Active Learning Strategies Using SVMs

Ming-Hen Tsai
Department of Computer Science and Information Engineering,
National Taiwan University, Taiwan
Email: b95028@csie.ntu.edu.tw.

Chia-Hua Ho
Department of Computer Science and Information Engineering,
National Taiwan University, Taiwan
Email: b95082@csie.ntu.edu.tw.

Chih-Jen Lin
Department of Computer Science and Information Engineering,
National Taiwan University, Taiwan
Email: cjlin@csie.ntu.edu.tw.

Abstract—In this paper, we decompose the problem of active learning into two parts, learning with few examples and learning by querying labels of samples. The first part is achieved mainly by SVM classifiers. We also consider variants based on transductive learning. In the second part, based on SVM decision values, we propose a framework to flexibly select points for query. Our experiments are conducted on the data sets of Causality Active Learning Challenge. With measurements of Area Under Curve (AUC) and Area under the Learning Curve (ALC), we find suitable methods for different data sets.

I. INTRODUCTION

In some supervised learning problems, labeling the training data may cost a lot of time or money. In addition, we may not need to label all the training data as some of them are not useful. Active learning is applied when labeling is costly or not all instances need to be labeled. The learner should try to find which instances are worth to be labeled.

This paper presents our results for the Causality Active Learning Challenge.1 We discuss methods considered for participating at the competition and compare their performances. In this competition, we are overall the second, and are the winners in one of the six data sets.

This paper is organized as follows. We introduce the Causality Active Learning Challenge in Section II. All the classifiers used for the competition are described in Section III. In Section IV, we introduce our training methods when only a few instances are labeled. Querying methods to request labels of more data points are in Section V. Experimental results on the development data sets are presented in Section VI. Our submission and results on the challenge data sets are shown in Section VII. We sum up in Section VIII.

II. CAUSALITY ACTIVE LEARNING CHALLENGE

Causality Active Learning Challenge is a competition for active learning. Participants are given all the training and testing feature values, but only one training instance is labeled. Participants can query labels of training instances, and the number of samples per query is not restricted. The goal of the competition is to achieve high area under curve (AUC) with as few queried samples as possible.

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

A. Evaluation

Before querying labels of some training instances, participants must submit the predicted labels of all instances (training and testing data) based on their current model. A learning curve is a line chart of AUC versus the log-scaled number of labeled training instances. The evaluation of competition results is based on the area under the learning curve (ALC).

B. Data Sets

There are six development data sets and six challenge data sets. Development data sets are used for participants to tune systems and algorithms, while the competition is based on results for the challenge data sets. The six development data sets are: HIVA, IBN, SINA, NOVA, ORANGE, SYLVA, and ZEBRA, and the six challenge data sets are named as A, B, C, D, E, and F. For every data set, the number of training instances is the same as the number of testing instances. Table I and II respectively describe details of development and challenge data sets.

The development data sets and the challenge data sets are in similar domains, but the domain information and the ratio of positive labels in challenge data sets are concealed. All these data sets can be downloaded on the competition website.

III. CLASSIFICATION METHODS

This section describes classification methods used for the competition. They include supervised (support vector classification and regression), semi-supervised (transductive support vector machine), and unsupervised (one-class support vector machine) techniques.

A. Support Vector Classification

Support vector classification [1], [2] is a useful learning technique. It finds a hyperplane to separate two classes of instances with the maximized margin. Given \( l \) training instances \((y_i, x_i) \in \{+1, -1\} \times \mathbb{R}^n, i = 1, 2, \ldots, l\), a mapping function \( \phi \), and a penalty parameter \( C \), SVC solves the following constrained minimization problem:

\[
\begin{align*}
\min_{w, b, \xi} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\
\text{subject to} & \quad y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i \\
& \quad \xi_i \geq 0.
\end{align*}
\]

\[y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.\]
For a testing instance \( x \), the predicted label is:

\[
\text{sgn}(w^T \phi(x) + b).
\]

