dynamic soft feature selection. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction Foundations and Applications: Studies in Fuzziness and Soft Computing*. Springer, 2006.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Table 1: Summary of the results from AISTATS 2010 Active Learning Challenge. Results are shown for the top-two competitors on each data set with our algorithm denoted as SQBF.

Data	Algorithm		AUC	Ebar	ALC	Rank
Set A	FLYINGSKY	pipifuyj	0.8622	0.0049	0.6289	1
	SQBF	IdealAnalyticsIntel	0.9520	0.0045	0.5273	5
Set B	ROFU	scan33scan33	0.7327	0.0034	0.3757	1
	SQBF	IdealAnalyticsIntel	0.7544	0.0044	0.3173	5
Set C	BRAIN	chrisg	0.7994	0.0053	0.4273	1
	SQBF	IdealAnalyticsIntel	0.8333	0.0050	0.3806	2
Set D	DATAM1N	datam1n	0.9641	0.0033	0.8610	1
	SQBF	IdealAnalyticsIntel	0.9730	0.0030	0.6397	7
Set E	DSL	yukun	0.8939	0.0039	0.6266	1
	SQBF	IdealAnalyticsIntel	0.9253	0.0037	0.4731	5
Set F	SQBF	IdealAnalyticsIntel	0.9990	0.0009	0.8018	1
	NDSU	NDSU	0.9634	0.0018	0.7912	2
Overall	SQBF	IdealAnalyticsIntel	0.9062	0.0015	0.5233	4.167
	ROFU	scan33scan33	0.8774	0.0014	0.5072	4.833

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, MA, 1984.

K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 59–66. AAAI Press, 2003.

G.C. Cawley. Some baseline methods for the active learning challenge. In N. Lawrence, editor, *JMLR: Workshop and Conference Proceedings*, volume 15, 2011.

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 3:273–304, October 1996.

Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2):133–168, 1997.

I. Guyon., G. Cawley, G. Dror, and V. Lemaire. Results of the active learning challenge. In N. Lawrence, editor, *JMLR: Workshop and Conference Proceedings*, volume 15, 2011.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

D.D. Lewis and W.A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, page 12. Springer-Verlag New York, Inc., 1994.

B. Settles. Active learning literature survey. Technical report, Computer Sciences Technical Report, 2009.

H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. *Advances in Information Retrieval*, pages 246–257, 2007.

X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.

Active Learning and Experimental Design with SVMs

Chia-Hua Ho B95082@CSIE.NTU.EDU.TW
Ming-Hen Tsai B95028@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science, National Taiwan University
Taipei 106, Taiwan

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

In this paper, we consider active learning as a procedure of iteratively performing two steps: first, we train a classifier based on labeled and unlabeled data. Second, we query labels of some data points. The first part is achieved mainly by standard classifiers such as SVM and logistic regression. We develop additional techniques when there are very few labeled data. These techniques help to obtain good classifiers in the early stage of the active learning procedure. In the second part, based on SVM or logistic regression decision values, we propose a framework to flexibly select points for query. We find that selecting points with various distances to the decision boundary is important, but including more points close to the decision boundary further improves the performance. Our experiments are conducted on the data sets of Causality Active Learning Challenge. With measurements of Area Under Curve (AUC) and Area under the Learning Curve (ALC), we find suitable methods for different data sets.

1. Introduction

In some supervised learning problems, labeling the training data is costly. In addition, we may not need to label all the training data as some of them are not useful. Active learning is applied in such situations. Users can request more labeled instances by paying some cost. The goal is to obtain an accurate model using as few queried instances as possible.

Querying methods in active learning have been studied by many works. Seung et al. (1992) proposed a querying method called "Query by Committee," but it requires at least two different learning models on the same data set. Tong and Koller (2002) proposed some querying methods depending on the decision values of a support vector machine (SVM) model. In this paper, we propose a general and inexpensive algorithm to use decision values (by SVM or logistic regression) for selecting query points. We conduct experiments to compare our querying methods with some existing methods. Another contribution of this paper is to investigate some methods for the situation where there is only one labeled point.

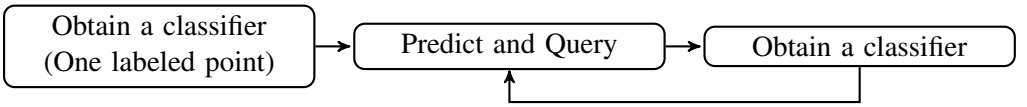


Figure 1: An active learning framework

This paper presents our methods, experiments and results for the Causality Active Learning Challenge.[1] Detailed settings of this competition can be found in Guyon et al. (2011). In this competition, we are second overall, and are the winner in one of the six data sets. This paper is an improved version of Tsai et al. (2010) by including some post-submission results.

2. Methods

This section first describes our framework for active learning problems and then shows details of each step.

2.1. The Framework

A typical active learning procedure iteratively performs the following two steps:

1. Train a classifier based on available labeled and unlabeled data.
2. Query labels of some data points.

If the performance is good enough or no resources are available to get more labeled data, the procedure is stopped.

For the first step, the easiest setting is to use only labeled data for training. In this work, we consider standard classifiers such as SVM and LR (logistic regression). However, when few instances are labeled, the resulting classifier may not perform well. We can either guess labels of some data to enlarge the training set or consider semi-supervised learning techniques. For the competition, we employ these techniques only when there is one labeled instance. Details are in Section 2.2. After the first query is made, subsequently we only use labeled data to train a classifier.

For the second step, each time we double the number of points for query. Therefore, if s is the number of points to be queried in the beginning, we then query $2s$, $2^2 s$, ... points until all labels are obtained. We choose points for query based on the prediction results of the current model. Details are in Section 2.3.

Figure 1 illustrates our framework.

2.2. Training with One Labeled Data Point

In the competition, we are given only one labeled data point in the beginning. The resulting classifier usually overfits this labeled data, To improve the predictability in the early stage, the following methods are considered.

1. The competition website is at http://www.causality.inf.ethz.ch/activelearning.php

2.2.1. Making the Unknown Data Have the Same Label

All data sets in this competition have much more negative instances than positive ones. A naïve approach is to treat all data with unknown labels as negative, and train the whole set by selected classifiers.

2.2.2. One-class SVM

One-class SVM (Schölkopf et al., 2001) is a method to estimate the support of a distribution. We run one-class SVM to obtain a region covering the only one labeled instance. The one-class SVM model then classifies points outside the boundary as in the other class.

2.2.3. One-class SVM + SVR

While one-class SVM can be used to label all the unknown instances, based on its predictions, we can further train another classifier. Here we consider SVR (Vapnik, 1998) by treating labels (+1 and −1) as the target regression values. The algorithm is outlined as follows.

1. Apply one-class SVM to train labeled data and get a classifier M_1, and use M_1 to predict unlabeled data.
2. Randomly choose m points with decision values in the bottom $p\%$, and treat them as negative. (p is small in general.)
3. Train an SVR model M_2, and use M_2 to predict all data.

Parameters p and m are selected by users.

2.2.4. Transductive Support Vector Machines

Transductive SVM (TSVM) (Joachims, 1999) is a semi-supervised technique for partially labeled data. We consider the formulation used in Sindhwani and Keerthi (2006). TSVM adjusts labels of the unlabeled instances to maximize the margin between two classes. Sindhwani and Keerthi (2006) impose a constraint in the optimization problem so that the ratio of unlabeled data assigned as positive is r. Users must provide this positive ratio.

2.3. Querying Strategies

We propose several querying strategies. In particular, a general decision value fitting algorithm is proposed in Section 2.3.2.

Assume we are given an upper bound m on the number of points to query, an index set Q of labeled points, and the set of decision values F. If $\boldsymbol{w}$ is the SVM or LR weight vector, any $f_i \in F$ is the decision value $\boldsymbol{w}^T \boldsymbol{x}_i$ of a data point $\boldsymbol{x}_i$. In the competition, F includes the decision values of all training instances, but in post submissions, we only consider the decision values of unlabeled training instances. We want to find a set of points S and query their labels. These newly labeled points in S should help to improve the performance.

2.3.1. Some Simple Existing Methods

We consider some naïve or existing querying methods.

- **No Active Learning**
  In this method, we consider $S = \{1 \le i \le l\} \backslash Q$. That is, we query all the unlabeled training points at a time.
- **Random Query**
  Let P be a set generated by randomly drawing a point from $\{1, \ldots, l\}$ m times. Then we consider $S = P \backslash Q$. Since repetition is possible, the set P may have less than m points. Results of this approach may not be very stable due to the randomness.
- **Simple Query**
  Tong and Koller (2002) proposed a querying method for active learning with SVM. It suggests querying points which are close to the SVM hyperplane. Since the distance between a point and the hyperplane is $|\boldsymbol{w}^T \boldsymbol{x}_i| / \sqrt{\boldsymbol{w}^T \boldsymbol{w}}$, we identify points with the smallest m values in $\{|f_1|, \ldots, |f_l|\}$ as the set P. Then $S = P \backslash Q$.

2.3.2. A Decision Value Fitting Algorithm

While the above "simple query" method finds points close to the decision boundary, here we proposed a general algorithm to flexibly select query points based on decision values. Assume all decision values in F are linearly scaled to an interval $[-\Delta, \Delta]$. We can apply any discretization method to obtain m grid points in this interval. Then data instances with scaled decision values close to these grid points are selected for querying their labels. For example, if most grid points are close to 0, then points with decision values close to the decision boundary are selected. Our procedure requires two functions: μ discretizes $[-\Delta, \Delta]$ to obtain grid points and ψ scales F to $[-\Delta, \Delta]$. The detailed procedure is described below.

1. Select a mapping function μ.
2. Set some scale function $\psi : F \to \text{Range}(\mu)$.
3. Set $T = \{\mu(-1 + 2i/m) | 0 \le i \le m\}$.
4. Set $P = \{j | j = \arg\min_{1 \le i \le l} |\psi(f_i) - t|, \text{where } t \in T\}$.
5. Set $S = P \backslash Q$. If S is empty, do "Random Query." Finally, Output S for query.

Assume that evaluating $\psi(f_i)$ takes $O(1)$ time. Then the complexity of the above algorithm is $O(l \log l)$. The implementation is by sorting $\psi(F)$ and T, and then going through the sorted sequences once to construct the set P.

In Sections 2.3.3 and 2.3.4, we show two examples of the above algorithm.

2.3.3. Uniformly-discretized Decision Value Query

We consider uniformly selecting points according to the distribution of decision values. The setting here covers points with different distances to the decision boundary.

We scale F to $[-1, 1]$ by the following function:

$$\begin{cases} \psi_{[-1,1]}(x) = 2\left(\frac{x - \min(F)}{\max(F) - \min(F)} - \frac{1}{2}\right) & \text{if } \max(F) \neq \min(F), \\ \psi_{[-1,1]}(x) = 0 & \text{otherwise,} \end{cases}$$

set μ to the identity function, and find the following points to uniformly discretize $[-1,1]$ to m intervals:

$$T = \left\{ -1 + \frac{2i}{m} \middle| 0 \le i \le m \right\}.$$

We choose points whose decision values are close to values in T:

$$P = \{j | j = \arg\min_{1 \le i \le l} |\psi_{[-1,1]}(f_i) - t|, \text{where } t \in T\}.$$

When participating in the competition, we include all decision values in the set F. Thus P may include labeled instances, and we let $S = P \backslash Q$ be the set of points for query. For post-competition submissions presented in this paper, F only includes decision values of unlabeled training instances, so $P \backslash Q$ is actually equal to P. Besides, $T = \{-1 + 2i/m | 1 \le i \le m-1\}$ is considered in our competition submission because points with too large decision values may not be informative or may be already labeled instances.

When m is large, T may be very dense. In this situation, it is possible that P contains less than m points.

2.3.4. Arcsin-discretized Decision Value Query

We propose a strategy combining both "simple query" and "uniformly-discretized decision value query." The idea is to query more points with small absolute decision values, but also query some points far away from the decision boundary. We achieve this by using a nonlinear function μ to discretize the space of decision values.

We first scale F to $[-\pi/2, \pi/2]$ by the following function:

$$\begin{cases} \psi_{[-\pi/2,\pi/2]}(x) = \pi\left(\frac{x - \min(F)}{\max(F) - \min(F)} - \frac{1}{2}\right) & \text{if } \max(F) \neq \min(F), \\ \psi_{[-\pi/2,\pi/2]}(x) = 0 & \text{otherwise.} \end{cases}$$

Next, we let μ be the arcsin function to have more discretized points around the origin:

$$T = \left\{ \arcsin\left(-1 + \frac{2i}{m}\right) \middle| 0 \le i \le m \right\}.$$

Then, we set P according to the definition in Section 2.3.2. S is set as $P \backslash Q$ likewise.

Following the same reason stated in Section 2.3.3, for our challenge submissions, $T = \{-1 + 2i/m | 1 \le i \le m-1\}$ is considered.

3. Experiments

In this section, we present experimental results. In the competition, all the training and testing feature values are available, but only one training instance is labeled. Participants can query labels of training instances, and the number of samples per query is not restricted. Before querying labels of some training instances, participants must submit the predicted decision values of all instances (training and testing data) based on their current model, so that AUC can be calculated. A learning curve is then constructed as

Table 1: Data information. Both training and testing sets contain the same number of instances.

(*a*) Development data sets.

data set	feature type	sparsity (%)	missing value (%)	positive ratio (%)
HIVA	binary	90.88	0	2.35
IBN_SINA	mixed	80.67	0	37.84
NOVA	binary	99.67	0	18.34
ORANGE	mixed	9.57	65.46	1.78
SYLVA	mixed	77.88	0	6.15
ZEBRA	continuous	0.04	0.004	4.58

(*b*) Challenge data sets.

data set	feature type	sparsity (%)	missing value (%)
A	mixed	79.02	0
B	mixed	46.89	25.76
C	mixed	8.6	0
D	binary	99.67	0
E	continuous	0.04	0.0004
F	mixed	1.02	0

a line chart of AUC versus the log-scaled number of labeled training instances. The evaluation of competition results is based on the area under the learning curve (ALC).

There are six development data sets and six challenge data sets. Development data sets are used for participants to tune systems and algorithms, while the evaluation is based on results for the challenge data sets. The six development data sets are: HIVA, IBN_SINA, NOVA, ORANGE, SYLVA, and ZEBRA, and the six challenge data sets are named as A, B, C, D, E, and F. For every data set, the number of training instances is the same as the number of testing instances. Tables 1(*a*) and 1(*b*) respectively describe details of development and challenge data sets.

The development and the challenge data sets are in similar domains, but the domain information and the ratio of positive labels in challenge data sets are concealed.

3.1. Classification Methods, Software, and Implementations

Other than in the situation where there is only one labeled point, we consider standard SVM (Boser et al., 1992; Cortes and Vapnik, 1995) or logistic regression to train labeled data. We solve

$$\min_{\boldsymbol{w},b} \quad \begin{cases} \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \|\boldsymbol{w}\|_1 \end{cases} + C^+ \sum_{i:y_i=1} \xi(\boldsymbol{w};\boldsymbol{x}_i,y_i) + C^- \sum_{i:y_i=-1} \xi(\boldsymbol{w};\boldsymbol{x}_i,y_i), \tag{1}$$

Table 2: AUC using different methods to handle the situation of having only one labeled training point.

Method	HIVA	NOVA	IBN_SINA	ORANGE	SYLVA	ZEBRA
All negative	0.530	0.656	0.424	0.514	0.774	0.402
OSVM linear kernel	**0.532**	0.672	0.424	0.514	0.774	0.402
OSVM RBF kernel	0.382	0.261	0.793	0.549	0.855	0.685
OSVM sigmoid kernel	0.532	0.672	0.424	0.514	0.774	0.402
OSVM Laplacian kernel	0.382	0.261	0.781	0.557	0.862	0.680
OSVM + SVR	0.505	**0.688**	0.798	0.534	0.840	0.705
TSVM $r = 0.1$	0.413	0.356	**0.883**	0.563	**0.943**	**0.707**
TSVM r =real postive ratio	0.432	0.332	0.823	**0.577**	0.932	0.704

where

$$\begin{aligned}
\xi^{\mathrm{L1}}(\boldsymbol{w}; \boldsymbol{x}_i, y_i) &= \max(1 - y_i(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b), 0), \\
\xi^{\mathrm{L2}}(\boldsymbol{w}; \boldsymbol{x}_i, y_i) &= \max(1 - y_i(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b), 0)^2, \text{ and} \\
\xi^{\mathrm{LR}}(\boldsymbol{w}; \boldsymbol{x}_i, y_i) &= \log(1 + e^{-y_i(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b)})
\end{aligned} \tag{2}$$

are respectively L1, L2, and LR (Logistic Regression) loss functions. Parameters C^+ and C^- are used for positive and negative classes as data are unbalanced (Vapnik, 1998). The function $\phi(\boldsymbol{x})$ maps data to a higher dimensional space. We may employ the kernel trick so that only $K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$ is needed. We experiment with different regularization terms and loss functions, which are referred to as L2RL1, L2RL2, L1RLR, and L2RLR. If not explicitly stated, L2RL1 uses the RBF kernel; other solvers use the linear kernel.

For convenience, we name methods in Section 2.2 as "All negative," "OSVM," "OSVM + SVR," and "TSVM," respectively. For querying methods in Section 2.3, they are named "NO AL," "random,""simple," "uniform," and "arcsin," respectively.

For L2RL1, OSVM, and SVR, we use the software LIBSVM (Chang and Lin, 2001) and its extension (Lin and Li, 2008), in which more kernels are implemented. For example, "L2RL1 Laplacian" indicates that the Laplacian kernel is used. For L2RL2, L1RLR, and L2RLR, we consider only the linear kernel and employ the software LIBLINEAR (Fan et al., 2008).[2] For TSVM, we use SVMlin as the solver (Sindhwani and Keerthi, 2006). Parameter selection is important for SVM and LR. If a nonlinear kernel is used, we search for C^+ and the kernel parameter by setting $C^- = C^+$.[3] For the linear kernel, we check various C^- and C^+/C^-. Unfortunately, during the competition, our parameter selection is not very systematic.

2. Note that LIBLINEAR does not consider the bias term b in (2).
3. If we do not fix C^-, the cost for parameter selection may be too high.

3.2. Results on the Development Sets

Before doing experiments, all six data sets except IBN_SINA are scaled so that each feature takes values in $[0, 1]$ (HIVA and NOVA already have binary features). We do not scale IBN_SINA because the results are even slightly worse after scaling. In the data set ORANGE, 187 of 230 features contain missing values, so additional indicator features are used. That is, for each instance, we use another 187 binary features to indicate if the corresponding value is missing (0) or not (1). ZEBRA has very few missing values (0.004%). We simply assign these missing values to zero. Only four of ZEBRA's 61,488 instances are affected.

3.2.1. The First Labeled Point

When there is only one labeled point, we use methods described in Section 2.2. We do not conduct parameter selection because of a concern on overfitting. For "All negative," we use L2RLR with $C^- = 10^{-11}$ and $C^+ = 1$. For OSVM + SVR, we set $p = 10$ and $m = 3$; RBF kernel is used for non-categorical data (IBN_SINA, ORANGE, SYLVA, and ZEBRA), while linear kernel is used for categorical data (HIVA, NOVA). We found through experiments that for categorical data linear kernel provides competitive results and enjoys fast training. For TSVM, two settings for the parameter r, positive class fraction of unlabeled data, are compared. One is the real positive ratio for each data set, which is only given in development data sets; the other is 0.1 for all data sets. A constant ratio is experimented because later we do not know the positive ratio of challenge data sets.

Table 2 shows testing AUC values by each method in Section 2.2. If the method involves some random selections (e.g., OSVM + SVR), we conduct experiments five times and present the average result. For each data set, the bold-faced value indicates the best classifier's AUC.

From Table 2, we find that TSVM outperforms other classifiers in most data sets, and setting the positive ratio to 0.1 may be better than using the real positive ratio. However, TSVM performs very poorly on HIVA and NOVA. In contrast, OSVM + SVR performs reasonably well on all problems.

3.2.2. Subsequent Queries

Once we have produced a classifier using the first labeled point, we can query labels of some training points. For subsequent binary classification, we employ L2RLR on categorical data sets (HIVA and NOVA), while nonlinear classifiers (L2RL1) on other non-categorical data sets (IBN_SINA, SYLVA, and ZEBRA). Though ORANGE is non-categorical, we employ a linear classifier L1RLR. The reason is that ORANGE contains missing values. Our previous study for KDD Cup 2009 shows that linear classifiers, especially L1RLR, with missing value indicators yields good results.

We conduct parameter selection on each data set with the classifiers described above, and obtain the best classifier. Table 4 presents the parameters we use.

Table 3: Comparison of mean and standard deviation of ALC values using various querying methods.

(*a*) Linear classifiers for categorical data

Method	HIVA	NOVA	ORANGE
NO AL	**0.320±0.000**	0.643±0.060	**0.378±0.000**
random	0.177±0.052	0.677±0.060	0.265±0.026
simple	0.083±0.000	0.694±0.000	0.265±0.000
uniform	0.168±0.000	0.751±0.000	0.249±0.000
arcsin	0.133±0.000	**0.753±0.000**	0.226±0.000

(*b*) Nonlinear classifiers for mixed (categorical and numeric) feature-valued data

Method	IBN_SINA	SYLVA	ZEBRA
No AL	0.874±0.000	0.941±0.000	**0.550±0.000**
random	0.897±0.018	0.940±0.010	0.395±0.033
simple	0.723±0.000	0.800±0.000	0.274±0.000
uniform	0.860±0.000	0.935±0.000	0.387±0.000
arcsin	**0.900±0.000**	**0.967±0.000**	0.308±0.000

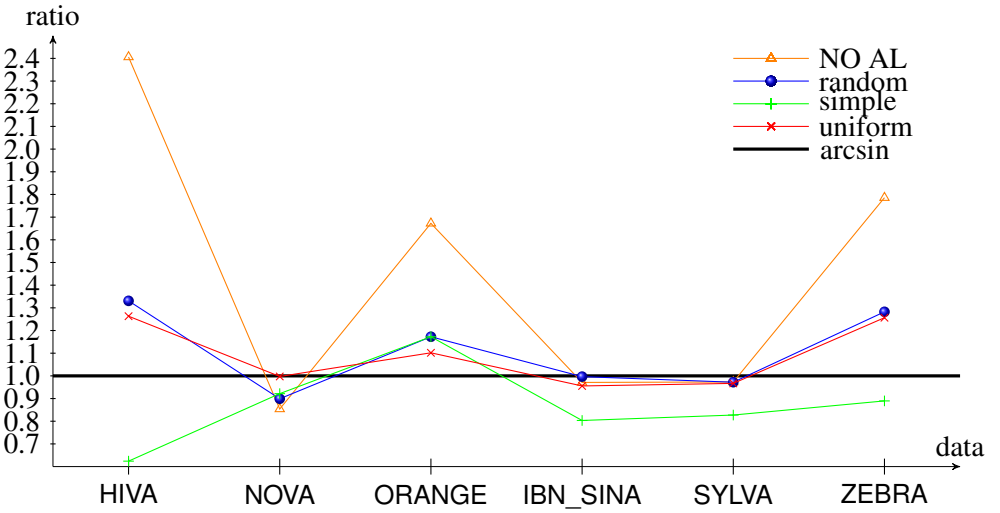


Figure 2: A comparison between arcsin-discretized and other querying methods. The ration means (ALC by any method)/(ALC by the arcsin method).

For each data set, the method for the first labeled point is either "TSVM $r = 0.1$" or "All negative" according to which one gives a higher AUC in Table 2, and we use the

Table 4: Final AUC values using the classifier (with parameter selection) that yields the highest ALC on each data set.

(a) Linear classifier

Data set	classifier	C^-	C^+	AUC
HIVA	L2RLR	2	2	0.790
NOVA	L2RLR	32	32	0.988
ORANGE	L1RLR	0.125	1	0.815

(b) Nonlinear classifier

Data set	classifier	C	γ	AUC
IBN_SINA	L2RL1	8	0.03125	0.991
SYLVA	L2RL1	1	1	0.999
ZEBRA	L2RL1	32	0.5	0.842

same querying method in the whole active learning process. As mentioned in Section 2.1, we sequentially query labels of s, $2s$, 2^2s, 2^3s, ... points until the AUC does not change significantly or there is no data with unknown labels. The value s is 16 in the experiment here, but we use different s when participating in the competition.

The resulting ALC of all querying methods are shown in Tables 3(a) and 3(b) according whether linear or nonlinear classifiers are used. Due to the randomness in the algorithms, each experiment is conducted five times, and the mean and standard deviation of ALC values are reported. In both tables, methods resulting the highest mean ALC values for each data set are marked in bold. Further, Figure 2 shows how the arcsin querying method is compared with others.

Table 4 presents the final AUC after all labels of the training set are available. That is, we train the whole training set and predict the testing set. Parameter selection is conducted on the training set.

From Tables 2-4, we observe that if the initial and final AUC values are low, the performance without active learning methods is better. Take HIVA as an example, without active learning, it has initial AUC 0.530, final AUC 0.790, and ALC 0.320. Other active learning methods' ALC values are below 0.190. Regarding querying methods, in general, arcsin-discretized query works better than uniformly discretized query, and simple query is even worse. However, the best querying methods seems to be data dependent.

3.2.3. Stableness of TSVM

From Table 2, we observe that TSVM yields very high and very low AUC values on different data sets. In addition, in most data sets, setting r in TSVM to a constant ratio is better than using the real positive ratio, which varies from 1.78% to 37.84%. We thus study how the positive ratio r in TSVM affects AUC.

We randomly sample 30 positive data in each data set as the first labeled point. Then, we run TSVM with $r = 0.05, 0.10, \ldots, 0.40$, and the real positive ratio. The resulting AUC values are shown in Table 5 and Figure 3. We observe that a small r tends to give a higher mean and a smaller variance on many data sets. This result seems to indicate that using a smaller r for TSVM is better.

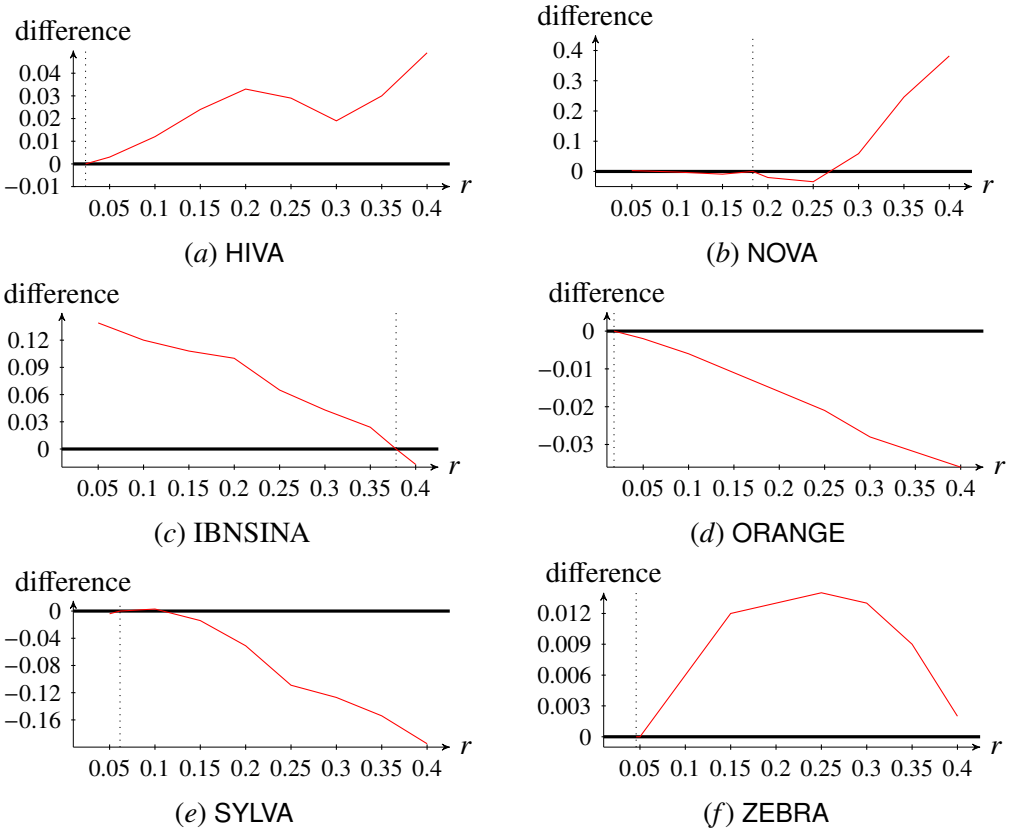


Figure 3: AUC value differences between using various r values and using the real positive ratio for TSVM. The dotted vertical line and the horizontal solid line respectively indicate the true positive ratio and the performance of using it.

Table 5: Mean and standard deviation of AUC for different r values in TSVM.

r	HIVA	NOVA	IBN_SINA	ORANGE	SYLVA	ZEBRA
0.05	0.453±0.054	0.372±0.011	0.783±0.086	0.534±0.038	0.878±0.063	0.612±0.111
0.10	0.462±0.048	0.367±0.011	0.764±0.106	0.530±0.036	0.885±0.072	0.618±0.115
0.15	0.474±0.054	0.360±0.011	0.752±0.141	0.525±0.034	0.868±0.072	0.624±0.119
0.20	0.483±0.048	0.349±0.013	0.744±0.162	0.520±0.030	0.831±0.067	0.625±0.127
0.25	0.479±0.044	0.335±0.019	0.709±0.192	0.515±0.026	0.773±0.058	0.626±0.133
0.30	0.469±0.025	0.428±0.245	0.687±0.217	0.508±0.021	0.755±0.047	0.625±0.140
0.35	0.480±0.040	0.615±0.324	0.668±0.215	0.504±0.017	0.728±0.056	0.621±0.149
0.40	0.499±0.053	0.751±0.295	0.627±0.195	0.500±0.017	0.687±0.033	0.614±0.160
real	0.450±0.052	0.369±0.164	0.644±0.200	0.536±0.039	0.882±0.065	0.612±0.111

3.3. Competition Results and Post Challenge Submissions

During the competition, our procedure is slightly ad hoc. After the competition, we prepare a more systematic procedure and obtain some new results. A comparison among our competition results, our post challenge submissions, and the best competition results by participants is reported.

3.3.1. Methods for Challenge Data Sets

From the experiments on development data sets, we know that selecting proper methods for each data set is important. Because challenge sets are from the same domains as development sets, we try to obtain one-to-one mappings between them. Then methods suitable for a development set can be applied to the corresponding challenge set. To do the mapping, we look for missing values, sparsity, and feature types in development and challenge data sets. We are able to identify the following relationships.

- A is IBN_SINA because they have the same feature numbers and similar sparsity.
- B is ORANGE because they have the largest number of missing values among all data sets.
- D is NOVA because they have the same sparsity and the same feature type.
- E is ZEBRA because they have both numerical feature type.

However, it is not clear what C and F are. We decide to use methods for SYLVA for these two sets in the competition. After the competition ended, we know from the organizer that C is HIVA and F is SYLVA. Such information is used in selecting methods for our post-competition submission.

Once the data mapping is determined, we port parameters and methods from development data. In selecting our final submission method, we try to the balance between stableness and high ALC. Table 6 shows methods used in the competition. Below we consider a more systematic procedure and use it to get some post-challenge results:

1. When there is only one labeled point, choose a better method between "All negative" and TSVM $r = 0.1$ using Table 2.
2. For $i = 0, 1, 2, \ldots$

Table 6: The methods and parameters on challenge data sets.

	competition submissions			post-competition submissions		
data	first point	querying method	s	first point	querying method	s
A	OSVM + SVR	uniform + L2RL1	8	TSVM 0.1	arcsin + L2RL1	16
B	TSVM real	No AL + L1RLR	All	TSVM 0.1	No AL + L1RLR	All
C	TSVM 0.11	uniform + L2RL2	2,3,256	All negative	No AL + L2RLR	All
D	OSVM + SVR	arcsin + L2RLR	16	All negative	arcsin + L2RLR	16
E	OSVM RBF	No AL + L2RL1	All	TSVM 0.1	No AL + L2RL1	All
F	OSVM RBF	uniform + L2RL1	8	TSVM 0.1	arcsin + L2RL1	16

Table 7: Competition results. The column iniAUC means the AUC value when there is only one labeled point, while finAUC means the final AUC when all training instances are labeled.

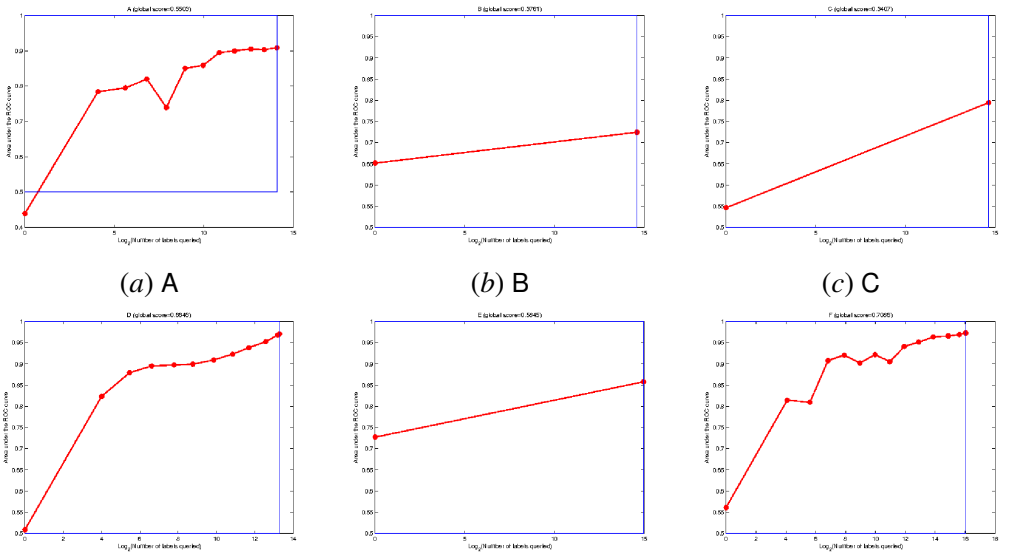
	post-competition submissions				our competition results				best results	
data	iniAUC	finAUC	ALC	rank	iniAUC	finAUC	ALC	rank	finAUC	ALC
A	0.439	0.908	0.550	3	0.439	0.928	0.553	3	0.962	0.629
B	0.652	0.724	0.376	1	0.643	0.733	0.376	1	0.767	0.376
C	0.546	0.794	0.341	4	0.428	0.779	0.199	11	0.833	0.427
D	0.509	0.970	0.665	3	0.433	0.970	0.662	3	0.972	0.861
E	0.727	0.858	0.585	2	0.726	0.857	0.584	2	0.909	0.627
F	0.561	0.973	0.709	6	0.534	0.997	0.669	9	0.998	0.802

- Query labels by the following methods, and use the corresponding solver in Tables 4(*a*) and 4(*b*) to train and predict.
  - See Table 3. If active learning seems to work (e.g., IBN_SINA,NOVA, and SYLVA), do arcsin-discretized query with $s = 16$. That is, query $2^i s$ points.
  - Otherwise, query all the remaining labels.
- If all training instances are labeled, stop.

Table 6 also lists methods applied by using the above procedure. They are slightly different from those used in the competition.

3.3.2. RESULTS

We compare our AUC and ALC value with the highest AUC and ALC among all contestants in Table 7. Here AUC means the final AUC using the whole training set. We also list our rank for each problem. Clearly, our post-challenge submissions give slightly better results than our competition submissions. The learning curves of the post-competition submissions are in Figure 4.



(*a*) A (*b*) B (*c*) C

(*d*) D (*e*) E (*f*) F

Figure 4: Learning curves of challenge data sets in post-competition submissions.

4. Discussion and Conclusions

In this work, we consider the active learning framework shown in Figure 1. Since the evaluation criteria is ALC with log-scaled x-axis, the performance at the first iteration is very important. We investigate various methods to obtain a good classifier when there is only one labeled point. We also compare all these methods on the six data sets. Results indicate that the best method seems to be data dependent, but semi-supervised methods, e.g., transductive SVM, can generate comparatively good results. Then for subsequent queries, we develop a general algorithm to select a set of points. The algorithm uses only decision values from classifiers, so little extra cost is needed. Users can choose a suitable mapping functions μ in Section 2.3.2 for their data sets. Unfortunately, we have neither theoretical justification for the performance of our methods, nor do we know how severely the results may be effected using different mapping functions. These issues are possible future works. However, while none of the methods is the best, we observe that querying methods based on arcsin-discretized and uniformly-discretized decision values are better than others in our experiments, and the uniformly-discretized approach generates pretty stable results.

References

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Corina Cortes and Vladimir Vapnik. Support-vector network. *Machine Learning*, 20: 273–297, 1995.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. URL http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf.

Isabelle Guyon, Gavin Cawley, Gideon Dror, and Vincent Lemaire. Results of the active learning challenge. In *Active Learning Challenge*, volume 15, 2011.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of International Conference on Machine Learning*, 1999.

Hsuan-Tien Lin and Ling Li. Support vector machinery for infinite ensemble learning. *Journal of Machine Learning Research*, 9:285–312, 2008.

Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alexander J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. *Proceedings of the Fifth Workshop on Computational Learning Theory*, pages 287–294, 1992.

Vikas Sindhwani and S. Sathiya Keerthi. Large scale semisupervised linear SVMs. *29th Annual International ACM SIGIR*, 2006.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2002.

Ming-Hen Tsai, Chia-Hua Ho, and Chih-Jen Lin. Active learning strategies using SVM. In *Proceedings of IJCNN*, 2010.

Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

JMLR: Workshop and Conference Proceedings 16:85–98, 2011 Active Learning Challenge

Stochastic Semi-supervised Learning on Partially Labeled Imbalanced Data

Jianjun Xie JIANJUNXIE@GMAIL.COM
CoreLogic, 703 Palomar Airport Road, Carlsbad, CA 92021, USA

Tao Xiong TAO.XIONG@GMAIL.COM
eBay Inc., 2145 Hamilton Avenue, San Jose, CA 95125, USA

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

In this paper, we describe the stochastic semi-supervised learning approach that we used in our submission to all six tasks in 2009-2010 Active Learning Challenge. The method is designed to tackle the binary classification problem under the condition that the number of labeled data points is extremely small and the two classes are highly imbalanced. It starts with only one positive seed given by the contest organizer. We randomly pick additional unlabeled data points and treat them as “negative" seeds based on the fact that the positive label is rare across all datasets. A classifier is trained using the “labeled" data points and then is used to predict the unlabeled dataset. We take the final result to be the average of n stochastic iterations. Supervised learning was used as a large number of labels were purchased. Our approach is shown to work well in 5 out of 6 datasets. The overall results ranked 3rd in the contest.

Keywords: Active Learning, Semi-supervised Learning, Gradient Boosting Decision Tree

1. Introduction

The 2009-2010 active learning challenge consisted of six real world datasets from six different domains: handwriting recognition (A), marketing (B), chemo-informatics (C), text processing (D), embryology (E) and ecology (F) (Guyon et al., 2011). Each data set has a different number of features, a different number of records and a different positive label percentage. They are all binary classification problems with an imbalanced distribution of the two classes. Each dataset has been split into training and testing randomly. An initial positive seed is given in the training set. The participants were asked to submit the prediction to all the samples with unknown labels based on the queries had been made.

The prediction performance metric is the Area under the Learning Curve (ALC) which is referred to as the global score. A learning curve plots the Area Under the ROC curve (AUC) computed on all the samples with unknown labels, as a function of the number of queried labels (including the initial seed). In order to emphasize the model performance with few known labels, the x-axis is $\log 2$ scaled.

Six development datasets were made available before the final contest datasets were released. This provided the participants the opportunity to develop query strategies as well as to select the best learning method. Even though the final datasets are extracted from the same domain as the development datasets, they are different enough so that participants are not able to directly apply what they learned from development datasets. The domain of each final dataset and the label distribution are anonymized.

Active learning algorithms have seen many applications during the past decade in such areas as text classification (Tong and Koller, 2001), image classification (Luo et al., 2005), software testing (Bowring et al., 2004) and so on. Generally speaking, there are three learning scenarios of active learners: (i) membership query synthesis, (ii) stream-based selective sampling, and (iii) pool-based sampling. Pool-based active learning is the setting used in this challenge. In pool-based active learning, there are typically three strategies of querying unlabeled instances (Settles, 2009). First is uncertainty sampling. The examples whose predicted label (based on the current classifier estimate) is most ambiguous are picked first for label inquiry. Among others, measures of uncertainty include disagreement among oracles in Query by Committee (Freund et al., 1997), confidence of classification (Lewis and Gale, 1994), and distance to a decision boundary in SVMs (Tong and Koller, 2001). Second is called reducing future error, Roy and McCallum (2001) proposed to pick examples that minimize the generalization error probability. Because it is impossible to know future generalization errors, it uses the current classifier to estimate the probabilities for each unlabeled example. The third type uses ensembles of active learners. Baram et al. (2004) developed a master algorithm that picks the best expert from an ensemble of active learners depending on their performance. A more comprehensive and detailed literature review of those strategies can be found in (Settles, 2009).

When classifiers are trained, active learning algorithm usually takes advantage of the fact that both labeled and unlabeled data instances are available. Typically some form of semi-supervised learning algorithm is used for better performance. Examples of semi-supervised learning techniques include co-training (Blum and Mitchell, 1998), self-training (Rosenberg et al., 2005), cluster-and-label (Demiriz et al., 1999; Dara et al., 2002)and so on. A detailed literature survey on semi-supervised learning can be found in (Zhu, 2005).

In this contest, we proposed a stochastic semi-supervised learning approach to handle the active learning challenge when the number of labeled data is extremely small. We borrowed the concept of self-training in our logistic regression approach and the idea of cluster-and-label in our k-means clustering approach. We incorporated these ideas into a stochastic sample-train-label process. The details of the approach will be given in Section 2. We summarize our results and the comparison with others in Section 3. Finally, conclusion of our work is given in Section 4.

2. Our Approach

The method we used is a stochastic semi-supervised learning process. It was proposed mainly based on the following two facts in this contest. First, the number of avail-

able labeled examples is extremely small, while the number of unlabeled examples is abundant. Second, the positively labeled exemplars are rare comparing with negatively labeled ones. In other words, the probability of getting a negatively labeled exemplar through random sampling from the unlabeled pool is much higher than the probability of getting a positively labeled exemplar.

Two classifiers are used in the stochastic semi-supervised learning process. One is clustering and another is logistic regression. The criteria we used to choose clustering or logistic regression are based on the following factors.

1. Number of features. If the number of feature is very large, say more than 800, clustering is preferred. This is because clustering is an unsupervised approach. We can use the whole dataset to do clustering with the initially available label as seed. While logistic regression is a supervised learning, we have to get enough labels at the beginning to build a meaningful model. A large number of features will increase the complexity of the model with very limited available examples. Since dataset C and D are two datasets with the largest number of features (i.e. 12000 for C and 851 for D), we choose clustering for these two datasets.

