Active Learning With Optimal Instance Subset Selection

Yifan Fu, Xingquan Zhu, and Ahmed K. Elmagarmid, *Fellow, IEEE*

***Abstract*—Active learning (AL) traditionally relies on some instance-based utility measures (such as uncertainty) to assess individual instances and label the ones with the maximum values for training. In this paper, we argue that such approaches cannot produce good labeling subsets mainly because instances are evaluated independently without considering their interactions, and individuals with maximal ability do not necessarily form an optimal instance subset for learning. Alternatively, we propose to achieve AL with optimal subset selection (ALOSS), where the key is to find an instance subset with a maximum utility value. To achieve the goal, ALOSS simultaneously considers the following: 1) the importance of individual instances and 2) the disparity between instances, to build an instance-correlation matrix. As a result, AL is transformed to a semidefinite programming problem to select a k-instance subset with a maximum utility value. Experimental results demonstrate that ALOSS outperforms state-of-the-art approaches for AL.**

***Index Terms*—Active learning, instance subset selection, machine learning.**

I. Introduction

Active learning [1] represents a family of approaches which selectively label training samples to build classifiers with maximum prediction accuracy (in this paper, samples and instances are interchangeable terms). Compared to passive learning, which labels samples in a random manner, an active learner intends to reduce labeling cost by focusing on informative (or *uncertain*) instances without compromising the accuracy of the classifiers trained from labeled data. Due to increasing capability for data collection and underlying costs involved for labeling, active learning (AL) has been popularly used in many applications, including text classification [2], [3], information networks [4], facial age classification [7], protein structure prediction [5], and stream data mining [6].

From a technical perspective, the key of AL is to find instances mostly needed for labeling, such that the inclusion of the instances into the labeled set can help improve learning models. In practice, the identification of "mostly needed" samples is done by assessing the utility values of the instances with respect to some models trained from labeled data. For example, uncertainty is a common utility measure assessing a model's certainty in classifying an unlabeled sample. If an instance x's uncertainty is high, it implies that the current model does not have enough knowledge to classify x, and presumably, including x into the training set can help improve the underlying learning model. Following this heuristic, the key challenge for AL is to design proper utility metrics to select instances with a maximal capability for labeling.

When assessing instances for labeling, existing AL methods can be roughly categorized into two groups: 1) individual assessment based and 2) set assessment based. The former treats unlabeled instances as independent and identically distributed (I.I.D.) samples, and each instance's utility value is calculated without taking others into consideration, whereas the latter intends to select an optimal subset with a maximal utility value, by using sample correlations/distributions to estimate utility value.

One possible problem with individual-assessment-based approaches is that they may label similar instances, which, in turn, results in labeling redundancy. Intuitively, assuming that a hiker is planning to spend $100 to buy necessary supplies/equipments for a summer trip. The most important items would be "water" related because they are essential for surviving. It is, however, not wise to spend all money on "water"-related items, such as bottled water, juice, coke, etc., because they complement each other. Items, such as food, footwear, map, etc., are all important to ensure a successful and pleasant trip. Take Fig. 1 as another example; when only considering the uncertainty of the samples for labeling, a labeling set may contain most uncertain samples, each of which has the maximum uncertainty value from a single-instance perspective, but samples in the set may contain redundant knowledge and do not form an ideal candidate set, as shown in Fig. 1(b). Uncertainty metrics based on a set assessment, on the other hand, utilizes some similarity measures to discriminate samples so selected instances may not be the "most uncertain" ones, whereas together, they form a good labeling set. As shown in Fig. 1(c), by considering sample correlations, decision boundaries generated from six selected candidates are much closer to the genuine boundaries, compared to the approach in Fig. 1(b). Batch mode AL [10], [15] represents the typical set-assessment-based methods.

For set-assessment-based AL, existing solutions have proposed to do the following: 1) incorporate a density measure to reduce information overlap in an optimal subset, such as [8] and [9], and 2) rely on some search process, such as hill-climbing

Manuscript received June 16, 2011; revised April 24, 2012; accepted July 1, 2012. Date of publication August 17, 2012; date of current version April 16, 2013. This work was supported by the Australian Research Council Future Fellowships under Grant FT100100971. This paper was recommended by Associate Editor H. Kargupta.

Y. Fu and X. Zhu are with the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney, N.S.W. 2007, Australia (e-mail: Yifan.Fu@student.uts.edu.au; Xingquan.Zhu@uts.edu.au).

A. K. Elmagarmid is with the Qatar Computing Research Institute, Doha, Qatar (e-mail: aelmagarmid@qf.org.qa).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2012.2209177

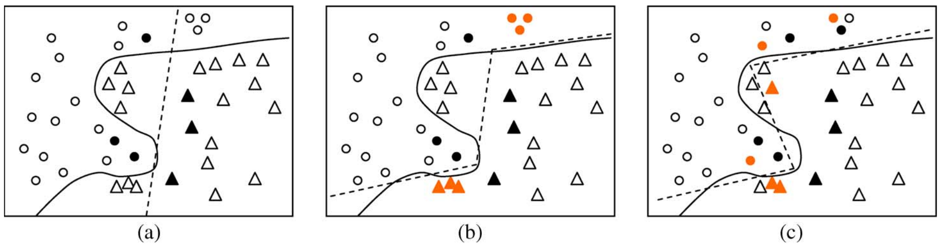

Fig. 1. Toy example demonstrating AL and labeling redundancy. Circles and triangles each denote one type of samples, with solid circles and triangles denoting labeled instances and the rest denoting unlabeled samples. The solid line denotes genuine decision boundaries, and dashed lines denote decision boundaries learned by learners (by using labeled samples). The uncertainty of the sample is calculated as $1/W(x)$, where $W(x)$ is the distance of the sample to the current decision boundary (i.e., samples close to the decision boundaries are the most uncertain ones [3]). (a) Decision boundaries learned from six labeled training samples. (b) By labeling six most uncertain samples, a learner refines its decision boundaries to approximate to the genuine decision boundaries. (c) By taking sample redundancy into consideration, a method chooses the most informative candidates with low redundancy among them, through which the learned decision boundaries are largely improved, compared to approaches considering uncertainty only.

search, to form an instance subset with a maximum modular score, such as [10]. Although these methods do take sample correlations into consideration for AL, in practice, they have at least two disadvantages as follows.

1) For clustering-based methods [8], [9], they normally separate instances into groups and select the centroid of each cluster as representative instances for labeling. Accordingly, AL crucially relies on the clustering results, whereas most clustering methods can only generate convex groups, and real-world data can distribute arbitrarily. In addition, because instance selection depends on cluster numbers and sizes and clustering is typically not a transparent process, these methods are inflexible for AL.
2) For batch mode AL [10], instance selection is made by using a search process, e.g., hill-climbing search, to choose an instance, one at a time, to maximize the objective function, and iterate k times to greedily include instance, one at a time, into the selected set to form a k-instance subset. Therefore, these works, in essence, still base on single-instance selection without considering each k-instance subset as a whole for AL.

In this paper, we propose a new paradigm for AL where instance correlation plays an essential role for selecting unlabeled samples for labeling. To capture instance correlations, we combine instance uncertainty and disparity to form a matrix with each element denoting correlation of instances indexed by the corresponding row and column. Using correlation matrix, AL can be regarded as an optimal subset selection problem to select a k-instance subset out of n samples, such that the selected subset has the maximum utility value. Compared to the existing *a priori*, the contribution of this paper is threefold:

1) Optimal subset *versus* the best individual: Individuals with the best capacity do not necessarily form an optimal subset, even if utility measures do take instance correlations into consideration. Our approach provides a new paradigm for AL by taking a group of instances as a whole for consideration.
2) A new utility measure for instance characterization: By considering instance correlations, the proposed measure combines instance uncertainty and disparity to capture instance level correlations. As a result, the selected subset will contain the best individuals with minimum redundant knowledge.
3) A general framework for AL: In addition to the clear optimization objective and theoretical basis, the theme of using instance correlations to form a matrix and employing semidefinite programming (SDP) to select an optimal subset can be applied to any utility measure and any learning algorithms.

The remainder of the paper is organized as follows. Section II reviews existing work on AL. The problem definition and system overview are introduced in Section III. The algorithm details are introduced in Section IV, followed by experiments in Section V. We conclude the paper in Section VI.

II. Related Work

The key point of AL is to find samples that are mostly critical to label, so that the inclusion of these instances into the labeled set will help improve the learning model. In practice, the assessment of each instance's "degree of importance" for labeling is undertaken by assessing the uncertainty of the unlabeled instances based on the model trained from the current labeled sample set. If an instance's uncertainty is high, it implies that the current model does not have knowledge in classifying the instance, and, presumably, including this sample into the training set can, therefore, help improve the model. A large number of methods have been proposed to quantify and assess sample uncertainty in different ways. Methods also exist to characterize "ghost" points for imbalance data classification [22] and quantify a subgroup of instances [30] and their inconsistency [17]. When reviewing these approaches from an instance-correlation perspective, they can be categorized into the following two categories: 1) AL based on individual assessment and 2) AL based on set assessment. In a recent survey paper [32], we have summarized the existing AL methods with or without considering instance correlations.

A. AL Based on Individual Assessment

Traditionally, AL is regarded as a query process where utility values of individual instances are calculated and sorted, according to some predefined criteria, with the objective of selecting the ones with maximum utility values for labeling.