To handle unbalanced data, we often give different penalty parameters on positive and negative data [3]. The optimization problem becomes

\[
\begin{align*}
\min_{w, b, \xi} & \quad \frac{1}{2} w^T w + C^+ \sum_{y_i=1} \xi_i + C^- \sum_{y_i=-1} \xi_i \\
\text{subject to} & \quad y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \\
& \quad \xi_i \geq 0.
\end{align*}
\]

Due to the high dimensionality of \( \phi(x) \), we employ the kernel trick so that only \( K(x, x') = \phi(x)^T \phi(x') \) is needed.

In our experiment, we consider the RBF kernel:

\[
K(x, x') = \exp(-\gamma \|x - x'\|^2_2),
\]

and the linear kernel

\[
K(x, x') = x^T x'.
\]

To obtain a good model, it is important to find suitable SVM parameters [4]. With RBF kernel (3), there are three parameters to choose: \( C^+ \), \( C^- \), and \( \gamma \). However, searching three parameters takes too much time. As a result, we just search for \( C^+ \) and \( \gamma \) by setting \( C^- = C^+ \), so the standard SVM problem (1) is solved. For linear SVM we search for \( C^- \) and \( C^+ / C^- \). The algorithm is shown as follows.

1) Consider a grid space of \((C, \gamma)\) with \( \log_2 C \in \{-5, -3, \ldots, 15\} \), \( \log_2 \gamma \in \{-15, -13, \ldots, 3\} \) for RBF kernel, and a grid space of \((C^- , C^+ / C^-)\) with \( \log_2 C^- \in \{-5, -3, \ldots, 5\} \), \( \log_2 C^+ / C^- \in \{0, 2, \ldots, 10\} \) for linear kernel.

2) For each hyperparameter pair \((C, \gamma)\) or \((C^-, C^+ / C^-)\) in the search space, conduct five-fold cross validation on the training set.

3) Choose the parameter that leads to the best AUC.

4) Use the best parameter to create a model as the predictor.

In some data, we use L2-loss SVC with linear kernel (4) where \( b = 0 \), and \( \xi_i \) in (1) and (2) is replaced by \( \xi_i^2 \). In this case, there is no \( \gamma \). The parameter search is on \( C^+ / C^- \) and \( C^- \).

B. Support Vector Regression

Support Vector Regression (SVR) [3] is used when the target value to predict is continuous. Given \((y_i, x_i) \in \mathbb{R} \times \mathbb{R}^n, i = 1, \ldots, l\), and a mapping function \( \phi \), SVR solves the following problem:

\[
\begin{align*}
\min_{b, w, \xi, \xi^*} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i + C^+ \sum_{i=1}^l \xi_i^* \\
\text{subject to} & \quad w^T \phi(x_i) + b - y_i \leq \epsilon + \xi_i, \\
& \quad y_i - w^T \phi(x_i) - b \leq \epsilon + \xi_i^*, \\
& \quad \xi_i, \xi_i^* \geq 0, i = 1, 2, \ldots, l.
\end{align*}
\]

For an instance \( x \), the predicted value \( y \) is

\[
w^T \phi(x) + b.
\]

We use a fixed \( \epsilon = 0.1 \) for experiment.

C. Logistic Regression

Logistic regression (LR) is also a popular method for classification. If the training points are \((y_i, x_i) \in \{+1, -1\} \times \mathbb{R}^n, i = 1, 2, \ldots, l\), LR assumes the following probability model:

\[
P(y = \pm 1 | x) = \frac{1}{1 + \exp(-y w^T x)}.
\]

To maximize the log likelihood with regularization, L2-regularized logistic regression solves the following problem:

\[
\begin{align*}
\min_{w, \xi} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i \\
\text{subject to} & \quad \xi_i = \log(1 + \exp(-y_i w^T x_i)).
\end{align*}
\]
We omit the bias term in SVC and SVR formulations because it is not considered in the software LIBLINEAR [5] used for experiments. LIBLINEAR supports L1-regularization, where the objective function in (5) is changed to
\[
\min_w \sum_{i=1}^n |w_i| + C \sum_{i=1}^l \xi_i
\]
(6)