2. Distribution of two classes. We prefer to use logistic regression if the two classes are extremely unbalanced. This is because if one classes is extremely rare, clustering method may treat them as outliers and ignore them. On the other hand, it is a lot easier to get right by randomly picking some unlabeled examples and labeling them as majority class. If the two classes are more balanced, we would choose clustering because it is too easy to be wrong for the initial labeling by random guess. Even though the class distribution was not available in the test dataset, it was not hard to figure out which development dataset it was corresponding to. We therefore chose clustering for dataset A since it corresponds to handwriting recognition which has the highest percentage of positive label in development dataset (37%). We decided to use logistic regression approach for the remaining 3 datasets.

Algorithm 1 details the steps we used in the contest before we did any label purchase. Each dataset is given one positive seed by the organizer to start with. For dataset A, C and D, we randomly pick another data point from the unlabeled data pool as a negative seed. We use these two seeds as our initial cluster centers for k-means clustering. We repeat this process n times, each time with a different randomly picked negative seed and the same positive seed. We label the cluster where the positive seed resides as positive cluster, the other one as negative cluster. We calculate the count of positive cluster membership of each data point after n iterations and use the normalized membership count as the prediction score.

For dataset B, E and F, we randomly pick 20 unlabeled data points as negative label for each positive label. This assumes that the positive label in these datasets was less than 5%. This is true in the corresponding development datasets. We build a logistic regression model using the “labeled" data points. We repeat n iterations of the above random sampling/modeling process and take the average score as the prediction score.

Algorithm 1: Stochastic semi-supervised learning process
Given one positive label, N unlabeled examples:

1. For dataset A, C and D
    2. Set $i = 1$
        3. Randomly pick one example from N unlabeled examples as “negative" example
        4. Use the positive label and the “negative" label as initial seeds, do k-means clustering on whole dataset with number of cluster = 2
        5. Label cluster where positive seed sits as positive, another one as negative
        6. Save cluster membership of each example $f_i(c)$ where $f(c = positive) = 1; f(c = negative) = 0$.
        7. Increase i by 1. If $i < 100$ return to step 3
    8. Calculate final predicted score for each example using $\frac{1}{M}\sum_{i=1}^{M} f_i(c)$ where $M = 100$.
9. For dataset B, E and F
    10. Set $i = 1$
        11. Randomly pick 20 examples from N unlabeled examples as “negative" examples
        12. Use the one positive label and the 20 “negative" labels as training set, build a logistic regression model
        13. Score the whole dataset, save score for each example f_i
        14. Increase i by 1. If $i < 100$ return to step 11.
    15. Calculate average score for each example using $\frac{1}{M}\sum_{i=1}^{M} f_i$ where $M = 100$.
    16. Label highest 1% of score as positive examples, lowest 1% of score as negative examples, rebuild the logistic regression model (self-training).
    17. Calculate final score for each example using above logistic regression model.

This score is used to label all other unlabeled data (the concept of self-training). Final logistic regression model is built on dataset with the "derived" labels.

When more labels are available through the query, we mix these labeled data with the sampled "negative" data together as initial seeds (for k-means) or modeling dataset (for logistic regression). We repeat the stochastic process after each label query. The known labels are always kept in the modeling dataset, while the "derived" labels are changing each time. We put more weights on the labeled data points this way. The random sampling process is repeated n time as described above. The final score is the arithmetic average of the n stochastic process. We take $n = 100$ in this contest.

The above semi-supervised learning approach is used when the number of available labels is extremely small. When the amount of the labeled data becomes large, we tend to use Gradient Boosting Decision Tree (TreeNet) (Friedman, 1999) as our classifier to generate prediction score. We use a switching threshold of approximately 200 in this work. This corresponds to the middle range of x value for all 6 datasets in the area under the learning curve plot because of the $\log 2$ scaling on the number of purchased labels. It is not possible to conduct an experiment to figure out the best threshold value in the contest since there is only one chance for each team to develop their approach. We heuristically obtained this value from our experiments on the development datasets. However, the test datasets were modified by the organizer so that they were different enough from the development sets even for the ones from the same domain. We took this value as a reference in the contest. For most of our label queries, we directly jumped to a large purchase (> 1000 labels) from a very small purchase (less than 100). We only built a supervised learning model on a single dataset (dataset A) with 233 purchased labels.

2.1. Dataset A: Handwriting Recognition

Every team had three chances to work on the datasets from this domain: development phase, contest phase and verification phase. Only the verification phase counted in the competition because every team got different labels during the contest stage for the same dataset, therefore the results were not comparable. This design was used to detect potential cheaters.

We will present the details of our experiments in this paper only on the contest phase and the verification phase. For these two phases, all the input fields are exactly same. The difference is that the training labels, as shown in Table 1 are altered. We can see that 688 negative training labels in verification phase were changed to positive labels in contest phase for our case. This was of course not known during the contest.

As stated in Algorithm 1, we used k-means algorithm (PROC FASTCLUS in SAS software) to do our initial classification with only one positive seed available. We randomly picked one data point as negative seed, together with the known positive seed as the initial cluster center for two classes. Only numerical features were used. Each feature was standardized by z-scaling before the clustering process. The distance between each data point and the seed is based on Euclidean distance. After the clustering process, we labeled the cluster where the positive seed lives as positive cluster, another

one as negative cluster. We stored the cluster membership of each data points. Above *k*-means clustering process was repeated 100 times. We calculated the count of positive cluster membership of each data point after 100 iterations and used the normalized membership count as the prediction score.

Table 1: Label difference of dataset A (training sample) between contest phase and verification phase.

Labels in contest phase	Labels in verification phase	Frequency	Percentage
-1	-1	15579	88.85%
1	-1	688	3.92%
1	1	1267	7.23%

During the contest phase of dataset A, we submitted 9 queries. Details of each query is listed in Table 2. We used the prediction score as uncertainty measure. The first query was carried out by randomly picking 2 data samples within the score range of 0.5 to 0.6 which is in 60 - 66 percentile of all training data samples. We got one positive label and one negative label. We used these 2 labels together with the first seed as the new seeds for the next round of *k*-means clustering. We randomly picked other unlabeled data from the pool as “negative" seeds as described in first step. We repeated our *k*-means process 100 times. Each time we got 2 clusters which were labeled by the known seeds. We again averaged the membership counts of each data points over 100 to create our prediction score. This semi-supervised learning process was used in the first 7 submissions. The total cost was 67 including the first seed. The query samples for submissions 2 through 6 were based on value of prediction score (in the range of 0.5 and 0.6). We started big label purchases from submission 7. Random sampling was used in these large queries. We had some concerns that the uncertainty sampling might have bias because it was based on the “current" model’s prediction which was built on small number of labels. We used gradient boosting decision tree as the classifier in our submissions 8 to 10. The typical parameter setting was as following: number of trees = 700, learning rate = 0.015, subsample rate = 0.7, number of nodes = 6, minimum number of child =100, percentage of testing = 0.25.

In the verification phase, we used exactly same classification approach as the contest phase. However, we changed the query strategy. Our experience during the contest phase, taught us that there were no significant improvements in the first several small queries. In fact, it actually lowered the global score. Therefore, we decided to conduct a relatively large query in the first submission. We used the exactly same prediction score in the first submission as in contest phase. This is because the dataset was exactly same, the first seed was exactly same, only some labels were different which we did not know without querying. We obtained 233 labels after the first query which enables us to use boosting decision tree algorithm to build classification model. We used the same algorithm in the last 3 submissions. Uncertainty and selective sampling was used

Table 2: Queries submitted in contest phase for dataset A. The final global score = 0.45.

Submission Sequence	Number of Samples in Query	Number of Queried Samples	AUC	Sampling Strategy
1	2	1	0.54	Uncertainty and selective
2	2	3	0.61	Uncertainty and Selective
3	4	5	0.61	Uncertainty and Selective
4	9	9	0.63	Uncertainty and Selective
5	31	18	0.64	Uncertainty and Selective
6	18	49	0.63	Uncertainty and Selective
7	5030	67	0.66	Random
8	7244	5097	0.90	Random
9	5194	12341	0.92	Get all
10	0	17535	0.91	

in the first two queries. Random sampling was used in the rest of large queries. Details of the queries are listed in Table 3. Our final global score = 0.62 which ranked 2nd in the competition. Figure 1 shows the learning curve of our submissions in verification phase.

Table 3: Queries submitted in verification phase for dataset A.

Submission Sequence	Number of Samples in Query	Number of Queried Samples	AUC	Sampling Strategy
1	232	1	0.67	Uncertainty and Selective
2	1959	233	0.82	Uncertainty and Selective
3	4286	2192	0.92	Random
4	11057	6478	0.94	Get all
5	0	17535	0.93	

2.2. Dataset C: Chemo-informatics

We took the same approach on dataset C as we did for dataset A, i.e., using the stochastic semi-supervised learning process for the first submission. However, we adjusted our query strategy: we purchased all the labels in one query. This is actually a passive learning process. Our purpose is to test how good this passive learning will be comparing with the active learning carried out by others. There are 851 variables in the

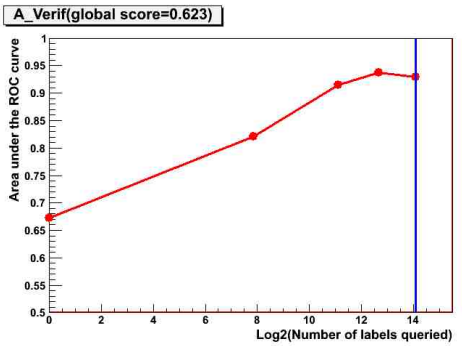


Figure 1: Learning curve of dataset A in verification phase

dataset. We did a very simple variable selection by filtering out the variables without noticeable variance (one value occupies more than 99% population) before we started our semi-superivsed clustering process. We then standardized all variables with mean of 0 and standard deviation of 1. The k-means clustering process is exactly same as that of in dataset A. The number of iterations n is set to 100. We used boosting decision tree algorithm to build the final model after the first (also the last) query. The final global score = 0.33 which ranked No.4 in the competition. Our passive learning approach did not achieve the best result.

2.3. Dataset D: Text Processing

Our approach encountered a large obstacle in dataset D. This dataset has some characteristics that are very different from other datasets.

1. It has 12000 features. This number is even bigger than the number of total training samples. Therefore, good variable selection has increased importance in order to have a better classification model.

2. It has 25.2% positive labels, while all other datasets have less than 10% positive labels.

3. The positive seed given actually has very low score which means it is on the other side of decision boundary after all labels are known.

Our method did not work well because of factors 2 and 3. However, these facts were not known during the contest. We used the same approach as in datasets A and C. One assumption in our approach is that the dataset is highly imbalanced. The positive label population is much smaller than the negative label population. This guarantees that in our stochastic process, the probability of getting negative seeds from unlabeled data pool is much higher than that of getting positive seeds. Our prediction score is based on the cluster membership counts of each data point. The cluster label is based on the given seed. The fact that the first positive seed is closer to majority of negative

examples than majority of positive seeds made our first prediction score worse than random guess (the AUC of our first submission was 0.46). We did 2 small purchases of labels using uncertainty sampling guided by prediction score and then followed a large purchase to get all training labels. Our final global score is 0.33 which ranked No. 18 out of 19 participated teams in this dataset. Details can be seen in Figure 2.

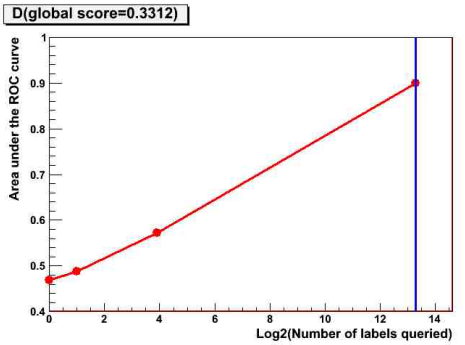


Figure 2: Learning curve of dataset D submission. Final global score = 0.33.

2.4. Dataset B: Marketing

It is not hard to figure out during the contest that dataset B came from marketing domain. We see that the corresponding development dataset has an extremely imbalanced class distribution (positive label is only 1.78%). As stated in algorithm 1, we use logistic regression as classifier instead of k-means clustering at the beginning of the label purchase.

There were a lot of missing values in the dataset. We first did a preprocessing by simply filling the missing value with 0. We then standardized all variables (except one categorical variable, column 14) with a mean of 0 and a standard deviation of 1. There were 250 variables in the dataset. Most of them were populated by several distinct values. We did a simple unsupervised variable selection based on Shannon entropy, which is defined as

$$Entropy = -\sum_i p_i \log p_i \tag{1}$$

where p_i is the distribution of i-th bin (or i-th entry for categorical variable). We kept all the variables with entropy value greater than 0.03. This left us 43 out of 250 variables. The rationale behind this approach is to get rid of all the variables with very skewed distribution (for example, large amount of population filled by one value). All constant variables or close to constant were removed by this method.

We took the initial positive seed given by organizer, then randomly sampled 20 unlabeled data points as “negative" labels. The reason we chose 20 is because we assume the percentage of positive labels is smaller than 5% in the dataset. Actually positive labels make up 9.16% after all labels are purchased, even though the corresponding

development dataset has only 1.78% positive labels. The organizer changed the class distribution purposely in order to test the robustness of every competitor's approach. We built an over-fitted logistic regression model on these 21 samples (we call it over-fit because the number of features is greater than the number of examples). This model was used to score all data points. We repeated our sampling process for "negative" labels n iterations like we did for k-means. At the end, each data point got n scores. The averaged score over n iterations ($n = 100$) was used to label the unlabeled data. We labeled all data points with score in lowest 10% as negative and in highest 10% as positive. Final logistic regression model was built on dataset with the "derived" labels. The score was used as final prediction in the first submission. We obtained two more labels after the first query. We repeated the process that we used in the first prediction. We purchased all the labels after the 2nd query. We then used gradient boosting decision tree for our final prediction. In order to make sure all good variables were included in the final model, we put back all 250 variables in the final training dataset. We did 3 rounds of bagging on boosting decision tree models. The results is shown in Figure 3. We did get the highest AUC for the final prediction score among all submissions. Our global score is 0.3754 which ranked No. 2 in the competition. The winner had a global score of 0.3757 with a passive learning approach. The winner's initial prediction had a better AUC and was their advantage.

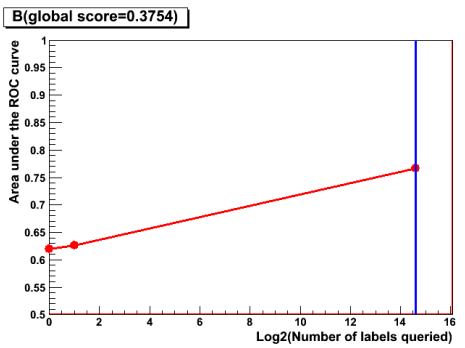


Figure 3: Learning curve of data B submission.

2.5. Dataset E: Embryology

There are 154 continuous variables in dataset E. We standardized each variable to a mean of 0 and a standard deviation of 1. We did not perform any unsupervised variable selection. We used exactly same approach as we did in dataset B. We tried a different query strategy for our first label purchase. In stead of uncertainty query we did most of the time, we chose a certainty query, i.e., we queried 2 data points in the highest 1% of the scores. The labels we obtained for those 2 data points is one positive and one negative, respectively. We repeated our stochastic semi-supervised learning process by keeping these newly purchased labels in each random sampling/learning iteration. The

learning curve of dataset E is listed in Figure 4. We can see that the AUC for second submission actually becomes worse than the first submission. The newly queried labels over corrected the first model mainly due to the negative label we got which had a high prediction score of 0.75 in first model. However, it had a score of 0.03 in the second model. We did 2 more small label purchases using uncertainty sampling. The whole process is shown in Table 4. We then queried all labels and built the last model using boosting decision tree. The final global score is 0.53 which ranked No. 3 in the competition.

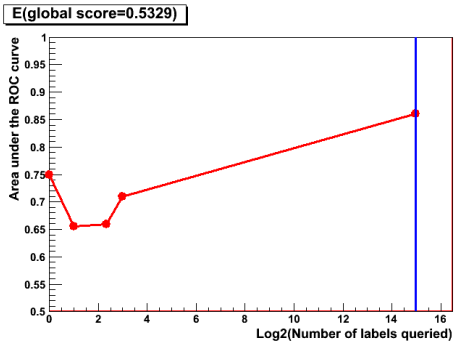


Figure 4: Learning curve of data E submission

Table 4: Queries submitted for dataset E. The final global score = 0.53.

Submission Sequence	Number of Samples in Query	Number of Samples Queried	AUC	Sampling Strategy
1	2	1	0.75	Certainty
2	3	3	0.66	Uncertainty and Selective
3	3	6	0.67	Uncertainty and Selective
4	32243	9	0.72	Get all
5	0	32252	0.86	

2.6. Dataset F: Ecology

Dataset F has only 12 variables. We did not perform any variable reduction/selection. We standardized all numerical variable to a mean of 0 and a standard deviation of 1. Once again, we used same approach as in dataset E except we modified the sampling strategy. For this dataset, we skipped one small label queries and did an additional large label queries. We realized that most of the small number of label queries damaged the global score. After the 3rd submission, we obtained 552 labels. We switched from

semi-supervised learning to supervised learning by using boosting decision tree algorithm. Table 5 lists each query steps. The learning curve is shown in Figure 5. Our final global score is 0.77, placed No. 4 in the competition.

Table 5: Queries submitted for dataset F. The final global score = 0.77.

Submission Sequence	Number of Samples in Query	Number of Samples Queried	AUC	Sampling Strategy
1	2	1	0.76	Uncertainty and Selective
2	7	3	0.73	Uncertainty and Selective
3	542	10	0.77	Uncertainty and Selective
4	5175	552	0.95	Random
5	61901	5727	0.98	Get all
6	0	67628	0.99	

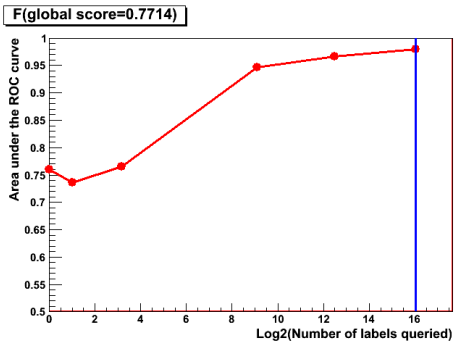


Figure 5: Learning curve of data F submission

3. Results and Discussion

Our final results and the comparison with the winners of each dataset are listed in Table 6. It is interesting to notice that six datasets have six different winners. None of the teams won more than one dataset, which means no team's method was completely general. Our overall ranking placed 3rd among 22 participated teams. Our approach worked relatively well on 5 out of 6 datasets. It is dataset D that degraded our overall performance. This degradation is largely because dataset D has the highest positive label percentage (25.2%, while all others are less than 10%).

We summarize some of the challenges we have seen during the competition in the following.

Table 6: Results of each dataset and the comparison with the winners.

Data set	Positive label %	AUC	ALC	Num of queries made	Rank	Winner's AUC	Winner's ALC
A	7.23	0.9250	0.6230	4	2	0.8622	0.6289
B	9.16	0.7670	0.3754	2	2	0.7327	0.3757
C	8.15	0.8137	0.3341	1	4	0.7994	0.4273
D	25.19	0.8897	0.3312	3	18	0.9641	0.7449
E	9.03	0.8650	0.5329	4	3	0.8939	0.6266
F	7.68	0.9883	0.7714	5	4	0.9990	0.8018

1. How to consistently get better performance with only a few (less than 10) known labels across different datasets. In this contest, this was very critical because of the way how the global score was calculated and the $\log 2$ scaling on number of queried samples. All top players of each dataset had good performance at the first submission. It is very hard to find a robust method working for all dataset. That is the primary reason why nobody won more than one dataset. Our stochastic approach attempts to improve the robustness of our method. It works relatively well overall.

2. How to consistently improve model performance with the increase of known labels in a given dataset. This is particularly hard when the number of known labels is small. We saw cases both in our own submission and in others that the model performance was getting worse when a few more labels were added into the existing model. This was because the initial model was heavily impacted by the initial labels. A few newly added labels may not be representative to the whole dataset, especially when the uncertainty sampling query is used.

3. It is not conclusive that active learning approach will always beat passive learning in these real world data sets based on the current global score measurement, particularly when the data dimension is high and the label distribution is imbalanced. Winners of dataset B, D and E all used passive learning. The $\log 2$ scaling in the global score calculation might give too much weight to models with only a few labels. The passive learning approach avoids the dips of learning curve and gains some "artificial" advantages when the dataset is hard to learn.

4. Conclusion

In summary, we have described the method we used in all six datasets in the active learning challenge. We propose a stochastic semi-supervised learning approach to tackle the classification problem under the condition that the number of labeled data points is extremely small and the two classes are highly imbalanced. We prefer to

use k-means clustering for datasets with a very large number of features (greater than 800 in this contest). We suggest using logistic regression for datasets with highly imbalanced class distribution. In the contest, we switched to supervised learning using gradient boosting decision tree algorithm when the number of known labels is greater than 200, which corresponds to the middle range of x value in the area under the learning curve. Both uncertainty sampling and density-based selective sampling were used for label queries. Our approach performed pretty well in 5 out of 6 datasets. We ranked 3rd overall in the contest.

Acknowledgments

We thank the organizers of the active learning challenge for setting up an excellent platform and providing real world datasets for this competition. We thank David Zaleta for proof reading the manuscript. J. Xie would like to thank Dr. Isabelle Guyon for providing him the travel support.

References

Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *JMLR*, pages 255–291, 2004.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

J. F. Bowring, J. M. Rehg, and M. J. Harrold. Active learning for automatic classification of software behavior. *In Proceedings of the International Symposium on Software Testing and Analysis*, pages 195–205, 2004.

R. Dara, S. Kremer, and D. Stacey. Clustering unlabeled data with soms improves clssification of labeled real-world data. In *Proceedings of the World Congress on Computation Intelligence(WCCI)*, May 2002.

A. Demiriz, K. Bennett, and M. Embrechts. Semi-supervised clustering using genetic algorithms. In *Proceedings of Artificial Neural Networks in Engineering*, November 1999.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 1999.

Isabelle Guyon, Gavin Cawley, Gideon Dror, and Vincent Lemaire. Results of the active learning challenge. *JMLR W& CP*, 15, 2011.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. *SIGIR*, pages 3–12, 1994.

T. Luo, K. Kramer, and D. B. Goldgof. Active learning to recognize multiple types of plankton. *JMLR*, 6:589–613, April 2005.

Charles Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshop on Applications of Computer Vision*, January 2005.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. *ICML*, pages 441–448, 2001.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2:45–66, November 2001.

Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

JMLR: Workshop and Conference Proceedings 16:99–112, 2011 Active Learning Challenge

An Active Learning Algorithm Based on Parzen Window Classification

Liang Lan LANLIANG@TEMPLE.EDU
Haidong Shi HAIDONGSHI@TEMPLE.EDU
Zhuang Wang ZHUANG@TEMPLE.EDU
Slobodan Vucetic VUCETIC@IST.TEMPLE.EDU
Department of Computer and Information Sciences
Temple University, Philadelphia, PA 19122 USA

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

This paper describes active learning algorithm used in AISTATS 2010 Active Learning Challenge as well as several of its extensions evaluated in the post-competition experiments. The algorithm consists of a pair of Regularized Parzen Window Classifiers, one trained on full set of features and another on features filtered using Pearson correlation. Predictions of the two classifiers are averaged to obtain the ensemble classifier. Parzen Window classifier was chosen because is an easy to implement lazy algorithm and has a single parameter, the kernel window size, that is determined by the cross-validation. The labeling schedule started by selecting random 20 examples and then continued by doubling the number of labeled examples in each round of active learning. A combination of random sampling and uncertainty sampling was used for querying. For the random sampling, examples were first clustered using either all features or the filtered features (whichever resulted in higher cross-validated accuracy) and then the same number of random examples was selected from each cluster. Our algorithm ranked as the 5th overall, and was consistently ranked in the upper half of the competing algorithms. The challenge results show that Parzen Window classifiers are less accurate than several competing learning algorithms used in the competition, but also indicate the success of the simple querying strategy that was employed. In the post-competition, we were able to improve the accuracy by using an ensemble of 5 Parzen Window classifiers, each trained on features selected by different filters. We also explored how more involved querying during the initial stages of active learning and the pre-clustering querying strategy would influence the performance of the proposed algorithm.

Keywords: Parzen Window, ensemble classifiers, clustering, feature filter, active learning

1. Introduction

Active learning addresses the problem in which large amounts of unlabeled data are available at low cost but where acquiring labels is expensive. The objective of active learning is to achieve high accuracy by using the least labeling effort. The AISTATS

2010 Active Learning Challenge (Guyon et al., 2010) considered the pool-based active learning scenario, where labels can be queried from a large pool of unlabeled data. In the challenge, the participants were given 6 binary classification tasks covering a wide variety of domains. All datasets were high-dimensional and with significant class imbalance.

In each task the participants were given a single seed example from a minority class and a large amount of unlabeled data to be queried. The virtual cash was used to acquire the class labels for selected examples from the server. Before acquiring labels, participants were asked to provide predictions for a separate test data set, which were used to measure performance of each active learning algorithm. At the completion of the challenge, the organizer released learning curves describing the evolution of the Area Under the ROC Curve (AUC) accuracy as a function of the number of labels, and also the Area under the Learning Curve (ALC) score summarizing the overall performance. The ALC score was designed to reward high accuracy when labeled datasets are small.

We used the Regularized Parzen Window Classifier (RPWC) (Chapelle, 2005) as the baseline classifier because of its ease of implementation and ability to solve highly nonlinear classification problems. We developed an ensemble of RPWC to enhance the overall performance. Due to the time constraints, we used only two component classifiers that differed in the features being used. Feature selection is a critical choice for Parzen Window classifiers because it influences the measure of distance between examples. For the challenge, we considered several feature selection filters.

Active learning has been a very active research topic in machine learning for many years, and we had a large choice of querying strategies. Uncertainty sampling (Lewis and Gale, 1994) is one of the most popular choices due to its simplicity and established effectiveness on many domains. In uncertainty sampling, a learner queries unlabeled examples that are the most uncertain, on which the existing models disagree the most (Seung et al., 1992) or that are expected to produce the greatest increase in accuracy (Cohn et al., 1996). By observing that pure uncertainty sampling can be too aggressive, a variety of approaches have been developed that combine uncertainty sampling with information about data density obtained from unlabeled data; see (Nguyen and Smeulders, 2004) and references within. Since classifiers trained on small labeled datasets could be very unreliable, pure random sampling during the initial stages of active learning can be very competitive to the more sophisticated active learning strategies.

The querying strategy used in our algorithm consisted of selecting a mixture of the most uncertain examples and randomly sampled examples. Instead of pure random sampling, which might over-emphasize dense regions of the feature space, we used random sampling of unlabeled data clusters. Due to the time limitations, we used an aggressive schedule that at each round of active learning doubled the number of labeled examples.

2. Methodology

In this section we describe the problem setup and explain the details of the active learning algorithm we used in the competition and for the post-competition experiments.

2.1. Problem Setup

Let us denote the labeled dataset as $L = \{(x_i, y_i), i = 1...N_l\}$, where x_i is an M-dimensional feature vector for the ith example and y_i is its class label. Initially, L contains a single seed example ($N_l = 1$). We are also given a pool of unlabeled data $U = \{(x_i), i = 1...N_u\}$. The unlabeled dataset is assumed to be an *i.i.d.* sample from some unknown underlying distribution $p(x)$. Each example x_i has a label y_i coming from an unknown distribution $p(y|x)$.

At each stage of the active-learning process, it is possible to select a subset Q of examples from U, label them, and add them to L. Then, using the available information in L and U a classifier is trained and a decision to label another batch of unlabeled examples is made. At each stage, accuracy of the classifier is tested. Accuracies from all stages are used to evaluate the performance of the proposed active learning algorithm. The success in active learning depends on a number of design decisions, including data preprocessing, choice of classification algorithm, and querying strategy.

2.2. Proposed Active Learning Approach

The proposed active learning algorithm is outlined in Algorithm 1. At each round of the algorithm, we first trained F different base classifiers from the available labeled examples. Each classifier was built using the Regularized Parzen Window Classifier, described in §2.4. The classifiers differed by the features that were used. Several different feature selection filters explained in §2.6 were used. The final classifier was obtained by averaging predictions of the individual classifiers. To query new unlabeled examples, we used a combination of uncertainty sampling and random sampling, as explained in §2.8. At each round we doubled the number of queried examples. This querying schedule was used to quickly cover a range of the labeled set sizes, consistent with the way the organizers evaluated the performance of competing active learning algorithms.

2.3. Data Preprocessing

Data preprocessing is an important step for successful application of machine learning algorithms to real word data. We used the *load_data* function provided by the organizers to load the data sets, which replaces the missing values with zeros and replaces Inf value with a large number. All non-binary features were normalized to have mean zero and standard deviation one.

2.4. Regularized Parzen Window Classifier

Choice of classification algorithm is very important in active learning, particularly during the initial stages when only a few labeled examples are available. For the active learning competition, our team selected the Regularized Parzen Window Classifier (RPWC) (Chapelle, 2005) as the base classifier. There are several justifications for our choice. RPWC is easy to implement and very powerful supervised learning algorithm

Algorithm 1: Outline of the Active Learning Algorithm

Input: labeled set L, unlabeled set U; querying schedule $q_t = 20 \times 2^t$
Initialization: $Q \leftarrow$ *randomly selected 20 unlabeled examples from U;*

for *t = 1 to 10* **do**
  $U \leftarrow U - Q$;
  $L \leftarrow L + \text{labeled}(Q)$;
  for *j = 1 to f* **do**
    $F_j = \text{feature\_filter}_j(L)$; /*feature selection filter*/
    $C_j = \text{train\_classifer}(L, F_j)$; /*train classifier C_j from L using features F_j, determine model parameters by CV on L */
    $A_j = \text{accuracy}(C_j, L)$; /*estimate accuracy of classifier C_j by CV on L*/
  end
  $C_{avg} \leftarrow \text{average}(C_j)$; /*build ensemble classifier C_{avg} by averaging*/
  $j^* = \arg\max A_j$; /* find the index of the most accurate classifier*/
  Apply k-means clustering on $L + U$ using features from F_{j^*};
  $Q \leftarrow 2q_t/3$ of the most uncertain examples in U by the ensemble C_{avg};
  $Q \leftarrow Q + q_t/3$ of random examples chosen from randomly selected clusters of U;
end

able to represent highly nonlinear concepts. It belongs to the class of lazy algorithms as it does not require training. RPWC gives an estimate of the posterior class probability of each example, which can be used to calculate the classification uncertainty for each unlabeled example.

RPWC can be represented as

$$p(y = 1|x) = \frac{\sum_{y_i=1} K(x, x_i) + \varepsilon}{\sum K(x, x_i) + 2\varepsilon}$$

where ε is a regularization parameter, which should be a very small positive number and is relevant when calculating the classification uncertainty for examples which are underrepresented by the labeled examples. We set $\varepsilon = 10^{-5}$ in all our experiments. K is the Gaussian Kernel of the form

$$K(x, x') = \exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right)$$

where the σ represents the kernel size.

2.5. Kernel Size Selection

The kernel size σ is a free parameter which has a strong influence on the resulting estimate. When σ is small RPWC resembles nearest neighbor classification, while it resembles trivial classifier that always predicts the majority class when σ is large. Therefore, it has large influence on the representational power of the RWPC and should

be selected with care. In this competition, we used leave-one-out cross-validation (CV) to select the optimal value for σ. We explored the following set of values for $2\sigma^2$: $M/9, M/3, M, 3M$, where M is the number of features.

2.6. Feature Selection

Feature selection becomes a critical question when learning from high-dimensional data. This is especially the case during the initial stages of active learning when the number of labeled examples is very small. Feature selection filters are an appropriate choice for active learning because they are easy to implement and do not require training of a classifier. Typically, feature filters select features that have different statistics in positive and negative examples. In this competition, we considered statistical feature selection filters (Radivojac et al., 2004) based on the Pearson correlation and the Kruskal-Wallis test. We used only the Pearson correlation in the competition, while we also used the Kruskal-Wallis in our post-competition experiments.

2.6.1. All Features

As the baseline feature selection method, we simply used all the available features. This choice can be reasonable during the initial stages of active learning when the number of labeled examples is too small even for the simple statistical filters to make an informed decision.

2.6.2. Pearson Correlation

The Pearson correlation measures correlation between a feature and the class label and is calculated as

$$r_j = \frac{\sum_{i=1}^{N}(x_i^j - \bar{x}^j)(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i^j - \bar{x}^j)^2 (y_i - \bar{y})^2}}$$

where $\bar{x}^j$ is the mean of the jth feature, $\bar{y}$ is the mean of the class label (negative examples were coded as 0 and positive as 1), and N the number of labeled examples. Value r_j measures linear relationship between jth feature and target variable and its range is between 1 or -1. Values of r close to 0 indicate the lack of correlation.

We used the Student's t-distribution to compute the p-value for the Pearson correlation that measures how likely it is that the observed correlation is obtained by the independent variables. We selected only features with p-values below 0.05. Since the number of selected features using the 0.05 threshold would largely depend on the size of the labeled dataset, as an alternative, we selected only the top M^* (e.g. $M^* = 10$) features with the lowest p-values.

2.6.3. Kruskal-Wallis Test

The t-statistics test for Pearson correlation assumes that the variables are Gaussian, which is clearly violated when the target is a binary variable. That is why the nonparametric Kruskal-Wallis (KW) test was also considered in our algorithms. For jth feature,

KW sorts its values in ascending order and calculates average rank of all positive examples (denoted as r_j^+) and all negative examples (denoted as r_j^-). Then, the KW statistics of jth feature is calculated as

$$kw_j = \frac{12}{N(N+1)}\left(N^+\left(r_j^+ - \frac{N+1}{2}\right)^2 + N^-\left(r_j^- - \frac{N+1}{2}\right)^2\right)$$

where N^+, N^- are the numbers of positive and negative labeled examples, respectively. The KW statistics becomes large when the average ranks deviate from the expected rank $(N+1)/2$. The p-value of the KW statistic is calculated easily because kw_j follows the standard χ^2 distribution.

Similarly to the feature filter based on the Pearson correlation, given the p-values, features can be selected in two ways. In first, all features with p-values below 0.05 are selected. In second, M^* (e.g. $M^* = 10$) features with the smallest p-values are selected.

2.7. Ensemble of RPWC

Ensemble methods construct a set of classifiers and classify an example by combining their individual predictions. It is often found that ensemble methods improve the performance of the individual classifiers. (Dietterich, 2000) explains why ensemble of classifiers can often perform better than any single classifier from statistical, computational, and representational viewpoints. The following describes how we constructed an ensemble of RPWCs.

2.7.1. Base Classifiers

For the active learning challenge, we considered five base classifiers: (1) RPWC using all available (normalized) features; (2) RPWC on *filter_data1* (p-value of the Pearson correlations < 0.05); (3) RPWC on *filter_data2* (10 features with the smallest p-value of the Pearson correlations); (4) RPWC on *filter_data3* (p-value of the Kruskal-Wallis test < 0.05); (5) RPWC on *filter_data4* (10 features with the smallest p-value of the Kruskal-Wallis test).

2.7.2. Averaging Ensemble

In general, an ensemble classifier can be constructed by weighted averaging of base classifiers. For example, cross-validation can be used on labeled data to determine what choice of weights results in the highest accuracy. Since in an active learning scenario, the size of labeled data is typically small, we felt that weight optimization might lead to overfitting. Therefore, we decided to construct the ensemble predictor by simple averaging of the base RPWC as

$$p(y = 1|x) = \frac{1}{f}\sum_{i=1}^{f} p(y = 1|x, C_i).$$

2.8. Active Learning Strategy - Exploration and Exploitation

Our active learning strategy outlined in Algorithm 1 combined exploration and exploitation. We used a variant of random sampling for exploration and uncertainty sampling for exploitation. The queried examples at each stage were mixture of randomly sampled and uncertainty sampled examples from the unlabeled dataset.

Uncertainty Sampling Uncertainty sampling is probably the most popular active learning strategy. It measures prediction uncertainty of the current classifier and selects the most uncertain examples for labeling. There are two common ways to estimate uncertainty. If an ensemble of classifiers is available, uncertainty is defined as the entropy of their classifications. Using this definition, selected examples are either near the decision boundary or within underexplored regions of feature space. If a classifier can give posterior class probability, $p(y = 1|x_i)$, the alternative approach is to select examples whose probability is the closest to 0.5. We used this approach in our experiments since RPWC provides posterior class probabilities. It is worth to note that, thanks to the use of regularization parameter ε, examples with $p(y = 1|x_i)$ close to 0.5 in RPWC are both those near the decision boundary and those that are within the underexplored regions of the feature space. Both types of examples are interesting targets for active learning.

Clustering-Based Random Sampling Relying exclusively on uncertainty-based selection may be too risky, especially when the labeled dataset is small and the resulting classifier is weak. In addition, the uncertainty sampling for RPWC decribed in previous section is prone to selecting a disproportionate number of outliers. Therefore, in our approach we also relied on exploration to make sure examples from all regions of the feature space are represented in labeled data.

A straightforward exploration strategy is random sampling. However, this strategy can be wasteful if data exhibits clustering structure. In this case, dense regions would be over-explored while the less dense regions would be under-explored. This is why we decided to use k-means clustering to cluster the unlabeled examples as a preprocessing step at each stage of the active learning procedure. Then, our exploration procedure selected the same number of random unlabeled examples from each cluster. In k-means, we used randomly selected k examples as cluster seeds.

The result of clustering greatly depends on the choice of distance metric. To aid in this, we leveraged the existing base RPWCs. For clustering, we used the normalized features of the most accurate base classifier.

Combination of Uncertainty and Clustering Sampling As the initial selection in all 6 data sets, we randomly sampled 20 examples from the unlabeled data. Only after 20 examples were labeled we trained the first ensemble of RPWCs and continued with the procedure described in Algorithm 1. The justification is that we did not expect RPWC to be accurate with less than 20 labeled examples. For the same reason, we reasoned that clustering-based random sampling would not be superior to pure random sampling this early in the procedure.

In the following rounds of active learning, we sampled $2/3$ of the examples using uncertainty sampling and $1/3$ using the clustering-based random sampling.

Alternative Sampling Strategy Our sampling strategy that combines uncertainty-based and random sampling is similar to the pre-clustering idea from (Nguyen and Smeulders, 2004). They proposed to measure interestingness of unlabeled examples by weighting uncertainty with density. Examples with the highest scores are those that are both highly uncertain and come from dense data clusters. In our post-competition experiments we implemented the pre-clustering algorithm with two modifications. First, we used RPWC instead of the logistic regression to compute the conditional density. Second, instead of selecting unlabeled examples with the highest scores, we randomly selected from a pool of the highest ranked examples to avoid labeling of duplicate examples.

3. Experiments and Results

In this section we will describe the results of the experiments performed during the competition and for the post-competition analysis.

3.1. Data Description

The organizers provided datasets from 6 application domains. The details of each dataset are summarized in Table 1. All datasets are for binary classification and each is characterized by large class imbalance, with ratio between positive and negative examples near 1:10. The number of features in different datasets ranges from a dozen to several thousand.

Table 1: Dataset Description

Data	Feature	# Feature	Sparsity (%)	MissValue (%)	# Train	Maj Class (%)
A	mixed	92	79.02	0	17535	92.77
B	mixed	250	46.89	25.76	25000	90.84
C	mixed	851	8.6	0	25720	91.85
D	binary	12000	99.67	0	10000	74.81
E	cont.	154	0.04	0.0004	32252	90.97
F	mixed	12	1.02	0	67628	92.32

3.2. Experimental Setup

We applied the approach outlined in Algorithm 1. For the competition, we used an ensemble of only two RPWCs: one used all features and the other used features whose Pearson correlation p-value was below 0.05. This choice was made to speed-up our submission process. Before ordering labels for the first 20 random examples, we provided predictions that were proportional to the distance from the seed example. For k-means clustering we used $k = 10$ in all rounds of active learning experiments for all 6 datasets.

For all the experiments, we used the custom-made Matlab code. We used Matlab because it allowed us to implement the proposed Algorithm 1 in only a few days and complete all tasks form the competition within a week. During code development, we used several functions from the Statistics Matlab toolbox, such as the k-means clustering, the Kruskal-Wallis test, and the t-test.

3.3. Performance Evaluation

The classifier performance at each round of the active learning procedure was calculated by the challenge organizers as the Area Under the ROC Curve (AUC) on test data whose labels were withheld from the challenge participants. Performance of an active learning algorithm was calculated using the Area under the Learning Curve (ALC) calculated using the trapezoid method on the $log2$ scale. The log-scale enforced that larger emphasis is given to accuracy during the initial stages of active learning. The final score for each dataset was calculated as

$$global\ score = (ALC - Arand)/(Amax - Arand)$$

where *Amax* is ALC of the *ideal learning curve*, which is obtained when the perfect predictions are made for the whole range of the labeled dataset sizes, (AUC =1). *Arand* represents the ALC of the *'lazy' learning curve*, which is obtained by always making random predictions (where the expected value of AUC is always 0.5). Therefore, *global score* close to 0 indicates that the resulting active learning algorithm is not better than the random predictor, while value close to 1 indicates that very accurate classifier could be made using very few labeled examples.

3.4. Competition Results

We summarize the performance of our active learning algorithm in Tables 2 and 3 and Figure 1. Table 2 lists the *global score* on all 6 datasets. It compares the results from the best performer, the highest and second highest overall ranked teams, INTEL and ROFU, and of our team (TUCIS). The ranking of our algorithm was in the top half among all competing algorithms on all 6 datasets. Thanks to this consistency, our algorithm was ranked as the 5-th best algorithm overall. Our algorithm was better than INTEL on dataset D and better than ROFU on datasets C and F.