Existing query strategies can be roughly divided into four categories. The simplest query framework is uncertainty sampling based on posterior probabilities, including margin sampling [12], entropy measure [13], and least confidence measures [14]. Query by committee [1] is another commonly used scheme, where a committee of classifiers is used to assess unlabeled samples, by measuring voting disagreements or divergences. A third category of methods use expected model change for discriminative probabilistic models, such as sequence labeling using conditional random fields [13]. In the fourth group of query strategies, the selection criterion is to find instances directly reducing model bias and/or variance [15], [16], such that learners trained from labeled instances are expected to achieve minimum error rates.

In the situation that an accurate labeler for an AL approach does not exist, "AL from crowds" [29] or some proactive cost-sensitive AL [31] approaches exist to identify the most useful annotator given that the labeling information may be provided by multiple "imperfect" annotators. For data with shared dependency structure information (such as graph or networked data where instances are linked through some relationships), AL methods [4] also exist to label the most informative node such that the classifier trained from labeled nodes can achieve the maximum accuracy. Because networked databased AL methods require instance correlations to be explicitly given, they cannot be generalized to handle generic data sets with no explicit relationships between instances. For all work in this area, the evaluation of the annotators and instances is based on individual assessment without considering group interaction between instances and annotators.

In summary, when determining which instance should be selected and labeled, all existing works in the aforementioned category treat instances as independent observations and disregard sample correlations. As a result, the selected label set may contain redundant instances and therefore limit learner performance, as we showed in Fig. 1.

B. AL Based on Set Assessment

Set-assessment-based methods calculate the utility values of an instance subset by taking the diversity of the instances of the subset (i.e., a batch) into consideration. For example, Wang *et al.* [26] propose *discriminative neighborhood metric learning* to build a correlation matrix from the entire unlabeled data set and then pick up the median point in each cluster based on the correlation matrix. Nguyen and Smeulders [27] utilize a *coarse-to-fine* adjustment mechanism in AL to avoid repeatedly selecting the same data in each cluster. In summary, most of these algorithms use clustering or greedy heuristics to ensure that selected instances in the subset are both diverse and uncertain. Brinker [8] and Xu *et al.* [9] propose AL approaches for support vector machines, which explicitly consider diversity by using clustering algorithm, and the centroid of each cluster is used to avoid redundancy in the selected subset.

To select an instance subset with low redundancy, Hoi *et al.* [10] uses hill-climbing search to greedily include instances, one at a time, into the selected set to form a k-instance subset. This method, in essence, is still based on single-instance selection without taking each k-instance subset as a whole for selection. Instead of employing a greedy search strategy, Guo and Greiner [15] employed expectation loss and Taylor approximation to find approximate solutions for instance subset selection. In order to find an instance subset with a minimum expected loss value, one has to retrain multiple models, by varying the selection of subset, to find the best selection. As a result, this method [15] is computationally expensive in practice.

In order to address the aforementioned issues, our work proposes to investigate a new AL strategy by taking an instance subset as a whole to assess its utility value, without involving clustering or heuristic search, and find the optimal subset for AL.

III. Problem Definition and System Overview

A. Problem Definition

Given is a data set $\mathcal{D}$ consisting of N instances with each instance x_i denoted by $x_i = \{x_{i,1}, \ldots, x_{i,m}; y_i\}$, where $x_{i,j}$ denotes the jth attribute value of the instance and y_i denotes x_i's class label. If x_i is unlabeled, we denote it by $x_i = \{x_{i,1}, \ldots, x_{i,m}; ?\}$. Assume that, at any stage, a labeled sample is moved from $\mathcal{D}$ to a subset $\mathcal{D}^L$, and the remaining unlabeled instances in $\mathcal{D}$ form an unlabeled subset $\mathcal{D}^U$, with $\mathcal{D} = \mathcal{D}^L \cup \mathcal{D}^U$ and $\mathcal{D}^L \cap \mathcal{D}^U = \emptyset$. The **aim** of optimal-subset-based AL is to select and label a batch (i.e., a subset Δ) of instances, one batch at a time, from $\mathcal{D}^U$, such that, when a user-requested number of instances are labeled, a classifier trained from $\mathcal{D}^L$ has the maximum prediction accuracy in classifying some previously unseen test instances T.

B. System Overview

In order to train classifiers from $\mathcal{D}^L$ with the maximum prediction accuracy, a commonly employed principle for AL is to label samples with high uncertainty. Following this principle, assume that a matrix $\mathcal{M}$ exists to capture each single instance's uncertainty as well as the disparity between any two instances x_i and x_j; the aforementioned AL **goal** can be regarded as the selection of an optimal subset of unlabeled samples Δ, such that the summation of instance uncertainty and disparity over Δ can reach the maximum (compared to any alternative subsets with the same size).

Accordingly, we employ an iterative procedure with the following three major steps.

1) **Classifier ensemble construction**: Use bootstrap sampling to train a committee of classifiers E from $\mathcal{D}^L$.
2) **Building correlation Matrix**: Applying E to unlabeled sample set $\mathcal{D}^U$ and build a correlation matrix $\mathcal{M}$ to capture instance uncertainty and disparity between instances.
3) **Optimal subset selection**: Use correction matrix $\mathcal{M}$ to select an optimal subset Δ which has the maximum utility value among all candidate sets with the same size.

This problem can be formulated as a quadratic integer programming problem as follows:

$$\max_{\mathbf{x}} \mathbf{x}^T \mathcal{M} \mathbf{x},$$
$$\text{s.t.} \sum_{i, x_i \in \mathbf{X}} x_i = k; \quad x_i \in \{0, 1\} \tag{1}$$

where $\mathbf{x}$ is an n-dimensional column vector and n is the size of unlabeled set $\mathcal{D}^U$. The constraint k defines the size of the subset for labeling, with $x_i = 1$ denoting that instance x_i is selected for labeling and $x_i = 0$ otherwise.

Assume that the objective function in (1) is properly solved; the optimal subset Δ will contain k instances with the maximum summation of the uncertainty and disparity, through which the instance labeling redundancy is reduced. Depending on the number of instances for labeling, the aforementioned process repeats until the data set is suitably labeled.

IV. ALOSS: Optimal Instance Subset Selection for AL

We now discuss technical details on correlation matrix construction and optimal subset selection for AL.

A. *Correlation Matrix Construction*

An important step of AL with optimal subset selection (ALOSS) is to construct a correlation matrix $\mathcal{M}$ with elements in $\mathcal{M}$ properly capturing each single instance's uncertainty as well as correlations between each instance pair. To build a correlation matrix $\mathcal{M} \in \mathbb{R}^{n \times n}$, where n denotes the number of instances in the unlabeled set $\mathcal{D}^U$, we separate elements in $\mathcal{M}$ into two parts. More specifically, assuming that $\mathcal{U}_{i,i}$ defines the uncertainty of instance x_i and $\mathcal{I}_{i,j}, i \neq j$ defines the disparity between instances x_i and x_j, the correlation matrix $\mathcal{M}$ is constructed using

$$\mathcal{M}_{i,j} = \begin{cases} \mathcal{U}_{i,j}, & \text{if } i = j \\ \mathcal{I}_{i,j}, & \text{if } i \neq j. \end{cases} \tag{2}$$

Classifier Weighting Matrix: To calculate instance uncertainty, we build a classifier ensemble E with π heterogeneous members, $\hbar_1, \ldots, \hbar_\pi$, each of which is trained from labeled sample set $\mathcal{D}^L$. Meanwhile, assuming that users intend to use a specific type of learning algorithm, for example, decision trees, to the final labeled samples and train a classifier to predict test instances, we also apply the same learning algorithm to $\mathcal{D}^L$ to train a benchmark classifier $\hbar^*$. In this paper, we call $\hbar^*$ a "benchmark" classifier because the AL goal is to ensure that a same type of classifier, such as decision tree, naive Bayes (NB), etc., trained from $\mathcal{D}^L$ will have the maximum accuracy. Because we use ensemble E with different types of base learners, the purpose of employing $\hbar^*$ is to make sure that the AL process indeed favors samples with respect to user-selected learning algorithm. In other words, if users want to build an NB classifier from the final labeled set $\mathcal{D}^L$ and expect this NB classifier to have the maximum accuracy, our AL process will use the NB classifier as the "benchmark" to guide the instance-selection process.

TABLE I
TOY EXAMPLE DEMONSTRATING THE CORRELATION MATRIX CONSTRUCTION PROCESS. THE DATA SET CONTAINS FOUR INSTANCES $\{x_1, x_2, x_3, x_4\}$ AND TWO CLASS LABELS TRUE (T) OR FALSE (F): (a) PREDICTION FROM A BENCHMARK CLASSIFIER $\hbar^*$ AND PREDICTIONS FROM THE THREE CLASSIFIERS OF AN ENSEMBLE $E = \{\hbar_1, \hbar_2, \hbar_3\}$ ON THE DATA SET, (b) $H_{i,j}$ VALUES FOR MATRIX H, AND (c) NORMALIZED MATRIX $\mathcal{H} = H^{\mathrm{T}} \times H/n$

(a)

Classifier	x_1	x_2	x_3	x_4
$\hbar^*$	T	F	F	T
$\hbar_1$	F	T	F	T
$\hbar_2$	T	T	F	T
$\hbar_3$	F	T	F	F

(b)

Instance	$\hbar_1$	$\hbar_2$	$\hbar_3$
x_1	0	1	0
x_2	0	0	0
x_3	1	1	1
x_4	1	1	0

(c)

Classifier	$\hbar_1$	$\hbar_2$	$\hbar_3$
$\hbar_1$	$\frac{2}{4}$	$\frac{2}{4}$	$\frac{1}{4}$
$\hbar_2$	$\frac{2}{4}$	$\frac{3}{4}$	$\frac{1}{4}$
$\hbar_3$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

Given classifier ensemble $E = \{\hbar_1, \ldots, \hbar_\pi\}$ and a benchmark classifier $\hbar^*$, for each ensemble member $\hbar_j$, we compare its prediction with the benchmark classifier $\hbar^*$ on each unlabeled sample in $\mathcal{D}^U$ (which contains n instances) and generate an n by π matrix $H \in \mathbb{R}^{n \times \pi}$ as follows.