D. Transductive SVM

Transductive SVM (TSVM) [6] is a semi-supervised technique for partially labeled data. We consider the formulation used in [7]. Given \( l \) labeled instances \( \{(y_i, x_i) \in \{+1, -1\} \times R^n, i = 1, 2, \ldots, l \} \) and \( u \) unlabeled instances \( x'_i \in R^n, i = 1, 2, \ldots, u \), the following optimization problem is solved.

\[
\min_{w,(y'_i)_{i=1}^u, \xi, \xi'} \frac{\lambda}{2} w^T w + \frac{1}{2l} \sum_{i=1}^l \xi_i^2 + \frac{\lambda'}{2u} \sum_{i=1}^u \xi_i'^2
\]
subject to 
\[
y_i (w^T x_i) \geq 1 - \xi_i, \xi_i \geq 0,
\]
\[
y'_i (w^T x'_i) \geq 1 - \xi'_i, \xi'_i \geq 0,
\]
\[
\frac{1}{u} \sum_{i=1}^u \max(0, (w^T x'_i)) = r,
\]
(8)

TSVM adjusts labels of the unlabeled instances to maximize the margin between two classes. The equality in (8) ensures that our label assignments for unlabeled data have a positive ratio \( r \). Parameters \( \lambda \) and \( \lambda' \), similar to \( C \) in SVC, are decided by users.

E. One-class Support Vector Machine

One-class SVM [8] is a method to estimate the support of a high-dimensional distribution. Given instances \( x_i \in R^n, i = 1, 2, \ldots, l \), one-class SVM solves the following problem:

\[
\min_{w, \xi, \rho} \frac{1}{2} w^T w - \rho + \frac{1}{\rho} \sum_{i=1}^l \xi_i
\]
subject to 
\[
w^T \phi(x_i) \geq \rho - \xi_i, \xi_i \geq 0.
\]
(9)

For a given testing instance \( x \), the predicted label is 
\[
\text{sgn}(w^T \phi(x) - \rho).
\]

One-class SVM is an unsupervised technique because no label information is used. We use many kernels for one-class SVM. In addition to linear kernel RBF kernel (3) and (4), we also use degree-3 polynomial kernel:
\[
K(x, x') = (\gamma x^T x')^3,
\]
sigmoid kernel:
\[
K(x, x') = \tanh(\gamma x^T x'),
\]
perceptron kernel:
\[
K(x, x') = -||x - x'||_2,
\]
Laplacian kernel:
\[
K(x, x') = \exp(-\gamma ||x - x'||_1),
\]
and exponential kernel:
\[
K(x, x') = \exp(-\gamma ||x - x'||_2).
\]

IV. Training with Few Data Points

It is difficult to achieve high AUC values using a small number of training points. However, since the \( x \)-axis (number of queried instances) of the learning curve is log scaled, AUC is severely affected by AUC values when few instances are labeled.

When there are only a few training points, two problems arise. First, training with SVM usually causes overfitting. Second, all data may be of the same label, a situation certainly happening when there is only one training instance. To conquer these problems, the following methods are experimented.

A. Making the Unknown Data Have the Same Label

All data sets in this competition have much more negative instances than positive ones. A naive approach is to treat all data with unknown labels as negative, and train the whole set by SVC, logistic regression, or any other classifiers.

Using this method to assign labels for training instances, most predicted labels for testing instances are negative.

B. One-class SVM

We run one-class SVM on labeled data to get a classifier when all labeled data have the same label. It yields satisfactory results on most development data. Especially, when there is only one labeled instance, this method produces pretty good results. However, it does not use the information of unlabeled data.