Table 3 summarizes the AUC accuracy of the final classifier trained using all purchased labels. As can be seen, RPWC was less accurate than the best reported classifier, the random forest used by INTEL, on all 6 datasets. The difference was the smallest on dataset F, while it was around 10

Figure 1 shows the learning curves that represent AUC classification accuracy as a function of the number of purchased labels. In all cases, a trend of increasing AUC with sample size can be observed. However, there are some significant differences in behavior of our algorithm on different datasets. On dataset *A*, the behavior is in accord to an intuitive notion that AUC should increase sharply until reaching the convergence region. On datasets *B*, *D*, and *E*, we can observe an almost linear increase in AUC with

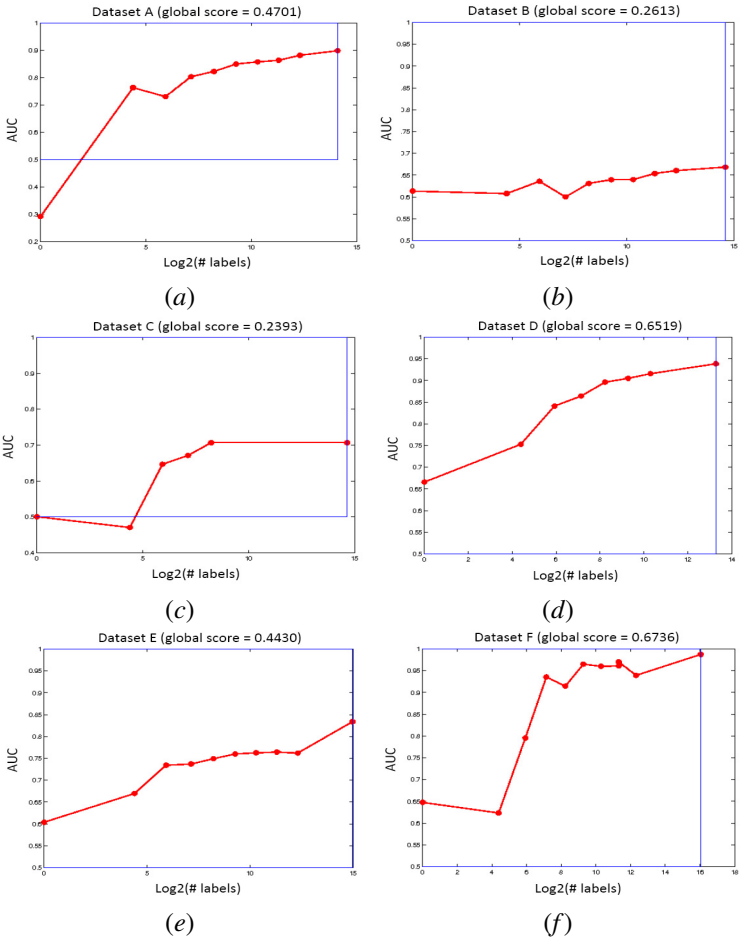


Figure 1: AUC versus the sample size on 6 datasets: the official results

a logarithm of the sample size. It is interesting to observe that the linear AUC growth on the log-scale was assumed to be the lower bound by the competition organizers (Guyon et al., 2010). Our results on datasets B, C, and E indicate that this assumption was overly optimistic. As a consequence, building only a single predictor on a fully labeled dataset (the strategy cleverly used by ROFU team) would yield the same *global score* as our active learning algorithm on datasets B, D, and E. Performance on datasets C and F was characterized by a drop in AUC accuracy after labeling the first 20 examples that was significantly improved in the following labeling stages. The observed behavior could possibly be attributed to a significant class imbalance that was 91.85% for dataset C and 92.32% for dataset F. This result clearly illustrates that classifiers trained on small samples could be very unreliable and should be used with care in uncertainty sampling.

Table 2: Summary of the Competition Results: the (*global score*)

Dataset	Best	INTEL (1st)	ROFU (2nd)	TUCIS (5th)	TUCIS RANK
A	0.629	0.527	0.553	0.470	7
B	0.376	0.317	0.375	0.261	8
C	0.427	0.381	0.199	0.239	7
D	0.861	0.640	0.662	0.652	5
E	0.627	0.473	0.584	0.443	7
F	0.802	0.802	0.669	0.674	8

Table 3: Summary of the Competition Results: the Final AUC scores

Dataset	Best	INTEL (1st)	ROFU (2nd)	TUCIS (5th)
A	0.962	0.952	0.928	0.899
B	0.767	0.754	0.733	0.668
C	0.833	0.833	0.779	0.707
D	0.973	0.973	0.970	0.939
E	0.925	0.925	0.857	0.834
F	0.999	0.999	0.997	0.987

3.5. Post-Competition Results

Upon completion of the active learning challenge, we performed several additional experiments to better characterize the proposed algorithm and explore some alternatives. In order to finish this task we first queried labels of all training examples from the challenge server. Then, we randomly selected 80% of the labeled examples for training and 20% for testing. To be consistent with the challenge rules, in all experiments we used the same seed as used in the competition. All shown post-competition results are based on repeating the active learning experiment 5 times. In order to quickly finish the experiments, each active learning experiment was terminated after 320 examples were labeled (after 5 rounds of Algorithm 1).

Early Start Our competition algorithm started by labeling 20 randomly sampled and building a classifier from them. Our first question was what would be the *global*

Table 4: Comparison of two querying strategies

dataset	first 20 labels one by one	begin with 20 labels
A	**0.537±0.036**	0.466±0.015
B	0.267±0.022	**0.273±0.021**
C	0.242±0.034	**0.261±0.039**
D	0.576±0.035	**0.601±0.027**
E	**0.445**±0.028	0.433±0.017
F	0.750±0.046	**0.757±0.062**

Table 5: Comparison of different ensemble methods

Data	1 classifier (full)	1 classifier(PR)	1 classifier(KW)	2 Classifiers	5 Classifiers
A	0.463±0.080	0.433±0.040	0.442±0.026	**0.466±0.015**	0.459±0.017
B	0.261±0.005	0.246±0.035	0.283±0.011	0.273±0.021	**0.304±0.023**
C	0.321±0.023	0.292±0.038	0.297±0.059	0.261±0.039	**0.337±0.050**
D	**0.670±0.027**	0.540±0.029	0.545±0.051	0.601±0.027	0.551±0.050
E	0.426±0.007	0.417±0.024	0.412±0.050	0.433±0.017	**0.450±0.007**
F	0.620±0.023	0.785±0.026	**0.792±0.039**	0.757±0.062	0.779±0.026

score if we trained classifiers after every single labeled example, $N_l = 1, 2, ..., 20$. Since *log*2 scale was used in calculating ALC, any success on the first 20 examples would be visible in the final score. Similarly, any failure to make a good learning progress would be reflected negatively. In Table 4, we compared the results of the submitted algorithm (tested on 20% of training examples, as explained above) to the same algorithm that built a series of classifiers until labeled data size reached 20 (after that, we followed exactly the same procedure as in Algorithm 1). Consistent with Algorithm 1, for the first 20 queries we used exclusively random sampling. As can be seen, the *global score* of the alternative approach increased only on dataset *A*, decreased on dataset *D*, and remained similar on the remaining 4 datasets. This result confirmed our original strategy from Algorithm 1. Another result from this set of experiments is worth mentioning. Table 4 also lists the standard deviation of *global score* after repeating each experiment 5 times. It can be seen that the variability is relatively low (around 0.046). The largest difference 0.05 was observed on dataset *F*.

To more clearly illustrate the evolution of AUC at different sample sizes, in Figure 2 we are showing the learning curves on the 6 datasets. It can be seen from Figure 2 that the variability in AUC values slowly decreases with the number of labels in all datsets. On dataset *A* it could be seen that the accuracy rapidly increases after only two additional examples are labeled and that it approaches the maximum achievable accuracy. This clearly explains the increase in ALC reported in the first row of Table 4. However, the similar behavior is not replicated on the remaining 5 datasets. In fact, on 4 of the datasets (*B*, *C*, *D*, *F*) the accuracy grows slower than the postulated log-scale linear growth. On three of them, we could even observe the initial drop in accuracy that explains significant drop in ALC reported in Table 3.

These results clearly demonstrate the challenges in design of a successful active learning strategy when the number of labeled examples is small. This is particularly the case when a dataset is highly dimensional, with a potentially large number of irrelevant, noisy, or weak features. In this case, there simply might be too little data to make informed decisions about feature selection, classifier training, and sampling strategy. This view is supported by the competition results where each dataset seems to have favored different active learning approach.

Larger Ensemble Due to the need to finalize all experiments before the challenge deadline, our ensemble consisted only of two RPWCs. Here, we explored what would

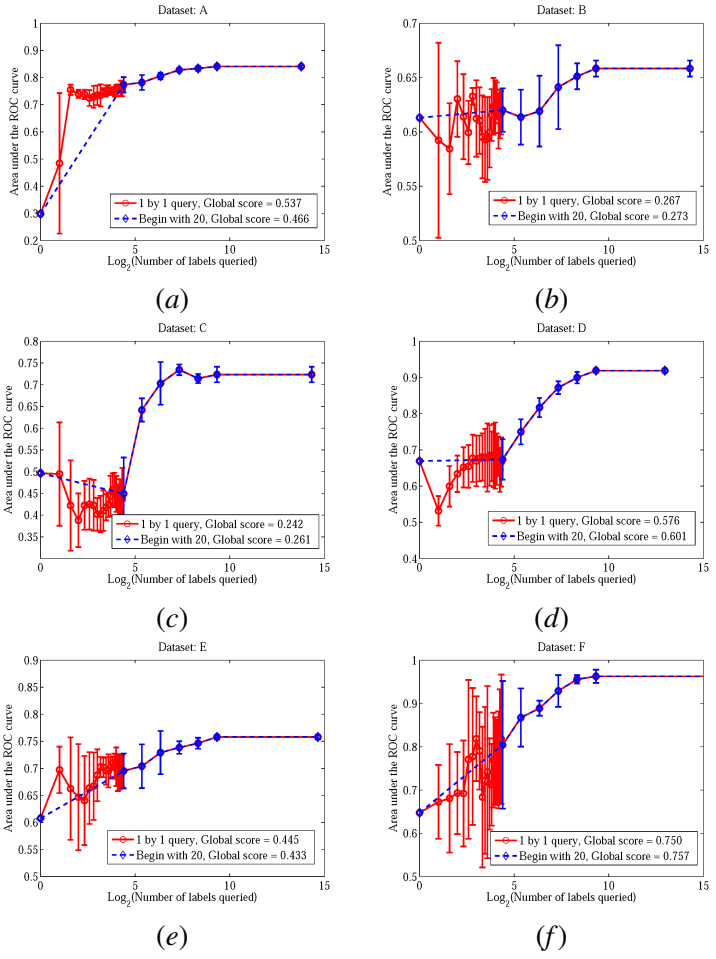


Figure 2: Competition and Post-Competition AUC curves

Table 6: Comparison of two querying strategies

dataset	1/3 RAND + 2/3 UNCTN	Precluster
A	**0.466±0.015**	0.447±0.012
B	**0.273±0.021**	0.170±0.057
C	0.261±0.039	**0.293±0.049**
D	**0.601±0.027**	0.507±0.036
E	**0.433±0.017**	0.378±0.058
F	**0.757±0.062**	0.709±0.062

be the impact of accuracy if we used 5 RPWC, each using a different feature selection filter, as explained in §2.7.1. Table 5 summarizes the results. It can be seen that the 5-classifier ensemble is more accurate than our competition algorithm on 4 challenge datasets and is less accurate only on dataset *D*. This behavior is consistent with the large body of work on ensemble methods Dietterich (2000). The obtained result indicates that using a larger variety of feature selection methods would result in improved performance of our approach. Table 5 also lists results of several individual classifiers. Their performance was overall worse than the performance of both ensembles. It is interesting to observe that RPWC that used all feature was the most successful individual classifier on 4 of the 6 datasets, and that its performance was particularly impressive on dataset *D* and particularly poor on dataset *F*.

Active Learning by Preclustering Finally, we explored the performance of preclustering algorithm by Nguyen and Smeulders (2004) outlined in §2.8. The results in Table 6 shows that preclustering was inferior than the 2/3 uncertain + 1/3 random approach used in our challenge algorithm. We explain this result by a strong reliance of pre-clustering on the classification model that might be unreliable when trained with small number of labeled examples. Nevertheless, this is a slightly surprising result deserving further analysis.

4. Conclusion

The results of the AISTATS 2010 Active Learning Challenge show that our proposed algorithm based on Parzen Window classification and mixture of uncertainty-based and random sampling gives consistent results, that are somewhat lower than the winning algorithms. Our analysis reveals that, although it was less accurate than the winning decision forests, Parzen Window classification was a reasonable choice for solving highly dimensional classification problems. In addition, since several of the challenge datasets could be accurately solved using linear classifiers Guyon et al. (2010), the representative power of Parzen Window classifiers becomes a disadvantage. Our results also show that accuracy of Parzen Window classifiers can be boosted by using their ensembles. On the other hand, the ease of implementation, coupled with respectable accuracy achieved on the challenge tasks, indicates that Parzen Window classifiers should be given consideration on any active learning problem.

Using the mixture of uncertainty and random sampling proved to be a successful strategy. It shows that good performance on active learning problems requires a right mix of exploration and exploitation strategies. Our results confirm that at initial stages of active learning, when there are few labeled examples, the exploration should be given precedence over exploitation. This can be easily attributed to low quality of knowledge that can be acquired from small labeled datasets. This paper also illustrated the benefits of analyzing properties of unlabeled data (e.g. through clustering, or semi-supervised learning) and importance of feature selection when addressing high-dimensional active learning problems.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation Grant IIS-0546155.

References

O. Chapelle. Active learning for parzen window classifier. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2005.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In *Journal of Artificial Intelligence research*, volume 4, 1996.

T. Dietterich. Ensemble methods in machine learning. In *J. Kittler and F. Roli, editors, the 1st International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*. Springer-Verlag, 2000.

I. Guyon, G. Cawley, G. Dror, and V. Lemaire. Results of the active learning challenge. In *Journal of Machine Learning Research: Workshop and Conference Proceedings*, volume 15, 2011.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, 1994.

H. Nguyen and A. W. Smeulders. Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

P. Radivojac, Z. Obradovic, A. K. Dunker, and S.Vucetic. Characterization of permutation tests for feature selection. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, 2004.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the 5th ACM Workshop on Computational Learning Theory (COLT)*, 1992.

Active Learning for Unbalanced Data in the Challenge with Multiple Models and Biasing

Yukun Chen YUKUN.CHEN@VANDERBILT.EDU
Subramani Mani* SUBRAMANI.MANI@VANDERBILT.EDU

Discovery Systems Lab, Department of Biomedical Informatics
*Department of Electrical Engineering and Computer Science**
Vanderbilt University
Nashville, TN 37232, USA

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

The common uncertain sampling approach searches for the most uncertain samples closest to the decision boundary for a classification task. However, we might fail to find the uncertain samples when we have a poor probabilistic model. In this work, we develop an active learning strategy called "Uncertainty Sampling with Biasing Consensus" (USBC) which predicts the unbalanced data by multi-model committee and ranks the informativeness of samples by uncertainty sampling with higher weight on the minority class. For prediction, we use Random Forests based multiple models that generate the consensus posterior probability for each sample as part of USBC. To further improve the initial performance in active learning, we also use a semi-supervised learning model that self labels predicted negative samples without querying. For more stable initial performance, we use a filter to avoid querying samples with high variance. We also introduce batch size validation to find the optimal initial batch size for querying samples in active learning.

1. Introduction

Active learning, unlike traditional supervised learning, takes into account the cost of labeling. When non-labeled data is abundantly available, active learners are seeking a way to maximize the learning performance while minimizing the cost of labeling samples from the data. In the active learning challenge (Guyon et al., 2011) our task is to develop active learning strategies to maximize the global score, which considers the overall learning performance as a function of the number of queried samples.

Although there are many types of active learning, this challenge uses pool-based active learning for classification scenarios to simulate its application in many domains where the size of available unlabeled data is large. In the challenge, we are given a data matrix where only one instance is provided with positive label. The prediction values and the instance(s) to be queried need to be uploaded to the system, which would return

the true label(s) for the queried instances. Iteratively, we need to make new prediction and ask for new instance(s) to query until an ending criteria is met, for example, all training samples are queried. The prediction performance is evaluated according to area under the learning curve (ALC) which plots the AUC (area under the ROC curve) score for the prediction on all the samples with unknown labels as a function of the number of labels queried. For the ALC score computation, please refer our first reference paper (Guyon et al., 2011).

In order to maximize the ALC score, the global score in the challenge, we need to consider two major components of the active learning method: classification algorithm and querying algorithm. The querying algorithm is designed to find the most informative instance based on the prediction and/or some intermediate information during prediction. When the training set is small, or the missing value rate is high, or the feature dimension is large, an accurate classifier is needed to support querying algorithms to output the informative samples while the samples queried subsequently should further improve the accuracy of prediction. For a discussion of the basic active learning strategies, the reader is referred to the survey by Burr Settles (Settles, 2009). In an earlier paper (Chen and Mani, 2010), we have discussed the preliminary results on the development datasets by basic and newly proposed active learning strategies such as uncertainty sampling based and information density based querying methods, along with a classification voting committee of Naïve Bayes', Support Vector Machines (SVM) and Random Forests. In this paper, we further investigate the uncertainty sampling based method and propose Uncertainty Sampling with Biasing Consensus (USBC) that improved our previous performance. Semi-supervised learning method is also incorporated when the training set is small, and batch size validation is proposed to find the minimal sufficient initial querying size.

The remainder of this paper is organized as follows. Section 2 describes the datasets in both development and final phases of the competition, and the preprocessing steps. Section 3 introduces all active learning strategies including USBC which consists of Random Forests based multiple models and uncertainty sampling based querying method, semi-supervised learning model and batch size validation. Section 4 presents the experiments and results for both development datasets and final datasets. In Section 5, we discuss the results and the strengths and the weaknesses of our methods. In Section 6, we summarize our work and point to some future directions.

2. Datasets in the Challenge

The final data consists of 6 datasets, named A to F, with the same domains as the data in development phase, but the identity of the domains and the fraction of positive labels were purposely omitted. The performance result for each single upload was also unknown before the end of the challenge. Table 1 and Table 2 present the information for development data and final data, respectively.

Dataset ORANGE and dataset B have a large percentage of missing values. For variables with more than 50% of missing values, we considered the missing value as another state. For example, if a binary variable X has only ten percent valid values, we

Table 1: Datasets in the Development Phase of Challenge in 6 domains (In feature type column “Feat Type", “b" represents binary, “c" continuous, and “m" mix of binary and continuous; the number (Num) of train and test samples is equal.)

Data Name	Domain Name	Feat Type	Feat Num	Sparse Rate	Missing Rate	Train/Test Num	Positive labels Rate
HIVA	Chemo-informatics	b	1617	90.88	0	21339	3.52
IBN_SINA	Handwriting recognition	m	92	80.67	0	10361	37.84
NOVA	Text processing	b	16969	99.67	0	9733	28.45
ORANGE	Marketing	m	230	9.57	65.46	25000	1.78
SYLVA	Ecology	m	216	77.88	0	72626	6.15
ZEBRA	Embryology	c	154	0.04	0.004	30744	4.58

Table 2: Datasets in the Final Phase of Challenge in 6 domains (In feature type column “Feat Type", “b" represents binary, “c" continuous, and “m" mix of binary and continuous; the number (Num) of train and test samples is equal.)

Data Name	Domain Name	Feat Type	Feat Num	Sparse Rate	Missing Rate	Train/Test Num	Positive labels Rate	Map Guess
A	?	m	92	79.02	0	17535	?	IBN_SINA
B	?	m	250	46.89	25.76	25000	?	ORANGE
C	?	m	851	8.6	0	25720	?	HIVA
D	?	b	12000	99.67	0	10000	?	NOVA
E	?	c	154	0.04	0.0004	32252	?	ZEBRA
F	?	m	12	1.02	0	67628	?	SYLVA

convert X into a variable with three states by including a missing state. For variables with no more than 50% missing values, we performed Gaussian imputation to impute the small number of the missing values randomly based on the distribution for non-missing values. For the datasets with very high number of features such as NOVA, HIVA, D and C, we performed preprocessing using Principle Component Analysis (PCA) to reduce the number of features to 100.

In the challenge, although the domain information of the final datasets was blocked, we could still broadly match them into 5 categories based on types of variables, size of variables, sample size, and number of missing values. For example, dataset Zebra and dataset E are considered in the same category because both datasets are continuous, which is a unique property convincing us to work on it separately; dataset NOVA and dataset D are in the same category because the number of features on both datasets is higher than the number of their samples, and feature types of both are binary; dataset ORANGE and dataset B are in the same category because both have a very high rate of missing values; dataset SYLVA and dataset F are similar because both have the largest number of samples; datasets IBN_SINA and HIVA and datasets A and C are considered similar because both have no obvious differences, but dataset C has a much higher number of variables and is more similar to HIVA. Our basic strategy is to find the best active learning model for each development dataset, and apply it to the final dataset in the corresponding category. The last column, Map Guess, in Table 2 summarizes our guess of the mapping between development data and final data.

3. Active Learning Strategies

Uncertainty Sampling with Bias Consensus (USBC) was the basic active learning algorithm we used for all datasets. This algorithm considers the posterior probability based on multiple models, the uncertainty value, as well as the bias factor based on the proportion of positive class in the training set at each iteration of active learning. It outputs the biasing uncertainty values for all instances, and the instances with the highest output value are queried for labeling. Two other strategies that we employed, semi-supervised learning support and batch size validation, are also introduced in this section.

The following notational convention is used for the description of our active learning strategies: The data feature matrix is denoted by $\mathbf{x}$; the outcome variable (class) column is denoted by $\mathbf{y}$; the model that generates the posterior probability of label y given data vector (instance) x is denoted by θ; the set of labeled and unlabeled training data is denoted by $\mathcal{L}$ and $\mathcal{U}$ respectively.

3.1. Random Forests Based Multiple Models

Random Forests Classifier (RF) proposed by Breiman (Breiman, 2001) is used as our basic classifier for training on the set of labeled samples $\mathcal{L}$. It is one of the best algorithms for learning predictive models with very low generalization error for most of the datasets in this challenge. But we also need to consider the variance of output generated by RF. Inspired by another basic active learning strategy query-by-committee (Seung

et al., 1992), we used the ensemble of multiple Random Forests models as the prediction committee to reduce the variance by computing the consensus posterior probability (CPP) of the multiple models, computed by the following function for each sample:

$$CPP(x) = \frac{1}{M}\sum_{m=1}^{M} P(y = 1|x; \theta_{(m)}) \tag{1}$$

where $y = 1$ represents the positive label; M is the number of models; $\theta_{(m)}$ is the Random Forest model with index m. On the other hand, we could assess the informativeness of samples by comparing the variance of multi-model outputs. This is another reason why we use multiple Random Forests instead of single one with large number of trees.

In terms of the parameter selections for RF, the method of cross validation would not be sound because we have to start from a small training set in active learning. Based on the recommendation from Breiman, we used a large number of trees (*ntree*) and square root of the number of variables as the default size of randomly selected subset of variables. It is also supported by our experimental findings: the higher number of trees for the RF classifier can generate a higher average prediction score and lower variance in prediction. But we could not use a very high number of trees due to time and memory constraints. Therefore, we picked *ntree* = 4000 and the default size of randomly selected subset of variables.

In terms of the number of RF models M in the multiple models, the prediction based on the consensus probability would be more stable when M is high. Samples with low variance values based on these RF prediction models also benefited our sample selection in the next step in active learning. We could have more accurate variance information if we use a higher number for M, however due to time constraint, we picked $M = 5$ RFs with the same parameters for our experiments. Additional discussion on the effect of variance information for active learning is included in Section 3.3.

3.2. Querying Methods

We focused on the uncertainty sampling based query method that could naturally solve the active learning problems based on its sound theoretical properties (Lewis and Gale, 1994). However, the active learning starts from a very small training set so we could hardly get a good model at the beginning. Looking for the true uncertain samples is sometimes as hard as finding a true decision boundary or model. Secondly, due to the fact that the negative class size is larger than the positive one, we most likely query more samples with negative labels based on the basic uncertainty sampling method. This would intensify the imbalance in the class distribution of training set and the classifier would tend to ignore the minority class at the beginning of the active learning process.

We prefer to query the minority class (positive samples in the challenge) because (1) we assume that a single sample with smaller-fraction label is more informative and (2) we would like to make the training set relatively balanced in the early iteration of active learning. We present our implementation of Least Confidence with Bias (LCB)

even when the prior of fraction of positive labels is not available. LCB is able to query the samples that are close to the decision boundary and are more likely to belong to the minority class. We expected LCB to converge and output the most uncertain samples more quickly than the basic uncertainty sampling with no bias. Moreover, our method does not rely on the constant prior of positive label percentage from the original data. The bias factor only considers the real-time fraction of positive label from the current training samples.

Let us define the query method $Q(\mathbf{x})$, the function we use to assess how informative each instance is in the unlabeled pool U. x^* is selected as the most informative sample according to the basic query function:

$$x^* = \arg\max_{x \in \mathcal{U}} Q(\mathbf{x}) \tag{2}$$

Let pp be the percentage of positive label in the current training set. We define P_{max} for the binary-class problem as follows: P_{max} is the consensus posterior probability that outputs the highest informative value in function $Q^{LCB}(\mathbf{x}, pp)$. We can also say that the instance with CPP equal to P_{max} is most informative. The LCB function follows:

$$Q^{LCB}(\mathbf{x}, pp) = \begin{cases} \frac{CPP(\mathbf{x})}{P_{\max}}; \text{if} CPP(\mathbf{x}) < P_{\max} \\ \frac{1-CPP(\mathbf{x})}{1-P_{\max}}; \text{otherwise} \end{cases} \tag{3}$$

$$P_{max} = mean(0.5, 1 - pp) \tag{4}$$

The output curve of LCB with P_{max} of 0.60 and 0.70 (or $pp = 0.3$ and 0.1) is shown in Figure 1. If the two classes are balanced, $pp = 0.5$ and P_{max} is 0.5, which is equivalent to the original Least Confidence (LC) function. If there are fewer positive samples than negative in the dataset, LCB is more likely to query the uncertain samples with positive label.

During the active learning process, the bias factor pp would be adaptively assigned based on two conditions: the positive fraction and the performance of the model. For condition one, pp is just the positive fraction of current training set. But since we were given the prior knowledge that there are fewer positive samples than negative ones for all challenge datasets, it is not appropriate to bias to the negative class if the current fraction of positive samples is over 0.5. Thus we set $pp = 0.5$ if the positive fraction is over 0.5. For condition two, intuitively if our model is good enough, we would switch to the basic uncertainty sampling method. Although we could not obtain the performance metric in the final phase, we assume that when the size of the training set is sufficiently large, our model is adequate for predicting the most informative samples around the decision boundary. This user-defined threshold to switch between uncertainty sampling with and without bias is precisely assigned in Algorithm 2 in Section 4.

3.3. High-variance Filter

We considered the variance of posterior probabilities from multi-models to examine the factor of unstable performance. Although the consensus posterior probability was

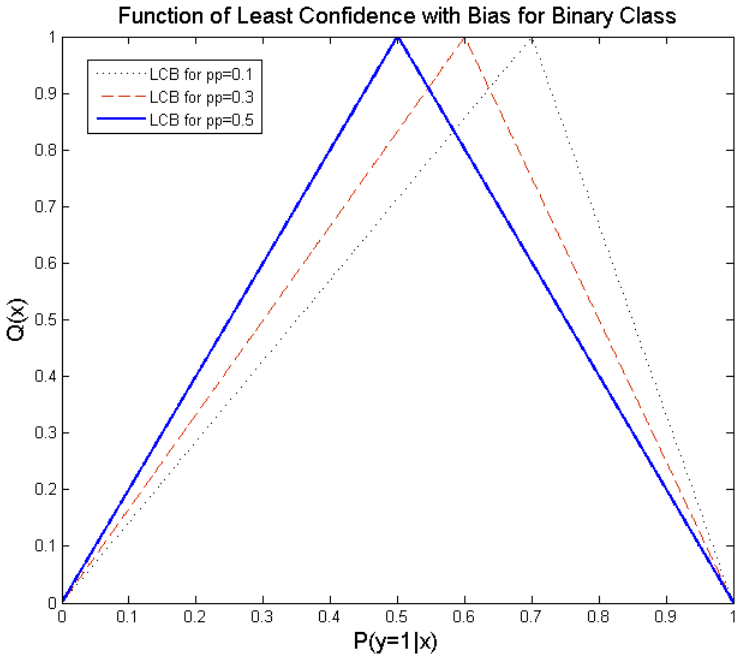


Figure 1: The function of least confidence with bias for different *pp*(Percentage of Positive label). If $pp = 0.5$, it is equivalent to original least confidence.

introduced, the beginning part of the learning curve still has a possibility of encountering a significant drop. For example, the point on the curve based on training with only two, four, eight, or sixteen training samples is much lower than the starting point. We did an independent experiment based on the variance from the multiple models. We queried and trained with two subsets of samples with the highest and lowest variance values of posterior probabilities from multiple RF models based on previous prediction. It turns out that the AUC decrease or increase at the beginning of learning curve for the high-variance subset happens much more significantly and frequently than the low-variance one. Hence we introduced a filter to remove the samples with very high variance from the current querying list to increase the stability of performance. Eventually we prefer to query the samples with high informativeness and low variance in the early iteration of active learning. Let us define the filter variance threshold t such that samples with variance value greater than t will not be queried in the current iteration of active learning.

When we have a larger training set, the variance value for each sample would be gradually decreased since the individual model in the prediction committee would be more accurate and generate less disagreement with each other. We manually assign t

for different development datasets such that there would be only one tenth of training samples which are not filtered out for querying at the first iteration of active learning.

3.4. Semi-supervised Learning Support

In our preliminary experiments, we only used cosine similarity function to do prediction by knowing just one positive sample. However, the performance was poor for most of the datasets since the cosine similarity is just not sufficient for label prediction. In the final phase, we performed an additional step for initial training: train the multiple Random Forests models with the positive seed along with a small number of predicted negative samples, which are predicted based on cosine similarity function. It generated better start points in the active learning framework for most of the development datasets. Here are the steps:

Algorithm 1: Semi-supervised learning for prediction based on one positive sample

---

Input: Data feature matrix $\mathbf{x}$ with one positive-labeled seed $y_1 = 1$
Output: CPP($\mathbf{x}$)

1. For all samples, estimate the cosine similarity to the positive-labeled seed;
2. Assign negative labels to K samples with the smallest cosine similarity values;
3. Train the multiple models with one given positive sample and K predicted negative samples and predict for other samples.

We also used this semi-supervised learning model to increase the size of training size by assigning more negative samples with the consensus posterior probability that was closest to negative label without purchasing the label. However, this approach could fail when the predicted negative samples are actually positive and the performance was adversely affected when we used too many predicted negative samples. Thus we need to limit the number of predicted negative samples in the early training.

3.5. Batch Size Validation

In the challenge, batch size (the number of instance to query at each iteration) is another user-defined factor. $B(i)$ represents the batch size at iteration i. Although we have proposed some ideas in order to improve the AUC score when the training set with labels is small, they may not work well uniformly for all datasets.

Batch size validation is proposed such that for each dataset we could find the smallest initial batch size by which our active learning method could generate the highest global score. The intuition is that we are unable to guarantee the performance when we just have a small number of purchased labels. The bad AUCs obtained with initial queries can adversely affect the global score, especially on the log2 scaling in the learning curve space, where the weight on each incremental query is decreased. The purpose of batch size validation for the classifier is to find the optimal initial batch size

to reduce the chance of being negatively affected by overfitting when the training set is not large enough. The optimal initial batch size is the one which can achieve the highest global score in the development phase. In the final phase, we use the minimum sufficient initial batch size for final datasets accordingly.

This method can be applicable when we have a development dataset and final dataset, like the data in the challenge. For the active learning in general, we need to estimate the minimal initial batch size for the specific classifier and dataset, based on experience or an experiment such as batch size validation. Moreover, the idea could be extended to find not only the optimal initial batch size, but also the batch sizes in all other iterations.

4. Experiments and Results

4.1. Active Learning Procedure

Algorithm 2 in the next page presents the procedure we followed combining all the methods discussed earlier:

4.2. Results

We show the global scores in Table 3 and the corresponding learning curves in Figure 2 that represent our most recent performance in the development dataset based on the active learning procedure described in Section 4.1. In Figure 2, the x-axis for each graph represents the number of labels queried in Log2 domain, and the y-axis is AUC score.

We did batch size validation for only ZEBRA, IBN_SINA and NOVA and the results are shown in Figure 3. We did not do the same for ORANGE since Gaussian imputation for missing values could make the variance of output very high and unreliable. SYLVA obviously does not need the validation since the start point is already very high. And we were not sure if HIVA could exactly map to dataset A or C, so the default batch size was used.

For ZEBRA, the ALC score gradually increases when the initial batch size increases. It is obvious that our prediction model is constrained until we have a sufficiently large training set. The highest ALC score is achieved at almost the highest initial batch size. This score is about 20% more than the ALC with default initial batch size. So we applied this optimal initial batch size to final dataset E which is also with continuous features.

For IBN_SINA, the best ALC score was obtained when we used the default batch size. The ALC decreases with initial batch size increases.

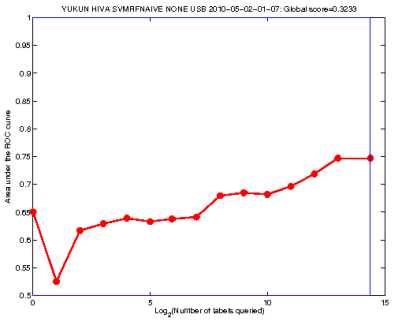
For NOVA, the ALC score goes up when we increase the initial batch size to sixteen and then gradually decreases. Therefore initial batch size of sixteen is the minimum sufficient training set for the prediction of dataset NOVA by our multiple models. We applied this initial batch size to the final dataset D which also contains a very large number of binary features.

Algorithm 2: Active Learning Procedure with uncertainty sampling with biasing consensus (USBC)

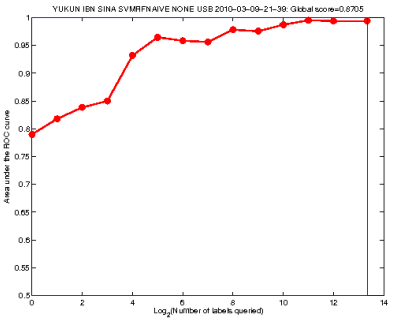
Input: Initial unlabeled set $\mathcal{U}$ containing data feature matrix **x** excluding the first labeled instance; the initial labeled set $\mathcal{L} = \{x_1, y_1 = 1\}$

Output: Global ALC score

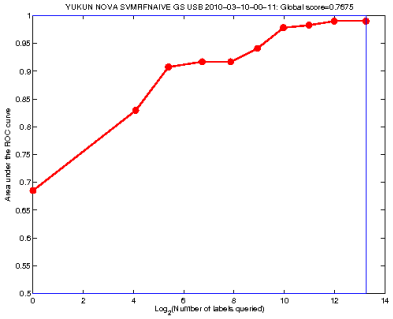
1. Initialization:
   (a) Run preprocessing steps (missing value imputation, PCA, etc) if needed;
   (b) Assign $B(i)$, the batch size as a function of iteration i, from 1 to i_{max} (the last iteration): the default is $B(i) = 2^{i-1}$, but it could be $B(i) = 2^{i+i_o}$, where i_o is 0, 1, 2, ..., i_{omax} (the max i_o that would query all samples at the beginning), depending on the batch size validation result;
2. CPP(**x**) = semi-supervised learning (**x**|$\mathbf{y_1} = \mathbf{1}$) and output initial AUC;
3. Run $Q^{LCB}(\mathbf{x}, \mathbf{0.5})$ and query $B(1)$ sample(s) with the max Q(**x**), where $x \in \mathcal{U}$; update $\mathcal{U}$ and $\mathcal{L}$;
4. Run uncertainty sampling with biasing consensus (USBC) for i from 2 to i_{max}:
   (a) Add at most three predicted negative samples into the training sets (if activated);
   (b) Train by five RF models, predict for all unlabeled samples in $\mathcal{U}$ by CPP, and output AUC(i);
   (c) Run high-variance filter with parameter $t(i)$, temporally remove those $t(i)$ samples from $\mathcal{U}$ but added them back after current iteration (if activated);
   (d) Run $Q^{LCB}(\mathbf{x}, pp)$ and query $B(i)$ samples with the max Q(**x**) for true labels, where $x \in \mathcal{U}$, pp is the positive fraction for samples in $\mathcal{L}$ with two exceptions: if the positive fraction is larger than 0.5, $pp = 0.5$; if $|\mathcal{L}|$ is larger than $(|\mathcal{U}| + |\mathcal{L}|)/10$, $pp = 0.5$;
   (e) Update $\mathcal{U}$ and $\mathcal{L}$; loop over for $i = i + 1$ until $i > i_{max}$;
5. Output global ALC score



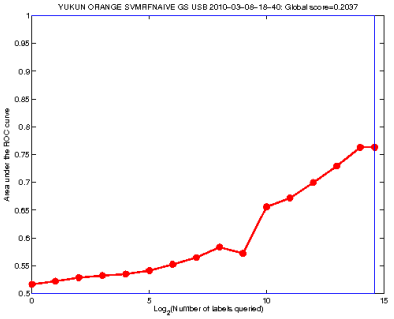
(*a*) Learning Curve HIVA



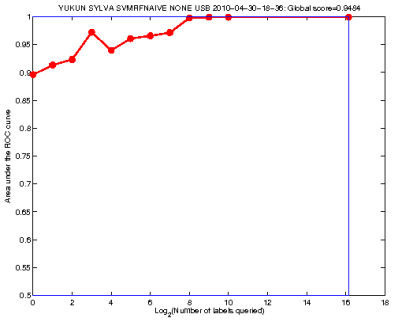
(*b*) Learning Curve IBN_SINA



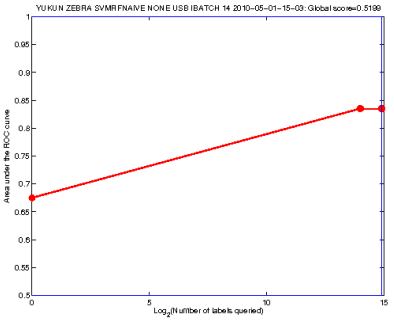
(*c*) Learning Curve NOVA



(*d*) Learning Curve ORANGE



(*e*) Learning Curve SYLVA



(*f*) Learning Curve ZEBRA

Figure 2: Learning Curves for Development Datasets.

Table 3: The experimental results for development datasets

Data Name	ALC Score	AUC Score	Initial AUC Score	Initial Batch Size	Use Filter	Use Predicted Negative
HIVA	0.3233	0.7468 ± 0.79%	0.6502 ± 0.65%	1	No	No
IBN_SINA	0.8705	0.9960 ± 0.09%	0.7900 ± 0.28%	1	No	Yes
NOVA	0.7675	0.9940 ± 0.14%	0.6853 ± 0.38%	16	Yes	Yes
ORANGE	0.2037	0.7630 ± 1.11%	0.5170 ± 0.78%	1	No	Yes
SYLVA	0.9484	0.9990 ± 0.04%	0.8958 ± 0.22%	1	No	No
ZEBRA	0.5199	0.8318 ± 0.56%	0.6751 ± 0.48%	16384	No	No

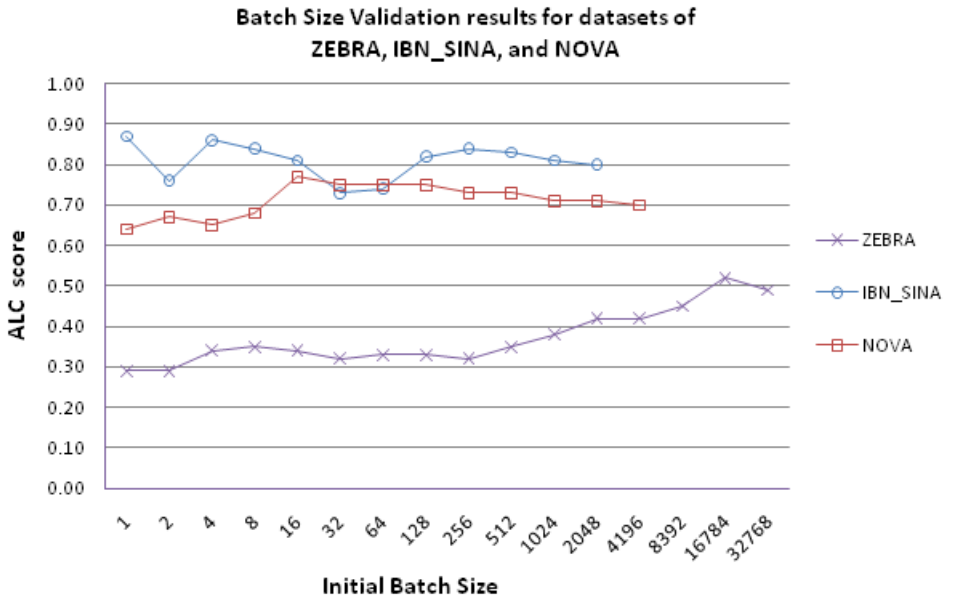


Figure 3: Batch size validation results for development datasets of ZEBRA, IBN_SINA, and NOVA

Table 4: The results for final datasets

Data Name	Our ALC	Best ALC	Our AUC(Initial)	Best AUC	Initial Batch	Use Filter/ Pred-Neg	Final Rank
A	0.3609	0.6289	0.9615 ± 0.39% (0.7500)	0.9615	1	No/Yes	9
B	0.1297	0.3757	0.6484 ± 0.44% (0.5000)	0.7670	1	No/Yes	12
C	0.1876	0.4273	0.7715 ± 0.52% (0.4500)	0.8137	1	No/No	12
D	0.5390	0.8610	0.9554 ± 0.33% (0.4500)	0.9641	16	Yes/Yes	12
E	0.6266	0.6266	0.8939 ± 0.39% (0.7300)	0.9090	30000	No/No	1
F	0.7853	0.8018	0.9976 ± 0.09% (0.5500)	0.9990	1	No/No	3

In Table 4, we show the final results for the datasets in the final phase. We compared our ALC scores and AUC scores (with initial AUC score) with the best ALC scores and AUC scores from participants in the challenge. Figure 4 shows the learning curves of our final results. In Figure 4, X-axis for each graph represents the number of labels queried in Log2 domain, and the y-axis is AUC score. These results are also available from the active learning challenge website: http://www.causality.inf.ethz.ch/activelearning.php?page=factsheet&id=157.

5. Discussion

Uncertainty sampling based querying method is highly dependent on the prediction model. Random Forest based prediction model may not be the best classifier for very-high-dimension, sparse and binary-feature datasets, like dataset ORANGE, HIVA, NOVA, B, C, and D. It did not perform well on datasets B and C. We actually anticipated the poor performance for datasets B and C because our multiple models did not have good prediction on ORANGE and HIVA which are similar to datasets B and C respectively. SVM based methods demonstrated reasonable performance for these types of datasets in this challenge from the other teams. However, Random Forest based prediction model turns out to be an excellent classifier for other datasets. Using our multiple Random Forests models for prediction based on training on almost all labeled training data, we obtained near-perfect performance on dataset F with over 99% AUC score and very good prediction on dataset A, D, and E.

We did not win on dataset D, which is similar to NOVA, because we had a very low initial AUC. It shows that our semi-supervised method did not work well in this type of datasets (see the initial AUC result for dataset B, C and D). However, the high-variance filter was effective on dataset D because the learning curve did not encounter any significant drop (see Figure 4(D)).

For dataset A, we did not perform well because our initial AUCs with no more than 32 training samples are lower than 0.5 (see Figure 4(A)). The use of predicted negative samples in the early iterations of active learning is not helpful and probably is the reason for our failure in dataset A as some of the predicted negative samples are

actually positive. But when we have more than 32 training samples, our performance is better.