1) $H_{i,j} = 1$, if $\hbar_j$ and $\hbar^*$ have the same prediction on x_i.
2) $H_{i,j} = 0$, otherwise.

By using the H matrix, which records agreements between each ensemble member and the benchmark classifier $\hbar^*$, we calculate a π by π matrix $\mathcal{H} \in \mathbb{R}^{\pi \times \pi}$ with $\mathcal{H} = H^{\mathrm{T}} \times H/n$. Clearly, each diagonal term $\mathcal{H}_{j,j}$ denotes the percentage of same predictions between $\hbar_j$ and $\hbar^*$, whereas each off-diagonal term $\mathcal{H}_{i,j}, i \neq j$ denotes the common agreements between classifiers $\hbar_i$ and $\hbar_j$. Table I demonstrates the correlation matrix construction process using a toy example.

Instance Uncertainty: To calculate uncertainty for each single instance x_i in $\mathcal{D}^U$, we apply each ensemble classifier $\hbar_j, j = 1, \ldots, \pi$ to x_i and build a vector $\mathbf{u}_i$ with each element $u_{i,j}, j = 1, \ldots, \pi$, recording the uncertainty of classifier $\hbar_j$ on instance x_i (in our experiments, we use *entropy* to measure $\hbar_j$'s uncertainty on x_i). As a result, we can build an n by π matrix $\mathbf{u}$ß$n\mathbb{R}^{n \times \pi}$ and calculate weighted instance uncertainty as follows:

$$\mathcal{U} = \mathbf{u} \times \mathcal{H} \times \mathbf{u}^{\mathrm{T}}. \tag{3}$$

According to (3), each diagonal term in $\mathcal{U}_{i,i}$ contains the weighted entropy of x_i with respect to all ensemble classifiers in E, which are trained from labeled samples $\mathcal{D}^L$.

Instance Disparity: The purpose of calculating the disparity between each pair of instances x_i and x_j is to capture the difference between x_i and x_j such that an optimal subset can contain instances with high uncertainty and high disparity (so there is low redundancy in the labeled samples). To achieve the goal, we employ two types of distance measures, prediction distance and feature distance, for disparity assessment.

Prediction distance ($\mathcal{P}$) intends to compare the prediction dissimilarity of a same set of classifiers on two instances. The purpose is to assess the behavioral difference between a pair of instances (x_i versus x_j) with respect to some classifiers. Given an instance x_i and a classifier $\hbar_\kappa$, assume that the labeling space has $\mathcal{Y}$ labels in total, so x_i can be predicted, by $\hbar_\kappa$, as any label $l \in [1, \mathcal{Y}]$. Denote $\hbar_\kappa^l(x_i)$ as the probability of x_i belonging to class l, as per classifier $\hbar_\kappa$'s prediction. For a pair of instances x_i and x_j, their prediction difference with respect to a classifier $\hbar_\kappa$ is denoted by

$$\mathbf{p}_{i,j}^{\hbar_\kappa} = \left(\left|\hbar_\kappa^1(x_i) - \hbar_\kappa^1(x_j)\right|, \ldots, \left|\hbar_\kappa^{\mathcal{Y}}(x_i) - \hbar_\kappa^{\mathcal{Y}}(x_j)\right|\right). \quad (4)$$

When combining prediction distance over all class labels $l \in [1, \mathcal{Y}]$ and all classifiers $\hbar_\kappa, \kappa = 1, \ldots, \pi$, we have

$$\mathcal{P}_{i,j} = \sum_{\kappa=1, \hbar_\kappa \in E}^{\pi} \sum_{l=1}^{\mathcal{Y}} \left|\hbar_\kappa^l(x_i) - \hbar_\kappa^l(x_j)\right| \times \mathcal{H}_{\kappa,\kappa} \quad (5)$$

where $\mathcal{H}_{\kappa,\kappa}$ denotes the weight of the classifier $\hbar_\kappa$, as we discussed in Section IV-A.

Feature distance ($\mathcal{F}$), as its name suggests, intends to capture the disparity of a pair of instances in the feature space. Given instance $x_i = \{x_{i,1}, \ldots, x_{i,m}; y_i\}$, where $x_{i,\kappa}$ denotes the κth feature value of x_i, the feature distance between x_i and x_j is calculated as follows:

$$\mathcal{F}_{i,j} = \sqrt{\sum_{\kappa=1}^{m} (x_{i,\kappa} - x_{j,\kappa})^2}. \quad (6)$$

Instance disparity ($\mathcal{I}$): Because prediction distance ($\mathcal{P}$) and feature distance ($\mathcal{F}$) each denote the difference between instances x_i and x_j from different perspectives, the final disparity between x_i and x_j is the product of the two distances as follows:

$$\mathcal{I}_{i,j} = \mathcal{P}_{i,j} \times \mathcal{F}_{i,j}. \quad (7)$$

By using the product of the prediction distance and feature distance to calculate the disparity between instances, as shown in (7), our intention is to simultaneously consider instances' behaviors (prediction distance) and their distance in feature space (feature distance). Assuming that prediction distance and feature distance each assess instance distribution from one dimension, the product therefore assesses the joint distribution from both dimensions and is a proper way of assessing instance disparity.

B. Optimal Subset Selection

Given a correlation matrix $\mathcal{M}$ with n instances, the purpose of optimal subset selection, as defined in the objective function (1), is to select a subset with k instances, such that the summation of all instances' uncertainty and their disparities is the maximum among all alternative subsets with the same size. This problem is actually a standard 0–1 optimization problem, which is NP-hard in general. SDP [18], fortunately, provides an approximate solution to solve similar NP-hard maximization problems with polynomial complexity. Accordingly, we transform the original problem in (1) to a "Max-Cut with size k" (MC-k) problem [18], [19], whose objective is to partition an edge-weighted graph (which contains N vertices) into two subsets, with one of which containing k vertices, such that the total weight of edges across the cut (i.e., partitioning) is maximized. A formal definition of the MC-k problem is given in (8), where N denotes the number of vertices in the graph and $W_{i,j}$ is the edge weight between vertices i and j

$$\max_y \frac{1}{2} \sum_{i\in[1,N], j\in[1,N], i<j}^{N} W_{i,j} \times (1 - y_i y_j), \\ \text{s.t.} \sum_i y_i = N - 2k; \quad y_i \in \{-1, 1\}. \quad (8)$$

To transform the original problem, defined in (1), into the MC-k problem as defined in (8), we transform variable x_i in (1) as follows:

$$x_i = \frac{y_i + 1}{2} \quad (9)$$

where $y_i \in \{-1, 1\}$. Replacing x_i in (1) using its form in (9), we have

$$\max_{\mathbf{y}} \frac{1}{4} (\mathbf{y} + \mathbf{e})^{\mathrm{T}} \mathcal{M} (\mathbf{y} + \mathbf{e}), \\ \text{s.t. } (\mathbf{y} + \mathbf{e})^{\mathrm{T}} I (\mathbf{y} + \mathbf{e}) = 4k; \quad y_i \in \{-1, 1\} \quad (10)$$

where $\mathbf{y}$ is an n-dimensional vector with values of either 1 or -1 and $\mathbf{e}$ is the same-sized column vector with all values being 1 s. The original cardinality constraint $\sum_i x_i = k$ is rewritten as a quadratic form $\mathbf{x}^{\mathrm{T}} I \mathbf{x} = k$, where I is an identity matrix.

To put transformed objective function in (10) and its cardinality constraints into quadratic form, we expand the vector $\mathbf{y} = (y_1, \ldots, y_n)$ into an extended form $\mathbf{y} = (y_0, y_1, \ldots, y_n)$ with $y_0 = 1$ and construct a new matrix $\mathcal{Q} \in \mathbb{R}^{(n+1)\times(n+1)}$ as follows:

$$\mathcal{Q} = \begin{pmatrix} \mathbf{e}^{\mathrm{T}} \mathcal{M} \mathbf{e} & \mathbf{e}^{\mathrm{T}} \mathcal{M} \\ \mathcal{M} \mathbf{e} & \mathcal{M} \end{pmatrix}. \quad (11)$$

Similarly, we can apply the same extension to the cardinality constraints and build a new constraint matrix $\mathcal{C} \in \mathbb{R}^{(n+1)\times(n+1)}$ as follows:

$$\mathcal{C} = \begin{pmatrix} n & e^{\mathrm{T}} \\ e & I \end{pmatrix}. \quad (12)$$

As a result, the original instance-selection problem in (1) is transformed into an MC-k problem as follows:

$$\max_{\mathbf{y}} \mathbf{y}^{\mathrm{T}} \mathcal{Q} \mathbf{y}, \\ \text{s.t. } \mathbf{y}^{\mathrm{T}} \mathcal{C} \mathbf{y} = 4k; \\ y_0 = 1; \quad y_i \in \{-1, 1\} \; \forall I \neq 0. \quad (13)$$

Solving MC-k Using SDP Programming: To solve (13), we denote $Y = \mathbf{y} \times \mathbf{y}^{\mathrm{T}}$, where $Y \in \mathbb{R}^{(n+1)\times(n+1)}$, and have an SDP form for (13) as follows:

$$\max_{\mathbf{y}} \mathcal{Q} \bullet Y, \\ \text{s.t. } \mathcal{C} \bullet Y = 4k; \\ y_0 = 1; \quad y_i \in \{-1, 1\} \; \forall I \neq 0. \quad (14)$$

In (14), $\bullet$ defines the dot product given as $A \bullet B = \sum_{i,j} A_{i,j} \times B_{i,j}$. We integrate the constraints on binary variable y_i. Because y_i has only two values 1 and -1, together with the constraint $y_0 = 1$, the diagonal terms in Y are all 1 s. Consequently, the constraints $y_i \in \{-1, 1\}$ can be expressed as $\mathrm{Diag}(Y) = I$, where I is an $(n+1)$-dimensional identity matrix.