C. One-class SVM + SVR

We can use a one-class model to label all the unknown training instances. Then, we train an SVR model. The algorithm is outlined as follows.

1) Use one-class SVM to train labeled data and get a classifier \( M_1 \).
2) Use \( M_1 \) to predict unlabeled data.
3) Randomly choose \( m \) points with decision values in the bottom \( \rho \% \), and assign negative label to them. (\( \rho \) is small in general.)
4) Train an SVR model \( M_2 \).
5) Use \( M_2 \) to predict all data.

\( p \) and \( m \) are parameters selected by users.

D. Transductive Support Vector Machines

TSVM can be used when there are only a few labeled instances. This method has incorporated testing data and unlabeled training data into consideration.
V. QUERYING STRATEGIES

In this section, we describe several querying strategies that we experiment with in the challenge. At the end of the section, a general and easily implemented algorithm is proposed.

Given an upper bound \( m \) on the number of points to query, a set of labeled points’ index \( Q \), and the set of decision values \( F = \{ f_1, f_2, f_3, \ldots, f_l \} \), where \( f_i \) is \( x_i \)'s decision value predicted by the current model, we want to find a set of points \( S \) and query their labels. These newly labeled points in \( S \) should help to improve the AUC.

A. No Active Learning

In this method, we consider \( S = \{ 1 \leq i \leq l \} \setminus Q \). That is, we query all the unlabeled training points at a time. The method is adopted when active learning methods do not seem to improve AUC significantly.

B. Random Query

Let \( P \) be a set of \( m \) random numbers from \( \{ 1, \ldots, l \} \). Then we consider
\[
S = P \setminus Q
\]
in this method. This approach is generally useful, but results may be unstable due to the randomness.

C. Simple Decision Value Query

Tong and Koller [9] proposed a querying method for active learning with SVM. It suggests querying points which are close to the SVM hyperplane. That is, we choose training instances with the smallest \( |f_i| \), \( i = 1, \ldots, l \). To get \( m \) labeled training points, we find a set \( P \) including points with the \( m \) smallest decision values and then have \( S = P \setminus Q \).

D. Uniformly-discretized Decision Value Query

Instead of finding points close to the decision boundary, we uniformly select points according to the distribution of decision values. This setting covers points with different distances to the decision boundary.

We scale \( F \) to \([-1, 1]\) by the following function:
\[
\begin{align*}
\psi(x) &= 2 \left( \frac{x - \min(F)}{\max(F) - \min(F)} - \frac{1}{2} \right) \quad \text{if } \max(F) \neq \min(F), \\
\psi(x) &= 0 \quad \text{otherwise},
\end{align*}
\]
and find the following points to uniformly discretize \([-1, 1]\) to \( m \) intervals:
\[
T = \left\{ -1 + \frac{2i}{m} \, |0 \leq i \leq m \right\}.
\]
We choose points whose decision values are close to values in \( T \):
\[
P = \{ j | j = \arg \min_{1 \leq t \leq l} |\psi(f_i) - t|, \text{where } t \in T \}.
\]
Finally, let \( S = P \setminus Q \) be the set of points for query.

In practice, \( T = \{-1 + 2i/m |1 \leq i \leq m-1\} \) is considered because points with too large too large decision values may not be informative.

E. Arcsin-discretized Decision Value Query

The previous two methods have their own advantages and disadvantages. A method combining both of them is proposed. The idea is to query more points with small absolute decision values, but also query some points far away from the decision boundary.

The method is similar to the previous method of discretizing the space of decision values, but we apply a nonlinear setting so that more points with small decision values are chosen.

We first scale \( F \) to \([-r \pi/2, r \pi/2]\) by the following function:
\[
\begin{align*}
\psi(x) &= r \pi \left( \frac{x - \min(F)}{\max(F) - \min(F)} - \frac{1}{2} \right) \quad \text{if } \max(F) \neq \min(F), \\
\psi(x) &= 0 \quad \text{otherwise},
\end{align*}
\]
where \( r \) is a parameter greater than 0.