The uncertainty sampling with biasing consensus worked well on dataset F (with a score 0.02 less than the winner of dataset F). The learning curve (see Figure 4(F)) is what we expect: in general, the curve is exponentially increasing almost to the top at the beginning iterations and then maintaining the performance to the end. But our first four points on the learning curve were not good enough.

We won in the dataset E by querying 30000 out of 32252 training samples at the beginning. This is the winning strategy because of the batch size validation for ZEBRA, which is similar to dataset E. The semi-supervised learning that generates the starting point with 73% AUC also improved the global score on dataset E. The batch size validation result also indicates when to start active learning for the particular dataset and which method we need to use. On the other hand, we may not be able to use active learning process effectively. For example, it is hard to build an appropriate model with a small training set in dataset E. In this case, we would rather not use active learning but query all samples to achieve the best global score.

6. Conclusion and Future Work

In this paper, we presented our active learning strategies used in the final phase of active learning challenge. Apart from prediction model and query model, our strategy involved semi-supervised learning and batch size validation for active learning. However, we did not win more than one dataset like the other winners. Our methods need further evaluation using additional datasets. The active learning challenge is still an open problem to solve. One possible future direction to explore is to automatically assign batch size as a function of predictive performance and informativeness. We would also like to pursue other algorithmic methods that can consistently increase the learning curve and optimize the global score.

References

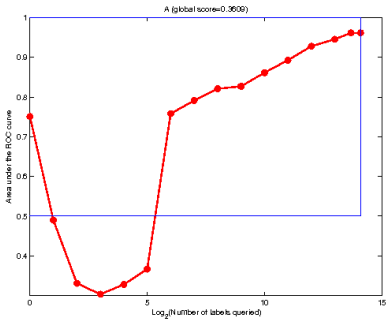
L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Y. Chen and S. Mani. Study of active learning in the challenge. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010.

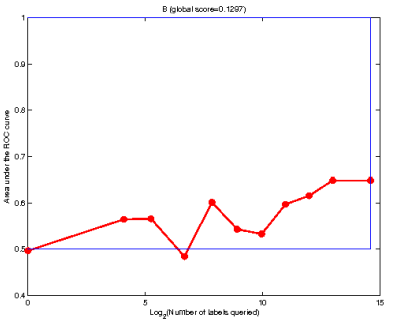
I. Guyon, G. Cawley, G. Dror, and V. Lemaire. Results of the active learning challenge. In *Active Learning Challenge*, volume 15, 2011.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

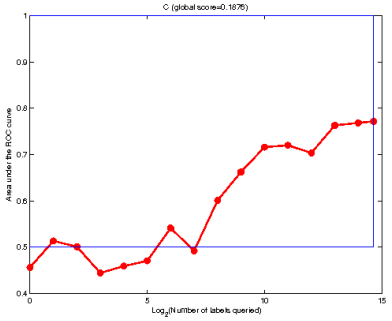
Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.



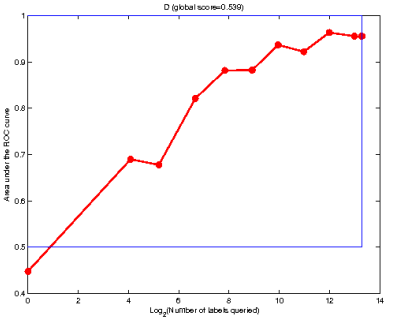
(*a*) Learning Curve A



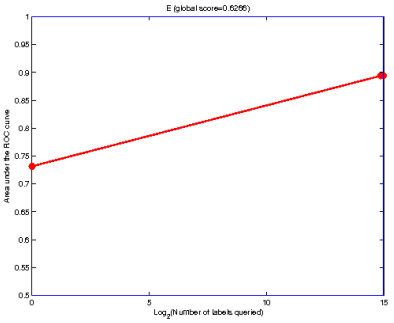
(*b*) Learning Curve B



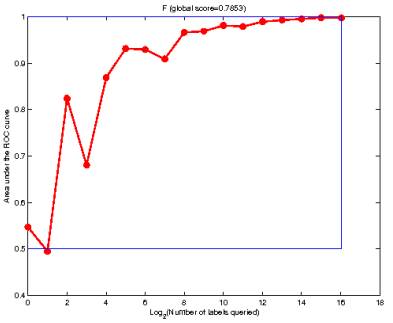
(*c*) Learning Curve C



(*d*) Learning Curve D



(*e*) Learning Curve E



(*f*) Learning Curve F

Figure 4: Learning Curves for Final Dataset A through F.

H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory*, pages 287–294, 1992.

Active Learning with Clustering

Zalán Bodó ZBODO@CS.UBBCLUJ.RO
Zsolt Minier MINIER@CS.UBBCLUJ.RO
Lehel Csató LEHEL.CSATO@CS.UBBCLUJ.RO
Faculty of Mathematics and Computer Science
Babeş–Bolyai University, Cluj-Napoca, Romania

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

Active learning is an important field of machine learning and it is becoming more widely used in case of problems where labeling the examples in the training data set is expensive. In this paper we present a clustering-based algorithm used in the Active Learning Challenge[1]. The algorithm is based on graph clustering with normalized cuts, and uses k-means to extract representative points from the data and approximate spectral clustering for efficiently performing the computations.

Keywords: active learning, large scale spectral clustering, normalized cuts, support vector machines

1. Introduction

In active learning the learner *queries* data points from a large *data pool* that are thought to be the most informative (Settles, 2009). Active learners are useful when obtaining the label of a point is expensive. For example we can consider text categorization problems with a large number of categories – order of thousands or so – where data is easily collected but the assignment of documents to categories requires background knowledge and careful examination, being very time consuming when performed manually.

To find the labels of unlabeled examples, *oracles* are queried in different ways. A popular scenario is pool-based active learning (Lewis and Gale, 1994), where we assume a large data set with only a few labeled and the majority unlabeled examples. An item is chosen by inspection from the unlabeled pool. Other scenarios include query synthesis (Angluin, 1988), where queries are synthesized and novel examples can be generated, or stream-based selective sampling (Atlas et al., 1990), where the examples are coming successively and for each example one has to decide independently whether it is informative or not.

The central problem in active learning is the selection procedure, which can be reduced to *measure* the information content of the unlabeled points. This problem is called the *query strategy*. These can be based on the probabilistic output of a classifier, on the agreement between the members of a committee, based on the estimated reduc-

1. http://www.causality.inf.ethz.ch/activelearning.php

tion of error, to name a few (see eg. Lewis and Gale, 1994; Seung et al., 1992; Roy and McCallum, 2001).

In this paper we propose an active learning method based on spectral clustering (Shi and Malik, 2000) and large-scale approximate spectral clustering (Yan et al., 2009). Our algorithm is based on graph clustering with normalized cuts and uses the property that normalized cuts partition the data using a hyperplane (Rahimi and Recht, 2004). Therefore informativeness can be measured with the unthresholded cluster indicator values as produced by the clustering algorithm; this output can be interpreted as the output of a maximum margin-based classifier.

Since the simple heuristics of using the distance of a point from the separating hyperplane as a measure of informativeness – the smaller the better – was efficient (Tong and Koller, 2001), we apply this strategy in our algorithm. We mention that other semi-supervised or constrained clustering methods could be used, our choice of constrained spectral clustering leads to the query strategy as above whose application is straightforward, and additionally the spectral graph transducer proved effective on various data sets (Joachims, 2003).

The paper is structured as follows. Section 2 presents the components of our algorithm: spectral clustering and large-scale approximate spectral clustering (Section 2.1), spectral graph transducer (Section 2.2) and support vector machines (SVMs) for classification and active learning (Section 2.3). In Section 3 the proposed algorithm is presented in details and Section 4 describes the experiments and discusses the results.

1.1. Problem setting and notation

Let the training data be $X = X_L \cup X_U = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\} \cup \{\mathbf{x}_{\ell+1}, \ldots, \mathbf{x}_{N:=\ell+u}\}$, where X_L is the labeled and X_U is the unlabeled part. We assume that data is sampled i.i.d. from an unknown distribution. The goal in the challenge is to query $s \leq u$ labels of yet unlabeled data X_U from an *oracle* that are the most informative for the learning algorithm.

In the Active Learning Challenge the algorithm is evaluated on a separate *test* data set X_T, $|X_T| = t$. The performance is measured based on the number of queried labels, by iteratively increasing the number of known labels, ℓ, from 1 to N. That is, after querying s labels Y_s of some points X_s the labeled and unlabeled data sets change: $X_L = X_L \cup (X_s, Y_s)$, $X_U = X_U \setminus X_s$. We denote vectors by small boldfaces $\mathbf{a}, \mathbf{b}$; matrices by capital boldfaces $\mathbf{A}, \mathbf{B}$; while scalars and sets are denoted by normal letters $a, b, \ldots, A, B$. Furthermore $\mathbf{A}_{i\cdot}$ and $\mathbf{A}_{\cdot j}$ denotes the i-th row and j-th column of $\mathbf{A}$ respectively. We use $\mathbf{A}'$ to for the transpose of $\mathbf{A}$, and $\|\cdot\|$ for the Euclidean norm.

2. Active learning with spectral clustering

2.1. Large-scale spectral clustering

Spectral graph clustering techniques (von Luxburg, 2006) became popular in the last decade owing to their simplicity and efficiency. They minimize an objective function

involving graph cuts. The two most popular cut objectives are the *ratio cut* and *normalized cut* (von Luxburg, 2006):

$$\text{rcut}(A_1, A_2) = \sum_{i=1}^{2} \frac{\text{cut}(A_i, \overline{A_i})}{|A_i|}, \quad \text{ncut}(A_1, A_2) = \sum_{i=1}^{2} \frac{\text{cut}(A_i, \overline{A_i})}{\text{vol}(A_i)}, \tag{1}$$

where a cut (A_1, A_2) is defined as sum of edge weights between the two sets of graph vertices A_1 and A_2, and the volume of a partition is the sum of edge weights within the partition and all the vertices of the graph. Since exactly solving the above problems is NP hard, usually the relaxed versions are solved (Shi and Malik, 2000).

For the relaxation we introduce the similarity matrix $\mathbf{W}$ and the diagonal degree matrix $\mathbf{D}$ with $D_{ii} = \sum_j W_{ij}$; the unnormalized graph Laplacian (Chung, 1997) which is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. With these notations, the discrete normalized cut problem can be relaxed to solving the following optimization problem:

$$\begin{aligned} \mathbf{y}^* = \arg\min_{\mathbf{y}} \quad & \mathbf{y}'\mathbf{L}\mathbf{y} \\ \text{s.t.} \quad & \mathbf{y}'\mathbf{D}\mathbf{y} = 1, \quad \mathbf{y}'\mathbf{D}\mathbf{1} = 0 \end{aligned} \tag{2}$$

where $\mathbf{y}$ is the cluster indicator vector. The solution is $\mathbf{y}^* = \mathbf{D}^{-1/2}\mathbf{v}_2$, where $\mathbf{v}_2$ is the eigenvector corresponding to the second smallest eigenvalue of the symmetrically normalized graph Laplacian $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. For crisp clusters the values in $\mathbf{y}^*$ are thresholded and treated as cluster indicators.

Having a time complexity of $O(N^3)$ and space complexity of $O(N^2)$, the application of (2) for large data sets is difficult, therefore efficient approximations are needed. A simple strategy is to reduce the number of points considered for clustering without losing too many of the characteristic features of the original data set.

Yan et al. (2009) proposed a fast approximate spectral clustering – k-means-based approximate spectral clustering or KASP – where the mis-clustering rate converges to zero as the number of extracted representative points grows. The representative points are obtained by using k-means clustering and the algorithm is as follows:

1. Perform k-means clustering on the whole data set.
2. Consider the output of k centers as the representative points.
3. Run a spectral clustering algorithm on the representative points.
4. Based on the clustering of the centers assign the initial points to the clusters determined by the spectral method.

For details of the algorithm see Yan et al. (2009). It was tested on some data sets and led to significant speedups and negligible degradation in clustering accuracy.

Normalized spectral clustering is a kernel method that shares similarities with the SVM (see Section 2.3). In (Rahimi and Recht, 2004) the authors showed that normalized spectral clustering can be expressed in terms of a hyperplane separating the unlabeled points maximizing the *gap* as given below:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \|\mathbf{w}'\mathbf{\Phi}\mathbf{D}^{-1/2}\|^2, \tag{3}$$

where $\mathbf{w}$ is the normal vector of the hyperplane and $\mathbf{\Phi}$ is the matrix of the transformed points, $\mathbf{\Phi}_{\cdot i} = \phi(\mathbf{x}_i)$, and ϕ is the feature mapping. The cluster indicator value for a point $\mathbf{x}_i$ equals $\mathbf{w}^{*\prime}\phi(\mathbf{x}_i)$, and similarly to the case of SVMs (using the *representer theorem* eg from Schölkopf and Smola, 2002) the *cluster indicator function* is written using kernels as:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \tag{4}$$

where $\boldsymbol{\alpha} = \mathbf{D}^{-1/2}\mathbf{v}_2$ from Eq. (2), and $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})'\phi(\mathbf{y})$ is the kernel function. Since the decision function parameters and the cluster indicators are equal ($\boldsymbol{\alpha} = \mathbf{y}^*$), the active learning heuristic of choosing the closest points to the decision surface can be applied (Settles, 2009). The result is that the cluster indicators can be used to predict the importance of a point: as the cluster indicator gets closer to zero, the point becomes increasingly important.

2.2. Constrained spectral clustering

The spectral graph transducer (SGT) method (Joachims, 2003) can be viewed as a constrained spectral clustering algorithm with explicit label constraints. The algorithm uses the ratio cut, but one can also define it using the normalized cut by simply changing the graph Laplacian to the symmetric normalized Laplacian. We obtain therefore a problem similar to the one presented in (Joachims, 2003):

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathbf{z}'\left(\mathbf{L}_{\text{sym}} + c\,\mathbf{D}^{-1/2}\mathbf{C}\mathbf{D}^{-1/2}\right)\mathbf{z} - 2c\,\mathbf{z}'\mathbf{D}^{-1/2}\mathbf{C}\gamma \\ \text{s.t.} \quad & \|\mathbf{z}\| = 1, \quad \mathbf{z}'\mathbf{D}^{1/2}\mathbf{1} = 0 \end{aligned} \tag{5}$$

where $\mathbf{z} = \mathbf{D}^{1/2}\mathbf{y}$, $\mathbf{y}$ is the resulting cluster indicator, $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ is the symmetric normalized graph Laplacian; γ contains the labels: $\gamma_i = \pm 1$ for labeled and 0 for unlabeled points, and $\mathbf{C}$ is a diagonal matrix with positive values only at the indexes of the labeled points.

The analysis from (Rahimi and Recht, 2004) can be applied in this case also, since the SGT narrows spectral clustering only by a quadratic constraint and therefore we can say that it also finds a separating hyperplane. Accordingly, the cluster indicator values returned by SGT can be viewed as decision function values $f(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x})$, where $\mathbf{w}$ is the normal of the separating hyperplane and ϕ is the feature map.

Consider the problem (5). Due to the representer theorem (Schölkopf and Smola, 2002) we again have the decision function as a linear combination of kernels as in Eq. (4). Moreover, we know that $\mathbf{y} = \mathbf{K}\boldsymbol{\alpha}$, and the resulting decision function (Belkin et al., 2006):

$$f(\mathbf{x}) = \sum_{i=1}^{N} (\mathbf{K}^{-1})_{i\cdot}\mathbf{y}\; k(\mathbf{x}_i, \mathbf{x}). \tag{6}$$

Figure 1 shows the separating hyperplanes obtained for the *two moons* data set using spectral clustering and spectral graph transducer.

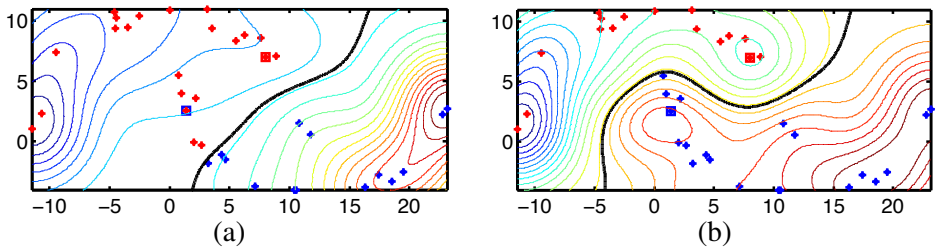


Figure 1: Separating hyperplanes – thick lines – for the *two moons* data set containing two labeled examples: (a) Normalized spectral clustering; (b) Normalized spectral graph transducer.

Similarly to the case of spectral clustering, the absolute value of the cluster indicators returned by the SGT – the distance to the separating hyperplane – can be used to predict the *importance* of a point – a fast and popular *uncertainty sampling* technique for active learning for separating hyperplane-based methods like SVMs (Tong and Koller, 2001).

2.3. Learning with SVMs

Support vector machines – in their original formulation as binary classifiers – find an optimal hyperplane with maximal margin separating the negative examples from the positive ones (Boser et al., 1992; Cortes and Vapnik, 1995). By maximizing the margin of the separating hyperplane, a bound on the actual risk is lowered. The optimization problem is as follows:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}'\mathbf{x}+b) \geq 1-\xi_i, \quad \xi_i \geq 0, \quad i=1,\ldots,\ell. \end{aligned} \tag{7}$$

where $\mathbf{w}$ is the normal vector to the separating hyperplane and ξ_i are the misclassification thresholds – or slack variables (Boser et al., 1992; Vapnik, 1995). The Lagrange formulation of this problem lowers the number of constraints, thus simplifies the optimization task (Boyd and Vandenberghe, 2004). The main advantage of the SVM formulation is its ability to deal with linearly non-separable data in a manner similar to the linear case. To handle linearly non-separable cases – instead of scalar products – we use kernel functions, two examples are the *polynomial* and *Gaussian (or RBF)* kernel functions (Schölkopf and Smola, 2002):

$$k_{\text{poly}}(\mathbf{x},\mathbf{z}) = (a\mathbf{x}'\mathbf{z}+b)^c, \qquad k_{\text{rbf}}(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{z}\|^2\right). \tag{8}$$

Due to the representer theorem (Boser et al., 1992; Vapnik, 1995), the optimal weight vector $\mathbf{w}^*$ can be written as $\mathbf{w}^* = \sum_i \alpha_i^* \phi_i$ and consequently the resulting op-

timal classification function has the form

$$f^*(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i^* y_i k(\mathbf{x}_i, \mathbf{x}) + b^*, \tag{9}$$

where $\boldsymbol{\alpha}^*$ and b^* denote the optimal weight parameters and the optimal bias respectively.

As mentioned in the previous section, we employ a similar method to the simple SVM-based active learning: we assume that a point with unknown label is the more *informative* the *closer* it is to the optimal separating hyperplane from Eq. (9).

We compute thus the distance of each unlabeled point $\mathbf{p}$ from the separating hyperplane $H^* = \{\mathbf{x} \mid f^*(\mathbf{x}) = 0\}$. This is $d(\mathbf{p}, H^*) = |f^*(\mathbf{p})|/\|\mathbf{w}^*\|$, and since $\mathbf{w}^*$ is constant for a given hyperplane, it is sufficient to consider $d(\mathbf{p}, H^*) \propto |f^*(p)|$ for comparison. We employ the fast method of querying the closest point(s) to the hyperplane, that is the points for which $|f(\mathbf{p})|$ is minimal, since the points near the separating hyperplane and near the margins tend to be more influential (Seung et al., 1992; Schohn and Cohn, 2000).

The choice of querying the points close to the separating hyperplane is once more motivated using the notion of *version space*, defined as the region in parameter space whose values classify *all* labeled data correctly. Tong and Koller (2001) implement an algorithm that selects points that reduce the version space as fast as possible by roughly halving it at each iteration. Since SVMs can be regarded as classifiers finding the center of the hypersphere with largest radius inside the version space – *version space duality* – choosing a point as close as possible to the center of the optimal hypersphere is often close to the center of the version space would practically halve the version space. Choosing the next query point as above leads thus to bisect and reduce the version space very fast.

3. The algorithm

Our proposed algorithm is a combination of constrained spectral clustering and k-means clustering. The algorithm based on a type of constrained spectral clustering, namely on the spectral graph transducer method since the amplitude of the decision function is a measure of informativeness. We implemented a *semi-supervised method* since we wanted to incorporate as much information as it was possible both from unlabeled and from test data sets. Therefore we use spectral graph transducer as long as there are unlabeled data. When all labels are known, support vector machines are applied.

We chose spectral clustering since it is a successful algorithm making no strong assumptions on the form of clusters, nor on the ratio of cluster sizes (Shi and Malik, 2000; von Luxburg, 2006). The spectral graph transducer – and thus our algorithm – is based on the semi-supervised *smoothness assumption*, which says that points in high-density regions should have similar labels (Chapelle et al., 2006).

Owing to the large number of training and test points, approximations are needed to speed up the computations. We decided to use a method that selects or generates

representative points from the data set, and uses only the representative points instead of the entire data set. To this end we used k-means based approximate spectral clustering, which generates the representative points as the centers of the resulting clusters.

When the number of labeled examples is small, we extract the representative points from the unlabeled and test sets and we add them to the training data set *as unlabeled points*.

When the number of labeled examples becomes large, representative points are also built for the labeled examples – this is done for computational reasons. In this second case the labeled points and the ones generated from the unlabeled and test sets constitute the training data.

The proposed algorithm divides learning into four cases, depending on the number of labeled points: as the number of labeled points increases different methods are needed for efficiently performing the computations; θ is a threshold on the number of labeled points controlling the treatment of labeled data. The distinct cases of the algorithm are shown below.

1. If labeled points have homogeneous labels
   - Perform k-means-based approximate spectral clustering on the whole data set.
2. Else If $\ell < \theta$
   - Perform k-means on the unlabeled and test data.
   - Form the new data set from the labeled points and the centers of the clusters.
   - Perform SGT on the new data set.
3. Else If $\ell \geq \theta$
   - Perform k-means on the unlabeled and test data.
   - Perform k-means on the labeled data separately in each of the two classes.
   - Form the new data set from the centers of the obtained clusters.
   - Perform SGT on the new data set.
4. Else If $\ell = u$
   - Perform bagging with linear SVMs.

In the first case, when the label of only one point is known and while the labeled points belong to the same class, we perform approximate spectral clustering as described in Section 2.1. The labels of individual points are determined by the label of the cluster the point resides in. The informativeness of a point is determined by the closeness of the cluster indicator value to zero.

When more than one label assignments are known, we separate training into two cases depending on the number of data points. If the number of labeled points is less

than a predetermined threshold θ, we first perform k-means clustering on the unlabeled and test data, and consider the resulting cluster centers as the new representative points. After forming the new data set from the centers and the labeled points we train a normalized SGT on it (Case 2).

If the number of labeled points is above the threshold we cannot deal anymore with these points separately because of their large number, therefore to reduce their number we cluster the labeled points in each of the two classes using k-means. Thus the new data set is formed by combining the cluster centers obtained from the unlabeled and test set with the cluster centers obtained from the labeled set. As in the previous case we train a normalized SGT using this data set (Case 3).

When all the data labels are known we use a bag of linear support vector machines for the binary classification task. Bagging is used to improve the learning algorithm, that is to reduce the average error of the model (Case 4).

4. Experiments and discussion of the results

The Active Learning Challenge was organized in the frame of the Pascal2 Challenge Program and is part of the AISTATS 2010 and WCCI 2010 conference competition programs. The goal of the challenge was to develop active learning methods for a pool-based learning scenario. The organizers provided $6 + 1$ development (6 development and 1 toy) and 6 final data sets.

The data sets are split in two, the first half contains the training and the second half contains the test data. The training, testing and querying steps proceed in an cycle: using the labeled and unlabeled data one trains the learning algorithm, and for all the examples provides prediction scores for the evaluation system. Based on some criterion a few examples are selected from the first half of the data set for querying its labels, and after obtaining them the process repeats until all the budget is spent – initially everybody is provided a sum of N ECU (Experimental Cash Units), where N is the total number of examples in the data set – or an AUC score of 1 is reached.

The algorithms used in our experiments were the following:

- ALG1 – the simple algorithm which initially uses normalized spectral clustering and then requests all the training labels and uses bagging with linear SVMs (no active learning).
- ALG2 – the algorithm described in Section 3, i.e. the method which starts with normalized spectral clustering when only the label of one point is known, then uses a normalized spectral graph transducer, and finally bagging with linear SVMs.
- SVM – the algorithm using linear SVMs as described in Section 2.3 or (Tong and Koller, 2001).

All the algorithms listed above are uncertainty sampling methods since they choose the most informative points based on how distant a point is from the separating hyperplane.

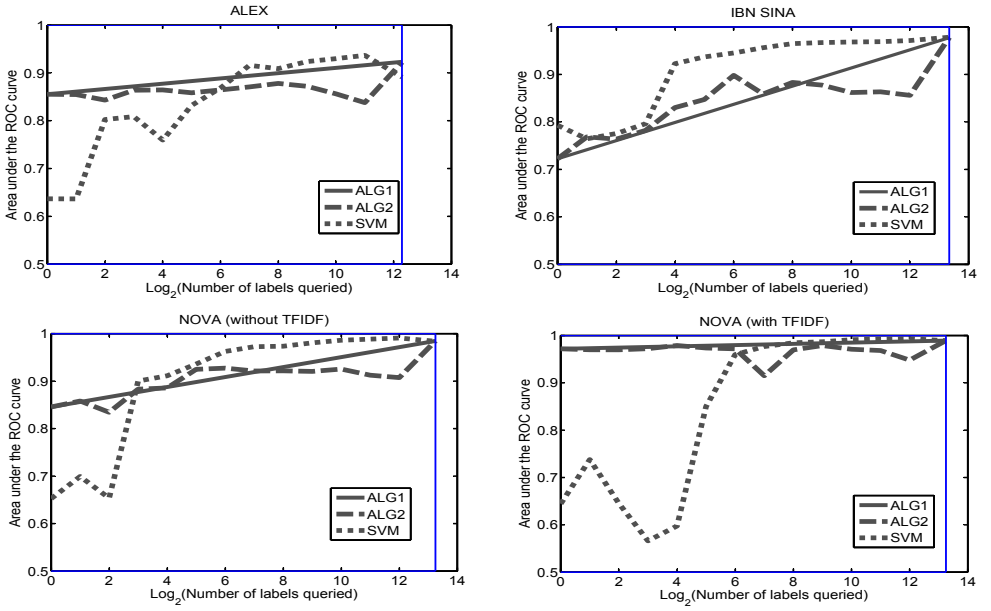


Figure 2: ROC curves for different development data sets: ALEX, IBN_SINA, NOVA, and NOVA using the tf×idf transformation.

We have already argued why and how normalized spectral clustering and spectral graph transducer can be used for this purpose in Sections 2.1 and 2.2.

In our experiments we used only the following data sets:

Data set	Domain	Features	Size
ALEX	Toy data set	11	5000
IBN_SINA	Handwriting recognition	92	10361
NOVA	Text categorization	16969	9733
A	Handwriting recognition	92	17535
D	Text categorization	12000	10000
F	Ecology	12	67628

because in the development phase we made experiments with data sets A, D and F, and when the final data sets appeared we chose the data sets most similar to these.

For our algorithm the threshold θ was set to 2^8, while at the final step we performed bagging with 20 linear support vector machines.

The k-means clustering has two parameters: the first k denotes the number of clusters formed from the unlabeled and test data, while k_{lab} is the number of clusters containing labeled points; k was set to $(|X_U|+|X_T|)/100$, while k_{lab} to 100.

Spectral clustering and SGT uses the affinity matrix $\mathbf{W}$ for calculating the graph Laplacian. Here we used the complete graph of the examples using the Gaussian simi-

	ALEX	IBN_SINA	NOVA	NOVA (with tf×idf)
ALG1	91.86/**77.40**	97.73/69.98	98.46/**83.03**	98.93/**96.12**
ALG2	92.23/72.63	97.79/68.70	98.46/80.75	98.93/93.35
SVM	89.12/68.46	97.83/**82.23**	98.41/81.44	98.92/71.28

Table 1: Table showing the exact results (AUC/ALC) obtained for the development data sets with algorithms ALG1 (the fast method without active learning), ALG2 (algorithm described in Section 3), and SVM (linear SVM using the distance of a point from the hyperplane as an informativeness measure). The best results are typeset in boldface.

	A	D	D (with tf×idf)	F
ALG1	84.96/4.19 (ranked 22/22)	*95.24/74.49*	96.41/86.10 (ranked 1/22)	96.27/38.07 (ranked 15/16)
ALG2	84.52/-13.99	95.20/67.30	96.38/63.22	96.28/28.16
SVM	55.92/12.91	95.23/68.42	96.35/84.24	96.52/60.24

Table 2: Results obtained (AUC/ALC) for the challenge data sets using algorithms ALG1, ALG2 and SVM. We included an additional test using data set D without the tf×idf transformation (second column). The algorithm used in the challenge (ALG1) is indicated by the rectangular frame.

larity,

$$W_{ij} = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \tag{10}$$

where the width parameter σ specifies the distance below which the neighborhood relationship means *similarity*; it was set as the mean norm of the feature vectors in the data set (Chapelle et al., 2006). Furthermore, we set the following parameters of the algorithm: $c = 1000$, $d = 80$ (eigenvalues of the normalized Laplacian) and $\mathbf{C}$ was set to the identity matrix, that is no differentiation between the labeled points was made. These parameters were set based on the experimental results and conclusions from (Joachims, 2003).

For the NOVA and D data sets we performed principal component analysis (PCA) and used only the first $r = 50$ principal components; using this value we obtained the best results on NOVA. Since the training data is textual, before performing PCA we transformed the feature vectors using the *tf×idf* transformation (term frequency × in-

verse document frequency) (Baeza-Yates and Ribeiro-Neto, 1999), but we also report results without applying this transformation. In these data sets we also normalized the vectors to unit length before the learning process.

When using the linear SVM for active learning and while the labeled set contains points from only one class – this includes the first step as well – we calculate the rank of the i-th data point as $1/(1+\|\mathbf{z}-\mathbf{x}_i\|^2)$, where $\mathbf{z}$ is the mean of the labeled points. Other solutions for the one-class problem would be the application of the Gaussian similarity or one-class SVMs (Schölkopf et al., 2001). We used the above similarity measure since no additional parameter is involved in this way, and it provided good results on the development data.

The methods were implemented in MATLAB using the sample code provided by Isabelle Guyon for the challenge[2]. For SVMs we used LIBSVM (Chang and Lin, 2001) and performing fast k-means was accomplished using the package written by Charles Elkan (Elkan, 2003)[3].

The time complexity of the algorithm presented in Section 3 is $O(N \cdot i \cdot (k + k_{\text{lab}} + n) + k^3 + p^3)$, assuming that $N \approx t$. In the formula k and k_{lab} denote the desired number of clusters as defined beforehand, n denotes the dimension of the data, i is the maximum iteration count in k-means and SVM (LIBSVM) instances, and $p = \max\{k_{\text{lab}}, \theta\}$.

The methods were evaluated using two evaluation measures: a *local* and a *global* score. The local score shows the performance (Area Under the ROC Curve, AUC) of the method in the last step of the query process. The global score (Area under the Learning Curve, ALC) characterizes the method by integrating the local score over the queries. To obtain the final global score, ALC is normalized using the following formula:

$$\frac{\text{ALC} - \text{A}_{rand}}{\text{A}_{max} - \text{A}_{rand}}, \tag{11}$$

where A_{max} is the area under the ideal learning curve and A_{rand} is the area under the "lazy" learning curve, that is the learning curve obtained using random predictions. Figure 2 shows the results obtained for the development data sets. The graphs are obtained by plotting the AUC values in terms of the number of labeled points using a $\log_2$ scaling for the x-axis.

Tables 1 and 2 show the AUC/ALC results (in percentage) obtained for some of the development and final data sets, respectively. Based on the results on the development sets we have chosen to run ALG1 on the final data sets; although it is not an active learning algorithm, it is fast and performs sufficiently well on some of the data sets. However, since the algorithm is evaluated in two points only (i.e. with a single label and with all the labels), if the initial clustering does not fit well, a low global score is obtained. This happened in the case of data sets A and F: at the first step we obtained AUC/ALC scores of 19.22%/ – 61.55% and 41.80%/ – 16.41%, respectively. The results obtained for the final data sets by ALG1 are evidently superior to the performances provided by ALG2; this can be explained by the fact that although the learning curves

2. http://www.causality.inf.ethz.ch/al_data/Sample_code.zip
3. http://cseweb.ucsd.edu/~elkan/fastkmeans.html

obtained for ALG2 have monotonically increasing tendency, at the beginning the increases are too small to beat ALG1. This can be caused by k-means, since for a smaller amount of data points the cluster centers are not sufficiently representative.

Other reasons of obtaining low scores can be the inadequate parameter settings. For example spectral clustering is very sensitive to the similarity graph: a suitable similarity function has to be chosen and its parameters have to be set carefully. Additionally, sparsification schemes can be considered for large data sets and better performance, which involves further important parameters.

Domain knowledge was also important in the challenge. For data set D – which shared similar characteristics with NOVA – we applied the tf×idf transformation for giving larger weights for some words based on their distribution in the corpus, used PCA to filter out noise and represent document vectors in a lower dimensional space, and finally normalized each vector to unit length. Applying these techniques used frequently in text categorization we achieved a performance improvement of almost 12% for ALG1. For ALG2 a lower global score is obtained since for 16 and 32 labeled points surprisingly low performances were recorded, in spite of the superior results in the remaining 12 evaluation points.

5. Future work

As a further research direction of this topic we plan to study other large-scale approaches for spectral clustering and SGT. We also plan to study how the application of the decision function in Eq. (4) influences the results of the KASP and RASP (Yan et al., 2009) algorithms. Another direction would be the application of kernel instead of "linear" k-means, however this introduces at least one new parameter. Finally, other graph construction methods are to be investigated, for example heuristics for computing the width parameter of the Gaussian similarity measure using the label information from the training data.

Acknowledgements

The authors acknowledge the partial support of the Romanian Ministry of Education and Research via grant PNII 11-039/2007.

We would also like to thank the reviewers for their work, for the comments, useful suggestions and supporting critiques.

References

Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

Les Atlas, David Cohn, Richard Ladner, M. A. El-Sharkawi, R. J. Marks, M. E. Aggoune, and D. C. Park. Training connectionist networks with queries and selective sampling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 566–573, 1990.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

Bernhard E. Boser, Isabelle Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. *Computational Learning Theory*, 5:144–152, 1992.

Steven P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

Fan Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20: 273, 1995.

Charles Elkan. Using the triangle inequality to accelerate k-means. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 147–153. AAAI Press, 2003.

Thorsten Joachims. Transductive learning via spectral graph partitioning. In *ICML*, pages 290–297. AAAI Press, 2003.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 3–12, London, UK, July 1994. Springer Verlag.

Ali Rahimi and Ben Recht. Clustering with normalized cuts is clustering with a hyperplane. *Statistical Learning in Computer Vision*, 2004.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, San Francisco, CA, 2001.

Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alexander J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proc. 5th Annual ACM Workshop on Comput. Learning Theory*, pages 287–294, New York, NY, 1992. ACM Press.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

Ulrike von Luxburg. A tutorial on spectral clustering. Technical Report 149, Max Planck Institute for Biological Cybernetics, August 2006.

Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *SIGKDD*, pages 907–916. ACM, 2009.

JMLR: Workshop and Conference Proceedings 16:141–155, 2011 Active Learning Challenge

Autonomous Experimentation: Active Learning for Enzyme Response Characterisation

Chris Lovell CJL07R@ECS.SOTON.AC.UK
Gareth Jones GJ07R@ECS.SOTON.AC.UK
Steve R. Gunn SRG@ECS.SOTON.AC.UK
Klaus-Peter Zauner KPZ@ECS.SOTON.AC.UK
School of Electronics and Computer Science
University of Southampton, UK

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

Characterising response behaviours of biological systems is impaired by limited resources that restrict the exploration of high dimensional parameter spaces. Additionally, experimental errors that provide observations not representative of the true underlying behaviour, mean that observations obtained from these experiments cannot be regarded as always valid. To combat the problem of erroneous observations in situations where there are limited observations available to learn from, we consider the use of multiple hypotheses, where potentially erroneous observations are considered as being erroneous and valid in parallel by competing hypotheses. Here we describe work towards an autonomous experimentation machine that combines active learning techniques with computer controlled experimentation platforms to perform physical experiments. Whilst the target for our approach is the characterisation of the behaviours of networks of enzymes for novel computing mechanisms, the algorithms we are working towards remain independent of the application domain.

Keywords: automatic hypothesis generation, closed-loop experimentation

1. Introduction

Nature exhibits biological systems that provide excellent computational mechanisms. Fundamental to that is the interactions of proteins, which can provide non-linear computational abilities (Zauner and Conrad, 2001). Understanding the behaviours exhibited by these interactions can only be achieved through physical experimentation. However, the scale and complexities of these domains mean that the number of experiments that can be performed is always heavily restricted in comparison to the size of the space being searched. Realistically, an experimenter may afford only a handful of experiments per parameter dimension. However, physical experimentation by nature implies that those experiments will produce observations with questionable accuracy. The variability of biological experimentation in particular, means some observations will be unrepresentative of the true underlying behaviour. As such, biological response characterisation exhibits the problems addressed by active learning, namely that learning

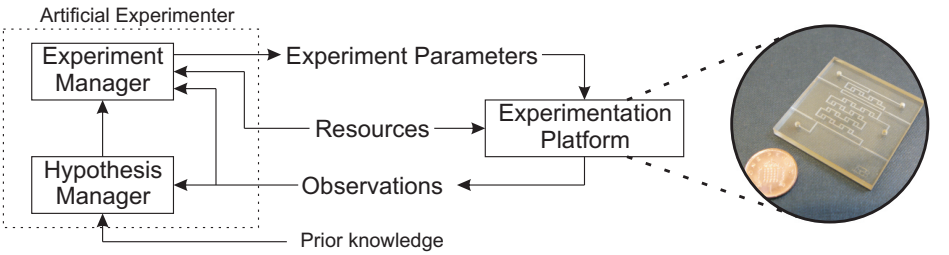


Figure 1: Flow of experimentation between an artificial experimenter and an automated experimentation platform. A prototype of the lab-on-chip platform in development is shown.

must occur with the minimal number of observations as performing experiments is expensive (Cohn et al., 1996). To minimise experimentation costs whilst maximising information gain, an autonomous experimentation machine is in development for biological response characterisation that combines active learning to reduce the number of experiments required to learn, with a resource efficient lab-on-chip automated experiment platform that minimises the volumes of reactants required per experiment. Autonomous experimentation is a closed-loop iterative process, where a computational system proposes hypotheses and actively chooses the next experiment, which is performed by an automated expeirmentation platform, with the results being fed back to the computational system, as illustrated in Figure 1. Here we consider the machine learning component, which is able to learn from a small number of actively chosen observations, where the observations are noisy and potentially erroneous. The lab-on-chip platform is currently in development (Jones et al., 2010).

The development of closed-loop autonomous experimentation machines is still in its infancy. Examples have existed within areas such as electro-chemistry (Żytkow et al., 1990), enzyme characterisation (Matsumaru et al., 2002) and identifying the functions of genes (King et al., 2004). Whilst another approach developed algorithms capable to guide an experimenter to rediscover the urea cycle (Kulkarni and Simon, 1990). Generally such systems, also described as computational scientific discovery systems in the literature, have applied more ad-hoc approaches to experiment design, with the exception of King et al. (2004) that considered a more mathematical active learning approach. However, both autonomous experimentation and many active learning techniques fail to address the problem of learning from only very small sets of experimental observations and that those observations may be unrepresentative of the actual behaviours that should be observed.

Presented here is a technique for producing likely response models from limited, noisy and potentially erroneous observations. To handle the uncertainty presented within this problem, a multiple hypotheses approach is utilised, where differing views about the validity of the observations are considered in parallel. In particular, in in-

stances where an observation does not agree with a hypothesis, new hypotheses are created that consider the observation as both valid and erroneous in parallel, until further experimental evidence is obtained to determine whether it was the observation or the hypothesis that was invalid. Additionally a surprise based active learning technique is presented that guides experimentation to quickly identify the features of the behaviour under investigation, whilst highlighting erroneous observations.

2. Problem Formulation

The biological domains of interest currently do not have significant documented behaviours that can be used to validate the techniques proposed. Therefore to evaluate the approaches presented, we consider a generalised problem that closely matches the target problem domain. First we assume that the true underlying behaviour exhibited by the biological system under investigation, can be modelled by some function $f(x)$. The goal for the system is to build a function $g(x)$, which matches the response of $f(x)$. However, the responses from queries to $f(x)$ can be distorted by experiment measurement and reading errors, causing noise to be applied to both the responses (through ϵ) and to the requested experiment parameters (through δ). Additionally, the lack of control of the biological materials, also present distortions to the responses of $f(x)$. In enzyme experimentation, the reactants can undergo undetectable physical or chemical change, which leads to experiments with those reactants yielding erroneous observations, unrepresentative of the true underlying behaviour. We model such instances as shock noise (through ϕ), which applies a large offset to the response value. Whilst ϵ and δ can occur on every experiment, ϕ will only be non-zero for a small proportion of experiments. We do not consider the case where ϕ occurs for a large number of experiments, as in this instance, the results from such experimentation would be disregarded from consideration anyway. We therefore represent a response characterisation experiment as:

$$y = f(x+\delta) + \epsilon + \phi \tag{1}$$

where parameter x and response y can be replaced with vectors for higher dimensionality.