Therefore, the SDP relaxation of (13) is denoted by

$$\begin{aligned} &\max_{\mathbf{Y}} \mathcal{Q} \bullet Y, \\ &\text{s.t.} \quad \mathcal{C} \bullet Y = 4k; \\ &\mathrm{Diag}(Y) = I; \quad Y \succeq 0 \end{aligned} \tag{15}$$

where $Y \succeq 0$ defines that symmetric matrix Y is positive semidefinite (i.e., all its eigenvalues are nonnegative). Following SDP problem formulation defined in (15), one can employ publicly available open source packages to solve (15). In our experiments, we use semi-definite programming algorithm [20], which is based on interior point method, to find solutions for (15).

C. ALOSS: System Framework

The whole process of ALOSS, as listed in Algorithm 1, is an iterative procedure. In each iteration, a small optimal subset Δ with k instances is selected for labeling. The labeled instances are then used to update models, including ensemble E and benchmark classifier $\hbar^*$, and help select another optimal subset Δ for labeling. The whole iterative process continues until the number of labeled instances $labeledSample$ reaches the user-defined value t.

Accelerated Process: In Algorithm 1, optimal subset selection is carried out on $\mathcal{M}$ which is a square matrix, and its size is determined by the size of the unlabeled data set $\mathcal{D}^U$. For a large-size matrix, finding solutions for SDP is computationally expensive. Alternatively, one can build a small matrix by removing samples whose uncertainty values are hopelessly small. Therefore, the instance subset selection only works on the remaining samples, which will, in turn, significantly reduce ALOSS's computational complexity. To achieve the goal, the key issue is to determine a proper threshold value and separate instances, according to their uncertainty values $\mathcal{H}$, into two subsets. For this purpose, we employ a histogram [28]-based automatic thresholding method to divide unlabeled instances into two groups: low-uncertainty group and high-uncertainty group, with the goal of separating samples into two groups such that their combined spread (intragroup variance) is minimal.

Given an unlabeled instance subset $\mathcal{D}^U$ with n instances, the uncertainty of each individual instance is denoted by $\mathcal{U}_{i,i}, I = 1, \ldots, n$, as given in (3). Assuming that the uncertainty values of all instances $\mathcal{D}^U$ are distributed between $[E_{\min}, E_{\max}]$, one can partition the range into L intervals with equal width, as defined by (16), and the number of instances whose entropies belonging to the jth interval is denoted by n_j

$$w = \frac{(E_{\max} - E_{\min})}{L}. \tag{16}$$

Assume that a threshold τ exists to separate instances in $\mathcal{D}^U$ into two groups $\mathcal{G}_{\mathrm{low}}(\tau)$ versus $\mathcal{G}_{\mathrm{high}}(\tau)$, as follows:

$$\begin{cases} \mathcal{G}_{\mathrm{low}}(\tau) = \{x_i | x_i \in \mathcal{D}^U; \quad \mathcal{U}_{i,i} \le E_{\min} + \tau \cdot w\} \\ \mathcal{G}_{\mathrm{high}}(\tau) = \{x_i | x_i \in \mathcal{D}^U; \quad \mathcal{U}_{i,i} > E_{\min} + \tau \cdot w\}. \end{cases} \tag{17}$$

Denote the percentage of samples in the lth interval by

$$P_l = \frac{\sum_{i=1;\quad x_i \in \mathcal{D}^U;\quad E_{\min}+l\cdot w<\mathcal{U}_{i,i}\le E_{\min}+(l+1)\cdot w}^{n} 1}{n}. \tag{18}$$

The respective percentages of samples in the groups $\mathcal{G}_{\mathrm{low}}(\tau)$ and $\mathcal{G}_{\mathrm{high}}(\tau)$, with respect to a given threshold τ, are then denoted by

$$\omega_{\mathrm{low}}(\tau) = \sum_{j=0}^{\tau} P_j \qquad \omega_{\mathrm{high}}(\tau) = \sum_{j=\tau+1}^{L-1} P_j. \tag{19}$$

The weighted means for each of group $\mathcal{G}_{\mathrm{low}}(\tau)$ and $\mathcal{G}_{\mathrm{high}}(\tau)$, with respect to the threshold τ, are then defined by

$$\mu_{\mathrm{low}}(\tau) = \sum_{j=0}^{\tau} \frac{(j+1) \cdot P_j}{\omega_{\mathrm{low}}(\tau)} \tag{20}$$

$$\mu_{\mathrm{high}}(\tau) = \sum_{j=\tau+1}^{L-1} \frac{(j+1) \cdot P_j}{\omega_{\mathrm{high}}(\tau)}. \tag{21}$$

Assume that the mean uncertainty level over the whole unlabeled data set $\mathcal{D}^U$ is calculated using

$$\mu = \sum_{j=0}^{L-1} (j+1) \cdot P_j. \tag{22}$$

The weighted intergroup variance, with respect to the given threshold τ, is then defined by

$$\sigma^2(\tau) = \omega_{\mathrm{low}}(\tau) \cdot [\mu_{\mathrm{low}}(\tau) - \mu]^2 + \omega_{\mathrm{high}}(\tau) \cdot [\mu_{\mathrm{high}}(\tau) - \mu]^2. \tag{23}$$

The main objective of the automatic thresholding method is to exhaustively search for an optimal threshold $\tau*$ that maximizes (23), which will, in turn, separate samples into two groups, $\mathcal{G}_{\mathrm{low}}(\tau*)$ and $\mathcal{G}_{\mathrm{high}}(\tau*)$, with maximum intergroup variance. The algorithm details for finding an automatic threshold to partition unlabeled samples $\mathcal{D}^U$ are shown in Algorithm 2.

Algorithm 1 ALOSS: Active Learning with Optimal Subset Selection

```
Input: (1) an unlabeled sample set $\mathcal{D}$; (2) $\pi$: # of classifiers
    to form an ensemble $E$; (3) $t$: # of samples selected for
    labeling; (4) $\hbar$: a learning algorithm for training final
    classifiers; and (5) $k$: the size of optimal subset
    (or batch size).
Output: an labeled sample set $\mathcal{D}^L$ with $t$ labeled samples.
1: $labeledSample \leftarrow$ A small random number;
2: $\mathcal{D}^L \leftarrow$ Randomly label $labeledSample$ instances from $\mathcal{D}$;
3: $\mathcal{D}^U \leftarrow \mathcal{D} \setminus \mathcal{D}^L$;
```

```
4: while $labeledSample < t$ do
5:     $\hbar^* \leftarrow$ Apply $\hbar$ to $D^L$ to train a benchmark classifier;
6:     $E = \{\hbar_1, \ldots, \hbar_\pi\} \leftarrow$ Apply bootstrap samplings to $D^L$
       and build ensemble $E$ with heterogeneous classifiers;
7:     $\mathcal{H} \leftarrow$ Apply $E$ and $\hbar^*$ to $\mathcal{D}^U$ and build instance
       uncertainty matrix;
8:     $\mathcal{D}^{U'} \leftarrow$ Refining $\mathcal{D}^U$ using accelerated search process in
       Algorithm 2;
9:     $\mathcal{I} \leftarrow$ Calculate disparity matrix for instances in $\mathcal{D}^{U'}$;
10:    $\mathcal{M} \leftarrow$ Build instance-correlation matrix;
11:    $\Delta \leftarrow$ Apply optimal subset selection to $\mathcal{M}$ and select a
       subset $\Delta$ with $k$ instances;
12:    $D^L \leftarrow D^L \bigcup \Delta$;   $D^U \leftarrow D^U \setminus \Delta$;
13:    $labeledSample \leftarrow labeledSample + k$;
14: end while
```

Algorithm 2 AP: Accelerated Process using instance selection

```
Input: (1) weighted uncertainty values for all unlabeled
   instances in $\mathcal{D}^U$, $\mathcal{U}_{i,i}, i = 1, \ldots, n$; (2) $L$: # of levels in
   separating instance uncertainty values $[0, L-1]$
Output: selected instance subset
1: $w \leftarrow$ determining the step value for separating uncertainty
   values as shown in (16).
2: $\tau* \leftarrow 0$; $\sigma(\tau*) \leftarrow 0$; initializing optimal threshold and
   corresponding maximum variance value as 0.
3: for $j = 0$ to $L - 1$ do
4:     $\tau \leftarrow j$; setting current threshold.
5:     $\mathcal{G}_{\text{low}}(\tau), \mathcal{G}_{\text{high}}(\tau) \leftarrow$ separating two groups as shown in
       (17).
6:     $\sigma^2(\tau) \leftarrow$ calculating intergroup variance as shown in
       (23)
7:     if $\sigma^2(\tau*) < \sigma^2(\tau)$ then
8:         $\tau* \leftarrow \tau$; updating threshold value.
9:         $\sigma^2(\tau*) \leftarrow \sigma^2(\tau)$; updating optimal variance.
10:        $\mathcal{G}_{\text{high}}(\tau*) \leftarrow \mathcal{G}_{\text{high}}(\tau)$; updating optimal subset.
11:    end if
12: end for
13: return $\mathcal{G}_{\text{high}}(\tau*)$.
```