Next, we use the arcsin function to have more discretized points around the origin:
\[
T = \{ \arcsin(-1 + \frac{2i}{m}) |0 \leq i \leq m \}.
\]
Then, similar to the setting in Section V-D,
\[
P = \{ j | j = \arg \min_{1 \leq t \leq l} |\psi(f_i) - t|, \text{where } t \in T \}
\]
and \( S = P \setminus Q \). More points with large absolute decision values are queried if \( r \) is small.

In practice, \( T = \{-1 + 2i/m |1 \leq i \leq m-1\} \) is considered with by the same reason stated above.

F. The Decision Value Fitting Algorithm

The above querying strategies can be generalized to the following algorithm:

1) Select a mapping \( M \).
2) Set some \( \psi \in F \rightarrow \text{Range}(M) \).
3) Set \( T = \{ M(-1 + 2i/m) |0 \leq i \leq m \} \)
or \( \{ M(-1 + 2i/m) |1 \leq i \leq m - 1 \} \).
4) Set \( P = \{ j | j = \arg \min_{1 \leq t \leq l} |\psi(f_i) - t|, \text{where } t \in T \} \).
5) Set \( S = P \setminus Q \).
6) If \( S \) is empty, randomly query \( m - |Q| \) points. (This merely happens.)
7) Output \( S \) for query.

For example, in the uniformly-discretized method, \( M \) is the identity function; in the arcsin-discretized method, \( M \) is the arcsin function. With this algorithm, various mapping functions can be easily used.

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

VI. EXPERIMENTAL RESULTS ON THE DEVELOPMENT SETS

In this section, we present experimental results on the development data sets of the contest.
Table III

Comparison of Mean and Standard Deviation of AUC Values Using Various Querying Methods. These Sets Have Categorical Data and We Consider Linear Classifiers

<table>
<thead>
<tr>
<th>Method</th>
<th>HIVA</th>
<th>NOVA</th>
<th>ORANGE</th>
<th>SYLVA</th>
<th>SYLVAexp</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO AL + L1RLR</td>
<td>0.325 ± 0.000</td>
<td>0.661 ± 0.042</td>
<td>0.389 ± 0.000</td>
<td>0.558 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>random + L1RLR</td>
<td>0.190 ± 0.030</td>
<td>0.577 ± 0.020</td>
<td>0.323 ± 0.047</td>
<td>0.876 ± 0.029</td>
<td></td>
</tr>
<tr>
<td>simple + L1RLR</td>
<td>0.115 ± 0.000</td>
<td>0.444 ± 0.047</td>
<td>0.204 ± 0.000</td>
<td>0.638 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>uniform + L1RLR</td>
<td>0.206 ± 0.000</td>
<td>0.514 ± 0.061</td>
<td>0.294 ± 0.000</td>
<td>0.794 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>arcsin r = 0.5 + L1RLR</td>
<td>0.224 ± 0.000</td>
<td>0.541 ± 0.044</td>
<td>0.214 ± 0.000</td>
<td>0.828 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>arcsin r = 1 + L1RLR</td>
<td>0.176 ± 0.000</td>
<td>0.640 ± 0.020</td>
<td>0.300 ± 0.000</td>
<td>0.873 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>NO AL + L2RLR</td>
<td>0.325 ± 0.000</td>
<td>0.733 ± 0.035</td>
<td>0.366 ± 0.000</td>
<td>0.941 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>random + L2RLR</td>
<td>0.194 ± 0.033</td>
<td>0.743 ± 0.025</td>
<td>0.315 ± 0.012</td>
<td>0.916 ± 0.017</td>
<td></td>
</tr>
<tr>
<td>simple + L2RLR</td>
<td>0.094 ± 0.000</td>
<td>0.564 ± 0.072</td>
<td>0.259 ± 0.000</td>
<td>0.731 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>uniform + L2RLR</td>
<td>0.189 ± 0.000</td>
<td>0.678 ± 0.075</td>
<td>0.281 ± 0.000</td>
<td>0.900 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>arcsin r = 0.5 + L2RLR</td>
<td>0.127 ± 0.000</td>
<td>0.722 ± 0.040</td>
<td>0.297 ± 0.000</td>
<td>0.929 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>arcsin r = 1 + L2RLR</td>
<td>0.136 ± 0.000</td>
<td>0.604 ± 0.071</td>
<td>0.274 ± 0.000</td>
<td>0.879 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>NO AL + L2RL2</td>
<td>0.315 ± 0.000</td>
<td>0.663 ± 0.038</td>
<td>0.367 ± 0.000</td>
<td>0.940 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>random + L2RL2</td>
<td>0.188 ± 0.017</td>
<td>0.683 ± 0.025</td>
<td>0.321 ± 0.018</td>
<td>0.909 ± 0.021</td>
<td></td>
</tr>
<tr>
<td>simple + L2RL2</td>
<td>0.109 ± 0.000</td>
<td>0.524 ± 0.055</td>
<td>0.222 ± 0.000</td>
<td>0.735 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>uniform + L2RL2</td>
<td>0.124 ± 0.000</td>
<td>0.663 ± 0.041</td>
<td>0.266 ± 0.000</td>
<td>0.889 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>arcsin r = 0.5 + L2RL2</td>
<td>0.147 ± 0.000</td>
<td>0.632 ± 0.059</td>
<td>0.257 ± 0.000</td>
<td>0.931 ± 0.000</td>
<td></td>
</tr>
<tr>
<td>arcsin r = 1 + L2RL2</td>
<td>0.167 ± 0.000</td>
<td>0.513 ± 0.020</td>
<td>0.259 ± 0.000</td>
<td>0.802 ± 0.000</td>
<td></td>
</tr>
</tbody>
</table>