2.1. Underlying Behaviours

Whilst models of existing behaviours do not currently exist for the domain of interest, we can define some properties of those behaviours that may be expected or would be potentially useful for engineering with these biomolecules. In Figure 2, a range of underlying behaviours, f_a,...f_g, are presented. These behaviours test, in figure order (a–g): (a) linear response, (b) non-linear response, (c) power law, (d) single peak, (e) two peaks, (f) two peaks where one peak is dominant over the other, (g) discontinuity between two distinct behaviours. Behaviours (a-c) are motivated from expectations that behaviours are often described in terms of linear systems or power laws, where (b) is similar to Michaelis-Menton kinetics (Nelson and Cox, 2008) and (c) is similar to responses where there is a presence of cooperativity between substrates and en-

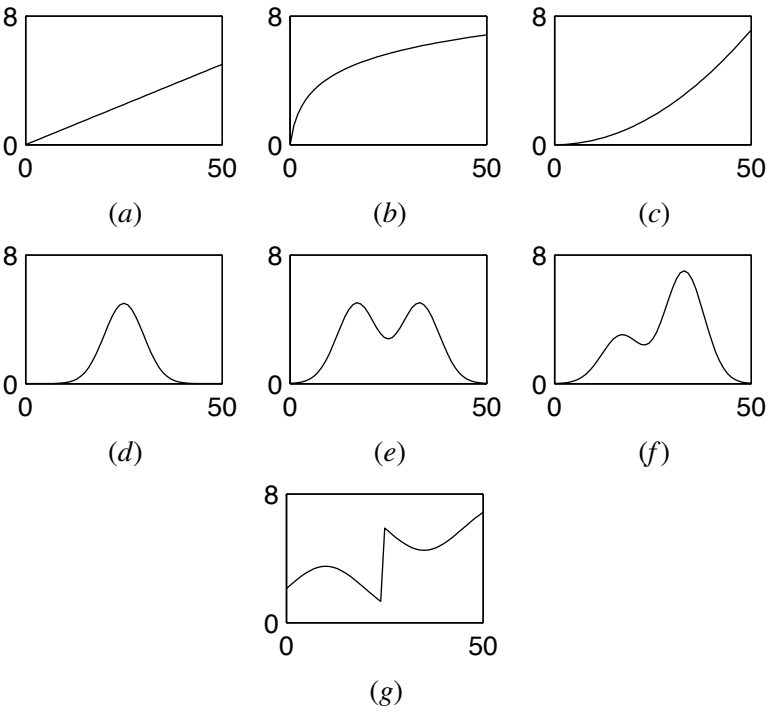


Figure 2: Underlying behaviours motivated from possible enzyme experiment responses.

zymes (Tipton, 2002). Behaviours (d-g) are motivated from the belief that expected behaviours in the domain being investigated may be nonmonotonic and could also include a phase change between distinct behaviours (Zauner and Conrad, 2001). We next discuss the implementation issues of the computational side of autonomous experimentation.

3. Hypothesis Management

A key problem for a hypothesis manager, is how to handle uncertainty in the form of erroneous observations. By accepting all observations as valid, errors can mislead the development of hypotheses. Determining the validity of observations is impeded by the limited resources, which prevent repeat experiments. In this situation, maintaining a single hypothesis appears inefficient in obtaining an accurate representation of the underlying behaviour. Alternatively, we can consider using multiple hypotheses that maintain different views of the validity of the observations in parallel. Whilst many multiple hypotheses based approaches produce hypotheses using random subsets of the data (Freund et al., 1997; Abe and Mamitsuka, 1998), we believe a more structured approach can be applied to deal with the uncertainty about the validity of the observations. That is, where an observation appears erroneous, separate hypotheses can be

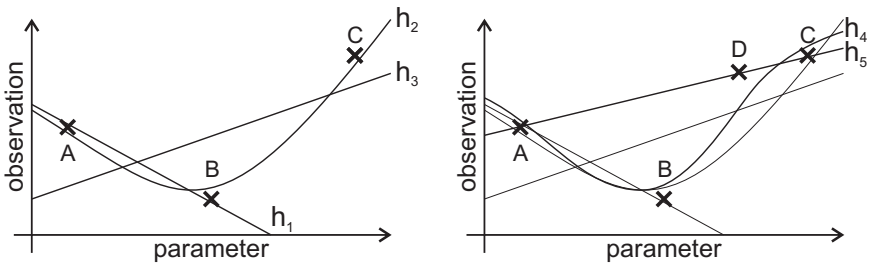


Figure 3: Validity of observations affecting hypothesis proposal. Hypotheses (lines) are formed after observations (crosses) are obtained. In (a), h_1 formed after A and B are obtained questions the validity of C. In (b), D looks to confirm the validity of C however causes h_4 and h_5 to differ in opinion about the validity of B.

used in parallel that consider the observation as erroneous or valid, with further experimentation providing the evidence to differentiate between the hypotheses.

To illustrate this, consider the situation presented in Figure 3, where observations are labelled alphabetically in the order obtained. After the first two observations are obtained, hypothesis h_1 appears as a reasonable hypothesis. On obtaining observation C however, a potential flaw in this hypothesis is found, suggesting that the hypothesis is erroneous, or with the expectation of erroneous observations, the observation itself could be erroneous. Continuing with the acquisition of observation D, the validity of observation C is now more likely, however observation B is now of questionable validity.

To achieve different views of observations with questionable validity, observations can be weighted differently in the regression calculation. In Zembowicz and Żytkow (1991) and Christensen et al. (2003), where the accuracy of observations is known, deliberate weighting of observations has been applied to obtain better predictions of the underlying behaviours. But in the present problem, obtaining accuracy information is restricted by resources. Multiple hypotheses allows different views about the validity of the observations to be considered in parallel, allowing any decisions about observation validity to be postponed until sufficient evidence is available. In the following section we describe this process in more detail.

3.1. Implementation

In practice a hypothesis is represented here by a smoothing spline. A smoothing spline is a piecewise cubic spline regression technique that can be placed within a Bayesian

framework (Wahba, 1990):

$$S_{w,\lambda}(f) = \sum_{i=1}^{n} w_i (y_i - f(x_i))^2 + \lambda \int_a^b (f''(x))^2 \, dx \tag{2}$$

where experiment parameter and observation pairs x_i and y_i are used to train a regression fit of the data. The parameter w_i is a weighting applied to each x_i, y_i pair, and the hyperparameter λ controls the amount of regularisation, with b and a being the maximum and minimum of the x_i values respectively.

The w and λ parameters are chosen by the hypothesis manager for each hypothesis, the method for this follows, such that a hypothesis, h, is the minimiser of the smoothing spline regression function for a particular w and λ:

$$h = \min S_{w,\lambda}(f) \tag{3}$$

The process of the hypothesis manager is as follows. After an experiment has been performed, a set of new hypotheses are proposed. New hypotheses are created from random subsets of the available observations, along with a randomly selected smoothing parameter, so as to allow for different initial views of the parameter space. All new hypotheses are added to the set of working hypotheses. The smoothing parameter is chosen from a set of possible parameters ($\lambda \in \{10, 50, 100, 150, 500, 1000\}$) that allow for a range of different fits of the data, corresponding to different initial views of the behaviour being investigated.

Next the hypothesis manager reviews the validity of the observations that have been obtained. To do this, the hypothesis compares all observations against all of the working hypotheses. Through the smoothing spline, each hypothesis is able to provide an indicative error bar for the prediction of the outcome of a particular experiment parameter (Wahba, 1990). This error bar is used to determine whether or not an observation agrees with a hypothesis, where if the observation falls outside of the error bar value, the observation is said to be in disagreement with the hypothesis. When such a disagreement occurs between a hypothesis and an observation, all of the parameters for that hypothesis are taken, and used to build two new refined hypotheses. These refined hypotheses differ from the original hypothesis, through altering the weighting parameter applied to the observation of questionable validity. One hypothesis will set the weighting applied to the observation to be 0 and the other to 100, so as to create a hypothesis that considers the observation to be erroneous and another that considers the observation to be true, where the high weight will force the outcome of the regression to pass closer to the observation. These two new hypotheses along with the original hypothesis, are kept in the working set of hypotheses.

After this process of refinement, all hypotheses in consideration are evaluated against the available observations using the following function:

$$C(h) = \frac{1}{N} \sum_{n=1}^{N} \exp\left(\frac{-\left(\hat{h}(x_n) - y_n\right)^2}{2\sigma^2} \right) \tag{4}$$

where $\hat{h}(x_n)$ is the hypotheses prediction for experiment parameter x_n, with y_n being the experimental observation for parameter x_n, σ is chosen a priori (currently 1.96), and N is the number of observations. Finally, for computational efficiency, the number of working hypotheses considered in parallel can be reduced. Removing the hypotheses that perform poorly in the evaluation stage, ensures that whilst the number of hypotheses considered in parallel remains large, it does not become computationally infeasible to inspect in the experiment selection stage. In the trials presented in Section 5, 200 new random hypotheses are created in each iteration, and the best 20% of all hypotheses under consideration are maintained into the next round of experimentation. From initial trials it appears that so long as the number of new hypotheses created is large, the number of hypotheses retained after each experiment can be altered as required for performance.

In review, the hypothesis manager maintains an expanding ensemble of working hypotheses throughout the experimentation conducted, where the above process of hypothesis proposal is conducted after every experiment performed. Hypotheses have parameters for the observation weightings, w_i, and smoothing parameter, λ. When creating new hypotheses, the hypothesis manager chooses initial random parameters, through selecting a random subset of the available observations to train from, giving those observations all initial weights of 1, with the rest 0, along with a randomly selected smoothing parameter. The parameter learning for the hypotheses comes through the refinement of the existing hypotheses, where the weight parameters of an existing hypothesis are changed in the new refined hypothesis to either 0 or 100, depending on whether the observation is believed to be erroneous, or valid but indicating a feature of the behaviour not characterised in the original hypothesis. The original hypothesis and the subsequent refinements are then all maintained in the working set of hypotheses, so as to test the new parameter settings in parallel. Only when sufficient experimental evidence is available that contradicts a particular hypothesis, is that hypothesis along with its set of parameters removed from consideration, whilst the more suitable hypotheses remain. Next we discuss how the set of hypotheses in consideration can be used to provide information for determining the experiments to perform.

4. Active Learning Experiment Management

The role of the experiment manager is to employ active learning techniques, to determine the next experiments to perform. The experiment manager uses the information available to it, namely the observations obtained and the hypotheses under consideration. With the hypothesis manager providing a set of competing hypotheses, the experiment manager adopts a query-by-committee style approach for determining the experiments to perform. In query-by-committee, labels or observations as referred to here, are chosen where the committee members most disagree (Seung et al., 1992). In other words, the experiment manager should select the experiments that are most likely to differentiate between and in turn disprove hypotheses under consideration, which agrees the experimental design methods suggested in philosophy of science literature (Chamberlin, 1890).

Experimental design T-optimal approaches exist for separating sets of hypotheses, however they can perform poorly if there is experimental noise (Atkinson and Fedorov, 1975). Alternatively, ensembles of hypotheses have been differentiated between by placing experiments where the variance of the predictions of the hypotheses is greatest (Burbidge et al., 2007). However, selecting experiments where the variance of the hypotheses predictions is greatest, can be misled by outlying hypothesis predictions, as shown in Figure 4.

Therefore we require an alternative active learning technique that will separate hypotheses efficiently. To achieve this we consider a strategy that chooses experiments where there is the maximal disagreement between any two of the hypotheses under consideration:

$$D = \arg\max_{x} \sum_{i=1}^{N} \sum_{j=1}^{N} \int \left(P_{h_i}(y|x) - P_{h_j}(y|x) \right)^2 dy \tag{5}$$

By replacing the y integral with the prediction of a hypothesis, $\hat{h}(x)$, the following equation will separate a set of hypotheses based on their predictions for different x:

$$D'(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} 1 - \exp\left(\frac{-\left(\hat{h}_i(x) - \hat{h}_j(x) \right)^2}{2\sigma_i^2} \right) \tag{6}$$

where $\hat{h}_i$ is the prediction of hypothesis i and σ_i^2 comes from the error bar of h_i for x. This discrepancy approach is more robust than a variance method, as shown in the example set of hypotheses shown in Figure 4, where a variance method would place an experiment where the prediction of the majority of the hypotheses is the same. Further to this, this discrepancy can adapt with the previous observations available, so as to differentiate between only the well performing and currently agreeing hypotheses:

$$D(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} C(h_i) C(h_j) A(h_i, h_j) \left(1 - \exp\left(\frac{-\left(\hat{h}_i(x) - \hat{h}_j(x) \right)^2}{2\sigma_i^2} \right) \right) \tag{7}$$

where $A(h_i, h_j)$ is the agreement between the hypotheses for the previous experiments:

$$A(h_i, h_j) = \frac{1}{N} \sum_{n=1}^{N} \exp\left(\frac{-\left(\hat{h}_i(x_n) - \hat{h}_j(x_n) \right)^2}{2\sigma_i^2} \right) \tag{8}$$

which could alternatively be calculated as a product of the agreement.

This discrepancy equation exploits the hypotheses to provide experiments that can effectively discriminate a set of hypotheses. However, it does not explore the experiment parameter space, which is needed to allow the hypothesis manager to build representative hypotheses in the first place. This can in part be addressed by performing an initial number of exploratory experiments, which also allows for the first hypotheses to be proposed. However, additional consideration must be given to handle this exploration-exploitation trade-off (Auer, 2002). In the following sections, two new experiment selection techniques are presented that consider this trade-off.

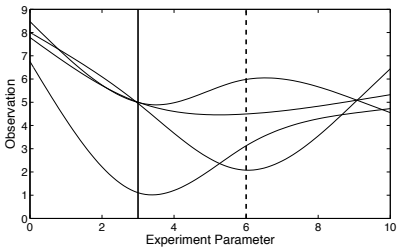


Figure 4: Location of experiments selected to maximise discrepancy between hypotheses. Solid bold vertical line is the experiment parameter the variance approach chooses. Dashed bold vertical line is the experiment parameter the maximum discrepancy approach in Equation (6) chooses. The curves show the predictions of the hypotheses across the parameter space.

4.1. Exploring Peaks in the Discrepancy Equation

By placing experiments where $D(x)$ is maximal, experiments may end up being placed within the same localised area of the experiment parameter space, repeatedly investigating one particular discrepancy, without any exploration. However, if we consider $D(x)$ over all possible experiment parameters, there will likely be local maxima, or peaks, in different areas of the parameter space. These local maxima show different features of the behaviour where the hypotheses disagree elsewhere in the parameter space. Therefore, instead of selecting the maximum of $D(x)$, the maxima can be used to select a set of experiments to perform across the parameter space, which investigate different reasons for hypothesis disagreement, whilst simultaneously allowing some additional exploration.

The process for this experiment selection technique is as follows. Starting with the initial observations and hypotheses, a set of experiments to perform are chosen as those at the peaks of $D(x)$, where experiments are not repeated. Those experiments are then chosen in order of their $D(x)$ value, from largest to smallest, so that if resources are depleted, then the experiments that are likely to differentiate between the hypotheses the most, will have been performed. After each experiment is conducted, new hypotheses are created, but the next set of experiments to perform are only chosen once the current set of experiments have been performed. This process continues until the maximum allowed number of experiments determined by the user have been performed.

4.2. Surprise Based Exploration-Exploitation Switching

Investigating surprising observations, defined as those observations that disagree with a well performing hypothesis, has been highlighted as a technique utilised by successful human experimenters and has also been considered in previous computational scientific discovery techniques (Kulkarni and Simon, 1990; Matsumaru et al., 2002). A surprising observation either highlights a failure in the hypothesis or an erroneous observation. If

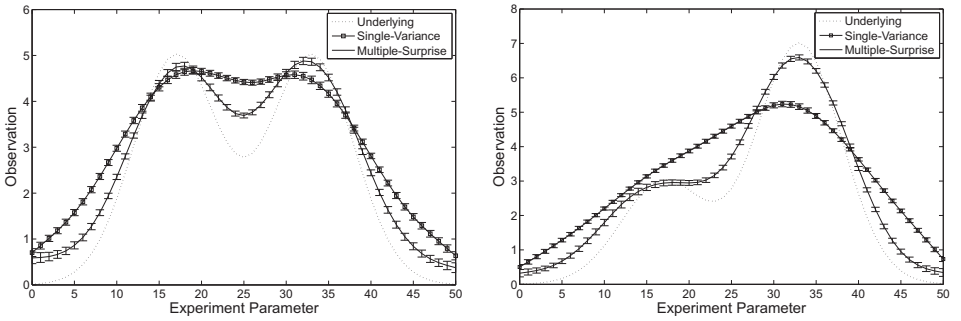
the observation is highlighting a failure of a hypothesis, especially an otherwise well performing hypothesis with a high prior confidence, then additional experiments should be performed to further investigate the behaviour where that observation was found, to allow the development of improved hypotheses. As such we consider the use of surprise to manage the exploration-exploitation trade-off, where obtaining surprising observations will lead to more exploitation experiments, and unsurprising observations lead to exploration experiments.

A Bayesian formulation for surprise has been considered previously in the literature, where a Kullback-Leibler divergence is used to identify surprising improvements to the models being formed (Itti and Baldi, 2009). However, the surprise in Itti and Baldi (2009) is scaled by higher posterior probabilities, but here we are more interested in those hypotheses with high prior confidences but lower posterior confidences as a result of the last experiment. Whilst looking for reductions in posterior probability may appear counter-intuitive, it is important to remember that successful refinement of those hypotheses will result in new hypotheses with higher confidences. Therefore, we interchange the prior and posterior terms to rework the Bayesian surprise function to be:

$$S = \sum_i C(h_i) \log \frac{C(h_i)}{C'(h_i)} \tag{9}$$

where $C(h)$ is the prior confidence of h before the experiment is performed, and $C'(h)$ is the posterior confidence of h after the experiment has been performed, calculated across all hypotheses under consideration using Equation (4), before any new hypotheses are added. Positive values of S states that the observation was surprising, as the overall confidence of the hypotheses have been reduced. Whilst a negative value states the observation was not surprising, as the overall confidence has increased. The result of S can therefore be used to control the switching between exploration and exploitation experiments, where a positive value will dictate that the next experiment will be exploitative, so as to allow investigation of the surprising observation. Whilst a negative value of S will lead to an exploration experiment next, to search for new surprising features of the behaviour.

The procedure for this experiment selection technique is as follows. The prior confidence of the current set of hypotheses before the experiment is performed, is compared with the posterior confidence of those same hypotheses after the experiment is performed, using the surprise function of Equation (9). If $S > 0$ then an exploitation experiment, the maximum of the discrepancy equation $D(x)$, will be performed on the next iteration. Otherwise an exploration experiment will be performed, which is defined as the experiment that has the maximum minimum distance to any previously performed experiment in the experiment parameter space. After S has been calculated, the hypothesis manager will go through the process of creating new hypotheses. This process of evaluating experiments using surprise to choose the next experiment type, is continued until the maximum number of experiments allowed has been performed.



(*a*) (*b*)

Figure 5: Comparison between the true underlying behaviour and the mean of the most confident hypotheses predictions for 100 trials, for the single hypothesis approach using prediction variance experiment selection, and the multiple hypotheses approach using the surprise technique for selecting exploration or exploitation experiments. Shown using behaviour f_e (a) and f_f (b).

5. Results and Discussion

Simulated experiments are conducted using the behaviours described in Section 2.1. All observations have additional Gaussian noise $\epsilon = N(0, 0.5^2)$. Parameter shift noise is kept here at $\delta = N(0,0)$, for clarity of results presented here, as such noise in initial trials appears to have little impact in the performance of the approaches tested here. Experiments are bounded between 0 and 50, and are discretised evenly over the parameter space with 51 possible different experiments, first to make experiment selection more tractable, but also to reflect that physical experiment parameter spaces have finite precision controlled by the laboratory hardware available. Initially 5 exploration experiments are performed that are equidistant to one another in the parameter space, to allow for an initial set of hypotheses to be proposed. One of these initial experiments in each trial has random shock noise $\phi = N(3,1)$ applied to it. The evaluation of the techniques occur over 15 actively selected experiments, where 3 of those experiments produce erroneous observations.

To contrast the multiple hypothesis approach, a single hypothesis approach is used. The single hypothesis is trained with all available observations, using cross-validation to determine the smoothing parameter. Experiments are chosen in the single hypothesis case through random selection, or where the error bar of the hypothesis is greatest. The multiple hypotheses method has experiments chosen through: random selection; choosing the maximum discrepancy value; choosing the peaks of the discrepancy function; and using the surprise method to switch between exploration and exploitation experiments.

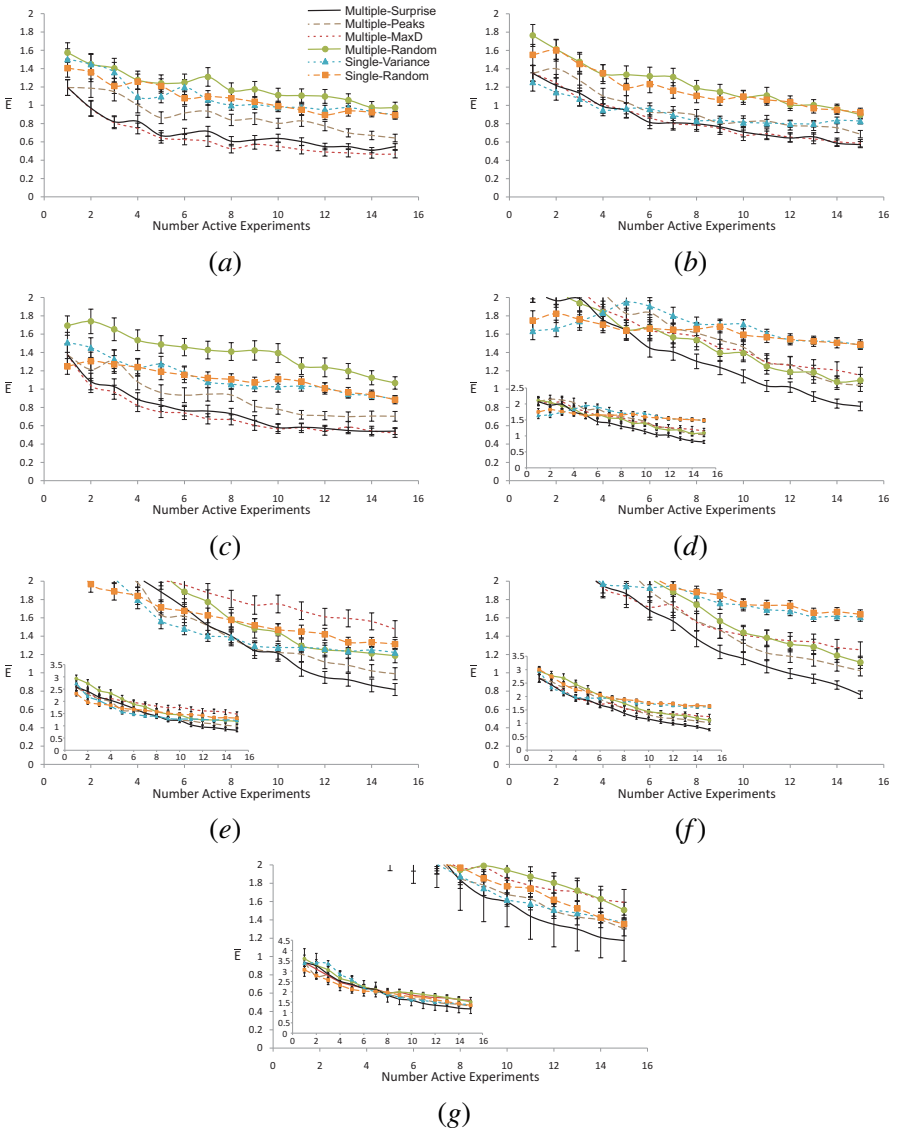


Figure 6: Performance of active learning and hypothesis management techniques. Shown is a comparison of error between the most confident hypothesis and the true underlying behaviour, over the number of actively chosen experiments, where 20% of the observations are erroneous, for 100 iterations. Shown in (a–g) are the corresponding results for the 7 behaviours shown in Figure 2.

To evaluate, the mean squared error between the most confident hypothesis of each trial is compared to the underlying behaviour being investigated:

$$E = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{b}(x_n) - f(x_n)\right)^2 \tag{10}$$

where $\hat{b}(x_n)$ is the prediction of the most confident hypothesis in the trial, for parameter values chosen across the whole parameter space. The mean of these trials over 100 iterations for the 7 different underlying behaviours considered, is shown in Figure 6.

Throughout, the single hypothesis techniques perform poorly in comparison to the multiple hypothesis techniques. Poor performance is due to the single hypothesis generally averaging through all of the data, which can result in features of the behaviours being missed, especially in the more complex nonmonotonic behaviours (d–g), as shown in Figure 5. In the monotonic cases, the difference in performance between the single and multiple hypotheses techniques comes from the single hypothesis averaging through all observations, including the erroneous ones, which allows the erroneous observations to affect the predicted responses, making the hypothesis less accurate.

The multiple hypotheses techniques generally outperform the single hypothesis methods, however the extent of which is dependent on the active learning technique employed. The random strategy performs poorly in the monotonic behaviours (a–c), as experiments are not performed specifically to evaluate the accuracy of observations, which allows for the hypotheses to be misled by the erroneous observations. Whilst this is still an issue in the nonmonotonic behaviours (d–g), the random strategy will generally explore the parameter space more, so identifying the different features of the behaviour being investigated, leading it to have a lower error rate than the single hypothesis techniques, and occasionally similar to the other multiple hypotheses techniques. The maximum discrepancy technique (MaxD) performs well in the simpler monotonic behaviours, as most of the differences between hypotheses will be caused by erroneous observations, which the technique will investigate and be able to produce an accurate representation of the behaviour. In the monotonic behaviours however, the technique may miss some of their features, where its success in identifying the features is dependent on the initial exploratory experiments, as it will perform no exploration on its own and may become stuck investigating the same feature repeatedly. Using the peaks of the discrepancy equation provides more exploration of the parameter space than choosing just the maximum of the equation, allowing for lower error values in the nonmonotonic behaviours. However, in the monotonic behaviours the strategy may spend more experiments investigating small differences between the hypotheses than investigating erroneous observations, meaning that the resultant hypotheses are not as accurate as using the maximum discrepancy for these behaviours. The surprise technique performs consistently well for all behaviours tested, by being able to evaluate the accuracy of the observations and suitability of the hypotheses through exploitation experiments, whilst performing a small number of additional exploratory experiments to further investigate the parameter space.

Over the 100 trials, the surprise technique used few exploration experiments per trial, with an average of 5 exploration experiments in the monotonic cases, normally in the latter stages of experimentation, and 4 exploration experiments in middle to latter stages for the nonmonotonic cases. As the hypotheses quickly produce a good representation of the underlying phenomena in the monotonic cases, additional exploratory experiments are performed as the observations obtained are not surprising to the hy-

potheses. If we allow the multiple peaks technique to have an additional 5 initial exploratory experiments but with 5 fewer exploratory experiments, we find that it has a similar performance to the surprise method with only the 5 initial exploratory experiments, except for a significant improvement in predicting f_g by the multiple peaks technique. However, this is due to the initial 10 exploratory experiments covering all features of the behaviour. The surprise technique is therefore more preferable than the multiple peaks technique, as it has a lower initial exploratory experiment requirement, instead deciding for itself whether additional exploration is required. As such the technique could be adapted to terminate experimentation after performing several unsurprising experiments, reducing the resources used further.

6. Conclusion

Presented is work towards an application for active learning, called autonomous experimentation. Our target domain is automatic enzyme characterisation, where the number of available experiments will be limited to a handful per parameter dimension and that those observations may be erroneous and unrepresentative of the true underlying behaviours. Our belief is that the uncertainty that exists within this problem, is best dealt with through a multiple hypotheses approach. In such an approach, decisions about the validity of observations can be delayed until more experimental evidence is available, through competing hypotheses with different views about the validity of the observations. These multiple hypotheses can be used for effective response characterisation when coupled to an active learning technique, which will outperform a single hypothesis based approach. A technique has been presented that evaluates the surprise of the previous experiment to determine whether the system will next perform an experiment that will explore the parameter space to find new features of the behaviour not yet represented by the hypotheses, or perform an experiment to exploit information held in the hypotheses so as to discriminate between them. The weakness of the multiple hypotheses technique has been shown to be where it is coupled with a random experiment strategy, where erroneous observations can be accepted as true, without experiments testing their validity, leading to the most confident hypotheses being created on inaccurate data. Our next step is to connect the algorithms presented here, with the microfluidic experimentation platform in development (Jones et al., 2010), to demonstrate fully autonomous experimentation.

Acknowledgments

This work was supported in part by a Microsoft Research Faculty Fellowship to KPZ.

References

N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *ICML '98*, pages 1–9, San Francisco, CA, USA, 1998. Morgan Kauffmann.

A. C. Atkinson and V. V. Fedorov. The design of experiments for discriminating between several models. *Biometrika*, 62(2):289–303, 1975.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

R. Burbidge, J. J Rowland, and R. D King. Active learning for regression based on query by committee. In *IDEAL 2007*, pages 209–218. Springer-Verlag, 2007.

T. C. Chamberlin. The method of multiple working hypotheses. *Science (old series)*, 15:92–96, 1890. Reprinted in: Science, v. 148, p. 754–759, May 1965.

S. W. Christensen, I. Sinclair, and P. A. S. Reed. Designing committees of models through deliberate weighting of data points. *JMLR*, 4:39–66, 2003.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

L. Itti and P. Baldi. Bayesian surprise attracts human attention. *Vision Research*, 49: 1295–1306, 2009.

G. Jones, C. J. Lovell, H. Morgan, and K.-P. Zauner. Characterising enzymes for information processing: Microfluidics for autonomous experimentation (abstract). In *9th International Conference on Unconventional Computation*, page 191, Tokyo, Japan, 2010.

R. D. King, K. E. Whelan, F. M. Jones, P. G. K. Reiser, C. H. Bryant, S. H. Muggleton, D. B. Kell, and S. G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 2004.

D. Kulkarni and H. A. Simon. Experimentation in machine discovery. In J. Shrager and P. Langley, editors, *Computational Models of Scientific Discovery and Theory Formation*, pages 255–273. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

N. Matsumaru, S. Colombano, and K.-P. Zauner. Scouting enzyme behavior. In D. B. Fogel et al., editor, *WCCI'02 – CEC*, pages 19–24, Honolulu, Hawaii, 2002. IEEE.

D. L. Nelson and M. M. Cox. *Lehninger Principles of Biochemistry*. W. H. Freeman and Company, New York, USA, 5th edition, 2008.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory*, pages 287–294, 1992.

K. F. Tipton. Principles of enzyme assay and kinetic studies. In R. Eisenthal and M. J. Danson, editors, *Enzyme Assays*, Practical Approach, chapter 1, pages 1–44. Oxford University Press, Oxford, England, 2nd edition, 2002.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference series in applied mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

K.-P. Zauner and M. Conrad. Enzymatic computing. *Biotechnol. Prog.*, 17:553–559, 2001.

R. Zembowicz and J. M. Żytkow. Automated discovery of empirical equations from data. In *ISMIS '91: Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems*, pages 429–440, 1991.

J. Żytkow, M. Zhu, and A.Hussam. Automated discovery in a chemistry laboratory. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 889–894, Boston, MA, 1990. AAAI Press / MIT Press.

JMLR: Workshop and Conference Proceedings 16:157–168, 2011 Active Learning Challenge

Managing Uncertainty within the KTD Framework

Matthieu Geist MATTHIEU.GEIST@SUPELEC.FR
Olivier Pietquin OLIVIER.PIETQUIN@SUPELEC.FR
IMS Research Group, Supélec, Metz, France

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

The dilemma between exploration and exploitation is an important topic in reinforcement learning (RL). Most successful approaches in addressing this problem tend to use some uncertainty information about values estimated during learning. On another hand, scalability is known as being a lack of RL algorithms and value function approximation has become a major topic of research. Both problems arise in real-world applications, however few approaches allow approximating the value function while maintaining uncertainty information about estimates. Even fewer use this information in the purpose of addressing the exploration/exploitation dilemma. In this paper, we show how such an uncertainty information can be derived from a Kalman-based Temporal Differences (KTD) framework and how it can be used.

Keywords: Value function approximation, active learning, exploration/exploitation dilemma

1. Introduction

Reinforcement learning (RL) (Sutton and Barto, 1996) is the machine learning answer to the well-known problem of optimal control of dynamic systems. In this paradigm, an agent learns to control its *environment* (*i.e.*, the dynamic system) through examples of actual interactions. To each of these interactions is associated an immediate reward which is a local hint about the quality of the current control policy. More formally, at each (discrete) time step i the dynamic system to be controlled is in a state s_i. The agent chooses an action a_i, and the dynamic system is then driven in a new state, say s_{i+1}, following its own dynamics. The agent receives a reward r_i associated to the transition (s_i, a_i, s_{i+1}). The agent's objective is to maximize the expected cumulative rewards, which it internally models as a so-called value or Q-function (see later). In the most challenging cases, learning has to be done online and the agent has to control the system while trying to learn the optimal policy. A major issue is then the choice of the behavior policy and the associated dilemma between exploration and exploitation (which can be linked to active learning). Indeed at each time step, the agent can choose an optimal action according to its (maybe) imperfect knowledge of the environment (exploitation) or an action considered to be suboptimal so as to improve its knowledge (exploration) and subsequently its policy. The ϵ-greedy action selection is a popular choice which consists in selecting the greedy action with probability $1-\epsilon$, and

an equally distributed random action with probability ϵ. Another popular scheme is the *softmax* action selection (Sutton and Barto, 1996) drawing the behavior action from a Gibbs distribution. Most successful approaches tend to use an uncertainty information to choose between exploration and exploitation but also to drive exploration. Dearden et al. (1998) maintain a distribution for each Q-value. They propose two schemes. The first one consists in sampling the action according to the Q-value distribution. The second one uses a myopic value of imperfect information which approximates the utility of an information-gathering action in terms of the expected improvement of the decision quality. Strehl and Littman (2006) maintain a confidence interval for each Q-value and the policy is greedy respectively to the upper bound of this interval. This approach allows deriving probably-approximately-correct (PAC) bounds. Sakaguchi and Takano (2004) use a Gibbs policy. However a reliability index (actually a form of uncertainty) is used instead of the more classic temperature parameter. Most of these approaches are designed for problems where an exact (tabular) representation of the value function is possible. Nevertheless, approximating the value in the case of large state spaces is another topic of importance in RL. There are some model-based algorithms which address this problem (Kakade et al., 2003; Jong and Stone, 2007; Li et al., 2009b). They imply approximating the model in addition to the value function. However we focus here an pure model-free approaches (just the value function is estimated). Unfortunately quite few value function approximator allow deriving an uncertainty information about estimated values. Engel (2005) proposes such a model-free algorithm, but the actual use of value uncertainty is left as a perspective. In this paper, we show how some uncertainty information about estimated values can be derived from the Kalman Temporal Differences (KTD) framework of Geist et al. (2009a,b). We also introduce a form of active learning which uses this uncertainty information in order to speed up learning, as well as some adaptations of existing schemes designed to handle the exploration/exploitation dilemma. Each contribution is illustrated and experimented, the last one on a real-world dialogue management problem.

2. Background

2.1. Reinforcement Learning

This paper is placed in the framework of Markov decision process (MDP). An MDP is a tuple $\{S, A, P, R, \gamma\}$, where S is the state space, A the action space, $P : s, a \in S \times A \rightarrow p(.|s,a) \in \mathcal{P}(S)$ a family of transition probabilities, $R : S \times A \times S \rightarrow \mathbb{R}$ the bounded reward function, and γ the discount factor. A policy π associates to each state a probability over actions, $\pi : s \in S \rightarrow \pi(.|s) \in \mathcal{P}(A)$. The value function of a given policy is defined as $V^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, \pi]$ where r_i is the immediate reward observed at time step i, and the expectation is done over all possible trajectories starting in s given the system dynamics and the followed policy. The Q-function allows a supplementary degree of freedom for the first action and is defined as $Q^\pi(s,a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi]$. RL aims at finding (through interactions) the policy π^* which maximises the value function for every state: $\pi^* = \text{argmax}_\pi(V^\pi)$. Two schemes among others can lead to

the optimal policy. First, *policy iteration* involves learning the value function of a given policy and then improving the policy, the new one being greedy respectively to the learnt value function. It requires solving the *Bellman evaluation equation*, which is given here for the value and Q-functions: $V^\pi(s) = E_{s',a|\pi,s}[R(s,a,s') + \gamma V^\pi(s')]$ and $Q^\pi(s,a) = E_{s',a'|\pi,s,a}[R(s,a,s') + \gamma Q^\pi(s',a')]$. The second scheme, *value iteration*, aims directly at finding the optimal policy. It requires solving the *Bellman optimality equation*: $Q^*(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{b\in A} Q^*(s',b)]$. For large state and action spaces, exact solutions are tricky to obtain and value or $Q-$function approximation is required.

2.2. Kalman Temporal Differences - KTD

Originally, the Kalman (1960) filter paradigm is a statistical method aiming at online tracking the hidden state of a non-stationary dynamic system through indirect observations of this state. The idea behind KTD is to cast value function approximation into such a filtering paradigm: considering a function approximator based on a familly of parameterized functions, the parameters are then the hidden state to be tracked, the observation being the reward linked to the parameters through one of the classical Bellman equations. Thereby value function approximation can benefit from the advantages of Kalman filtering and particularly uncertainty management because of statistical modelling.

The following notations are adopted, given that the aim is the value function evaluation, the Q-function evaluation or the Q-function direct optimization:

$$t_i = \begin{cases} (s_i, s_{i+1}) \\ (s_i, a_i, s_{i+1}, a_{i+1}) \\ (s_i, a_i, s_{i+1}) \end{cases} \quad g_{t_i}(\theta_i) = \begin{cases} \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_b \hat{Q}_{\theta_i}(s_{i+1}, b) \end{cases} \tag{1}$$

where $\hat{V}_\theta$ (resp. $\hat{Q}_\theta$) is a parametric representation of the value (resp. Q-) function, θ being the parameter vector. A statistical point of view is adopted and the parameter vector is considered as a random variable. The problem at sight is stated in a so-called *state-space formulation*:

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = g_{t_i}(\theta_i) + n_i \end{cases} \tag{2}$$

Using the vocabulary of Kalman filtering, the first equation is the evolution equation. It specifies that the searched parameter vector follows a random walk which expectation corresponds to the optimal estimation of the value function at time step i. The evolution noise v_i is centered, white, independent and of variance matrix P_{v_i}. The second equation is the observation equation, it links the observed transitions and rewards to the value (or Q-) function through one of the Bellman equations. The observation noise n_i is supposed centered, white, independent and of variance P_{n_i}.

KTD is a second order algorithm: it updates the mean parameter vector, but also the associated covariance matrix after each interaction. It breaks down into three steps. First, *predictions* of the parameters first and second order moments are obtained according to the evolution equation and using previous estimates. Then some *statistics of*

interest are computed. The third step applies a *correction* to predicted moments of the parameters vector according to the so-called Kalman gain K_i (computed thanks to the statistics obtained in second step), the predicted reward $\hat{r}_{i|i-1}$ and the observed reward r_i (their difference being a form of temporal difference error).

Statistics of interest are generaly not analytically computable, except in the linear case. This does not hold for nonlinear parameterizations such as neural networks and for the Bellman optimality equation (because of the max operator). Nevertheless, a derivative-free approximation scheme, the unscented transform (UT) of Julier and Uhlmann (2004), allows estimating first and second order moments of a nonlinearly mapped random vector. Let X be a random vector (typically the parameter vector) and $Y = f(X)$ its nonlinear mapping (typically the g_{t_i} function). Let n be the dimension of the random vector X. A set of $2n+1$ so-called sigma-points and associated weights are computed as follows:

$$\begin{cases} x^{(0)} = \bar{X} & j = 0 \\ x^{(j)} = \bar{X} + (\sqrt{(n+\kappa)P_X})_j & 1 \leq j \leq n \\ x^{(j)} = \bar{X} - (\sqrt{(n+\kappa)P_X})_{n-j} & n+1 \leq j \leq 2n \end{cases} \quad \text{and} \quad \begin{cases} w_0 = \frac{\kappa}{n+\kappa} & j = 0 \\ w_j = \frac{1}{2(n+\kappa)} & 1 \leq j \leq 2n \end{cases} \tag{3}$$

where $\bar{X}$ is the mean of X, P_X is its variance matrix, κ is a scaling factor which controls the accuracy, and $(\sqrt{P_X})_j$ is the j^{th} column of the Cholesky decomposition of P_X. Then the image of each sigma-point through the mapping f is computed: $y^{(j)} = f(x^{(j)}), 0 \leq j \leq 2n$. The set of sigma-points and their images can then be used to compute the following approximations: $\bar{Y} \approx \bar{y} = \sum_{j=0}^{2n} w_j y^{(j)}$, $P_Y \approx \sum_{j=0}^{2n} w_j (y^{(j)} - \bar{y})(y^{(j)} - \bar{y})^T$ and $P_{XY} \approx \sum_{j=0}^{2n} w_j (x^{(j)} - \bar{X})(y^{(j)} - \bar{y})^T$.

Thanks to the UT, practical algorithms can be derived. At time-step i, a set of sigma-points is computed from predicted random parameters characterized by mean $\hat{\theta}_{i|i-1}$ and variance $P_{i|i-1}$. Predicted rewards are then computed as images of these sigma-points using one of the observation functions (1). Then sigma-points and their images are used to compute statistics of interest. This gives rise to a generic algorithm valid for any of the three Bellman equations and any parametric representation of V or Q summarized in Alg. 1, p being the number of parameters. More details as well as theoretical results (such as proofs of convergence) about KTD are provided by Geist and Pietquin (2010).

3. Computing Uncertainty over Values

The parameters being modeled as random variables, the parameterized value for any given state is a random variable. This model allows computing the mean and associated uncertainty. Let $\hat{V}_\theta$ be the approximated value function parameterized by the random vector θ of mean $\bar{\theta}$ and variance matrix P_θ. Let $\bar{V}_\theta(s)$ and $\hat{\sigma}^2_{V_\theta}(s)$ be the associated mean and variance for a given state s. To propagate the uncertainty from the parameters to the approximated value function a first step is to compute the sigma-points associated to the parameter vector, that is $\Theta = \{\theta^{(j)}, 0 \leq j \leq 2p\}$, as well as corresponding weights, from $\bar{\theta}$ and P_θ as described before. Then the images of these sigma-points are computed using the parameterized value function: $\mathcal{V}_\theta(s) = \{\hat{V}^{(j)}_\theta(s) = \hat{V}_{\theta^{(j)}}(s), \quad 0 \leq j \leq 2p\}$.

Algorithm 1: KTD

Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$

for $i \leftarrow 1, 2, \ldots$ **do**

Observe transition t_i and reward r_i

Prediction Step $\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$ $P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$

Sigma-points computation $\Theta_{i|i-1} = \{\hat{\theta}^{(j)}_{i|i-1}, 0 \leq j \leq 2p\}$ /* from $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ */

$\mathcal{W} = \{w_j, 0 \leq j \leq 2p\}$ $\mathcal{R}_{i|i-1} = \{\hat{r}^{(j)}_{i|i-1} = g_{t_i}(\hat{\theta}^{(j)}_{i|i-1}), 0 \leq j \leq 2p\}$ /* see Eq. (1) */

Compute statistics of interest $\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$ $P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$ $P_{r_i} = \sum_{j=0}^{2p} w_j (\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})^2 + P_{n_i}$

Correction step $K_i = P_{\theta r_i} P_{r_i}^{-1}$ $\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i(r_i - \hat{r}_{i|i-1})$ $P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$

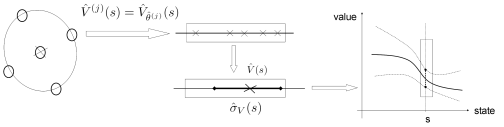


Figure 1: Uncertainty computation.

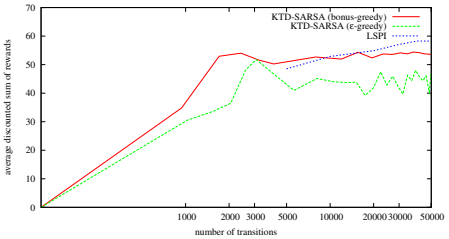


Figure 2: Dialog management results.