D. Time Complexity Analysis

The total time complexity of ALOSS includes two major parts: 1) building instance-correlation matrix $\mathcal{M}$ and 2) applying SDP to $\mathcal{M}$ to select a k-instance subset. We assume that a learning algorithm with quadratic complexity $O(t^2)$ is used, where t is the maximum number of labeled instances. In each iteration, there are one benchmark classifier and π ensemble members that need to be trained. In addition, the calculation of the disparity matrix needs pairwise instance correlation, which requires $O(n^2)$ time complexity, where n denotes the number of unlabeled instances. Therefore, in total, the time complexity for building instance-correlation matrix is

$$O(\mathcal{M}) = (\pi + 1)O(t^2) + O(n^2). \tag{24}$$

TABLE II
SIMPLE DESCRIPTION OF THE BENCHMARK DATA

ID	Dataset	Instances	Features	Classes
1	horse	368	23	2
2	auto-mpg	398	8	3
3	balance	625	5	3
4	pima	768	9	2
5	vehicle	846	19	4
6	german	1000	21	2
7	cmc	1473	10	3
8	car	1728	7	4
9	segment	2310	19	7
10	abalone	3196	37	2

Due to the accelerated search process (Algorithm 2), we can reduce the correlation matrix from $\mathcal{M} \in \mathbb{R}^{n \times n}$ to $\mathcal{M} \in \mathbb{R}^{\alpha \cdot n \times \alpha \cdot n}$, where $\alpha \in [0, 1]$ is the percentage of reduction. The SDP process requires $O(SDP) = O(k^2 + (\alpha \cdot n)^3)$ [33] complexity to solve the $n \times n$ matrix and selects a k-instance subset. Because the size of the instance subset k is much smaller than $\alpha \cdot n$, we can regard that SDP's time complexity is bounded by $O((\alpha \cdot n)^3)$.

Because the whole AL process requires the repetitive training of the classifiers, the while loop between steps 4 and 14 in Algorithm 1 needs to repeat t/k times, so the total time complexity of ALOSS is given as follows:

$$O(ALLOS) = \frac{t}{k}\left[(\pi+1)O(t^2) + O(n^2) + O\big((\alpha \cdot n)^3\big)\right]. \tag{25}$$

The aforementioned complexity analysis indicates that the bottleneck time complexity of ALOSS is asymptotically bounded by the SDP process.

V. EXPERIMENTS

We implement ALOSS and a number of baseline approaches using Java and WEKA data mining tools [23] and comparatively study their performance on ten benchmark data sets collected from the University of California, Irvine (UCI) machine learning data repository [24]. A simple description of the benchmark data sets is summarized in Table II. To comparatively study the algorithm performance, we compare classifiers trained from sample sets labeled by different methods. If a classifier trained from a sample set labeled by method A has a higher accuracy than the classifier trained from a sample set labeled by method B, we conclude that A has a better AL performance than B. To make fair comparisons, all methods are compared based on the same training and test instances (the initial randomly labeled instances are also the same for all methods). All experiments are based on ten times tenfold cross-validation. To build ensemble E with diverse classifiers, we employ four learning algorithms, including the following: 1) decision trees; 2) NB; 3) ZeroR (a classifier predicting samples to the majority class); and 4) multilayer perception, to build an ensemble with $\pi = 8$ classifiers, with each of them contributing two classifiers.

A. Benchmark Methods

In addition to the proposed ALOSS approach, we also implement a number of mainstream AL methods [25].

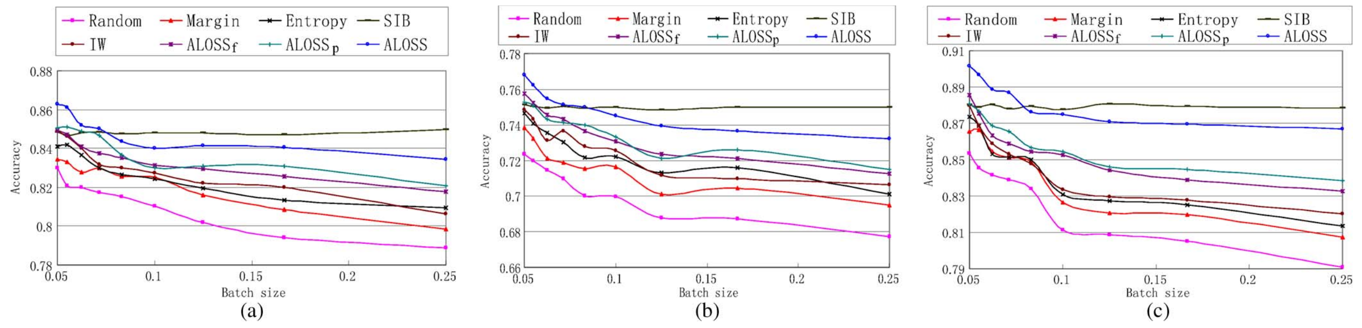

Fig. 2. Accuracy comparisons with respect to different batch sizes for AL. (a) Horse (two classes). (b) Auto-mpg (three classes). (c) Car (four classes).

Random is a simple approach, which randomly selects user-requested number of instances for labeling.

Entropy is a query-by-uncertainty AL method, which uses entropy as the uncertainty measure. Each instance x_i's entropy is calculated by using class distributions predicted from a classifier, as defined by (26), where $P(y_l|x_i)$ is the probability of x_i belonging to class y_l

$$x_H^* = \arg\max_{x_i} -\sum_l P(y_l|x_i)\log P(y_l|x_i). \tag{26}$$

All unlabeled samples are sorted according to their uncertainty values. A number of samples ranked on the top of the list form a batch, which is selected for labeling. The labeling process repeats until the user-requested number of instances are labeled.

Margin is identical to Entropy except the underlying uncertainty measure. Instead of using entropy, Margin aims to seek the difference between the two most likely class labels on a specific instance, as defined by

$$x_H^* = \arg\min_{x} P(y_1|x) - P(y_2|x) \tag{27}$$

where y_1 and y_2 are the first two most probable class labels, with x being classified by a classifier. Intuitively, an instance with the least margin value is the one which is mostly ambiguous (i.e., uncertain). Similar to Entropy, Margin also employs a batch-based iterative process for AL.

SIB is a single-instance batch method which is identical to *Entropy* except that, for SIB, the batch size is one, which means that SIB selects and labels an instance one at a time. The main purpose of using SIB as a baseline method is to check that if we try to remove the redundancy during the batch-based AL process, what is the best performance that the existing method can possibly achieve? Because SIB has the smallest batch size, it is the most computationally expensive method.

IW is a most recently developed instance weighting (IW)-based method [25]. For each unlabeled instance x_t, IW generates a weight value for x_t according to its maximum loss value on a set of benchmark classifiers, as defined in (28), where $\mathcal{L}()$ defines a loss function and f and g each denote a classifier in a set $\hbar$. The higher the p_{x_t}, the more likely x_t is going to be selected for labeling

$$p_{x_t} = \max_{f,g\in\hbar; y\in Y} \left(\mathcal{L}\left(f(x_t),y\right) - \mathcal{L}\left(g(x_t),y\right)\right). \tag{28}$$

$\mathbf{ALOSS}_p$ and $\mathbf{ALOSS}_f$ are variants of the proposed ALOSS approach, where instance disparity $\mathcal{I}$ only considers prediction distance (ALOSS_p) or feature distance (ALOSS_f), respectively, as introduced in Section IV-A.

For fairness of comparison, all baseline algorithms, except SIB, are designed to work in a "set assessment" mode by selecting the top k instances with the largest utility values one at a time. SIB labels one instance each iteration and will repeat k times to form a k-instance subset.

In the following sections, we first study algorithm performance in different parameter settings, including different batch sizes, different labeling portions, etc., and then report their performance on all benchmark data sets.

B. *Experimental Results With Different Batch Sizes*

In Fig. 2, we report the performance of different algorithms on three data sets, where the x-axis shows the batch size and the y-axis reports the accuracies of the classifiers trained from final labeled samples (by using different AL methods). In our experiments, we use decision trees (using J4.8 implementation) as the benchmark learner, and AL is used to label 50% of the samples. For example, for batch size 0.05 (which means 5% of the samples), an AL algorithm needs to repeat ten times in order to label 50% of the samples (in our experiments, we vary the number of labeling iterations from ten to two with step -1 to label 50% of the samples, which correspond to batch sizes, $0.5/10 = 0.05, 0.5/9 = 0.056, \ldots, 0.5/2 = 0.25$).