A. Data Preprocessing

Before doing experiments, all six data sets except IBN_SINA are scaled so that each feature takes values in [0, 1] (HIVA and NOVA have binary features). In the data set ORANGE, 187 of 230 features contain missing values. For each instance, we use another 187 binary features to indicate if the corresponding value is missing (0) or not (1). Every categorical feature in SYLVA is expanded as several binary features. The new expanded data set is named SYLVAexp. ZEBRA has a few missing values (0.004%). We simply assign these missing values to zero. Only four of the 61,488 instances are affected.

B. The First Point

When all labeled data are in the same class, we use methods described in Section IV. Such a situation happens when there is only one labeled point.

We do not conduct parameter selection to avoid overfitting. For the method assigning negative labels to unlabeled data, logistic regression is used for classification. For one-class SVM, LIBSVM extensions [10], [11] that support stump kernel, perceptron kernel, Laplacian kernel, and exponential kernel are used. For one-class SVM + SVR, RBF kernel is used for non-categorical data (IBN_SINA, ORANGE, SYLVA, and ZEBRA), while linear kernel is used for categorical data (HIVA, NOVA, and SYLVAexp). For each problem, the same kernel is used for one-class SVM and SVR. For TSVM, we use the software SVMlin [7]. Two experiments with different values of the parameter “positive class fraction of unlabeled data” are compared. One is the real positive ratio for each data set; the other is 0.1 for all data sets. A constant ratio is experimented because we do not know the positive ratio of testing data sets.

Table III shows testing AUC values under different strategies to handle the situation of having only one labeled training point. Due to the randomness in the algorithms, each experiment is conducted five times, and the mean and standard deviation of AUC values are reported. In the table, “All negative” means all training data except the first given positive instance are labeled as negative. OSVM stands for one-class SVM. “OSVM linear” means one-class SVM with linear kernel, and other entries can be interpreted analogously. For “OSVM + SVR,” p = 10 and m = 3; see Section IV-C. “SVMlin real” stands for running SVMlin with the real positive ratio in the whole data set (training and testing data). We know that ratio because the labels of development data sets are given.
“SVMlin 0.1” stands for running SVMlin with the parameter “positive ratio” assigned as 0.1. For each data set, The bold-faced value indicates the best classifier’s AUC.