Knowing these images and corresponding weights, the statistics of interest are computed: $\bar{V}_\theta(s) = \sum_{j=0}^{2p} w_j \hat{V}_\theta^{(j)}(s)$ and $\hat{\sigma}^2_{\hat{V}_\theta}(s) = \sum_{j=0}^{2p} w_j (\hat{V}_\theta^{(j)}(s) - \bar{V}_\theta(s))^2$. This is illustrated on Fig. 1. Extension to Q-function is straightforward. So, as at each time-step uncertainty information can be computed in the KTD framework.

4. A Form of Active Learning

4.1. Principle

It is shown here how this available uncertainty information can be used in a form of active learning. The KTD algorithm derived from the Bellman optimality equation, that is Alg. 1 with third equation of Eq. (1), is named KTD-Q. It is an off-policy algorithm: it learns the optimal policy π^* while following a different behaviorial policy b. A natural question is: what behaviorial policy to choose so as to speed up learning? Let i be the current temporal index. The system is in a state s_i, and the agent has to choose an action a_i. The predictions $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ are available and can be used to approximate the uncertainty of the Q-function parameterized by $\theta_{i|i-1}$ in the state s_i and for any action a. Let $\hat{\sigma}^2_{Q_{i|i-1}}(s_i, a)$ be the corresponding variance. The action a_i is chosen according to the following heuristic:

$$b(.|s_i) = \frac{\hat{\sigma}_{Q_{i|i-1}}(s_i, .)}{\sum_{a \in A} \hat{\sigma}_{Q_{i|i-1}}(s_i, a)} \tag{4}$$

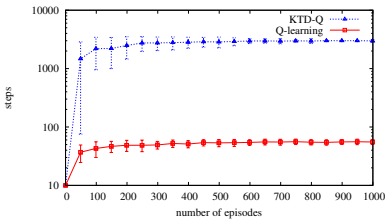


Figure 3: Optimal policy learning.

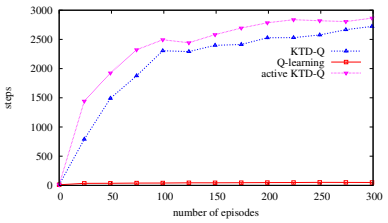


Figure 4: Random and active learning.

This totally explorative policy favours uncertain actions. The corresponding algorithm which is called active KTD-Q (Alg. 1 with 3rd Eq. of (1) and policy (4)).

4.2. Experiment

The second experiment is the inverted pendulum benchmark. This task requires maintaining a pendulum of unknown length and mass at the upright position by applying forces to the cart it is attached to. It is fully described by Lagoudakis and Parr (2003) and we use the same parameterization (a mixture of Gaussian kernels). The goal is here to compare two value-iteration-like algorithms, namely KTD-Q and Q-learning, which aim at learning directly the optimal policy from suboptimal trajectories (off-policy learning). As far as we know, KTD-Q is the first second-order algorithm for Q-function approximation in a value iteration scheme, the difficulty being to handle the max operator (Yu and Bertsekas (2007) propose also such an algorithm, however for a restrictive class of MDP). That is why we compare it to a first-order algorithm. The active learning scheme is also experimented: it uses the uncertainty computed by KTD to speed up convergence.

For Q-learning, the learning rate is set to $\alpha_i = \alpha_0 \frac{n_0+1}{n_0+i}$ with $\alpha_0 = 0.5$ and $n_0 = 200$, according to Lagoudakis and Parr (2003). For KTD-Q, the parameters are set to $P_{0|0} = 10I$, $P_{n_i} = 1$ and $P_{v_i} = 0I$. For all algorithms the initial parameter vector is set to zero. Training samples are first collected online with a random behavior policy. The agent starts in a randomly perturbed state close to the equilibrium. Performance is measured as the average number of steps in an test episode (a maximum of 3000 steps is allowed). Results are averaged over 100 trials. Fig. 3 compares KTD-Q and Q-learning (the same random samples are used to train both algorithms). Fig. 4 adds active KTD-Q for which actions are sampled according to (4). Average length of episodes with totally random policy is 10, whereas it is 11 for policy (4). Consequently the increase in length can only slightly help to improve speed of convergence (at most 10%, much less than the real improvement which is about 100%, at least at the beginning).

According to Fig. 3, KTD-Q learns an optimal policy (that is balancing the pole for the maximum number of steps) asymptotically and near-optimal policies are learned after only a few tens of episodes (notice that these results are comparable to the LSPI algorithm). With the same number of learning episodes, Q-learning with the same linear parameterization fails to learn a policy which balances the pole for more than a few tens of time steps. Similar results for Q-learning are obtained by Lagoudakis and Parr

(2003). According to Fig. 4, it is clear that sampling actions according to uncertainty speeds up convergence. It is almost doubled in the first 100 episodes. Notice that this active learning scheme could not have been used for Q-learning with value function approximation, as this algorithm cannot provide uncertainty information.

5. Exploration/Exploitation Dilemma

In this section, we present several approaches designed to handle the dilemma between exploration and exploitation (which can be linked to active learning). The first one is the well known ϵ-greedy policy, and it serves as a baseline. Other approaches are inspired from the literature and use the available uncertainty information (see Sec. 3 for its computation). The corresponding algorithms are a combination of KTD-SARSA (Alg. 1 with 2$^{\text{nd}}$ Eq. of (1)) with policies (5-8).

5.1. ϵ-greedy Policy

With an ϵ-greedy policy (Sutton and Barto, 1996), the agent chooses a greedy action respectively to the currently estimated Q-function with a probability $1-\epsilon$, and a random action with a probability ϵ (δ is the Kronecker symbol):

$$\pi(a_{i+1}|s_{i+1}) = (1-\epsilon)\delta(a_{i+1} = \underset{b\in A}{\operatorname{argmax}}\, \bar{Q}_{i|i-1}(s_{i+1},b)) + \epsilon\delta(a_{i+1} \neq \underset{b\in A}{\operatorname{argmax}}\, \bar{Q}_{i|i-1}(s_{i+1},b)) \quad (5)$$

This policy is perhaps the most basic one, and it does not use any uncertainty information. An arbitrary Q-function for a given state and 4 different actions is illustrated on Fig. 6. For each action, it gives the estimated Q-value as well as the associated uncertainty (that is $\pm$ estimated standard deviation). For example, action 3 has the highest value and the lowest uncertainty, and action 1 the lowest value but the highest uncertainty. The probability distribution associated to the ϵ-greedy policy is illustrated on Fig. 5.a. The highest probability is associated to action 3, and other actions have the same (low) probability, despite their different estimated values and standard deviations.

5.2. Confident-greedy Policy

The second approach we propose consists in acting greedily according to the upper bound of an estimated confidence interval. The approach is not novel (Kaelbling, 1993), however some PAC (probably approximately correct) guarantees have been given recently by Strehl and Littman (2006) for a tabular representation (for which the confidence interval is proportional to the inverse of the square root of the number of visits to the considered state-action pair). In our case, we postulate that the confidence interval width is proportional to the estimated standard deviation (which is true if the parameters distribution is assumed to be Gaussian). Let α be a free positive parameter, we define the confident-greedy policy as:

$$\pi(a_{i+1}|s_{i+1}) = \delta\Big(a_{i+1} = \underset{b\in A}{\operatorname{argmax}}\, (\bar{Q}_{i|i-1}(s_{i+1},b) + \alpha\hat{\sigma}_{Q_{i|i-1}}(s_{i+1},b))\Big) \quad (6)$$

The same arbitrary Q-values are considered (see Fig. 6), and the confident-greedy policy is illustrated on Fig. 5.b which represents the upper bound of the confidence interval. Action 1 is chosen because it has the highest score (despite the fact that it has the lowest estimated value). Notice that action 3, which is greedy respectively to the estimated Q-function, has only the third score.

5.3. Bonus-greedy Policy

The third approach we propose is inspired from the method of Kolter and Ng (2009). The policy they use is greedy respectively to the estimated Q-function plus a bonus, this bonus being proportional to the inverse of the number of visits to the state-action pair of interest (which can be interpreted as a variance, instead of the square-root of this quantity for interval estimation-based approaches which can be interpreted as a standard deviation). The bonus-greedy policy we propose uses the variance rather than the standard deviation, and is defined as (β_0 and β being two free parameters):

$$\pi(a_{i+1}|s_{i+1}) = \delta\Big(a_{i+1} = \underset{b\in A}{\operatorname{argmax}}\,(\bar{Q}_{i|i-1}(s_{i+1},b) + \beta\frac{\hat{\sigma}^2_{Q_{i|i-1}}(s_{i+1},b)}{\beta_0 + \hat{\sigma}^2_{Q_{i|i-1}}(s_{i+1},b)})\Big) \tag{7}$$

The bonus-greedy policy is illustrated on Fig. 5.c, still using the arbitrary Q-values and associated standard deviations of Fig. 6. Action 2 has the highest score, it is thus chosen. Notice that the three other actions have approximately the same score, despite the fact that they have quite different Q-values.

5.4. Thompson Policy

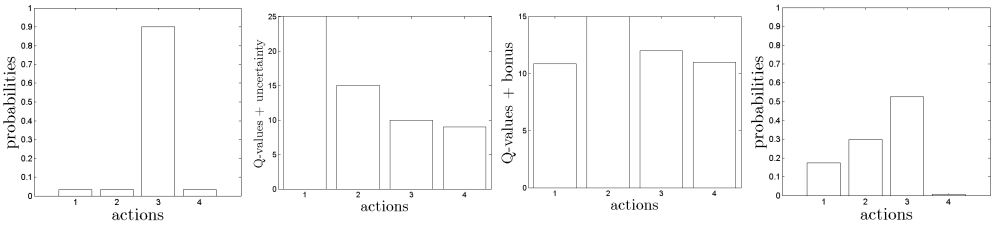
Recall that the KTD algorithm maintains the parameters mean vector and variance matrix. Assuming that the parameters distribution is Gaussian, we propose to sample a set of parameters from this distribution, and then to act greedily according to the resulting sampled Q-function. This type of scheme was first proposed by Thompson (1933) for a bandit problem, and it has been recently introduced into the reinforcement learning community in the tabular case (Dearden et al., 1998; Strens, 2000). Let the Thompson policy be:

$$\pi(a_{i+1}|s_{i+1}) = \underset{b\in A}{\operatorname{argmax}}\,\hat{Q}_\xi(s_{i+1},b) \text{ with } \xi \sim \mathcal{N}(\hat{\theta}_{i|i-1}, P_{i|i-1}) \tag{8}$$

We illustrate the Thompson policy on Fig. 5.d by showing the distribution of the greedy action (recall that parameters are random, and thus the greedy action too). The highest probability is associated to action 3. However, notice that a highest probability is associated to action 1 than to action 4: the first one has a lower estimated Q-value, but it is less certain.

5.5. Experiment

The bandit problem is an MDP with one state and N actions. Each action a implies a reward of 1 with probability p_a, and a reward of 0 with probability $1 - p_a$. For an



a. ϵ-greedy. b. Confident-greedy. c. Bonus-greedy. d. Thompson.

Figure 5: Policies.

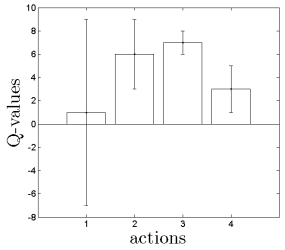


Figure 6: Q-values and associated uncertainty.

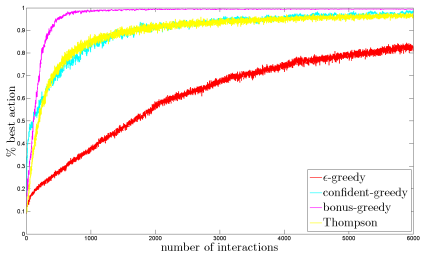


Figure 7: Bandit results.

action a^* (randomly chosen at the beginning of each experiment), the probability is set to $p_{a^*} = 0.6$. For all other actions, the associated probability is uniformly and randomly sampled between 0 and 0.5: $p_a \sim \mathcal{U}_{[0,0.5]}, \forall a \neq a^*$. Presented results are averaged over 1000 experiments. The performance of a method is measured as the percentage of time the optimal action has been chosen, given the number of interactions between the agent and the bandit. A tabular representation is adopted for KTD-SARSA, and the following parameters are used[1]: $N = 10$, $P_{0|0} = 0.1I$, $\theta_{0|0} = I$, $P_{n_i} = 1$, $\epsilon = 0.1$, $\alpha = 0.3$, $\beta_0 = 1$ and $\beta = 10$. As the considered bandit has $N = 10$ arms, a random policy has a performance of 0.1. Notice also that a purely greedy policy would choose systematically the first action for which the agent has observed a reward.

Results presented in Fig. 7 compare the four schemes. The ϵ-greedy policy serves as a baseline, and all proposed schemes using the available uncertainty performs better. Thompson policy and confident-greedy policy perform approximately equally well, and the best results are obtained by the bonus-greedy policy. Of course, these quite preliminary results do not allow to conclude about guarantees of convergence of the proposed schemes. However, they tend to show that the computed uncertainty information is meaningful and that it can provide useful for the dilemma between exploration and exploitation.

1. For an empirical study of the sensitivity of performance of the proposed policies as a function of parameter setting, see Geist (2009).

6. Dialogue management application

In this section is proposed an application to a real world problem: spoken dialogue management. A spoken dialog system (SDS) generally aims at providing information to a user through natural language-based interactions. An SDS has roughly three modules: a speech understanding component (speech recognizer and semantic parser), a dialogue manager and a speech generation component (natural language generator and speech synthesis). Dialogue management is a sequential decision making problem where a dialogue manager has to select which information should be asked or provided to the user when in a given situation. It can thus be cast into the MDP framework (Levin et al., 2000; Singh et al., 1999; Pietquin and Dutoit, 2006). The set of *actions* a dialog manager can select is defined by so called *dialog acts*. There can be different dialog acts such as: greeting the user, asking for a piece of information, providing a piece of information, asking for confirmation about a piece of information, closing the dialog *etc.* The *state* of a dialog is usually represented efficiently by the Information State paradigm (Larsson and Traum, 2000). In this paradigm, the dialogue state contains a compact representation of the history of the dialogue in terms of dialog acts and user responses. It summarizes the information exchanged between the user and the system until the considered state is reached. A dialogue management strategy π is therefore a mapping between dialogue states and dialogue acts. According to the MDP framework, a reward function has to be defined. The immediate reward is often modeled as the contribution of each action to the user's satisfaction (Singh et al., 1999). This is a subjective reward which is usually approximated by a linear combination of objective measures.

The considered system is a form-filling spoken dialog system. It is oriented toward tourism information, similarly to the one described by Lemon et al. (2006). Its goal is to provide information about restaurants based on specific user preferences. There are three slots in this dialog problem, namely the location of the restaurant, the cuisine type of the restaurant and its price-range. Given past interactions with the user, the agent asks a question so as to propose the best choice according to the user preferences. The goal is to provide the correct information to the user with as few interactions as possible. The corresponding MDP's state has 3 continuous components ranging from 0 to 1, each representing the averaging of filling and confirmation confidence scores (provided by the automatic speech recognition system) of the respective slots. There are 13 possible actions: ask for a slot (3 actions), explicit confirmation of a slot (3 actions), implicit confirmation of a slot and ask for another slot (6 actions) and close the dialog by proposing a restaurant (1 action). The corresponding reward is always 0, except when the dialog is closed. In this case, the agent is rewarded 25 per correct slot filling, -75 per incorrect slot filling and -300 per empty slot. The discount factor is set to $\gamma = 0.95$. Even if the ultimate goal is to implement RL on a real dialog management problem, in this experiment a user simulation technique was used to generate data (Pietquin and Dutoit, 2006). The user simulator was plugged to the DIPPER dialogue management system (Lemon et al., 2006) to generate dialogue samples. The Q-function is represented using one RBF network per action. Each RBF network has three equi-spaced Gaussian

functions per dimension, each one with a standard deviation of $\sigma = \frac{1}{3}$ (state variables ranging from 0 to 1). Therefore, there are 351 (*i.e.*, $3^3 \times 13$) parameters.

KTD-SARSA with ϵ-greedy and bonus-greedy policies are compared on Fig.2 (results are averaged over 8 independent trials, and each point is averaged over 100 past episodes: a stable curve means a low standard deviation). LSPI, a batch and off-policy approximate policy iteration algorithm (Lagoudakis and Parr, 2003), serves as a baseline. It was trained in an off-policy and batch manner using random trajectories, and this algorithm provide competitive results among the state of the art (Li et al., 2009a; Chandramohan et al., 2010). Both algorithms provide good results (a positive cumulative reward, which means that the user is generally satisfied after few interactions). However, one can observe that the bonus-greedy scheme provides faster convergence as well as better and more stable policies than the uninformed ϵ-greedy policy. Moreover, results for the informed KTD-SARSA are very close to LSPI after few learning episodes. Therefore, KTD-SARSA is sample efficient (it provides good policies while the insufficient number of transitions prevents from using LSPI because of numerical stability problems), and the provided uncertainty information is useful on this dialogue-management task.

7. Conclusion

In this paper, we have shown how an uncertainty information about estimated values can be derived from KTD. We have also introduced an active learning scheme aiming at improving speed of convergence by sampling actions according to their relative uncertainty, as well as some adaptations of existing schemes for exploration/exploitation. Three experiments have been proposed. The first one shown that KTD-Q, a second-order value-iteration-like algorithm, is sample efficient. The improvement gained by using the proposed active learning scheme was also demonstrated. The proposed schemes for exploration/exploitation were also successfully experimented on a bandit problem and the bonus-greedy policy on real-world problem. This is a first step toward combining the dilemma between exploration and exploitation with value function approximation.

The next step is to adapt more existing approaches dealing with the exploration/exploitation dilemma designed for tabular representation of the value function to the KTD framework, and to provide some theoretical guarantees for the proposed approaches. This paper focused on model-free reinforcement learning, and we plan to compare our approach to model-based RL approaches.

Acknowledgments

The authors thank the European Community (FP7/2007-2013, grant agreement 216594, CLASSiC project : www.classic-project.org) and the Région Lorraine for financial support.

References

S. Chandramohan, M. Geist, and O. Pietquin. Sparse Approximate Dynamic Programming for Dialog Management. In *Proceedings of the 11th SIGDial Conference on Discourse and Dialogue*, pages 107–115, Tokyo (Japan), September 2010. ACL.

R. Dearden, N. Friedman, and S. J. Russell. Bayesian Q-Learning. In *AAAI/IAAI*, pages 761–768, 1998.

Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University, April 2005.

M. Geist. *Optimisation des chaînes de production dans l'industrie sidérurgique : une approche statistique de l'apprentissage par renforcement*. Phd thesis in mathematics, Université Paul Verlaine de Metz (en collaboration avec Supélec, ArcelorMittal et l'INRIA), Novembre 2009.

M. Geist and O. Pietquin. Kalman Temporal Differences. *Journal of Artificial Intelligence Research (JAIR)*, 2010.

M. Geist, O. Pietquin, and G. Fricout. Kalman Temporal Differences: the deterministic case. In *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, Nashville, TN, USA, April 2009a.

M. Geist, O. Pietquin, and G. Fricout. Tracking in reinforcement learning. In *International Conference on Neural Information Processing (ICONIP 2009)*, Bangkok (Thailand), December 2009b. Springer.

N. Jong and P. Stone. Model-Based Exploration in Continuous State Spaces. In *Symposium on Abstraction, Reformulation, and Approximation*, July 2007.

S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

L. P. Kaelbling. *Learning in embedded systems*. MIT Press, 1993.

S. Kakade, M. J. Kearns, and J. Langford. Exploration in Metric State Spaces. In *International Conference on Machine Learning (ICML 03)*, pages 306–312, 2003.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

J. Z. Kolter and A. Y. Ng. Near-Bayesian Exploration in Polynomial Time. In *international conference on Machine learning (ICML 09)*, New York, NY, USA, 2009. ACM.

M. G. Lagoudakis and R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

S. Larsson and D. R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 2000.

O. Lemon, K. Georgila, J. Henderson, and M. Stuttle. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In *Meeting of the European chapter of the Associaton for Computational Linguistics (EACL'06)*, Morristown, NJ, USA, 2006.

E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, 2000.

L. Li, S. Balakrishnan, and J. Williams. Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection. In *Proceedings of the International Conference on Speech Communication and Technologies (InterSpeech'09)*, Brighton (UK), 2009a.

L. Li, M. Littman, and C. Mansley. Online exploration in least-squares policy iteration. In *Conference for research in autonomous agents and multi-agent systems (AAMAS-09)*, Budapest, Hungary, 2009b.

O. Pietquin and T. Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):589–599, March 2006.

Y. Sakaguchi and M. Takano. Reliability of internal prediction/estimation and its application: I. adaptive action selection reflecting reliability of value function. *Neural Networks*, 17(7):935–952, 2004.

S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement learning for spoken dialogue systems. In *Conference on Neural Information Processing Society (NIPS'99), Denver, USA*. Springer, 1999.

A. L. Strehl and M. L. Littman. An Analysis of Model-Based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 2006.

M. Strens. A Bayesian Framework for Reinforcement Learning. In *International Conference on Machine Learning*, pages 943–950. Morgan Kaufmann, San Francisco, CA, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1996.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of two samples. *Biometrika*, 3/4(25):285–294, 1933.

H. Yu and D. P. Bertsekas. Q-Learning Algorithms for Optimal Stopping Based on Least Squares. In *European Control Conference*, Kos, Greece, 2007.

JMLR: Workshop and Conference Proceedings 16:169–181, 2011 Active Learning Challenge

Inspecting Sample Reusability for Active Learning

Katrin Tomanek KATRIN.TOMANEK@UNI-JENA.DE
Jena University Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena, Germany

Katharina Morik KATHARINA.MORIK@TU-DORTMUND.DE
Department of Computer Science - Artificial Intelligence Group
Technical University of Dortmund, Germany

Editor: I. Guyon, G. Cawley, G. Dror, V. Lemaire, and A. Statnikov

Abstract

Active Learning (AL) exploits a learning algorithm to selectively sample examples which are expected to be highly useful for model learning. The resulting sample is governed by a sampling selection bias. While a bias towards useful examples is desirable, there is also a bias towards the learner applied during AL selection. This paper addresses sample reusability, i.e., the question whether and under which conditions samples selected by AL using one learning algorithm are well-suited as training data for another learning algorithm.

Our empirical investigation on general classification problems as well as the natural language processing subtask of Named Entity Recognition shows that many intuitive assumptions on reusability characteristics do not hold. For example, using the same algorithm during AL selection (called selector) and for inducing the final model (called consumer) is not always the optimal choice. We investigate several putatively explanatory factors for sample reusability. One finding is that the suitability of certain selector-consumer pairings cannot be estimated independently of the actual learning problem.

Keywords: active learning, uncertainty sampling, sample selection bias, covariate shift

1. Introduction

While supervised machine learning methods are de-facto standards for a variety of real-world problems, their greediness for large amounts of labeled training data is one of the major obstacles along the path to applications. Training data are usually not available in real-world applications. Human experts of the specific domain in focus need to create such labeled examples which is extremely costly. This holds, for example, for a range of natural language processing (NLP) tasks where (parts of) natural language text need to be classified. For the creation of training data the human annotator has to read through a set of (usually randomly selected) textual examples and manually assigns the corresponding categories to the constituent of interest (e.g., words). Such annotation is

costly and usually requires domain experts, for example when biomedical publications are to be annotated.

Active learning (AL) tackles the challenge of economic training data creation. In the AL paradigm, only examples of high utility for classifier training are selected for manual annotation in an iterative manner. AL has been shown to be a promising solution to annotation cost reduction, especially for scenarios where large amounts of unlabeled data are available at no or relatively low costs. Since AL exploits a learning algorithm to selectively sample examples which are expected to be highly useful for model learning, the resulting sample is governed by a sampling selection bias, also known as covariate shift. While a bias towards useful examples is generally intended and desirable, one must also keep in mind that utility is assessed with respect to the learner applied during AL selection so that the resulting sample is somewhat biased to this particular learner.

Approaches to AL are typically based on the assumption that the learning algorithm used during selection – called *selector* – and the learning algorithm used to induce the final model – called *consumer* – to are identical. Yet, there are settings where selector and consumer intentionally diverge. Firstly, the interaction with the annotating expert demands a fast learning algorithm embedded in AL. Hence, a less complex learner might be used as selector, while the final consumer remains the high-accuracy, more complex learning algorithm. Secondly, the optimal learner for a new problem is often unknown during data acquisition so that the selector is likely to differ from the final consumer. Thirdly, we may want to annotate just once and use the example set for many different learning problems. We call settings, where selector and consumer diverge *foreign-selection* (in contrast to *self-selection* as the default setting). Foreign-selection constitutes a scenario of AL *sample reuse*. We say that a sample S_{AL} obtained by AL is *reusable* by a particular consumer, if this consumer yields a higher classification accuracy when trained on S_{AL} than it would (on average) achieve when trained on a random sample S_{RD}.

The question is, whether and under which conditions a sample selected with AL exploiting a specific learner is suitable (i.e., reusable) for training another learning algorithm, or – more generally – how sampling efficiency of AL is affected by foreign-selection settings. To the best of our knowledge, this question has neither been posed nor studied in context of AL before. Most research in AL is restricted to a self-selection scenario. Despite its practical importance, the reusability issue has not yet been investigated.

For our investigation of reusability we state a set of hypotheses on a) expected reusability characteristics of specific foreign-selection scenarios and b) relevant factors assumed to influence reusability in foreign-selection scenarios. These hypotheses are empirically tested on several general classification problems as available from the UCI repository as well as the NLP task of Named Entity Recognition (NER) which is a well acknowledged prerequisite for tailored information services and so an inherently realistic application scenario of AL (Tomanek et al., 2007).

The rest of this paper is structured as follows: Section 2 motivates the hypotheses we aim to test. Section 3 then describes our experimental setting, including data sets, learning algorithms, AL approaches, and a novel measure for reusability. Results are reported and discussed in Section 4. Related work is discussed in Section 5 and Section 6 concludes.

2. Hypotheses

Expected reusability characteristics The first two hypotheses state what seems to be common-sense:

- **H1:** Samples obtained by AL with a particular selector are rather *unlikely to be reusable* by another learning algorithm due to adversarial ties to the selector.
- **H2:** For a particular consumer, self-selection constitutes the *upper bound* for AL sampling efficiency.

Expected factors influencing reusability These hypotheses cover four factors which possibly influence the reusability characteristics of a specific scenario.

- **H3:** Are there selector-consumer pairings exhibiting *general reusability characteristics* which hold for most learning problems? H3 states, that there are selectors which are in general well suited for certain consumers, and vice-versa.
- **H4:** Since the selector classifies examples according to its model, the similarity of the consumer's and the selector's model could determine reusability. H4 states that a high degree of *model similarity or model relatedness* leads to high reusability.
- **H5:** Since the resulting selection is what counts, H5 states that the *similarity of samples* chosen by self-selection and foreign-selection is important for reusability.
- **H6:** Since the example input space is changed by a learner's feature weights, H6 states that the *similarity of the feature ranking* in self- and foreign-selection is important for reusability.

3. Experimental Setup

This section outlines the experimental setup used for empirically investigating the hypotheses on AL sample reuse.

UCI data sets

data set	# examples	# attributes	attribute types	classes
CAR	1,728	6	nominal	4
MUSHROOM	8,124	22	nominal	2
NURSERY	12,960	8	nominal	5
SEGMENT	2,310	19	real	7
SICK	3,772	30	mixed	2

NER data sets

data set	# examples	# attributes	attribute types	classes
MUC7	3,022	≈ 50K	binary	8
PBGENE	10,570	≈ 50K	binary	4

Table 1: Data sets used for AL sample reuse experiments. For NER, *examples* refers to the number of sentences contained in the respective data set.

3.1. Learning Problems and Data

General classification problems as well as the NER learning problem are chosen. The data sets are chosen such that results are reproducible and comparable to previous work. Within the UCI repository (Asuncion and Newman, 2007), five data sets were selected according to *(a)* size (data sets should have more than 1,000 examples so that AL can actually select), and *(b)* diversity (data sets should contribute different numbers of features and example/feature ratios as well as different numbers of target classes). As for NER, we chose the MUC7 and the PBGENE corpus. Both corpora consist of natural language sentences annotated with respect to the particular entity classes of interest.[1] Table 1 gives an overview of the selected data sets and corpora.

3.2. Learning Algorithms

For experiments on the UCI data sets, we chose four well-known learning algorithms: Naïve Bayes (NB), Multinomial Logistic Regression (MaxEnt), C4.5 Decision Trees (DT), and linear kernel Support Vector Machines (SVM). The respective implementations of these algorithms in the WEKA toolkit are used with their default parameters (Witten and Frank, 2005). For NER experiments, we applied the following algorithms: Conditional Random Fields (CRF), MaxEnt, NB, Hidden Markov Models (HMM), and SVMs. We used standard features for NER (Nadeau and Sekine, 2007).

3.3. Active Learning and Utility Measures

In the scenario inspected here, the expert is in the loop of AL. This requires a fast processing of AL. Fast utility estimates come along with the price of possibly not finding a globally optimal sample. Statistically optimal approaches to AL (such as in Cohn

1. MUC7 (see http://www.ldc.upenn.edu/Catalog) has 7 entity types; PBGENE is a sub-corpus derived from the PENNBIOIE corpus (see http://bioie.ldc.upenn.edu/) and has 3 gene entity types.

Algorithm 1: Greedy Active Learning

input :
$\mathcal{L}$: set of labeled examples $l = (x, y) \in \mathcal{X} \times \mathcal{Y}$;
$\mathcal{P}$: set of unlabeled examples $p = (x) \in \mathcal{X}$;
$T(\mathcal{L})$: a learning algorithm;
$u(p, \theta)$: utility function;

repeat
learn model: $\theta = T(\mathcal{L})$;
select $p^* = \text{argmax}_{p' \in \mathcal{P}} u(p', \theta)$;
query label y for p^*: $l^* = (x, y)$;
$\mathcal{L} = \mathcal{L} \cup l^*, \mathcal{P} = \mathcal{P} \setminus p^*$;
until *stopping criterion met*;
return $\mathcal{L}^* = \mathcal{L}$

et al. (1996) or Roy and McCallum (2001)) usually require model retraining for each unlabeled example to be tested in each AL iteration. In contrast, Uncertainty Sampling (Lewis and Gale, 1994) requires only one model training step in each AL iteration. Uncertainty Sampling correlates utility with model confidence: the utility of an example is based on the uncertainty (as the inverse of the confidence) of the current classifier in its prediction. For our experiments we thus decided for Uncertainty Sampling instead of statistically optimal approaches to fit the practical requirement of low selection times when an (annotation) expert is in the loop.

In each iteration, AL greedily selects example $p = (x)$ with the highest utility score $u(p, \theta)$ which is based on the current model θ. Such a locally optimal selection depending on the history of previous selections is performed with the hope that it will lead to a good global solution. The true class label y for a selected example is queried from a human expert and the labeled example is then added to the training set $\mathcal{L}$ and the next AL iteration starts. After stopping, the sample $\mathcal{L}^*$ containing all labeled examples is returned. This sample is then used to train the final model. Algorithm 1 formalizes this procedure. When applied on the UCI data sets, we indeed only selected one example per AL iteration. Applied to the more complex learning problem of NER, we modify Algorithm 1 so that $b > 1$ examples with the highest utility scores are selected. This aims at keeping selection time low.

For AL with a NB and a MaxEnt-based selector, the confidence is estimated as the margin between the best and the second best label. The margin utility function (Scheffer and Wrobel, 2001) is given by:

$$u_{\text{MA}}(p, \theta) = 1 - \left(\max_{y' \in \mathcal{Y}} P_\theta(y'|x) - \max_{\substack{y'' \in \mathcal{Y} \\ y' \neq y''}} P_\theta(y''|x) \right) \quad (1)$$

For maximum margin classification, the decision value $d(\text{x}) = \langle \boldsymbol{w}, \text{x} \rangle + b$ indicates the distance of an example to the hyperplane. Larger distances can be interpreted as higher confidence of the classifier in its classification. For the SVM-based selector, the margin

utility function is accordingly defined

$$u_{\text{SVM}}(p, \boldsymbol{w}, b) = -(d_{y^*}(\mathrm{x}) - d_{y^{**}}(\mathrm{x})) \tag{2}$$

with $y^* = \text{argmax}_{y' \in \mathcal{Y}} d_{y'}(\mathrm{x})$ and $y^{**} = \text{argmax}_{\substack{y'' \in \mathcal{Y} \\ y' \neq y''}} d_{y''}(\mathrm{x})$. Due to the well-known instability of decision trees, Uncertainty Sampling should not be applied (Dwyer and Holte, 2007). Instead, a variant of AL known as *Query-by-Committee* (Seung et al., 1992) is promising. The utility of an example is derived from the disagreement within a committee of classifier models $C = (\theta_1, \ldots, \theta_c)$. In the experiments, committees with $|C| = 3$ and member θ_i are trained on a subsample $\mathcal{L}'$ of the available training data $\mathcal{L}$ with $|\mathcal{L}'| = \frac{|C|-1}{|C|}|\mathcal{L}|$. The Vote Entropy utility function quantifies disagreement (Engelson and Dagan, 1996)

$$u_{\text{VE}}(p, C) = -\sum_{y' \in \mathcal{Y}} \frac{V(y', x)}{|C|} \log \frac{V(y', x)}{|C|} \tag{3}$$

where $V(y', x)$ is the number of committee members θ_i predicting class y'.

As for the NER learning problems, the example grain size is set to complete sentences. However, since sentences consist of single tokens, we calculate the utility scores for each token separately and then average over all tokens of a sentence to get the sentence-level utility score.[2] Moreover, for the NER learning problems, we apply batch-mode AL where $b = 20$ sentences are selected in each AL iteration. Batch-mode AL is here applied to reduce computational complexity of AL because model training for NER is rather complex due to the high-dimensional feature space ($\approx$ 50,000 in this case).

In all experiments, the data sets described above where each split into a pool of AL selected (90%), and a held-out test set (10%) used to calculate learning curves. The results reported in the following are averages over 20 independent AL runs. For each run, another random split was generated. All experiments are based on the same 20 splits. AL runs were stopped once $|\mathcal{L}| = 150$ was reached (UCI), or once $\mathcal{L}$ consisted of 50,000 tokens (NER).

3.4. Quantification of Sample Reusability

To quantify sample reusability on a continuous scale we introduce a novel measure based on the Area Under the learning Curve (AUC). For a baseline sampling scenario S_{base} (usually random sampling), the learning curve of AL self-selection S_{self}, and that of AL foreign-selection S_{frgn}, the REU score is given by

$$\text{REU}(S_{\text{frgn}}, S_{\text{self}}, S_{\text{base}}, a, b) = \frac{\text{AUC}(S_{\text{frgn}}, a, b) - \text{AUC}(S_{\text{base}}, a, b)}{\text{AUC}(S_{\text{self}}, a, b) - \text{AUC}(S_{\text{base}}, a, b)} - 1 \tag{4}$$

2. Note that while CRFs and HMMs are actually used to model the sentence as a sequence of tokens $\boldsymbol{x} = (x_1, \ldots, x_n)$, we still calculated the utility score as an average over all token-level utility scores. The token-level score is based on the marginal probability at position i for a sequence $\boldsymbol{x}$. See e.g. (Tomanek and Hahn, 2009) for details.

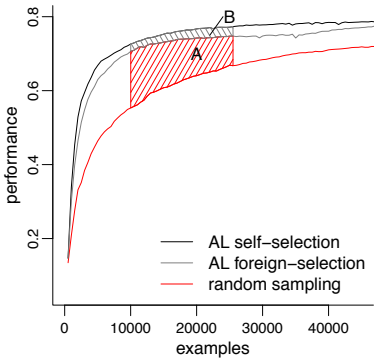


Figure 1: Quantification of sample reusability through the REU score which is here calculated by $\frac{A}{A+B} - 1$. In this example, REU = −0.17 indicates good reusability.

on an interval $[a,b]$ in the learning curve. This score indicates the percentage decrease of AL self-selection sampling efficiency by foreign-selection relative to the baseline sampling scenario, when compared to self-selection. If REU = 0, foreign- and self-selection are equally efficient and in the case of REU > 0, foreign-selection would be even better than self-selection. A negative score with $-1 \ll \text{REU} < 0$ indicates that reusability is in evidence but foreign-selection is less efficient than self-selection. Further, we say that reusability is "high" for negative REU scores close to 0, "low" for negative REU scores beyond or just slightly above −1. If REU < −1 we say that reusability is not given since foreign-selection is worse than random selection. Figure 1 visualizes the calculation of the REU score.

Note that a learning curve shows model performance as a function of data acquisition cost. Data acquisition cost may be application-specific. Obviously, the interval $[a,b]$ of the REU score must be chosen on the appropriate unit. As for the UCI data sets, we assume a unit cost per example and set the interval for REU score calculation to $[50;150]$ so as to exclude the "start-up" phase of AL where the learning curves are naturally very steep as well as to exclude the "convergence-phase" where learning curves usually flatten out. As for the NER scenario where complete sentences are selected, we say that the cost is a function of the sentence length (measured in number of tokens contained). We set the interval for REU score calculation to $[10000;30000]$ – again, in order to exclude start-up and convergence phase.

4. Results

The REU scores shown in Tables 2 and 3 are used to discuss the hypotheses.

4.1. Hypothesis H1

As for NER, reusability can be recorded for all AL foreign-selection scenarios and REU scores rarely fall below −0.5, indicating a high degree of reusability for this special learning problem. The only exception is that of foreign-selection for a NB-based

car	consumer				mushroom	consumer			
selector	DT	MaxEnt	NB	SVM	selector	DT	MaxEnt	NB	SVM
DT	0.00	−0.30	−0.47	−0.42	DT	0.00	−0.59	−0.69	−0.19
MaxEnt	−0.84	0.00	−1.04	−0.27	MaxEnt	−0.05	0.00	−0.47	0.12
NB	−0.84	−0.11	0.00	−0.28	NB	0.00	−0.07	0.00	−0.02
SVM	−0.90	0.26	−0.86	0.00	SVM	−0.91	−0.43	−1.17	0.00

nursery	consumer				segment	consumer			
selector	DT	MaxEnt	NB	SVM	selector	DT	MaxEnt	NB	SVM
DT	0.00	−1.13	−2.13	−0.93	DT	0.00	−0.72	−0.47	−0.95
MaxEnt	−0.33	0.00	−1.46	−0.07	MaxEnt	−0.95	0.00	−1.24	−3.07
NB	0.48	−0.09	0.00	0.14	NB	−2.56	−2.04	0.00	−1.77
SVM	−0.36	−0.35	−1.19	0.00	SVM	−4.35	−3.53	−2.39	0.00

sick	consumer			
selector	DT	MaxEnt	NB	SVM
DT	0.00	−0.83	−0.54	−0.90
MaxEnt	0.26	0.00	−0.43	−0.77
NB	0.31	−0.90	0.00	−2.18
SVM	0.34	−0.18	−0.25	0.00

Table 2: Reusability scores (REU) on UCI data sets. Colors: REU ≥ 0 and REU ≤ -1

Muc7 (NER)

	consumer				
selector	NB	HMM	MaxEnt	SVM	CRF
NB	0.00	0.07	−0.19	0.13	−0.15
HMM	−0.48	0.00	−0.40	−0.29	−0.39
MaxEnt	−0.39	−0.05	0.00	0.12	−0.12
SVM	−0.40	−0.07	−0.20	0.00	−0.24
CRF	−0.38	0.01	0.02	0.05	0.00

PBgene (NER)

	consumer				
selector	NB	HMM	MaxEnt	SVM	CRF
NB	0.00	−0.02	−0.01	−0.17	−0.13
HMM	−1.51	0.00	−0.29	−0.38	−0.35
MaxEnt	−3.47	−0.57	0.00	−0.08	−0.09
SVM	−2.22	−0.24	−0.22	0.00	−0.25
CRF	−3.58	−0.58	−0.06	−0.36	0.00

Table 3: Reusability scores on NER data sets. Colors: REU ≥ 0 and REU ≤ -1

consumer on the PBgene corpus. On the UCI data sets, in only 15 out of 60 foreign-selection scenarios, sample efficiency considerably drops below that of random selection with REU scores ≤ -1. Moreover, there are only two cases (both on the segment data set) where a sample actively selected by a specific selector is not reusable in by another consumer.

The hypothesis that reusability were a rare scenario (H1) is thus rejected. In contrast, reusability is observed in the majority of cases.

4.2. Hypothesis H2

Most surprising, the results reveal that self-selection sampling efficiency is occasionally outperformed by foreign-selection. As for NER, this is the case in 6 out of the 40 foreign-selection scenarios; for the UCI data sets, 8 out of the 60 foreign-selection scenarios exceed the assumed upper bound. Look, for instance, at the combination of an SVM selector and a MaxEnt consumer processing the car data set. This falsifies the upper-bound hypothesis leading to the remarkable finding that there are scenarios where a learner T_2 estimates the utility of an example for learner T_1 more appropriately than T_1 itself.

4.3. Hypothesis H3

The shown REU scores also contradict the assumption that there are certain pairings of learning algorithms for which general reusability characteristics hold. Inconsistent reusability characteristics for the selector-consumer pairings have to be ascertained over the five UCI data sets. NB, for example, is a good selector for the MaxEnt consumer on some data sets (mushroom, nursery and car), but not on others (sick and segment).

4.4. Hypothesis H4

Additional experiments test whether model similarity explains reusability. In line with Baldridge and Osborne (2004), H4 states that sample reusability depends on the degree of relatedness between selector and consumer regarding their *models*. Relatedness is usually measured by the degree of correlation between the *predictions* of models. However, in the context of AL, more interesting than the consistency of predictions is how similar the utility rankings of the unlabeled examples achieved with two different models are. This is because these rankings determine which examples are selected. For use in our AL scenario, we thus say that two models are related when they lead to a highly correlated utility ranking of unlabeled examples.

The Spearman's rank correlation coefficient r_S compares the utility rankings of two models (Baldridge and Osborne, 2004). We trained all learners on random samples (of 10,000 tokens in case of NER, and on 150 examples in case of the UCI data sets).[3] Utility rankings of the examples in the test set are then compared for all tuples of models.

3. We also tested random samples of different sizes but did not obtain essentially different results.

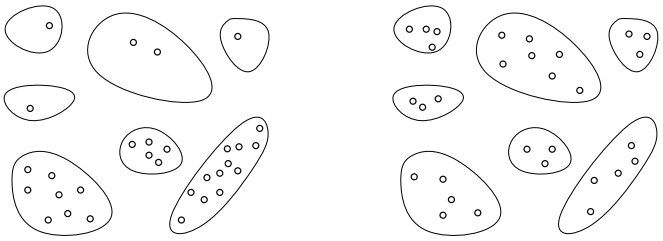


Figure 2: Distribution of samples S_1 (left) and S_2 (right) over common clustering.