As the batch size increases, the performance of all methods, except SIB, deteriorates (the batch size of SIB is fixed to one so its performance remains stable across all batch sizes). This is because the labeled samples are fixed to 50%, and for a smaller batch size, an active learner will have more iterations to update its instance-selection process. Interestingly, the results in Fig. 2 also show that, as the batch size increases, the performance gain of ALOSS, in comparison with other methods, continuously improves. For example, when using batch size 0.05, the absolute performance gain of ALOSS, compared to Margin, is 3.7%, from 86.5% to 90.2%. When increasing the batch size to 0.25, the performance gain increases to 6.1%, from 80.7% to 86.8%. The same results can be observed from other data sets in Fig. 2. This asserts that the optimal subset selection procedure in ALOSS does play an effective role for avoiding redundancy and selecting

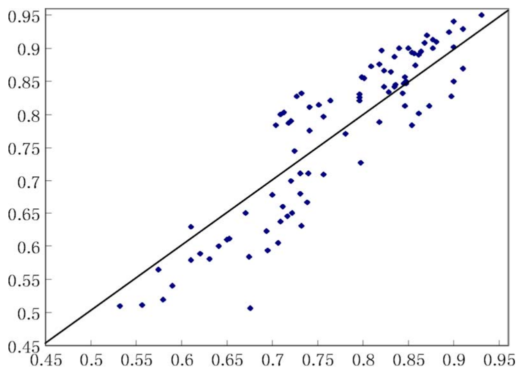

Fig. 3. Head-to-head comparisons between prediction-distance- and feature-distance-based instance disparity measures. The x-axis denotes the accuracy of ALOSS$_p$, and the y-axis denotes the accuracy of ALOSS$_f$. A value above the $y = x$ line indicates that ALOSS$_f$ outperforms ALOSS$_p$ and vice versa.

informative samples for labeling. For small batch sizes, the effect of redundant samples may be marginal because for a batch only contains several instances the redundancy among the instances may not be significant, in addition, a learning algorithm may also need the redundant information to build correct decision logics. For large batch sizes, simply sorting all samples according to their uncertainty, without considering their correlations, will introduce significant redundancy in the labeled samples.

To demonstrate algorithm performance in most rigorous conditions, in the following experiments, we use batch size 0.05 for all experiments.

C. Comparisons Between Different Distance Measures

In Fig. 3, we compare the performance of ALOSS$_p$ and ALOSS$_f$ on all benchmark data sets. In our experiments, we fix the batch size to 0.05, and the initial randomly labeled samples is set as 0.05. Then, we apply ALOSS$_p$ and ALOSS$_f$ to label different percentages of samples, with the labeling percentages varying from 10% to 50%. Because batch size is 0.05, for each data set, it will generate nine results, which correspond to the labeling percentages 10%, 15%, 20%, ..., 50%. From ten benchmark data sets, we can generate 90 pairs of accuracy values in total. We report all 90 pairs of accuracies in Fig. 3, where a value above the $y = x$ line indicates that ALOSS$_f$ outperforms ALOSS$_p$ and vice versa.

Among all 90 observations, ALOSS$_p$ outperforms ALOSS$_f$ on 55 cases, which asserts that, instead of considering feature values to assess instance correlations, like existing correlation-based AL algorithms do [27], using behaviors of instances with respect to different classifiers is an effective way of assessing instance correlations. Interestingly, for classifiers with relatively high accuracies, ALOSS$_f$ appears to have a better chance of outperforming ALOSS$_p$ (notice that, when the accuracies of the classifiers are less than 70%, ALOSS$_f$ only outperforms ALOSS$_p$ once out of 18 tests). This is mainly because ALOSS$_f$ relies on Euclidean distance, which is essentially a nearest neighborhood approach, to assess sample correlations. If the accuracies of the classifiers on the data sets are relatively low, it means that the decision concepts of the data sets are relatively

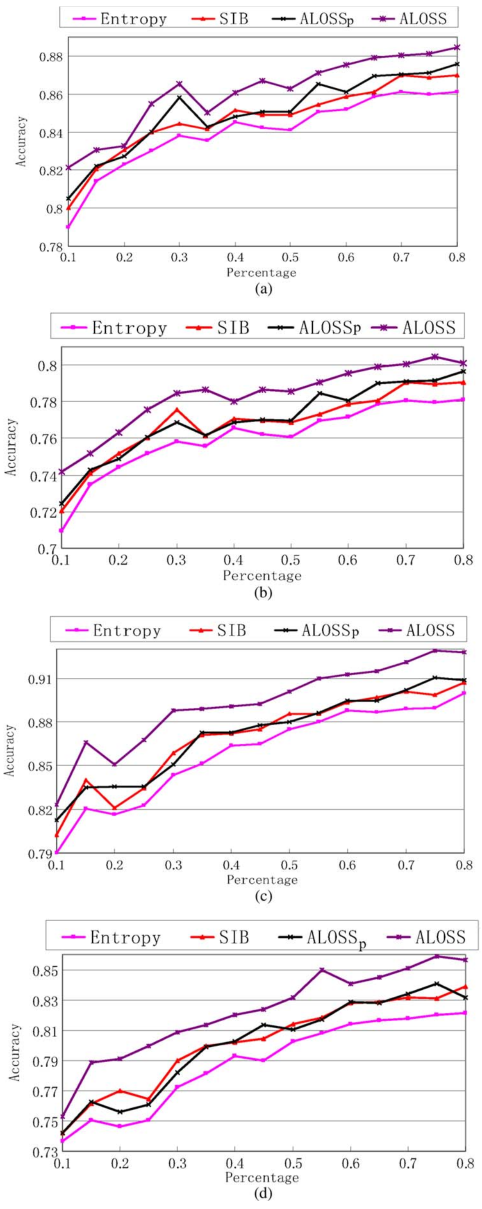

Fig. 4. Accuracy comparisons with respect to different portions of labeled instances. (a) Horse (decision trees). (b) Horse (NB). (c) Car (decision trees). (d) Car (NB).

complex, so using feature-based distance function is ineffective to differentiate samples from different classes. Imaging a two-class data set (positive and negative) with completely random instances and class labels, the best classification accuracy we can achieve for the data set is 50% (because instances and their class labels are completely random). Under such circumstances, Euclidean distance function is not able to assess the genuine disparity between instances. This is because, even if ALOSS$_f$ shows that instance pair (x_i, x_j) has a larger disparity value than (x_i, x_k), x_j, by no means, is a better selection than x_k (because everything is random, so all instances are supposed to have the same disparity to x_i). On the other hand, for the same random data set, the prediction-distance-based disparity calculated by ALOSS$_p$ will conclude that (x_i, x_j) has the same disparity value as (x_i, x_k) because classifiers have random predictions for all instances (x_i, x_j, and x_k).

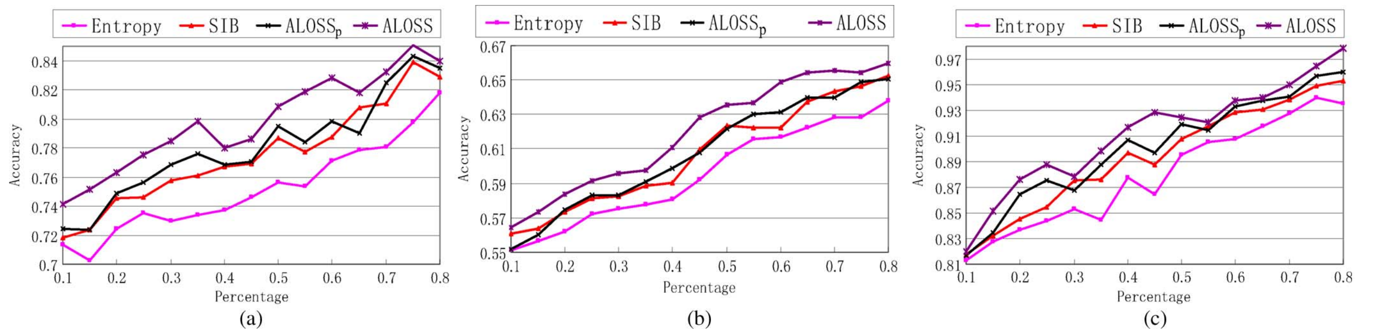

Fig. 5. Accuracy comparisons with respect to different portions of labeled samples (the batch size is fixed to 0.05). (a) Pima (two classes). (b) Auto-mpg (three classes). (c) Segment (seven classes).

