JMLR: Workshop and Conference Proceedings 16 (2011) 59–69 Workshop on Active Learning and Experimental Design

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 $T_i = T_i(G), i = 1 \dots R$, 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 $q(x_1) > q(x_2) > \dots > q(x_{n_u})$, where n_u is the number of remaining unlabeled instances.*

3. *Sample the next batch randomly from $x_1, \dots x_{\alpha n_u}$. 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 (2010) 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. (2010). 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 (2010). 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.

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.

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

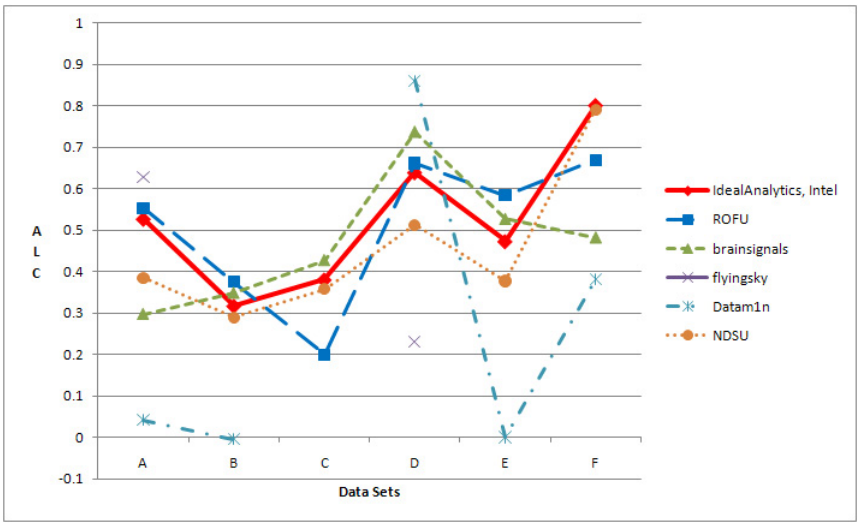

Figure 1: The ALC performance of SQBF (IdealAnalytics) and selected competitors over all the datasets where results were provided. The competitors selected achieved a top-two result on at least one dataset.

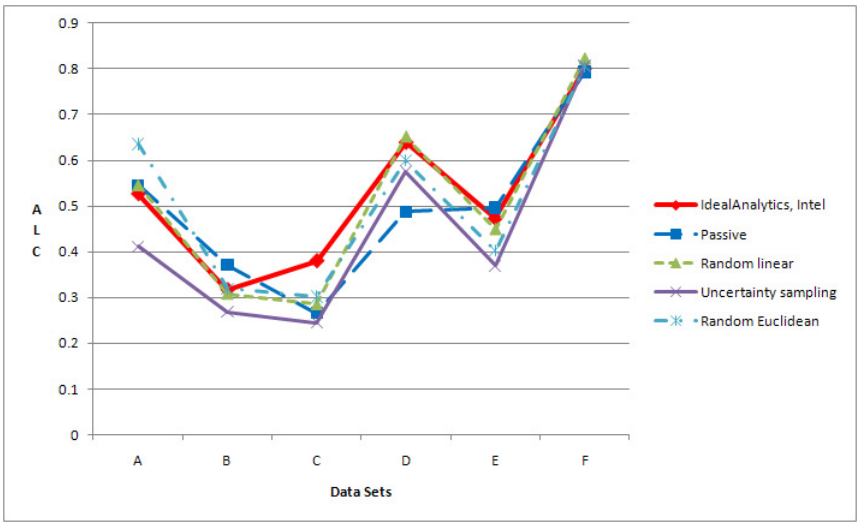

Figure 2: The ALC performance of SQBF (IdealAnalytics) and baseline methods (described by Cawley (2010) over all the datasets.

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

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	4.1667
	ROFU	scan33scan33	0.8774	0.0014	0.5072	4.8333

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.

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 1, 2010.

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 1, 2010.

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.