Hypothesis H4 is operationalized by the assumption that relatedness scores positively correlate with reusability scores. Table 4 shows correlation coefficients between the REU score and model similarity (relatedness of models). It indicates that there is either no or even a negative correlation between reusability and model similarity. By definition, relatedness scores are symmetrical. However, reusability, according to Table 2, is not. When the learners of selector and consumer are exchanged, reuse scores differ. Looking at the relatedness scores (omitted here due to space limitations), we observe many such cases. As an example, consider the pair of a SVM and a MaxEnt learner for which we obtained a high relatedness score on the PBgene corpus. A sample obtained by AL with a MaxEnt-based selector is perfectly reusable by an SVM-based consumer. However, when SVM is used to select for a MaxEnt consumer, reusability drops to a REU score of −0.22 (cf. Table 2). As another example, the very good reusability of a sample obtained by AL with a NB selector for a HMM consumer is in contrast to the rather low relatedness score for HMM and NB on the Muc7 corpus. A low relatedness score thus does not necessarily imply a low level of reusability. While a high rank correlation coefficient often accompanies reusability (as for the MaxEnt-CRF tuple), one cannot conclude the opposite from low correlation coefficients. This emphasizes that different samples can also lead to similar model performances.

4.5. Hypothesis H5

H5 hypothesizes that the similarity of samples obtained in self- and foreign-selection mode is a relevant factor for reusability. Different selectors may select from other parts of the instance space. The more the covered space of a foreign-selector diverges from that of the self-selector, the lower the REU scores are according to our assumption. A situation with REU ≤ -1 would then mean that the AL sample does not cover the relevant areas for the consumer. Comparing the sample distributions over the input space is performed by agglomerativ clustering over all unlabeled examples in the pool $\mathcal{P}$. The distance between two clusters is calculated according to the average linkage method based on the Euclidean distance. The hierarchical clustering is flattened down to $k = 20$ clusters. The examples in a sample S are then assigned to clusters in this clustering according to an example's proximity to a cluster centroid (Everitt et al., 2001).

D_S represents the distributions of the examples of sample S over the clustered input space. This distribution gives the percentage of a sample's examples falling in each cluster. Figure 2 visualizes this for two samples S_1 and S_2 obtained by two dif-

correlation of reusability and model similarity

	CAR	MUSHROOM	NURSERY	SEGMENT	SICK
r_P	0.04	-0.31	0.07	-0.22	-0.47
r_S	0.02	-0.31	0.01	-0.39	-0.24

correlation of reusability and sample similarity

	CAR	MUSHROOM	NURSERY	SEGMENT	SICK
r_P	0.30	0.24	0.03	0.37	0.23
r_S	0.29	0.19	0.08	0.40	0.14

correlation of reusability and feature ranking similarity

	CAR	MUSHROOM	NURSERY	SEGMENT	SICK
r_P	0.16	-0.35	0.46	-0.38	-0.06
r_S	0.04	-0.49	0.45	-0.42	-0.21

Table 4: Pearson's (r_P) and Spearman's (r_S) correlation coefficients for REU score and other variables (model similarity, sample similarity, and feature ranking similarity).

ferent selectors. The similarity of the two samples S_1 and S_2 is estimated based on the divergence of their distributions D_{S_1} and D_{S_2} which is calculated by the Jensen-Shannon divergence (JSD). The JSD score ranges in the interval of $[0,1]$; lower scores indicate higher distributional similarity. The similarity is calculated by $\mathrm{SIM}(S_1,S_2) = 1-\mathrm{JSD}(D_{S_1},D_{S_2})$. In the above example, a similarity of $\mathrm{SIM}(S_1,S_2) = 0.48$ is obtained.

Now, H5 becomes the testable statement that similarity (SIM) correlates with reusability (REU). Table 4 shows correlation coefficients between reusability and sample similarity. Pearson's correlation coefficients range between 0.03 and 0.37, which indicates a comparatively low (linear) relationship. Spearman's correlation coefficients are also very low on average ranging from 0.08 to 0.4. SIM scores are symmetrical, where reusability is not. These experiments show that the distributional similarity of samples does not explain reusability.

4.6. Hypothesis H6

Hypothesis H6 states that feature weighting is a relevant factor for reusability. This sensitivity can be expressed by comparing feature rankings obtained from foreign-selection and from self-selection. Feature rankings are obtained by a wrapper approach based on simple hill climbing (Kohavi and John, 1997). Subsequently, tuples of feature rankings are compared. A tuple always consists of the feature ranking obtained from a model learned on a foreign-selection sample and the feature ranking of a model learned on the self-selected sample. Comparison of feature rankings is based on a weighted

version of Spearman's rank correlation coefficient.[4] Accordingly, the *feature ranking score* $\mathrm{FR}(S_{T_1}, S_{T_2})$ shows the correlation of the feature rankings of a model induced by learner T_2 on a foreign-selection sample from AL with a selector based on T_1 and a self-selection sample where the selector was based on T_2.

Now, H6 means that the FR scores correlate highly with the REU scores in the foreign-selection scenarios. However, the experiments disprove this assumption. Table 4 shows correlation coefficients between reusability and feature ranking similarity (FR score). Correlation coefficients are mostly low or even negative. Overall, this outcome shows that the FR score is inadequate for predicting the REU score (Pearson's coefficient) as well as for ranking the selectors according to their appropriateness for a particular consumer (Spearman's coefficient). A twisted feature ranking may still lead to a model with similar accuracy compared to a model which is induced from a self-selected sample.

Reusability cannot be explained by the fact that models learned on different samples exhibit similar feature rankings. Note, a model θ induced from a foreign-selection sample may perform similarly well or even better than a model θ' induced by the same learner but from a self-selection sample.

5. Previous Work

While there is a huge body of work on AL for the self-selection scenario (see Settles (2009) for an overview), there is only little on scenarios of AL sample reuse and foreign-selection. A scenario of sample reuse motivated by the need to reduce the computational complexity of sampling was first described by Lewis and Catlett (1994) for a text classification problem. There, the consumer was based on decision trees and the selector was a logistic regression algorithm. Positive findings about sample reusability were reported. For the NLP task of statistical parsing, controversial findings on reusability have been published. Hwa (2001) reported positive results, Baldridge and Osborne (2004), on the other hand, presented and discussed scenarios where the AL foreign-selection bias considerably impairs reusability.

Previous work on AL sample reuse addresses AL sample reuse only in very specific scenarios. There is to-date no comprehensive study of the true nature of reusability, requirements for the presence of reusability, or prohibitive factors. To the best of our knowledge, this paper is the first approach in this direction.

Under a more general consideration and without explicit reference to AL, Fan et al. (2005) study how sensitive learning algorithms are in general to sample selection bias. A learner is called *local* if it is invariant to this bias, and *global* otherwise. They found that all learning algorithms can be both local and global depending on the combination of the data set, modeling assumptions made by the learner, and the learner's appropriateness to model the particular data set. This observation fits to the findings presented in this paper.

4. A weighted rank correlation is simply calculated based on weighted covariance. The weights are assigned inverse to their ranks.

6. Summary and Conclusions

This paper describes the problem of sample reusability in context of AL foreign-selection scenarios. Several hypotheses on reusability characteristics and explanatory factors for reusability are empirically investigated. Experiments were performed both on general classification problems and on the NER task which constitutes a special class of learning problems.

Based on the results of our experiments we have to reject the dominant self-selection assumptions (H1, H2). In particular for the NER learning problem, reusability is evident in all practical scenarios. Self-selection does not constitute the upper bound of sampling efficiency but can sometimes be outperformed by foreign-selection. None of the assumed influencing factors – *viz.* model similarity, sample similarity, and similarity of the feature ranking – were supported by our experiments. Reusability could even be observed when all these assumptions were violated. Most importantly, experiments showed that one cannot generalize which combinations of learners generally work well together in settings of AL foreign-selection. Hence, whether reusability is in evidence for a particular selector-consumer pairing appears to depend on the combination of learning problem, data set, and appropriateness of the particular learning algorithm.

Overall, our study points out that reusability is a relevant and challenging problem. Future work in this direction should focus on the quantification of a learner's sensitivity to sample selection bias given a specific learning problem in order to estimate – ideally prior to AL sample selection – whether a sample obtained by AL and a specific selector may be reusable by (which?) consumers. Our measure to quantify sample reusability is ready to use for such further investigations.

References

Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

Jason Baldridge and Miles Osborne. Active learning and the total cost of annotation. In *Proc. of EMNLP'04*, pages 9–16, 2004.

David Cohn, Zoubin Ghahramani, and Michael Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

Kenneth Dwyer and Robert Holte. Decision tree instability and active learning. In *Proc. of ECML'07*, pages 128–139, 2007.

Sean Engelson and Ido Dagan. Minimizing manual annotation cost in supervised training from corpora. In *Proc. of ACL'96*, pages 319–326, 1996.

Brian Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Wiley, 4th edition, 2001.

Wei Fan, Ian Davidson, Bianca Zadrozny, and Philip Yu. An improved categorization of classifier's sensitivity on sample selection bias. In *Proc. of ICDM'05*, pages 605–608, 2005.

Rebecca Hwa. On minimizing training corpus for parser acquisition. In *Proc. of ConLL'01*, pages 1–6, 2001.

Ron Kohavi and George John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

David Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proc. of ICML'94*, pages 148–156, 1994.

David Lewis and William Gale. A sequential algorithm for training text classifiers. In *Proc. of SIGIR'94*, pages 3–12, 1994.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, 2007.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. of ICML'01*, pages 441–448, 2001.

Tobias Scheffer and Stefan Wrobel. Active learning of partially hidden markov models. In *Proc. of the ECML/PKDD Workshop on Instance Selection*, 2001.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proc. of COLT'92*, pages 287–294, 1992.

Katrin Tomanek and Udo Hahn. Semi-supervised active learning for sequence labeling. In *Proc. of ACL/IJCNLP'09*, pages 1039–1047, 2009.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. An approach to text corpus construction which cuts annotation costs and maintains corpus reusability of annotated data. In *Proc. of EMNLP-CoNLL'07*, pages 486–495, 2007.

Ian Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

Appendix A
Datasets of the Active Learning Challenge

Report prepared by Isabelle Guyon with information from the data donors listed below:

Chemo-informatics (HIVA and C datasets): The National Cancer Institute (USA) provide the data used in the HIVA dataset. Charles Bergeron, Kristin Bennett and Curt Breneman (Rensselaer Polytechnic Institute, New York) contributed the C dataset.

Handwriting recognition (IBN-SINA and A datasets): Reza Farrahi Moghaddam, Mathias Adankon, Kostyantyn Filonenko, Robert Wisnovsky, and Mohamed Chériet (Ecole de technologie supérieure de Montréal, Quebec) contributed the datasets of Arabic manuscripts, IBN-SINA and A.

Text processing (NOVA and D datasets): - Tom Mitchell (USA) and Ron Bekkerman (Israel) provided the data used in the NOVA and D datasets (known as the Twenty Newsgroups).

Marketing (ORANGE and B datasets): Vincent Lemaire, Marc Boullé, Fabrice Clérot, Raphael Féraud, Aurélie Le Cam, and Pascal Gouzien (Orange, France) contributed the ORANGE and B datasets, previously used in the KDD cup 2009.

Ecology (SYLVA and F datasets): Jock A. Blackard, Denis J. Dean, and Charles W. Anderson (US Forest Service, USA) contributed the data used for the SYLVA and F datasets (Forest cover type).

Embryology (ZEBRA and E datasets): Emmanuel Faure, Thierry Savy, Louise Duloquin, Miguel Luengo Oroz, Benoit Lombardot, Camilo Melani, Paul Bourgine, and Nadine Peyriéras (Institut des systèmes complexes, France) contributed the ZEBRA and E datasets.

Introduction

Two times six datasets from various domains were made available for the Active Learning challenge (plus one toy dataset ALEX for practice purpose). The first six (Table 1) were made available during the development period. This gave the opportunity to the participants to practice without restriction and get performance feed-back on the results of their experiments from the on-line platform. Six other matching datasets (Table 2)

were made available for final testing. Only one experiment could be made with the final datasets to enter the challenge.

Table 1: **Development datasets.** ALEX is a toy dataset given for illustrative purpose. The other datasets match the final datasets by application domain (see text).

Dataset	Domain	Feat. type	Feat. num.	Sparsity (%)	Missing (%)	Pos. lbls (%)	Tr & Te examples
ALEX	Toy	binary	11	0	0	72.98	5000
HIVA	Chemo-informatics	binary	1617	90.88	0	3.52	21339
IBN SINA	Handwriting recog	mixed	92	80.67	0	37.84	10361
NOVA	Text processing	binary	16969	99.67	0	28.45	9733
ORANGE	Marketing	mixed	230	9.57	65.46	1.78	25000
SYLVA	Biology	mixed	216	77.88	0	6.15	72626
ZEBRA	Embryology	continuous	154	0.04	0.0038	4.58	30744

Table 2: **Final test datasets.** The fraction of positive labels was not available to the participants.

Dataset	Domain	Feat. type	Feat. num.	Sparsity (%)	Missing (%)	Pos. lbls (%)	Tr & Te num.
A	Handwriting recog	mixed	92	79.02	0	13.35	17535
B	Marketing	mixed	250	46.89	25.76	9.14	25000
C	Chemo-informatics	mixed	851	8.6	0	8.1	25720
D	Text processing	binary	12000	99.67	0	25.52	10000
E	mbryology	continuous	154	0.04	0.0004	9.04	32252
F	Biology	mixed	12	1.02	0	7.58	67628

Data formats

All the data sets are in the same format and include 7 files in text format:

```
dataname.param    % Parameters and statistics about the
                  % data
dataname.data     % Unlabeled data (matrix of space
                  % delimited numbers,  patterns in
                  % lines, features in columns).
dataname.mat      % The same data matrix in Matlab format.
dataname.label    % Target values.
dataname.labelid  % Identity of the labels (variables
                  % that are target values, i.e.,
                  % columns of the label matrix.)
dataname.featid   % Identity of the features (variables
```

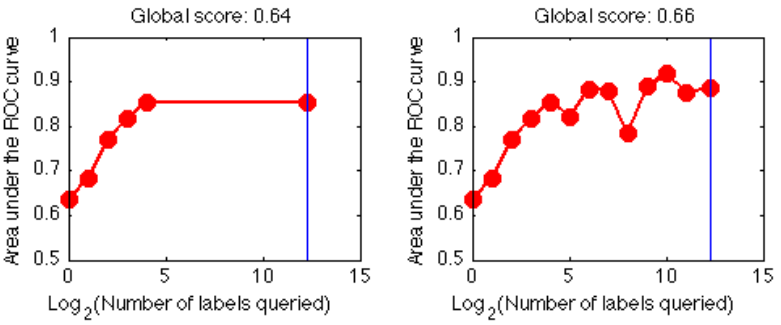
```
                    % that are not target values, i.e.
                    % columns of the data matrix)
dataname.dataid     % Identity of the samples (lines of
                    % the data matrix)
```

The participants used the following formats to send queries and results:

```
dataname.sample     % Sample numbers, one per line. Use to
                    % query labels.
dataname.predict    % Prediction values.
```

All problems were 2-class classification problems. The target classification values are therefore binary labels. All the unlabeled data (training and test data) were available from the outset of the challenge. For each dataset, only one labeled training example was initially provided (called seed example). The rest of the labels for training examples were available for purchase for virtual cash from the challenge platform. The test labels were never disclosed.

The evaluation was performed by computing learning curves from predictions made by the participants: Every time a participant required a set of labels from the platform, he has to turn in predictions of class categories for all the examples. The Area under the ROC curve (AUC), a classical metric used to assess classification performance, was computed for examples not used yet for training (i.e. whose labels were not made available so far to the participant), including unlabeled training examples and test examples. The global scoring metric was the Area under the Learning Curve (ALC), appropriately normalized between 0 and 1: $global\_score = (ALC - Arand)/(Amax - Arand)$ where Arand is the expected value of the ALC for random predictions and Amax is the largest achievable ALC.



(*a*) (*b*)

Figure 1: Examples of learning curves for 5 and 13 experimental points.

A log scale in number of examples used for training was used. The learning curve was interpolated linearly between two experimental points and extrapolated horizontally to the total number of training examples whose labels were available for purchase (see Figure 1).

The datasets were incorporated in the Virtual Lab of the Causality Workbench http://www.causality.inf.ethz.ch/workbench.php. For each dataset, a wrapper was written in object-oriented Matlab to make is available in the GLOP package (Generative Lab Object Package) http://www.causality.inf.ethz.ch//repository.php?id=23

The names of the objects are: `@alex`, `@hiva`, `@ibn_sina`, `@nova`, `@orange`, `@sylva`, `@zebra`, `@A`, `@B`, `@C`, `@D`, `@E`, `@F`. Here is an example of using the GLOB package:

```
L=alex;                              % instantiate an alex
                                     % model
data_profile(L);                     % show the dataset
                                     % profile
label_profile(L);                    % show the profile of
                                     % the labels
task_n_pricing(L);                   % show a description of
                                     % the task
save_profile(L);                     % show the entire dataset
                                     % profile
Qin=query(query_file);               % instantiate a query
[Qout, L]=process_query(L, Qin);     % process query, return
                                     % learning curve
```

In what follows, we present the design of the various datasets and show the learning curves produced either by the organizers using reference methods such as Least Square Support Vector Machines (LSSVM) and Selective Naïve Bayes (SNB) or the overall winners (by average rank over all final evaluation datasets) Ideal Analytics, Intel, using gradient tree boosting. More details on these methods are found in JMLR W&CP volume 15. Note that these are not necessarily the best results. Other results can be viewed on the website of the challenge, which remains open for post-challenge submissions: http://www.causality.inf.ethz.ch/activelearning.php?page=results#cont.

Handwriting recognition: IBN_SINA and A datasets

Topic

The IBN_SINA and A datasets provides a feature representation of Arabic Historical Manuscripts. The letter A is mnemotechnical for Avicenna, the Latin name of the Arab scholar Ibn Sina.

Sources

- **Original owners:** The dataset is prepared on manuscript images provided by The Institute of Islamic Studies (IIS), McGill.

Manuscript author: Abu al-Hasan Ali ibn Abi Ali ibn Muhammad al-Amidi (d. 1243 or 1233).

Manuscript title: Kitab Kashf al-tamwihat fi sharh al-Tanbihat (Commentary on Ibn Sina's al-Isharat wa-al-tanbihat).

Brief description: Among the works of Avicenna, his al-Isharat wa-al-tanbihat received the attention of the later scholars more than others. The reception of this work is particularly intensive and widespread in the period between the late twelfth century to the first half of the fourteenth century, when more than a dozen comprehensive commentaries on this work were composed. These commentaries were one of the main ways of approaching, understanding and developing Avicenna's philosophy and therefore any study of Post-Avicennian philosophy needs to pay specific attention to this commentary tradition. Kashf al-tamwihat fi sharh al-Tanbihat by Abu al-Hasan Ali ibn Abi Ali ibn Muhammad al-Amidi (d. 1243 or 1233), one of the early commentaries written on al-Isharat wa-al-tanbihat, is an unpublished commentary which still await scholars' attention.

- **Donors of the database:** Reza Farrahi Moghaddam, Mathias Adankon, Kostyantyn Filonenko, Robert Wisnovsky, and Mohamed Cheriet.

- **Contact:** Mohamed Cheriet - Synchromedia Laboratory
ETS, Montréal, (QC) Canada H3C 1K3
mohamed.cheriet@etsmtl.ca
Tel: +1(514)396-8972
Fax: +1(514)396-8595

- **Date received:** November 2009.

Reference

Reza Farrahi Moghaddam, Mohamed Cheriet, Mathias M. Adankon, Kostyantyn Filonenko, and Robert Wisnovsky. 2010. IBN SINA: a database for research on processing and understanding of Arabic manuscripts images. In Proceedings of the 9th IAPR International Workshop on Document Analysis Systems (DAS '10). ACM, New York, NY, USA, 11-18.

Experimental design

The features were extracted following the procedure described in the JMLR W&CP paper: IBN SINA: A database for handwritten Arabic manuscripts understanding research, by Reza Farrahi Moghaddam, Mathias Adankon, Kostyantyn Filonenko, Robert Wisnovsky, and Mohamed Chériet. The data include 92 numeric features and 15 classes with at least 1000 positive examples. We created a number of binary classification problems for the development and final test datasets:

For IBN_SINA (development dataset), we selected the separation of the class aL vs. the rest.

For A (Avicenna, final evaluation dataset), we created 14 classification problems separating 2 classes vs. the rest. The two classes always included EU:

EU+nL
EU+qL
EU+bL
EU+lL
EU+tL
EU+kL
EU+vL
EU+fL
EU+mL
EU+rL
EU+hL
EU+dL
EU+yL

This allowed us to provide a seed example belonging to the class EU that was an example of the positive class for all 14 problems. We assigned at random a different problem to each participant. In this way we created a trap to catch eventual cheater who would exchange labels. After the challenge was over, we asked the participants to submit results again, this time all of them on the same problem.

Data statistics

See Tables 1 and 2. The samples in dataset A are different from those in IBN_SINA.

Table 4: Variables

Index	Name	Access	Type	Min	Max
0	Target	observable	binary	-1	1
1	Aspect ratio	observable	continuous	0.039409	6.8387
2	Horizontal frequency	observable	categorical	1	12
3	Vertical CM ratio	observable	continuous	-0.94089	9.4262
4	Singular points	observable	categorical	0	24
5	Height ratio	observable	continuous	0.25714	5.8
6	Hole feature	observable	binary	0	1
7	End points	observable	categorical	0	15
8	Dot feature	observable	binary	0	1
9	BP_hole_1	observable	binary	0	1
10	BP_EP_1	observable	binary	0	1
11	BP_BP_1	observable	binary	0	1
12	BP_hole_2	observable	binary	0	1

Continued on next page

Table 4 – continued from previous page

Index	Name	Access	Type	Min	Max
13	BP_EP_2	observable	binary	0	1
14	BP_BP_2	observable	binary	0	1
15	BP_hole_3	observable	binary	0	1
16	BP_EP_3	observable	binary	0	1
17	BP_BP_3	observable	binary	0	1
18	BP_hole_4	observable	binary	0	1
19	BP_EP_4	observable	binary	0	1
20	BP_BP_4	observable	binary	0	1
21	BP_hole_5	observable	binary	0	1
22	BP_EP_5	observable	binary	0	1
23	BP_BP_5	observable	binary	0	1
24	BP_hole_6	observable	binary	0	1
25	BP_EP_6	observable	binary	0	1
26	BP_BP_6	observable	binary	0	1
27	EP_BP_1	observable	binary	0	1
28	EP_EP_1	observable	binary	0	1
29	EP_VCM_1	observable	categorical	0	2
30	EP_BP_2	observable	binary	0	1
31	EP_EP_2	observable	binary	0	1
32	EP_VCM_2	observable	categorical	0	2
33	EP_BP_3	observable	binary	0	1
34	EP_EP_3	observable	binary	0	1
35	EP_VCM_3	observable	categorical	0	2
36	EP_BP_4	observable	binary	0	1
37	EP_EP_4	observable	binary	0	1
38	EP_VCM_4	observable	categorical	0	2
39	EP_BP_5	observable	binary	0	1
40	EP_EP_5	observable	binary	0	1
41	EP_VCM_5	observable	categorical	0	2
42	EP_BP_6	observable	binary	0	1
43	EP_EP_6	observable	binary	0	1
44	EP_VCM_6	observable	categorical	0	2
45	BP_dot_UP_1	observable	binary	0	1
46	BP_dot_DOWN_1	observable	binary	0	1
47	BP_dot_UP_2	observable	binary	0	1
48	BP_dot_DOWN_2	observable	binary	0	1
49	BP_dot_UP_3	observable	binary	0	1
50	BP_dot_DOWN_3	observable	binary	0	1
51	BP_dot_UP_4	observable	binary	0	1
52	BP_dot_DOWN_4	observable	binary	0	1
53	BP_dot_UP_5	observable	binary	0	1
54	BP_dot_DOWN_5	observable	binary	0	1
55	BP_dot_UP_6	observable	binary	0	1
56	BP_dot_DOWN_6	observable	binary	0	1
57	EP_dot_1	observable	binary	0	1
58	EP_dot_2	observable	binary	0	1
59	EP_dot_3	observable	binary	0	1
60	EP_dot_4	observable	binary	0	1

Continued on next page

Table 4 – continued from previous page

Index	Name	Access	Type	Min	Max
61	EP_dot_5	observable	binary	0	1
62	EP_dot_6	observable	binary	0	1
63	Dot_dot_1	observable	binary	0	1
64	Dot_dot_2	observable	binary	0	1
65	Dot_dot_3	observable	binary	0	1
66	Dot_dot_4	observable	binary	0	1
67	Dot_dot_5	observable	binary	0	1
68	Dot_dot_6	observable	binary	0	1
69	EP_S_Shape_1	observable	categorical	0	2
70	EP_clock_1	observable	categorical	0	3
71	EP_UP_BP_1	observable	binary	0	1
72	EP_DOWN_BP_1	observable	binary	0	1
73	EP_S_Shape_2	observable	categorical	0	2
74	EP_clock_2	observable	categorical	0	3
75	EP_UP_BP_2	observable	binary	0	1
76	EP_DOWN_BP_2	observable	binary	0	1
77	EP_S_Shape_3	observable	categorical	0	2
78	EP_clock_3	observable	categorical	0	3
79	EP_UP_BP_3	observable	binary	0	1
80	EP_DOWN_BP_3	observable	binary	0	1
81	EP_S_Shape_4	observable	categorical	0	2
82	EP_clock_4	observable	categorical	0	3
83	EP_UP_BP_4	observable	binary	0	1
84	EP_DOWN_BP_4	observable	binary	0	1
85	EP_S_Shape_5	observable	categorical	0	2
86	EP_clock_5	observable	categorical	0	3
87	EP_UP_BP_5	observable	binary	0	1
88	EP_DOWN_BP_5	observable	binary	0	1
89	EP_S_Shape_6	observable	categorical	0	2
90	EP_clock_6	observable	categorical	0	3
91	EP_UP_BP_6	observable	binary	0	1
92	EP_DOWN_BP_6	observable	binary	0	1

Baseline results

The balanced error rates (BER) for separating one class vs. all others were computed by training a Support Vector Machine (SVM) with kernel $K(x,y) = \exp(-\gamma d(x,y))$ $\gamma = 0.02$. Training and testing were done using the training and test sets of IBN_SINA.

Figure 2 depicts the results on IBN_SINA and dataset A with the reference method LSSVM produced by Gavin Cawley. See: http://www.causality.inf.ethz.ch/activelearning.php?page=results#cont for more results.

Table 3: Targets

Index	Name	Access	Type	Min	Max	FracPos (%)
1	EU	observable	binary	-1	1	6.53
2	aL	observable	binary	-1	1	37.84
3	bL	observable	binary	-1	1	5.18
4	dL	observable	binary	-1	1	4.86
5	fL	observable	binary	-1	1	5.79
6	hL	observable	binary	-1	1	10.57
7	kL	observable	binary	-1	1	5.39
8	lL	observable	binary	-1	1	24.85
9	mL	observable	binary	-1	1	13.16
10	nL	observable	binary	-1	1	12.71
11	qL	observable	binary	-1	1	5.43
12	rL	observable	binary	-1	1	5.88
13	tL	observable	binary	-1	1	5.42
14	vL	observable	binary	-1	1	13.75
15	yL	observable	binary	-1	1	13.96

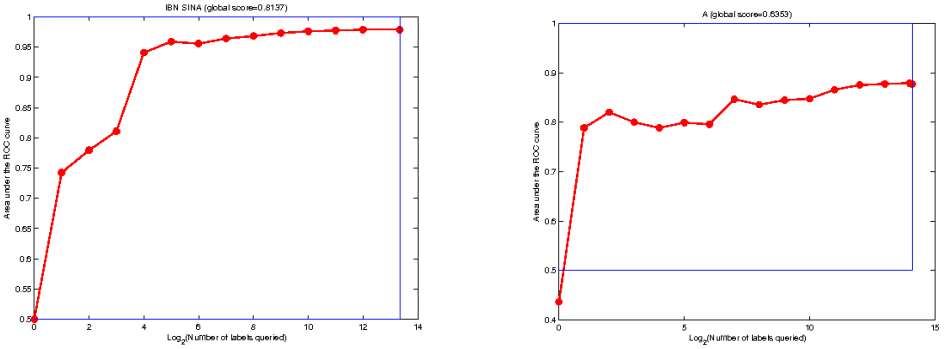
Table 5: Baseline results for IBN SINA

Problem no	Labels	SVM(BER)
1	EU	11.295
2	aL	3.671
3	bL	19.282
4	dL	10.657
5	fL	19.898
6	hL	7.057
7	kL	12.878
8	lL	7.66
9	mL	14.133
10	nL	16.663
11	qL	16.075
12	rL	14.875
13	tL	10.409
14	vL	8.57
15	yL	17.808

Marketing: ORANGE and B datasets

Topic

Customer Relationship Management (CRM) is a key element of modern marketing strategies. The datasets ORANGE and B (mnemotechnical for Banana) were extracted



(a) (b)

Figure 2: Reference results by Gavin Cawley for IBN_SINA (a) and dataset A (b).

from a large marketing database from the French Telecom company Orange. The goal is to predict the propensity of customers to switch provider (churn), buy new products or services (appetency), or buy upgrades or add-ons proposed to them to make the sale more profitable (up-selling). The difficulties include heterogeneous noisy data (numerical and categorical variables), and unbalanced class distributions. For the ORANGE dataset (development dataset), we asked the participants to predict "appetency". For the B dataset (final evaluation dataset), we asked the participants to predict "[appetency OR upselling] AND NOT churn".

Source

The research team at Orange France who prepared the data includes Vincent Lemaire, Marc Boullé, Fabrice Clérot, Raphael Féraud, Aurélie Le Cam, and Pascal Gouzien. Contact: Vincent Lemaire `vincent.lemaire@orange-ftgroup.com`.

- **Donor of database:** This version of the database was prepared for the "Active Learning Challenge" by Isabelle Guyon, 955 Creston Road, Berkeley, CA 94708, USA (`isabelle@clopinet.com`).
- **Date received (original data):** November 2008, for the KDD cup 2009.
- **Date prepared for the challenge:** November 2009.

Past usage

The ORANGE dataset in a large and small version (more or less features) was used in the KDD cup 2009. Scoring was done using the Area under the ROC curve (AUC). The score is the average of the results on the 3 tasks (churn, appetency, and upselling). The best results (in score) were obtained by the IBM team, using the large dataset.

Table 6: Best performance for the tasks of Orange dataset

	Churn	Appetency	Upselling	Score
Fast track	0.7611	0.883	0.9038	0.849312
Slow track	0.7651	0.8819	0.9092	0.852062

For the small dataset, it is uncertain what the results are because some teams "unscrambled" the data and submitted large dataset results in lieu of small dataset results. The small dataset results were worse than the large dataset results. See for more details: Analysis of the KDD Cup 2009: Fast Scoring on a Large Orange Customer Database, Isabelle Guyon, Vincent Lemaire, Marc Boullé, Gideon Dror and David Vogel; JMLR W&CP 7: 1-22, 2009.

Experimental design

The following information was obtained from Orange: "A datamart of about one million Orange customers was used, with about ten tables and hundreds of fields. The first step was to resample the dataset, to obtain 100,000 instances with less unbalanced target distributions. For practical reasons (the challenge participants had to download the data), the same data sample was used for the three marketing tasks. In a second step, the feature construction language was used to generate 20,000 features and obtain a tabular representation. After discarding constant features and removing customer identifiers, we narrowed down the feature set to 15,000 variables (including 260 categorical variables). In a third step, for privacy reasons, data was anonymized, discarding variables names, randomizing the order of the variables, multiplying each continuous variable by a random factor and recoding categorical variable with randomly generated category name. To encourage participation, an easier task was also built from a reshuffled version of the datasets with only 230 variables." For the Active Learning challenge, we used the small dataset version with 230 variables. We randomly re-ordered the features and the examples. In addition, for dataset B, the features were disguised by random shifts and scaling for continuous values and by randomly assigning category values for categorical variables. Twenty "distracter" features were added using real variable whose values were randomly shuffled. These steps made it difficult to match the samples with the original data and guess the labels.

Number of examples and class distribution

The samples are the same in both datasets, but both samples and features are ordered differently. In addition the features in dataset B are disguised and some distracters have been added. See also Tables 1 and 2.

Fraction of positive examples (test and training sets):

- Churn: 7.34

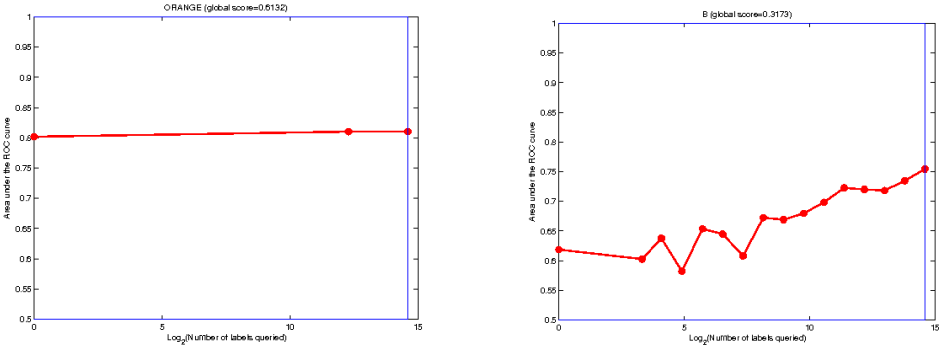
- Appetency: 1.78
- Upselling: 7.36

Type of input variables and variable statistics

Both continuous and categorical variables were found in data. There are 40 categorical variables and 190 continuous variables in the ORANGE data. Details can be obtained from the GLOP package by typing:
`save_profile(orange);`

Baseline results

The best reference results were produced by Marc Boullé using Selective Naïve Bayes (SNB). We also show the results of the overall winners on dataset B. See: http://www.causality.inf.ethz.ch/activelearning.php?page=results#cont for more results.



(a) (b)

Figure 3: Reference results: March Boullé on the Orange dataset (a) and Ideal Analytics, Intel on the B dataset (b).

Ecology: SYLVA and F datasets

Topic

The tasks of the SYLVA and F datasets are to classify forest cover types. Both tasks were carved out of data from the US Forest Service (USFS). The task of SYLVA is to classify Ponderosa pines vs. other classes of trees. The task of F is to classify Krummholz vs. other classes of trees.

Sources

- **Original owners:**
  Remote Sensing and GIS Program
  Department of Forest Sciences
  College of Natural Resources
  Colorado State University
  Fort Collins, CO 80523

  Jock A. Blackard
  USDA Forest Service
  3825 E. Mulberry
  Fort Collins, CO 80524 USA
  jblackard/wo_ftcol@fs.fed.us

  Dr. Denis J. Dean
  Associate Professor
  Department of Forest Sciences
  Colorado State University
  Fort Collins, CO 80523 USA
  denis@cnr.colostate.edu

  Dr. Charles W. Anderson
  Associate Professor
  Department of Computer Science
  Colorado State University
  Fort Collins, CO 80523 USA
  anderson@cs.colostate.edu

  Acknowledgements, Copyright Information, and Availability:
  Reuse of this database is unlimited with retention of copyright notice for Jock A. Blackard and Colorado State University.

- **Donor of database:**
  This version of the database was prepared for the "Active Learning Challenge" by Isabelle Guyon, 955 Creston Road, Berkeley, CA 94708, USA (isabelle@clopinet.com).

- **Date original data received:** August 28, 1998, UCI Machine Learning Repository, under the name Forest Cover Type.

- **Date prepared for the challenge:** November 2009.

Past usage

Blackard, Jock A. 1998. "Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types." Ph.D. dissertation. Department of Forest Sciences. Colorado State University. Fort Collins, Colorado. Classification performance with first 11,340 records used for training data, next 3,780 records used for validation data, and last 565,892 records used for testing data subset: – 70% backpropagation – 58% Linear Discriminant Analysis. The subtask SYLVA was prepared for the WCCI 2006 "Performance Prediction Challenge" and the IJCNN 2007 "Agnostic Learning vs. Prior Knowledge" (ALvsPK) challenge is a 2-class classification problem. The best results were obtained with Logitboost by Roman Lutz with 0.4% balanced error rate (BER) in the PK track and 0.6% error in the AL track (http://clopinet.com/isabelle/Projects/agnostic/Results.html).

Experimental design

The original data comprises a total of 581012 instances (observations) grouped in 7 classes (forest cover types) and having 54 attributes (features) corresponding to 12 measures (10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables). The actual forest cover type for a given observation (30 x 30 meter cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables were derived from data originally obtained from US Geological Survey (USGS) and USFS data. Data is in raw form (not scaled) and contains binary (0 or 1) columns of data for qualitative independent variables (wilderness areas and soil types).

Variable Information

Given is the variable name, variable type, the measurement unit and a brief description. The forest cover type is the classification problem. The order of this listing corresponds to the order of numerals along the rows of the database.

Code Designations and Distribution

Wilderness Areas:

1. Rawah Wilderness Area
2. Neota Wilderness Area
3. Comanche Peak Wilderness Area
4. Cache la Poudre Wilderness Area

Soil Types: 1 to 40, based on the USFS Ecological Landtype Units for this study area.

Table 7: Variables of Sylva and F datasets

Name	Data Type	Units	Description
Elevation	quantitative	meters	Elevation in meters
Aspect	quantitative	azimuth	Aspect in degrees azimuth
Slope	quantitative	degrees	Slope in degrees
Horz_Distance_To_Hydr	quantitative	meters	Horz Dist to nearest surface water feature
Vert_Distance_To_Hydr	quantitative	meters	Vert Dist to nearest surface water feature
Horz_Distance_To_Rd	quantitative	meters	Horz Dist to nearest roadway
Hillshade_9am	quantitative	0 - 255	Hillshade index at 9am, summer solstice
Hillshade_Noon	quantitative	0 - 255	Hillshade index at noon, summer soltice
Hillshade_3pm	quantitative	0 - 255	Hillshade index at 3pm, summer solstice
Horz_Distance_To_FP	quantitative	meters	Horz Dist to nearest wildfire ignition point
Wilderness_Area (4 cols)	qualitative	0 /1	Wilderness area designation
Soil_Type (40 cols)	qualitative	0 /1	Soil Type designation
Cover_Type (7 types)	integer	1 to 7	Forest Cover Type designation

Table 8: Class codes and distribution

Name	code	number of records
Spruce/Fir	1	211840
Lodgepole Pine	2	283301
Ponderosa Pine	3	35754
Cottonwood/Willow	4	2747
Aspen	5	9493
Douglas-fire	6	17367
Krummholz	7	20501

Data preprocessing and data split

We carved a binary classification task out these data. For SYLVA Ponderosa pine is separated from all other trees and for F, Krummholz is separated from all other trees. For SYLVA, we created patterns containing the concatenation of 4 patterns: two of the target class and two randomly chosen from either class. In this way there are pairs of redundant features and half of the features are non-informative. For F, we reverted to the original features, but recoded the categorical variables (Wilderness_Area and Soil_Type) with one variable taking integer values randomly assigned to the categories. We then randomized the order of the features and patterns and subsampled the patterns. In both cases, half of the data were reserved for training and half for testing.

Number of examples and class distribution

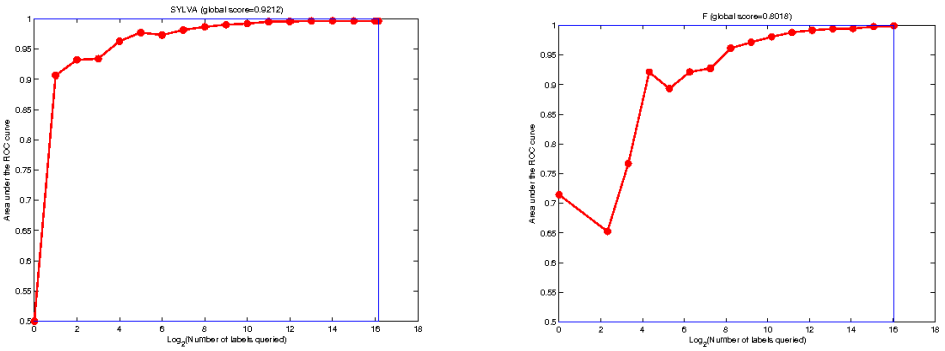
See Tables 1 and 2.

Table 9: Type of input variables and variable statistics

Index	Name	Access	Type	Min	Max
0	target	observable	binary	-1	1
1	Hillshade_Noon	observable	continuous	0	254
2	Soil_Type	observable	categorical	1	40
3	Slope	observable	continuous	0	65
4	Wilderness_Area	observable	categorical	1	4
5	Aspect	observable	continuous	0	360
6	Horizontal_Distance_To_Hydrology	observable	continuous	0	1397
7	Hillshade_9am	observable	continuous	0	254
8	Hillshade_3pm	observable	continuous	0	253
9	Vertical_Distance_To_Hydrology	observable	continuous	-166	599
10	Horizontal_Distance_To_Fire_Points	observable	continuous	0	7172
11	Horizontal_Distance_To_Roadways	observable	continuous	0	7117
12	Elevation	observable	continuous	1859	3858

Baseline results

We show below baseline results on SYLVA and the results obtained on the F dataset by the overall winners of the challenge. See: http://www.causality.inf.ethz.ch/activelearning.php?page=results#cont for more results.



(a) (b)

Figure 4: Reference results: Gavin Cawley on Sylva (a) and Ideal Analytics, Intel on the F dataset (b).

Chemo-informatics: HIVA and C datasets

Topic

The tasks of HIVA and C are to predict chemical activity of molecules. These are two-class classification problems. The variables represent properties of the molecule inferred from its structure. The problem is therefore to relate structure to activity (a QSAR=quantitative structure-activity relationship problem) to screen new compounds before actually testing them (a HTS=high-throughput screening problem). For HIVA the task is to identify compounds that are active against the AIDS HIV infection. For the C dataset the problem is to predict the activation of pyruvate kynase, a well characterized enzyme, which regenerates ATP in glycolysis by catalyzing phosphoryl transfer from phosphoenol pyruvate to ADP to yield pyruvate and ATP. We next describe HIVA and C separately.

Sources

- **Original owners:**
  For the HIVA dataset, the data was made available by the National Cancer Institute (NCI), via the DTP AIDS Antiviral Screen program at: http://dtp.nci.nih.gov/docs/aids/aids_data.html. The DTP AIDS Antiviral Screen has checked tens of thousands of compounds for evidence of anti-HIV activity. Available are screening results and chemical structural data on compounds that are not covered by a confidentiality agreement.

- **Donor of database:**
  This version of the database was prepared for the "Active Learning Challenge" by Isabelle Guyon, 955 Creston Road, Berkeley, CA 94708, USA (isabelle@clopinet.com).