TABLE III
DETAILED ALGORITHM PERFORMANCE COMPARISONS (DECISION TREES ARE USED AS THE BENCHMARK LEARNER, AND THE LABELING PERCENTAGE IS 15%

Dataset	Random	Margin	Entropy	SIB	IW	ALOSS$_p$	ALOSS
horse	$82.85_{\pm1.20}$	$83.12_{\pm1.09}$	$82.98_{\pm1.70}$	$83.02_{\pm1.28}$	$82.95_{\pm1.76}$	$83.17_{\pm1.69}$	**83.78**$_{\pm1.31}$
auto-mpg	$67.73_{\pm2.16}$	$67.75_{\pm2.96}$	$67.97_{\pm1.12}$	$68.52_{\pm1.68}$	$67.04_{\pm2.62}$	$68.21_{\pm2.13}$	**69.77**$_{\pm1.87}$
balance	$62.13_{\pm1.19}$	$63.20_{\pm2.9}$	$62.91_{\pm1.28}$	$63.94_{\pm1.94}$	$63.36_{\pm1.2}$	$63.94_{\pm1.23}$	**64.32**$_{\pm2.24}$
pima	$69.55_{\pm1.48}$	$70.11_{\pm1.88}$	$70.24_{\pm1.68}$	$70.96_{\pm1.34}$	$70.89_{\pm1.96}$	$70.75_{\pm1.94}$	**71.88**$_{\pm1.91}$
vehicle	$67.98_{\pm1.12}$	$69.03_{\pm1.93}$	$68.89_{\pm1.14}$	$69.56_{\pm1.98}$	$68.05_{\pm1.43}$	$69.17_{\pm1.45}$	**70.59**$_{\pm2.12}$
german	$68.54_{\pm1.13}$	$67.21_{\pm1.73}$	$68.76_{\pm1.38}$	$69.64_{\pm1.71}$	$68.56_{\pm1.56}$	$70.09_{\pm2.05}$	**70.13**$_{\pm1.57}$
cmc	$57.23_{\pm1.16}$	$58.90_{\pm1.78}$	$57.98_{\pm2.06}$	$58.98_{\pm1.76}$	$59.01_{\pm2.19}$	$59.23_{\pm1.82}$	**60.12**$_{\pm2.17}$
car	$83.35_{\pm1.93}$	$84.76_{\pm1.75}$	$84.68_{\pm1.25}$	$85.07_{\pm1.73}$	$83.58_{\pm2.87}$	$85.45_{\pm1.25}$	**86.78**$_{\pm2.15}$
segment	$82.29_{\pm3.08}$	$82.75_{\pm2.12}$	$83.84_{\pm2.17}$	**85.06**$_{\pm1.25}$	$84.98_{\pm1.25}$	$83.78_{\pm1.69}$	$84.57_{\pm1.95}$
analone	$80.78_{\pm1.56}$	$82.85_{\pm1.2}$	$81.67_{\pm1.34}$	$82.56_{\pm1.76}$	$80.21_{\pm3.21}$	$83.23_{\pm1.35}$	**84.72**$_{\pm2.18}$
Average	$72.24_{\pm1.60}$	$72.96_{\pm1.93}$	$72.99_{\pm1.51}$	$74.63_{\pm1.69}$	$72.86_{\pm2.00}$	$73.72_{\pm1.66}$	**75.31**$_{\pm2.04}$

TABLE IV
DETAILED ALGORITHM PERFORMANCE COMPARISONS (DECISION TREES ARE USED AS THE BENCHMARK LEARNER, AND THE LABELING PERCENTAGE IS 30%

Dataset	Random	Margin	Entropy	SIB	IW	ALOSS$_p$	ALOSS
horse	$83.31_{\pm1.08}$	$83.96_{\pm1.13}$	$84.07_{\pm1.29}$	$84.31_{\pm1.38}$	$84.11_{\pm1.43}$	$84.53_{\pm2.43}$	**85.47**$_{\pm1.76}$
auto-mpg	$68.36_{\pm1.81}$	$71.08_{\pm1.60}$	$69.97_{\pm1.05}$	$70.88_{\pm1.34}$	$69.04_{\pm1.68}$	$71.85_{\pm2.08}$	**72.06**$_{\pm1.45}$
balance	$63.54_{\pm1.2}$	$64.73_{\pm2.39}$	$64.85_{\pm1.69}$	$65.09_{\pm1.07}$	$65.18_{\pm1.38}$	$65.25_{\pm2.06}$	**66.25**$_{\pm1.46}$
pima	$70.91_{\pm1.10}$	$71.15_{\pm1.76}$	$70.96_{\pm1.25}$	$71.68_{\pm1.60}$	$71.72_{\pm1.01}$	$71.97_{\pm1.56}$	**72.78**$_{\pm1.38}$
vehicle	$69.06_{\pm1.15}$	$70.87_{\pm1.25}$	$70.08_{\pm1.95}$	$71.51_{\pm2.06}$	$71.06_{\pm2.2}$	$71.37_{\pm1.12}$	**72.56**$_{\pm2.56}$
german	$69.64_{\pm1.01}$	$69.75_{\pm1.02}$	$68.98_{\pm1.60}$	$69.62_{\pm1.71}$	$70.29_{\pm2.03}$	$71.79_{\pm1.45}$	**72.89**$_{\pm1.96}$
cmc	$58.01_{\pm1.09}$	$60.07_{\pm1.98}$	$59.80_{\pm2.31}$	$61.52_{\pm1.83}$	**62.43**$_{\pm2.32}$	$61.04_{\pm2.24}$	$62.26_{\pm2.12}$
car	$85.35_{\pm1.85}$	$86.12_{\pm1.22}$	$85.93_{\pm1.85}$	$86.72_{\pm1.74}$	$85.58_{\pm2.87}$	$87.05_{\pm2.12}$	**88.78**$_{\pm2.15}$
segment	$85.24_{\pm1.03}$	$85.62_{\pm2.29}$	$86.02_{\pm1.05}$	$86.73_{\pm1.60}$	$85.23_{\pm2.83}$	$86.27_{\pm1.56}$	**87.98**$_{\pm1.05}$
analone	$81.78_{\pm1.43}$	$83.65_{\pm1.47}$	$82.56_{\pm1.82}$	$83.71_{\pm1.91}$	$82.13_{\pm2.21}$	$85.34_{\pm1.72}$	**86.12**$_{\pm1.65}$
Average	$73.52_{\pm1.27}$	$74.7_{\pm1.61}$	$74.32_{\pm1.58}$	$75.17_{\pm1.62}$	$74.67_{\pm1.99}$	$75.64_{\pm1.83}$	**76.82**$_{\pm1.75}$

D. Results With Different Percentages of Labeled Samples

In Figs. 3 and 4, we report the algorithm performance with respect to different percentages of labeled samples, which vary from 10% to 80% (as indexed by the x-axis). The y-axis in the figures shows the accuracies of the classifiers trained from the corresponding labeled samples. For each benchmark data set, we use two types of benchmark learners, decision trees and NB, and report their accuracies in Fig. 4. Meanwhile, in Fig. 3, we compare the performances of different algorithms by using two-class, three-class, and seven-class benchmark data sets. Therefore, we can observe algorithm performance with respect to an increasing number of class labels. For all experiments, we fix the batch size to 0.05, and the initial randomly labeled subset is 0.05. Detailed results of each method on all benchmark data sets are reported in the next section (Fig. 5).

The results in Figs. 3 and 4 show that the Entropy method, without considering sampling redundancy, is the least effective algorithm mainly because the instance uncertainty is calculated based on each sample's own information, and the correlation between samples is ignored. By employing optimal subset selection, we observe that ALOSS constantly outperforms SIB. Because the batch size for SIB is set as one and a smaller batch often results in a better labeling quality, SIB represents the performance upper bound of individual-assessment-based active methods. By combing instance uncertainty and instance disparity together to select optimal subsets for AL, ALOSS is shown to outperform the upper bound to a large extent.

E. Detailed Comparisons for All Methods

In Tables III–V, we report detailed comparisons of all methods on the ten benchmark data sets. The detailed results reported in the tables are based on different percentages of labeled instances, varying between 15% (Table III),

TABLE V
Detailed Algorithm Performance Comparisons (Decision Trees Are Used as the Benchmark Learner, and the Labeling Percentage Is 50%

Dataset	Random	Margin	Entropy	SIB	IW	ALOSS$_p$	ALOSS
horse	$84.37_{\pm 1.51}$	$84.75_{\pm 1.07}$	$84.87_{\pm 1.45}$	$85.12_{\pm 1.47}$	$84.98_{\pm 2.56}$	$85.34_{\pm 1.60}$	**86.18**$_{\pm 1.06}$
auto-mpg	$74.08_{\pm 1.98}$	$75.17_{\pm 1.69}$	$75.01_{\pm 2.08}$	$75.92_{\pm 1.69}$	$74.05_{\pm 1.2}$	$76.07_{\pm 1.12}$	**77.09**$_{\pm 2.29}$
balance	$65.92_{\pm 1.03}$	$65.75_{\pm 2.35}$	$65.89_{\pm 1.48}$	$66.34_{\pm 1.19}$	$66.94_{\pm 1.39}$	$66.89_{\pm 2.36}$	**67.21**$_{\pm 1.96}$
pima	$72.96_{\pm 1.10}$	$72.92_{\pm 1.29}$	$73.05_{\pm 1.12}$	$73.99_{\pm 2.01}$	$73.07_{\pm 1.19}$	$73.87_{\pm 1.23}$	**74.92**$_{\pm 1.23}$
vehicle	$70.98_{\pm 1.13}$	$72.01_{\pm 2.8}$	$71.45_{\pm 1.34}$	$72.89_{\pm 1.65}$	$72.09_{\pm 2.1}$	$73.56_{\pm 1.18}$	**74.78**$_{\pm 3.56}$
german	$70.78_{\pm 1.01}$	$70.96_{\pm 1.04}$	$71.08_{\pm 1.56}$	$71.78_{\pm 1.67}$	$71.09_{\pm 2.53}$	$72.67_{\pm 1.95}$	**73.27**$_{\pm 1.65}$
cmc	$59.03_{\pm 1.75}$	$61.09_{\pm 2.08}$	$60.45_{\pm 1.34}$	$61.86_{\pm 1.12}$	$62.03_{\pm 1.23}$	$62.34_{\pm 1.71}$	**63.54**$_{\pm 2.56}$
car	$86.97_{\pm 1.93}$	$88.15_{\pm 1.74}$	$87.37_{\pm 2.16}$	$88.36_{\pm 1.92}$	$87.09_{\pm 3.21}$	$88.96_{\pm 1.46}$	**90.13**$_{\pm 2.96}$
segment	$86.24_{\pm 1.25}$	$86.02_{\pm 2.89}$	$87.12_{\pm 1.12}$	$87.89_{\pm 1.67}$	$87.23_{\pm 2.13}$	$88.31_{\pm 1.72}$	**89.98**$_{\pm 1.34}$
analone	$83.78_{\pm 1.23}$	$84.87_{\pm 2.12}$	$84.65_{\pm 1.51}$	$85.71_{\pm 1.65}$	$84.23_{\pm 1.22}$	$86.12_{\pm 1.34}$	**88.12**$_{\pm 1.89}$
Average	$75.51_{\pm 1.39}$	$76.17_{\pm 1.91}$	$76.09_{\pm 1.51}$	$76.98_{\pm 1.60}$	$76.28_{\pm 1.87}$	$77.41_{\pm 1.57}$	**78.52**$_{\pm 2.05}$

30% (Table IV), and 50% (Table V). In all tables, the batch size is fixed to 0.05.