- **Date prepared for the challenge:** November 2009.

Past usage

An earlier release of the database was uses in an Equbits case study: http://www.limsfinder.com/community/articles_comments.php?id=1553_0_2_0_C75. The feature set was obtained by a different method. An earlier version of HIVA prepared by Isabelle Guyon for the WCCI 2006 "Performance Prediction Challenge" and the IJCNN 2007 "Agnostic Learning vs. Prior Knowledge" (ALvsPK) challenge. Depending on whether prior knowledge was used or not and depending on the dataset variants, the best performance was between 26% and 28% Balanced Error Rate (BER). The best result on HIVA in the WCCI 2006 Performance Prediction Challenge was obtained by Gavin Cawley (Test BER=0.275695, Test AUC=0.7671). See http://clopinet.com/isabelle/Projects/agnostic/Results.html for details.

Experimental design

The screening results of the May 2004 release containing the screening results for 43,850 compounds were used. The results of the screening tests are evaluated and placed in one of three categories:

- **CA - Confirmed active**
- **CM - Confirmed moderately active**
- **CI - Confirmed inactive**

We converted this into a two-class classification problem: Inactive (CI) vs. Active (CA or CM.) Chemical structural data for 42,390 compounds was obtained from the web page. It was converted to structural features by the program ChemTK version 4.1.1, Sage Informatics LLC. Four compounds failed parsing. The 1617 features selected include:

- unbranched_fragments: 750 features
- pharmacophores: 495 features
- branched_fragments: 219 features
- internal_fingerprints: 132 features
- ring_systems: 21 features

Only binary features having a total number of ones larger than 100 (>400 for unbranched fragments) and at least 2% of ones in the positive class were retained. In all cases, the default program settings were used to generate keys (except for the pharmacophores for which "max number of pharmacophore points" was set to 4 instead of 3; the pharmacophore keys for Hacc, Hdon, ExtRing, ExtArom, ExtAliph were generated, as well as those for Hacc, Hdon, Neg, Pos.) The keys were then converted to attributes.

We briefly describe the attributes/features:

Branched fragments: each fragment is constructed through an "assembly" of shortest-path unbranched fragments, where each of the latter is required to be bounded by two atoms belonging to one or more pre-defined "terminal-atom".

Unbranched fragments: unique non-branching fragments contained in the set of input molecules.

Ring systems: A ring system is defined as any number of single or fused rings connected by an unbroken chain of atoms. The simplest example would be either a single ring (e.g., benzene) or a single fused system (e.g., naphthalene).

Pharmacophores: ChemTK uses a type of pharmacophore that measures distance via bond connectivity rather than a typical three-dimensional distance. For instance, to describe a hydrogen-bond acceptor and hydrogen-bond donor separated by five connecting bonds, the corresponding key string would be "HAcc.HDon.5". The pharmacophores were generated from the following features:

- Neg. Explicit negative charge.
- Pos. Explicit positive charge.
- HAcc. Hydrogen-bond acceptor.
- HDon. Hydrogen-bond donor.
- ExtRing. Ring atom having a neighbor atom external to the ring.
- ExtArom. Aromatic ring atom having a neighbor atom external to the ring.
- ExtAliph. Aliphatic ring atom having a neighbor atom external to the ring.

Internal fingerprints: small, fixed catalog of pre-defined queries roughly similar to the MACCS key set developed by MDL.

We matched the compounds in the structural description files and those in the compound activity file, using the NSC id number. We ended up with 42678 examples.

Number of examples and class distribution

See Table 1.

Type of input variables

All variables are binary. The data was saved as a non-sparse matrix, even though it is 91% sparse because dense matrices load faster in Matlab and the ASCII format compresses well.

Baseline results

We show below baseline results for HIVA using the reference method LSSVM. See: http://www.causality.inf.ethz.ch/activelearning.php?page=results#cont for more results.

The C (Chemo-informatics) dataset

Sources

- **Original owners:**
  The C dataset is a dataset for assessing the toxicity of kinases that was downloaded from PubMed http://pubchem.ncbi.nlm.nih.gov/. This dataset assayed pyruvate kinase in 51441 compounds, and the qHTS experimental results appear under assay identification number (AID) 361 on PubChem. Note

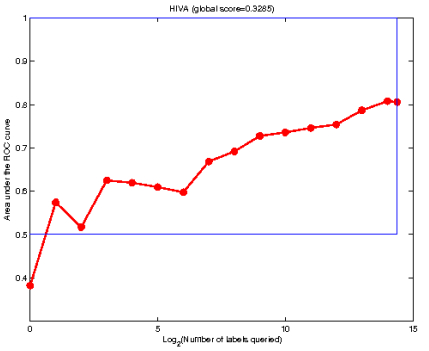


Figure 5: Reference results of Gavin Cawley on Hiva

that a Google search for '361' and '51441' returns a PubChem page as a top search result.

- **Donors of database:**
  Curt Breneman, Professor in the Department of Chemistry and Chemical Biology, Director of Rensselaer Exploratory Center for Cheminformatics Research, at the Rensselaer Polytechnic Institute, Troy, New York, and his students Micheal Krein and Charles Bergeron generated molecular descriptors for the datasets we have identified together as most suitable. The data preprocessing was designed in collaboration with Kristin Bennett, professor in the Department of Mathematical Science and Department of Computer Sciences, working at the same institute.

- **Date prepared for the challenge:** November 2009.

Past usage

None in the context of a challenge.

Experimental design

The data relates to drug discovery. The first step in drug design requires identifying a small number of screening hits that are effective at modulating a disease-specific biological pathway. Traditionally, a large number of compounds are assayed at a single concentration; this is called high-throughput screening (HTS), a mainstay of pharmaceutical development. A recent technique called quantitative high-throughput screening (qHTS) obtains more complete dose-response information by assaying compounds at multiple concentrations in a single experiment. The half-maximal activity concentration pAC_50 is the (negative log-10) concentration at which the midpoint of the activity range is attained.

This dataset assayed pyruvate kinase in 51441 compounds, and the qHTS experimental results appear under assay identification number (AID) 361 on PubChem. QSAR

descriptors (MOE and TAE-RECON) were generated at RPI by Micheal Krein, a PhD student with Prof. Breneman. The substance identification (SID) that may be looked up on PubChem to obtain the molecular structures that are used to generate computational chemistry descriptors.

The team of prof. Breneman calculated their own pAC_50's that are more reliable than the ones reported on the PubChem website. Compounds displaying no activity over the tested concentration range are assigned pAC_50=0. The same is true for a small number of irregular samples that do not follow the expected dose-response behavior. These are arbitrary choices, as 'some unknown number below $\simeq$ 3.5' and 'some unknown real number' would be more accurate statements for inactive and irregular, respectively.

For a classification task, samples having pAC_50$\geq$ 4.94 are interpreted as screening hits and the others are not. PubChem suggests an intermediate category that I call 'junior screening hits' with 4.24 <pAC_50< 4.94. Most samples in this category have pAC_50's that are uncertain, a problem that is significantly improved by the new, more reliable method for calculating pAC_50's.

For the challenge, all positive values of pAC_50 were associated with a positive target value and the others with a negative target value.

Number of examples and class distribution

See Table 2.

Type of input variables

Most variables are continuous, some are binary. To obtain the full dataset profile from the GLOP package, type at the Matlab prompt:

```
save_profile(C);
```

Baseline results

We show the results on the C dataset from the overall challenge winners. See: http://www.causality.inf.ethz.ch/activelearning.php?page=results#cont for more results.

Document classification: NOVA and the D dataset

Topic

The task of the NOVA and D datasets is Document classification from the 20-Newsgroup data. We selected the separation of politics and religion topics from all the other topics for NOVA and the separation of all newsgroups relating to computers vs. others for the D dataset. In both cases these are two-class classification problem with sparse binary input variables using a bag-of-word representation with a vocabulary of approximately 17000 words.

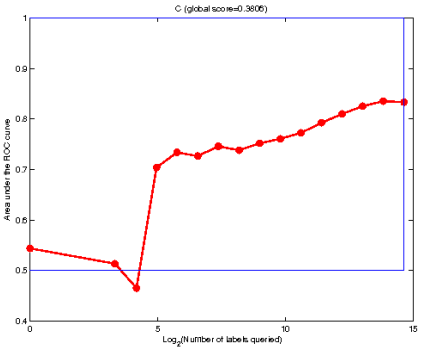


Figure 6: Results on the C dataset by Ideal Analytics, Intel.

Sources

- **Original owners:**
  Tom Mitchell
  School of Computer Science
  Carnegie Mellon University
  tom.mitchell@cmu.edu
  Available from the UCI machine learning repository. The version we are using was preprocessed by Ron Bekkerman http://www.cs.technion.ac.il/~ronb/thesis.html into the "bag-of-words" representation.

- **Donor of database:**
  This version of the database was prepared for the Active Learning challenge on performance prediction by Isabelle Guyon, 955 Creston Road, Berkeley, CA 94708, USA (isabelle@clopinet.com).

- **Date prepared for the challenge:** November 2009.

Past usage

T. Mitchell. Machine Learning, McGraw Hill, 1997.
T. Joachims (1996). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, Computer Science Technical Report CMU-CS-96-118. Carnegie Mellon University.
Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, and Yoad Winter. Distributional Word Clusters vs. Words for Text Categorization. JMLR 3(Mar):1183-1208, 2003.

Versions of NOVA were prepared for the WCCI 2006 "Performance Prediction Challenge" and the IJCNN 2007 "Agnostic Learning vs. Prior Knowledge" (ALvsPK). The best results were between 4% and 6% balanced error rate (BER), see http://clopinet.com/isabelle/Projects/agnostic/Results.html.

Experimental design

We selected 8 newsgroups relating to politics or religion topics as our positive class (Table C.1.) Vocabulary selection includes the following filters:

remove words containing digits and convert to lowercase
remove words appearing less than twice in the whole dataset.
remove short words with less than 3 letters.
exclude 2000 words found frequently in all documents.
truncate the words at a max of 7 letters.

Both NOVA and the D dataset come from the same original data. D is disguised compared to NOVA by removing a few features and adding distracters (random permutations of the original features). Some examples are repeated in D to make the size of the dataset different. Finally the order of the features and samples is randomized.

Table 10: Twenty Newsgroups categories, together with the positive/negative targets for the Nova and D datasets.

Newsgroup	Number of examples	Nova targets	D targets
alt.atheism	1114	+	-
comp.graphics	1002	-	+
comp.os.ms-windows.misc	1000	-	+
comp.sys.ibm.pc.hardware	1028	-	+
comp.sys.mac.hardware	1002	-	+
comp.windows.x	1000	-	+
misc.forsale	1005	-	-
rec.autos	1004	-	-
rec.motorcycles	1000	-	-
rec.sport.baseball	1000	-	-
rec.sport.hockey	1000	-	-
sci.crypt	1000	-	-
sci.electronics	1000	-	-
sci.med	1001	-	-
sci.space	1000	-	-
soc.religion.christian	997	+	-
talk.politics.guns	1008	+	-
talk.politics.mideast	1000	+	-
talk.politics.misc	1163	+	-
talk.religion.misc	1023	+	-

Number of examples and class distribution

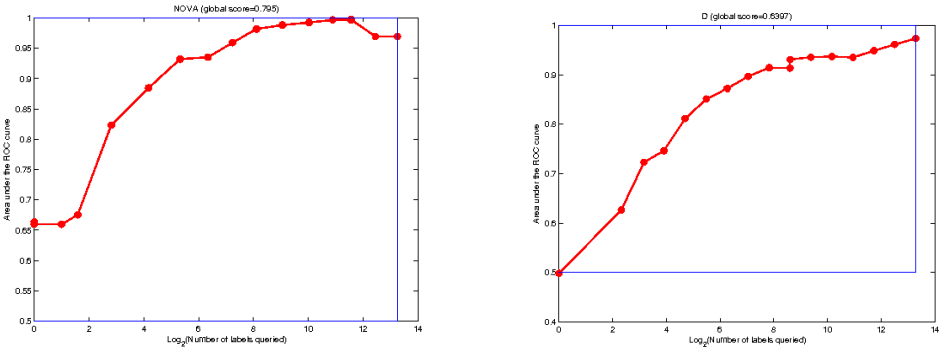
See Tables 1 and 2.

Type of input variables and variable statistics

All variables are binary. There are no missing values. The data is very sparse. Over 99% of the entries are zero. The data was saved as a sparse-binary matrix.

Baseline results

We show below sample results by the organizing team on NOVA and by the overall challenge winners on dataset D. See: http://www.causality.inf.ethz.ch/activelearning.php?page=results#cont for more results.



(a) (b)

Figure 7: Reference results on NOVA by Gideon Dror (a) and on dataset D by Ideal Analytics, Intel (b).

Embryology: ZEBRA and E datasets

Topic

The ZEBRA and E datasets provide a feature representation of cells of zebrafish embryo to determine whether they are in division (meiosis) or not. All the examples in this subset are manually annotated.

Sources

- **Original owners:**
  Emmanuel Faure, Thierry Savy, Louise Duloquin, Miguel Luengo Oroz, Benoit Lombardot, Camilo Melani, Paul Bourgine and Nadine Peyrieras.

Acknowledgements, Copyright Information, and Availability
Thanks to the Embryomics Consortium.
Thanks to IN2P3 (French National Calcul Center)

- **Donor of database:**
  Emmanuel Faure.

- **Date received:**
  November 2009.

Past usage

Nothing with these features. Other versions of these datasets were used in various publications, including: Cell Lineage Reconstruction of Early Zebrafish Embryos Using Label-Free Nonlinear Microscopy, Nicolas Olivier, Miguel A. Luengo-Oroz, Louise Duloquin, Emmanuel Faure, Thierry Savy, Israël Veilleux, Xavier Solinas, Delphine Débarre, Paul Bourgine, Andrés Santos, Nadine Peyriéras and Emmanuel Beaurepaire Science 20 August 2010: 329 (5994), 967-971.

Experimental design (Emmanuel Faure)

Our Embryomics project is devoted to the morphodynamical "reconstruction" of the cell lineage tree underlying the processes of animal embryogenesis. We designed a set of strategies, methods and algorithms to "sequence" the cell lineage tree as a branching process annotated in space and time. Our goal is to fully reconstruct the dynamics of cell divisions and movements from time-lapse series of high-resolution optical sections obtained by multiphoton laser scanning microscopy throughout embryonic development of live animals. Embryomics allows the automated tracking of events such as cell division and cell death in live embryos and give us access to parameters such as the rate of cell proliferation in time and space.
Embryo staining and mounting:
Wild type (070418a) and Zoep (081018a) zebrafish embryos were injected at the one cell stage with 200 pg mcherry/H2B RNA and 200 pg eGFP-ras prepared from PCS2+ constructs. Injected embryos were raised at 28.5 deg C for the next 3 hours. Embryos were mounted in a 3cm Petri dish with a glass slide bottom, sealing a hole of 0,5 mm at the Petri dish centre where a Teflon tore (ALPHAnov) with a hole of 780 microns received the dechorionated embryo. Embryo was maintained and properly oriented by infiltrating around the embryo 0.5% low melting point agarose (SIGMA) in embryo medium. Temperature control in the room (22 deg C) insured about 26 deg C under the objective and development slowly drifted from the standard 28.5 deg C developmental table.
Image acquisition:
The volume was imaged with a Leica DM6000 upright microscope SP5 MLSM equipped with a 20x/0,95NA W deeping lens Olympus objective. Field size is 700x700 in x, y.

Voxel size is 1.37 microns. Simultaneous excitation with 1030 nm and 980 nm femtosecond pulsed laser beams was obtained from a single source (Amplitude t pulse 20) with part of the beam modified through a photonic crystal fiber (Amplitude).
Variable Information:

```
(0-2)      Identity information for each cell
(3 - 4)    Neighbors were calculated by Voronoi
           tessellation.
(8 - 11)   Displacement was calculated by vector field
           from original image.
(12 - 32)  Using membrane segmentation, extract from
           the shape information.
(32 - 46)  Intensity information by matching membrane
           shape & original image.
(47 - 81)  Neighbors correlation on Membrane information
(82 - 151) Same as Membrane for nuclei segmentation.
```

Workflow of image processing:

- Filtering: Oimage filtering of both channels membranes and nucleus was performed through the methodology described in (Drblikova 2008) (O. Drblikova, K. Mikula, Semi-implicit diamond-cell finite volume scheme for 3D nonlinear tensor diffusion in coherence enhancing image filtering, in Finite Volumes for Complex Applications V: Problems and Perspectives (Eds. R. Eymard, J. M. Herard), ISTE and WILEY, London, 2008, pp. 343–35)

- Nucleus centre detection was performed through the methodology described in (Frolkovic 2007) (Frolkovic, P., Mikula, K., Peyrieras, N., & Sarti, A. 2007. Counting Number of Cells and Cell Segmentation Using Advection-Diffusion Equations. KYBERNETIKAPRAHA.)

- Segmentation: of membranes and nuclei shape was performed through the methodology described in (Mikula 2008) for nuclei and (Luengo 2008) for membranes. (K.Mikula, N.Peyrieras, M.Remesikova, A.Sarti, 3D embryogenesis image segmentation by the generalized subjective surface method using the finite volume technique, in Finite Volumes for Complex Applications V: Problems and Perspectives (Eds. R.Eymard, J.M.Herard), ISTE and WILEY, London, 2008, pp. 585-592) (Luengo-Oroz, M, Duloquin, L, Castro, C, Savy, T, Faure, E, Lombardot, B, Bourgine, P, Peyriéras, N, & Santos, A. 2008. Can voronoi diagram model cell geometries in early sea-urchin embryogenesis ? Biomedical Imaging : From Nano to Macro.)

- Tracking of cells and detection of mitoses was performed with the methodology described in Melani, C, Peyrieras, N, Mikula, K, & Zanella, C. 2007. Cells tracking in a live zebrafish embryo. Engineering in Medicine and Biology Society, Jan. 2007.

- Manual mitosis annotation was performed by 3 different biologist experts.

Data split:
Only manually annotated data were used. For the development dataset ZEBRA a subset of samples corresponding to cells before mitosis or not in mitosis was selected and the task was to separate these two types of samples. For the final evaluation set called "E", the samples not in mitosis and after mitosis were selected to create another binary classification problem. Hence the two datesets overlapped. The orders of the features and samples were randomized to make it difficult to identify the common examples.

Data statistics

See Tables 1 and 2.

Table 11: **Variables of Embryology datasets.** All variables are continuous.

Index	Min	Max	Name
0	0	319552192	Cell ID
1	0	4	Annotation value (target)
2	0	479	Cell Time
3	0	56	Number of neighbors
4	-1024	5.3887	Neighborhoods tensor Deformation
5	30.14	661.71	X Coord
6	53.43	680.89	Y Coord
7	5.48	139.74	Z Coord
8	-21.9182	12.539	X Deplacement
9	-21.3561	20.1138	Y Deplacement
10	-10.0698	16.4175	Z Deplacement
11	0.0098	25.465	Velocity
12	0.0041	29.3297	Membrane Distance Center Gravity
13	0.0001	170803	Membrane Volume Segmentation
14	0	49407.3984	Membrane Volume Pixel
15	0.0127	5564.98	Membrane Surface Area
16	0.2294	6.8166	Membrane Normalize Shape Index
17	22.3713	483.334	Membrane X Gravity Center
18	39.674	498.43	Membrane Y Gravity Center
19	5.2	101.964	Membrane Z Gravity Center
20	0	0.6635	Membrane X Ellipse Axes Length
21	0	0.6835	Membrane Y Ellipse Axes Length
22	0	0.4838	Membrane Z Ellipse Axes Length
23	0.0526	inf	Membrane Elipse Ratio Elongation (Axes Max/Min)
24	0.5774	1	Membrane 0.0 Eigen Vectors Covariance Matrix
25	-0.7071	0.7071	Membrane 0.1 Eigen Vectors Covariance Matrix
26	-0.7071	0.7071	Membrane 0.2 Eigen Vectors Covariance Matrix
27	-0.8156	0.8146	Membrane 1.0 Eigen Vectors Covariance Matrix
28	0.4129	1	Membrane 1.1 Eigen Vectors Covariance Matrix
29	-0.7071	0.7071	Membrane 1.2 Eigen Vectors Covariance Matrix
30	-0.8153	0.815	Membrane 2.0 Eigen Vectors Covariance Matrix
31	-0.828	0.8287	Membrane 2.1 Eigen Vectors Covariance Matrix

Continued on next page

Table 11 – continued from previous page

Index	Min	Max	Name
32	0.3348	1	Membrane 2.2 Eigen Vectors Covariance Matrix
33	0	240.395	Membrane Mean Intensity
34	0	255	Membrane Max Intensity
35	0	255	Membrane Min Intensity
36	0	82.1449	Membrane Sigma Intensity
37	0	2096780	Membrane Sum Intensity
38	0	230.859	Membrane Mean Countour Intensity
39	0	255	Membrane Max Countour Intensity
40	0	255	Membrane Min Countour Intensity
41	0	88.0933	Membrane Sigma Countour Intensity
42	0	751864	Membrane Mean Sphere Intensity
43	0	240.395	Membrane Max Sphere Intensity
44	0	255	Membrane Min Sphere Intensity
45	0	255	Membrane Sigma Sphere Intensity
46	0	82.1449	Membrane Sum Sphere Intensity
47	0	2096780	Neigbhors Membrane Distance Center Gravity
48	0.037	12.6397	Neighbors Membrane Volume Segmentation
49	1.1349	6568.5601	Neighbors Membrane Volume Pixel
50	4.8358	6605.0698	Neighbors Membrane Surface Area
51	8.0992	2495.23	Neighbors Membrane Normalize Shape Index
52	0.0199	1.5098	Neighbors Membrane X Gravity Center
53	1.336	467.737	Neighbors Membrane Y Gravity Center
54	1.748	475.114	Neighbors Membrane Z Gravity Center
55	0.298	98.8438	Neighbors Membrane X Ellipse Axes Length
56	0	0.2288	Neighbors Membrane Y Ellipse Axes Length
57	0	0.1974	Neighbors Membrane Z Ellipse Axes Length
58	0	0.1334	Neighbors Membrane Elipse Ratio Elongation (Axes Max/Min)
59	0.0561	inf	Neighbors Membrane 0.0 Eigen Vectors Covariance Matrix
60	0.015	1	Neighbors Membrane 0.1 Eigen Vectors Covariance Matrix
61	-0.6301	0.7071	Neighbors Membrane 0.2 Eigen Vectors Covariance Matrix
62	-0.6743	0.6786	Neighbors Membrane 1.0 Eigen Vectors Covariance Matrix
63	-0.7086	0.7306	Neighbors Membrane 1.1 Eigen Vectors Covariance Matrix
64	0.015	1	Neighbors Membrane 1.2 Eigen Vectors Covariance Matrix
65	-0.6982	0.684	Neighbors Membrane 2.0 Eigen Vectors Covariance Matrix
66	-0.7155	0.7279	Neighbors Membrane 2.1 Eigen Vectors Covariance Matrix
67	-0.7667	0.7565	Neighbors Membrane 2.2 Eigen Vectors Covariance Matrix
68	0.0148	1	Neighbors Membrane Mean Intensity
69	0.9515	87.6384	Neighbors Membrane Max Intensity
70	2.9334	214	Neighbors Membrane Min Intensity
71	0	29.4444	Neighbors Membrane Sigma Intensity
72	0.4493	40.0442	Neighbors Membrane Sum Intensity
73	391	228298	Neighbors Membrane Mean Countour Intensity
74	0.8438	58.7286	Neighbors Membrane Max Countour Intensity
75	3.2611	201	Neighbors Membrane Min Countour Intensity
76	0	28.8889	Neighbors Membrane Sigma Countour Intensity
77	0.4767	40.816	Neighbors Membrane Mean Sphere Intensity
78	1652.0422	177889	Neighbors Membrane Max Sphere Intensity
79	0.9515	87.6384	Neighbors Membrane Min Sphere Intensity

Continued on next page

Table 11 – continued from previous page

Index	Min	Max	Name
80	2.9334	214	Neighbors Membrane Sigma Sphere Intensity
81	0	29.4444	Neighbors Membrane Sum Sphere Intensity
82	0.4493	40.0442	Nucleus Distance Center Gravity
83	391	228298	Nucleus Volume Segmentation
84	0.0041	29.3297	Nucleus Volume Pixel
85	0.0001	170803	Nucleus Surface Area
86	0	49407.3984	Nucleus Normalize Shape Index
87	0.0127	5564.98	Nucleus X Gravity Center
88	0.2294	6.8166	Nucleus Y Gravity Center
89	22.3713	483.334	Nucleus Z Gravity Center
90	39.674	498.43	Nucleus X Ellipse Axes Length
91	5.2	101.964	Nucleus Y Ellipse Axes Length
92	0	0.6635	Nucleus Z Ellipse Axes Length
93	0	0.6835	Nucleus Elipse Ratio Elongation (Axes Max/Min)
94	0	0.4838	Nucleus 0.0 Eigen Vectors Covariance Matrix
95	0.0526	inf	Nucleus 0.1 Eigen Vectors Covariance Matrix
96	0.5774	1	Nucleus 0.2 Eigen Vectors Covariance Matrix
97	-0.7071	0.7071	Nucleus 1.0 Eigen Vectors Covariance Matrix
98	-0.7071	0.7071	Nucleus 1.1 Eigen Vectors Covariance Matrix
99	-0.8156	0.8146	Nucleus 1.2 Eigen Vectors Covariance Matrix
100	0.4129	1	Nucleus 2.0 Eigen Vectors Covariance Matrix
101	-0.7071	0.7071	Nucleus 2.1 Eigen Vectors Covariance Matrix
102	-0.8153	0.815	Nucleus 2.2 Eigen Vectors Covariance Matrix
103	-0.828	0.8287	Nucleus Mean Intensity
104	0.3348	1	Nucleus Max Intensity
105	0	240.395	Nucleus Min Intensity
106	0	255	Nucleus Sigma Intensity
107	0	255	Nucleus Sum Intensity
108	0	82.1449	Nucleus Mean Countour Intensity
109	0	2096780	Nucleus Max Countour Intensity
110	0	230.859	Nucleus Min Countour Intensity
111	0	255	Nucleus Sigma Countour Intensity
112	0	255	Nucleus Mean Sphere Intensity
113	0	88.0933	Nucleus Max Sphere Intensity
114	0	751864	Nucleus Min Sphere Intensity
115	0	240.395	Nucleus Sigma Sphere Intensity
116	0	255	Nucleus Sum Sphere Intensity
117	0	255	Neighbors Nucleus Distance Center Gravity
118	0	82.1449	Neighbors Nucleus Volume Segmentation
119	0	2096780	Neighbors Nucleus Volume Pixel
120	0.0172	21.8132	Neighbors Nucleus Surface Area
121	0.0271	4100.3735	Neighbors Nucleus Normalize Shape Index
122	0	356.0759	Neighbors Nucleus X Gravity Center
123	0.2093	112.4329	Neighbors Nucleus Y Gravity Center
124	0.0197	inf	Neighbors Nucleus Z Gravity Center
125	1.2045	94.0378	Neighbors Nucleus X Ellipse Axes Length
126	1.5092	122.5083	Neighbors Nucleus Y Ellipse Axes Length
127	0.2852	24.9129	Neighbors Nucleus Z Ellipse Axes Length

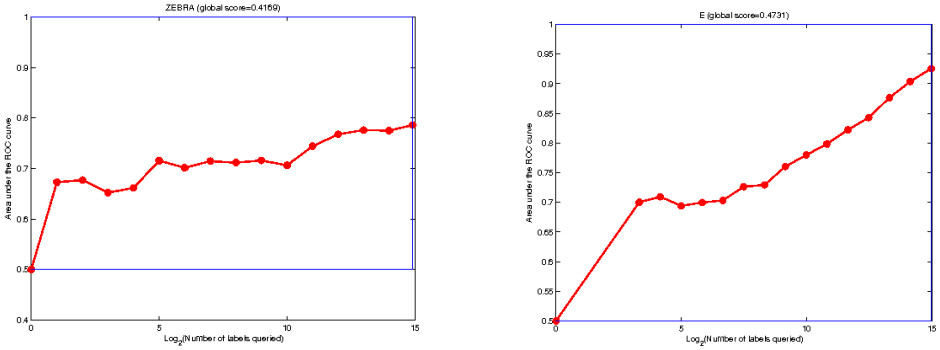
Continued on next page

Table 11 – continued from previous page

Index	Min	Max	Name
128	0	0.1269	Neighbors Nucleus Elipse Ratio Elongation (Axes Max/Min)
129	0	0.0993	Neighbors Nucleus 0.0 Eigen Vectors Covariance Matrix
130	0	0.0428	Neighbors Nucleus 0.1 Eigen Vectors Covariance Matrix
131	0.004	inf	Neighbors Nucleus 0.2 Eigen Vectors Covariance Matrix
132	0.0155	0.2103	Neighbors Nucleus 1.0 Eigen Vectors Covariance Matrix
133	-0.1081	0.1219	Neighbors Nucleus 1.1 Eigen Vectors Covariance Matrix
134	-0.0737	0.0683	Neighbors Nucleus 1.2 Eigen Vectors Covariance Matrix
135	-0.122	0.106	Neighbors Nucleus 2.0 Eigen Vectors Covariance Matrix
136	0.0156	0.2164	Neighbors Nucleus 2.1 Eigen Vectors Covariance Matrix
137	-0.0771	0.083	Neighbors Nucleus 2.2 Eigen Vectors Covariance Matrix
138	-0.0683	0.0793	Neighbors Nucleus Mean Intensity
139	-0.0844	0.0842	Neighbors Nucleus Max Intensity
140	0.0155	0.2467	Neighbors Nucleus Min Intensity
141	0	24.3792	Neighbors Nucleus Sigma Intensity
142	0	54.3125	Neighbors Nucleus Sum Intensity
143	0.0343	42.5	Neighbors Nucleus Mean Countour Intensity
144	0	12.7434	Neighbors Nucleus Max Countour Intensity
145	0	11538	Neighbors Nucleus Min Countour Intensity
146	0	18.4295	Neighbors Nucleus Sigma Countour Intensity
147	0	55.1875	Neighbors Nucleus Mean Sphere Intensity
148	0.0208	42.5	Neighbors Nucleus Max Sphere Intensity
149	0	12.8225	Neighbors Nucleus Min Sphere Intensity
150	0	16145.333	Neighbors Nucleus Sigma Sphere Intensity
151	0	24.3792	Neighbors Nucleus Sum Sphere Intensity

Baseline results

We show below a reference learning curve obtained for ZEBRA by the organizers and the learning curve of the overall winners for dataset E. More results are found on the webpage: http://www.causality.inf.ethz.ch/activelearning.php?page=results#cont.



(a) (b)

Figure 8: Reference results by Gavin Cawley for Zebra (a) and Results of IdealAnalytics, Intel on dataset E (b).

Appendix B
Matlab Implementation and Sample Code

Introduction

The Virtual Lab is a web-based platform implemented in PHP/MySQL with a Matlab® backend, see http://www.causality.inf.ethz.ch/workbench.php. It allows users to carry out virtual experiments on data generating systems, which emulate real systems. The users are given a budget of virtual cash to run their experiments. They can place queries to gain access to data, which are answered by the system in exchange for virtual cash.

For the Active Learning challenge, 13 datasets were wrapped into Matlab objects in the GLOP package (Generative Lab Object Package): six development datasets, six datasets used for final testing, and one toy dataset used for illustration purpose. The GLOP package managed the user's cash account, computed performances, and answered queries. The GLOP package is available for download from http://www.causality.inf.ethz.ch//repository.php?id=23.

The goal of active learning is to learn a task as fast as possible while using as few labeled examples as possible. In active learning experiments, one starts with an unlabeled dataset. A single labeled experiment is initially provided (seed). Other labels must be purchased for virtual cash. The performance of the participants is monitored with their learning curve, which plots the classification performance on test (or validation) data, as a function of the amount of virtual cash spent (which is proportional to the number of examples). The website of the Active Learning challenge is available at http://www.causality.inf.ethz.ch/activelearning.php.

Architecture of the Virtual Lab

The Virtual Lab is an environment accessible via the Internet. Users must be registered to place queries and get answers. Each user is known to the outside world via his WorkbenchID and can remain anonymous. Registered users have a private area called "My Lab" in which they can monitor the progress of their experiments. A summary of the results is displayed on a public leaderboard.

Software architecture

Users interact with the Virtual Lab via a web interface, supported by scripts written in the PHP scripting language embedded into HTML. The interface has a MySQL database backend, which holds tables of users, experiments and results. Submissions are performed via an upload page, which also records the use name, experiment name,

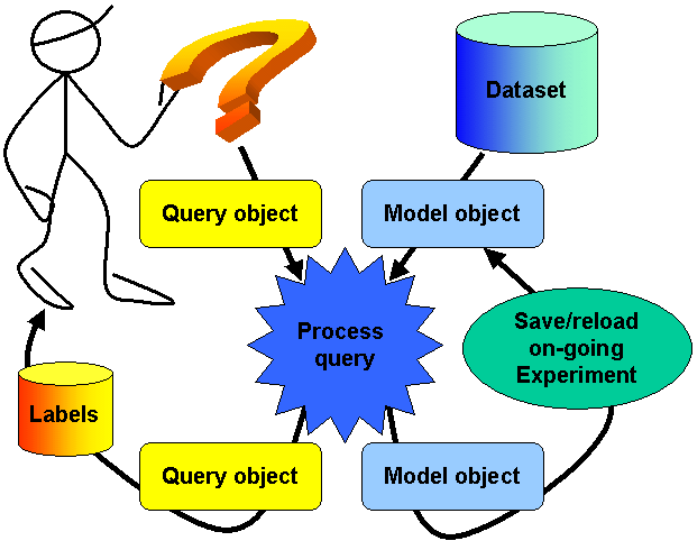


Figure 9: **Conducting experiments in the Virtual Lab.**

model (or dataset name), and date. The submitted zip archive is then renamed to include this information. For example, a query submitted by user John on dataset SYLVA with experiment name “verif", on 03/25/2010 at 15h 34m 03s would be named:

```
John_sylva_verif_2010-03-25-153403.zip
```

When a query is submitted to the Virtual Lab (in the form of a number of files following a specific format and bundled in the zip archive), it is entered in a queue. Jobs are processed in the order received. For every job Matlab gets launched. A Matlab script then parses the file name to recuperate information and instantiates a **query object** (see Section B) that loads the query information. A **model object** for the correct model/dataset (SYLVA in our example) is also instantiated. If the user has already started an experiment with the same name, a saved version of the corresponding Matlab object gets reloaded in the model object.

Matlab then processes the query. The model object is updated to record changes in the virtual cash account and prediction performances. The query object is modified to hold the query answer. Both objects are saved and the Matlab session is terminated. A script periodically checks whether query answers have become available and updates the database.

The overall process is summarized in Figure 9.

Custom version of the Virtual Lab

We have implemented several challenges using the Virtual Lab. Each time, we created a slightly customized version of the interface to run the challenge. Examples are found at: http://www.causality.inf.ethz.ch/challenges.php. For the Active Learning challenge, registered users were allowed to:

- Download data from the Data page http://www.causality.inf.ethz.ch/activelearning.php?page=datasets.
- Submit results and queries bundled as a zip archive from the Submit page http://www.causality.inf.ethz.ch/activelearning.php?page=submit.
- Retrieve their results from the personal Mylab page http://www.causality.inf.ethz.ch/activelearning.php?page=mylab or from the leaderboard page http://www.causality.inf.ethz.ch/activelearning.php?page=leaderboard.

The queries and answers follow a specific format described on the Instruction page http://www.causality.inf.ethz.ch/activelearning.php?page=instructions#cont. This is a simplified version of the more general query-answer format of the virtual lab, described at http://www.causality.inf.ethz.ch/workbench.php?page=info.

Submitted queries must be bundled in a .zip archive and include the following text files:

- **dataname.sample -** Row numbers of the samples for which you want labels (one number per line). You can ask only for the labels of training examples (the first T lines in the data matrix for T training examples).
- **dataname.predict -** Prediction scores of the target variable (label) for all the examples in the dataset (one number per line; as many lines as there are lines in the data table). The scores can be any number, larger numerical values indicating higher confidence in positive class membership. See the Evaluation page for more details.

We asked the participants to always send predictions when they placed a query. Predictions could be sent without requesting labels (i.e. an archive may contain only dataname.predict and not dataname.sample) but no request for labels was answered unless a valid prediction file dataname.predict was provided.

The answers to the query were made available through the My Lab page, as a zip archive containing the following files:

- **dataname.README -** A message indicating whether the query was processed successfully.

- **dataname.sample -** The identity of the samples for which the labels are provided (line number in the data matrix).
- **dataname.label** - The corresponding labels (target values).
- **dataname.score** - The AUC score for the predictions made (see Evaluation).
- **dataname.ebar** - An estimate of the error bar on the AUC.
- **dataname.global_score** - The area under the learning curve (see Evaluation).
- **dataname.totalcost** - The cost of the query. Usually equal to the number of labels provided, but may be less if some labels have been queried before and already paid for.

In addition, the graph of the learning curve was made available.

Usage

For the Active Learning challenge, step-by-step instructions were provided to the participants:

1. Get the data from the Data page.
2. Prepare your experiment: Choose one of the datasets, e.g., ALEX. Eventually preprocess the data. Give a name to your experiment, e.g., "myexp1". An experiment is a set of query/answers, starting from an initial budget allowing you to purchase all the labels, and ending when you run out of virtual cash. A new experiment is created when you submit your first query.
3. Iterate:
    - **Predict -** Using the examples with known labels (at the first iteration use the seed example, which has a positive label), train a predictor and provide prediction scores for ALL the examples of the dataset (including those used for training). Any sort of numeric prediction score is allowed, larger numerical values indicating higher confidence in positive class membership.
    - **Sample -** Choose among the remaining unlabeled examples those for which you want the purchase labels. You may only query examples in the first half of the dataset (training examples). If you query test examples, you will not receive those labels (and not be charged either).
    - **Submit a query -** Format your predictions and chosen samples using the Query Format. Submit it via the Submit page. Select the name of the dataset (e.g., ALEX). Enter the name of a new experiment to create it (e.g., "myexp1"), or select the name of an on-going experiment. Give a description of this entry to remember what you did (the description will be for your eyes only, it will not show on the leaderboard).

- **Retrieve the labels -** After submitting your query, you are redirected to the My Lab page. Refresh periodically the page until the results of your query are ready for download.

During the development period, the challenge participants used the six development datasets. They could experiment on the same dataset as many times as they wanted. Every time they obtained a new budget allowing them to purchase all the training labels. After the first experiment, they obtained all the training labels and could use them to develop algorithms without going through the website. This may have facilitated their work. But we urged them to try the submission system to make sure it worked for them because during final testing, they could only perform one experiment on each final dataset.

Sample Matlab code

We provided sample queries, and sample Matlab code to help the participants prepare their queries. The sample code processed the queries and generated learning curves for the example dataset ALEX. The sample Matlab code also works in conjunction with GLOP (see Section B) and can emulate the functioning of the Virtual Lab. It may be downloaded from: http://www.causality.inf.ethz.ch/al_data/Sample_code.zip.

The Generative Lab Object Package (GLOP)

We developed an object-oriented interface to easily incorporate new data generating models, implemented in Matlab. Data generating models may include artificial models, realistic simulators of real systems, or wrappers of datasets of real data. For the Active Learning challenge, we used only wrappers of real datasets.

GLOP is based on two simple abstractions:

- query object, and
- model object.

The query object holds the query provided by the user or the query answer (including data delivered by the data generating model). It has a fixed structure. The model object is a template from which new data generating models can be derived.

GLOP model objects are equipped with a number of generic methods. At the Matlab prompt, the following commands allow the user to find information on the models:

```
> whoisglop              % List the models
> default(model)         % Get default values of a model
> methods(model)         % List the methods
> properties(model)      % List the properties (data members)
```

The following commands allow the user to create a query of a given type. The query types presently supported include TRAIN, TEST, OBS, SURVEY, EXP, PREDICT, AL, and PREPRO. The type AL was created for the Active Learning challenge. Here are a few examples of instantiations of query objects:

```
> q=query('OBS 20');     % Create a query to get
                         % observational data
> q=query('TRAIN');      % Create a "train" query to get
                         % the training set
> q=query('TEST');       % Create a "test" query to get
                         % the test set
```

For the challenge, a Matlab script gets called every type a query is placed by a participant in the form of a "dataname.sample" and a "dataname.predict" files bundled in a zip archive. The script instantiate an AL query object using the following command:

```
> q=query('file');       % Create a query by loading
                         % the data in file.zip
```

There are presently 28 models in GLOP, 13 of which have been programmed for the Active Learning challenge. Those include the development dataset models:

`ibn_sina, hiva, nova, orange, sylva, and zebra,`

the final evaluation datasets:

`a, b, c, d, e, f,`

and the toy data model "alex" (which stands for "active learning example").

Here are a few examples of model instantiations:

```
> a=alarm({'cost_per_sample=1', ...
  'cost_per_var_observation=1', ...
  'cost_per_var_manipulation=2'});
> a=sprinkler;
```

For the Active Learning challenge, all models are derived from the class *al_model*, which itself inherits from the generic object *model*. A given model is usually associated with a task to be completed. Important methods of the *model* object include:

```
> initial_budget(a)              % Get virtual cash budget
                                 % allocated to task
> task_n_pricing(a)              % Text description of the
                                 % task
> [q, a]=process_query(a, q);    % Processes the query q
```

Note that in Matlab syntax, a method is called with the object name as first argument. Other arguments follow. Hence, *process_query* is a method of the objects of class *model* and has a query as argument. The method *process_query* is overloaded for objects of type *al_model*. In particular, it updates the calculation of the learning curve using the latest results provided with the query.

Conclusion

The Virtual Lab of the Causality Workbench was implemented with the idea of benchmarking causal discovery algorithm. It is also a very effective tool to organize sophisticated machine learning challenges unrelated to causality but requiring query/answer interactions between the participant and a central server. Additionally, the Matlab backend allows us to perform easily a variety of result analyzes and provide on-line performance feed-back in the form of graphs, such as learning curves. This greatly expands the possibilities in terms of challenge design. The website of the Active Learning challenge remains open for post-challenge submissions at `http://www.causality.inf.ethz.ch/activelearning.php`. A more detailed technical report on the Virtual Lab is available in (Guyon et al., 2009).

References

I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, O. Guyon, J.-P. Pellet, P. Spirtes, and A. Statnikov. The causality workbench virtual lab, 2009.

Acknowledgments

This development of the Virtual Lab of the Causality Workbench was funded by the U.S. National Science Foundation under Grant N0. ECCS-0725746. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We are very grateful to all the members of the causality workbench team for their contributions and in particular to our co-founders Constantin Aliferis, Greg Cooper, André Elisseeff, Jean-Philippe Pellet Peter Spirtes, and Alexander Statnikov. The website implementation was done by MisterP.net who provided exceptional support.