Among all methods, ALOSS achieves the best performance gain; ALOSS$_p$ and SIB are the second tier. The IW-based approach, however, marginally outperforms random sample selection. As we have discussed in (28), the IW in IW is essentially an individual-assessment-based measure, where the importance (i.e., the weight value) of each instance is calculated according to its loss value with respect to some classifiers. Without taking the instance correlation into consideration, IW only aims to label individual instances with the objective of minimizing the total loss of the system. Our experiments assert that instance correlations play an important role for AL methods to select informative and less redundant instances for labeling.

Intuitively, SIB intends to avoid sample labeling redundancy by selecting an instance one at a time for labeling. Such a hill-climbing instance labeling approach, however, is still inferior to ALOSS, although SIB indeed outperforms most other methods (despite the high computational costs of SIB). Indeed, although SIB intends to minimize redundancy in the sample selection process, it has no mechanism to ensure that samples selected in a consecutive number of iterations can form an optimal subset. This is similar to most hill-climbing search methods which constantly move toward each local optimal but may be eventually stuck to the local optimal and fail to find global optimal solutions. ALOSS inherently avoids the problem through the selection of an optimal subset by taking sample correlations into consideration.

VI. Conclusion

In this paper, we have proposed a new AL paradigm using optimal subset selection (ALOSS). Instead of treating each unlabeled instances as I.I.D. objects and assessing their utility values without considering sample correlations, ALOSS regards sample correlation as a key aspect for selecting the most important instance subset for labeling. To achieve the goal, ALOSS uses instance uncertainty and instance disparity to build an instance-correlation matrix, so the AL problem is transformed into an SDP problem which selects a subset with an optimal utility value. The employment of the correlation matrix and the corresponding optimization goal lay solid theoretical foundations to explain why ALOSS outperforms its peers. Experimental comparisons have demonstrated that ALOSS outperforms state-of-the-art active learners.

References

[1] H. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proc. COLT*, 1992, pp. 287–294.
[2] A. McCallum and K. Nigam, "Employing EM in pool-based active learning for text classification," in *Proc. ICML*, 1998, pp. 359–367.
[3] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," in *Proc. ICML*, 2000, pp. 999–1006.
[4] M. Bilgic, L. Mihalkova, and L. Getoor, "Active learning for networked data," in *Proc. ICML*, 2010, pp. 79–86.
[5] H. Osmanbeyoglu, J. Wehner, J. Carbonell, and M. Ganapathiraju, "Active machine learning for transmembrane helix prediction," *J. BMC Bioinformat.*, vol. 11, no. S58, pp. 1–9, Jan. 2010.
[6] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from stream data using optimal weight classifier ensemble," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 6, pp. 1607–1621, Dec. 2010.
[7] J. Wang, E. Sung, and W. Yau, "Active learning for solving the incomplete data problem in facial age classification by the furthest nearest neighbor criterion," *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 2049–2062, Jul. 2011.
[8] K. Brinker, "Incorporating diversity in active learning with support vector machines," in *Proc. ICML*, 2003, pp. 59–66.
[9] Z. Xu, R. Akella, and Y. Zhang, "Incorporating diversity and density in active learning for relevance feedback," in *Proc. ECIR*, 2007, pp. 246–257.
[10] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu, "Batch mode active learning and its application to medical image classification," in *Proc. ICML*, 2006, pp. 417–424.
[11] Y. Guo and D. Schuurmans, "Discriminative batch mode active learning," in *Proc. NIPS*, 2007, pp. 1–8.
[12] H. Abe and H. Mamitsuka, "Query learning strategies using boosting and bagging," in *Proc. ICML*, 1998, pp. 1–9.
[13] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proc. Conf. Emp. Methods NLP*, 2008, pp. 1–10.
[14] A. Culotta and A. McCallum, "Reducing labeling effort for structured prediction tasks," in *Proc. AAAI*, 2005, pp. 746–751.
[15] Y. Guo and R. Greiner, "Optimistic active learning using mutual information," in *Proc. IJCAI*, 2007, pp. 823–829.
[16] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proc. ICML*, 2001, pp. 441–448.
[17] K. Mu, W. Liu, and Z. Jin, "A general framework for measuring inconsistency through minimal inconsistent sets," *Knowl. Inf. Syst.*, vol. 27, no. 1, pp. 85–114, Apr. 2011.
[18] M. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, Nov. 1995.
[19] Y. Zhang, S. Burer, and W. Street, "Ensemble pruning via semi-definite programming," *J. Mach. Learn. Res.*, vol. 57, pp. 1315–1338, Dec. 2006.
[20] SDPA (SemiDefinite Programming Algorithm). [Online]. Available: http://sdpa.indsys.chuo-u.ac.jp/sdpa/
[21] J. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[22] S. Kknar-Tezel and L. Latecki, "Improving SVM classification on imbalanced time series data sets with ghost points," *Knowl. Inf. Syst.*, vol. 28, no. 1, pp. 1–23, Jul. 2011.
[23] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA: Morgan Kaufmann, 2005.

[24] D. Newman, S. Hettich, C. Blake, and C. Merz, UCI repository of machine learning databases, Irvine, CA: UCI Mach. Learn. 1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[25] A. Beygelzimer, S. Dasgupta, and J. Langford, "Important weighted active learning," in *Proc. ICML*, 2009, pp. 49–56.

[26] F. Wang, J. Sun, T. Li, and N. Anerousis, "Two heads better than one: Metric +active learning and its applications for IT service classification," in *Proc. IEEE ICDM*, Miami, FL, 2009, pp. 1022–1027.

[27] H. Nguyen and A. Smeulders, "Active learning using pre-clustering," in *Proc. ICML*, 2004, pp. 79–86.

[28] N. Otsu, "A threshold selection method from gray-level histogram," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.

[29] Y. Yan, R. Rosales, G. Fung, and J. Dy, "Active learning from crowds," in *Proc. ICML*, 2011, pp. 1161–1168.

[30] F. Herrera, C. Carmona, P. Gonzalez, and M. Jesus, "An overview on subgroup discovery: Foundations and applications," *Knowl. Inf. Syst.*, vol. 29, no. 3, pp. 495–525, Dec. 2011.

[31] P. Donmez and J. Carbonell, "Proactive learning: Cost-sensitive active learning with multiple imperfect oracles," in *Proc. CIKM*, 2008, pp. 619–628.

[32] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowl. Inf. Syst.*, 2012. doi:10.1007/s10115-012-0507-8, to be published.

[33] M. Yamashita, K. Fujisawa, and M. Kojima, "Implementation and evaluation of SDPA 6.0," *Optim. Methods Softw.*, vol. 18, no. 4, pp. 491–505, 2003.

Yifan Fu received the M.E. degree in software engineering from Northeast Normal University, Changchun, China, in 2009. She has been working toward the Ph.D. degree in the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia, since 2010.

Her research interests focus on active learning, ensemble methods, and graph mining.

Xingquan Zhu received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2001.

He is a Professor with the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. His research mainly focuses on data mining, machine learning, and multimedia systems. Since 2000, he has published more than 120 referred journal and conference proceedings papers in these areas.

Dr. Zhu is a recipient of the Australian Research Council Future Fellowships. He has been an Associate Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING since 2009, was a Program Committee Cochair for the 9th International Conference on Machine Learning and Applications (ICMLA 2010) and the 23rd IEEE International Conference on Tools with Artificial Intelligence (2011), and is a General Cochair of the 11th ICMLA (2012).

Ahmed K. Elmagarmid (F'10) received the B.S. degree in computer science from the University of Dayton, Dayton, OH, in 1977 and the M.S. and Ph.D. degrees in computer science from the Ohio State University, Columbus, in 1981 and 1985, respectively.

He is the Qatar Foundation's Inaugural Executive Director of the Qatar Computing Research Institute, Doha, Qatar. Before joining Qatar Foundation, he was the Director of both the Indiana Center for Database Systems and the Cyber Center, Discovery Park, Purdue University, West Lafayette, IN, and a Professor of computer science with Purdue University, where he taught and researched for 22 years. He is the Editor-in-Chief of *Distributed and Parallel Databases: An International Journal* and is on the editorial board of six journals. His expertise is in database technology, data integration, and quality.

Dr. Elmagarmid is an Association for Computing Machinery (ACM) Distinguished Scientist. Early in his career, in 1988, he was the recipient of the National Science Foundation's Presidential Young Investigator award from President Ronald Reagan. The Ohio State University (1993) and the University of Dayton (1995) have both named him among their distinguished alumni. He has chaired and served on several program committees. He was the General Chair of the ACM Special Interest Group on Management of Data conference in 2010.