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

C. Subsequent Queries

Once we have assigned training instances to be positive or negative using the first labeled point, we can start the active learning procedure by querying labels of some training points. For subsequent binary classification, we employ linear classifiers on categorical data sets (HIVA, NOVA, and SYLVAnexp) and the non-categorical set ORANGE. The reason to use linear classifiers for ORANGE is that our previous study in KDD 2009 shows that linear classifiers with missing value indicators yields good results. The software used is LIBLINEAR [5] with L1-regularized logistic regression (L1RLR), L2-regularized logistic regression (L2RLR), and L2-regularized L2-loss SVC (L2RL2). We use a non-linear classifier (LIBSVM with RBF kernel) on other non-categorical data sets (IBNSINA, SYLVAn, and ZEBRA).

For each data set, the method we use for the first point is the one with best AUC in Table III, and we use the same query method in the whole active learning process. We sequentially query labels of s, 2s, 2^2s, 2^3s, ... points until the AUC does not change significantly or there is no data with unknown label. The value s is eight in the experiment, but we use different s in the challenge data sets.

The resulting ALC of all querying methods are shown in Tables IV and V according to whether linear or nonlinear classifiers are used. Due to the randomness in the algorithms, each experiment is conducted five times, and the mean and standard deviation of ALC values are reported. In both table, methods resulting the highest two mean ALC values are marked in bold.

The final AUC is the AUC obtained by training the training data and predicting the testing data. We conduct parameter selection on each training data set, and use the best parameters to obtain a model for evaluating final AUC. The results are shown in Tables VI and VII.

From the tables, we observe that if the initial and final AUC values are low, the performance without using active learning methods is better. Take HIVA as an example, without active learning, it has initial AUC 0.532, final AUC 0.790, and ALC 0.325. Other active learning methods’ ALC are below 0.230 by mean. For such cases, active learning may not be helpful. Regarding querying methods, in general, arcsin-discretized query works better than uniformly-discretized query, and simple query is even worse. However, the best querying method seems to be data-dependent.

### Table V

<table>
<thead>
<tr>
<th>Method</th>
<th>IBN_SINA</th>
<th>SYLVA</th>
<th>ZEBRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>No AL + RBF</td>
<td>0.870 ± 0.000</td>
<td>0.941 ± 0.000</td>
<td>0.422 ± 0.000</td>
</tr>
<tr>
<td>random + RBF</td>
<td>0.889 ± 0.007</td>
<td>0.944 ± 0.008</td>
<td>0.368 ± 0.012</td>
</tr>
<tr>
<td>simple + RBF</td>
<td>0.685 ± 0.000</td>
<td>0.798 ± 0.000</td>
<td>0.410 ± 0.000</td>
</tr>
<tr>
<td>uniform + RBF</td>
<td>0.842 ± 0.000</td>
<td>0.918 ± 0.000</td>
<td>0.275 ± 0.000</td>
</tr>
<tr>
<td>arcsin r = 0.5 + RBF</td>
<td>0.852 ± 0.000</td>
<td>0.925 ± 0.000</td>
<td>0.302 ± 0.000</td>
</tr>
<tr>
<td>arcsin r = 1 + RBF</td>
<td>0.848 ± 0.000</td>
<td>0.946 ± 0.000</td>
<td>0.361 ± 0.000</td>
</tr>
</tbody>
</table>

### Table VI

<table>
<thead>
<tr>
<th>Data set</th>
<th>classifier</th>
<th>C^−</th>
<th>C^+</th>
<th>final AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIVA</td>
<td>L2RLR</td>
<td>2</td>
<td>2</td>
<td>0.790</td>
</tr>
<tr>
<td>NOVA</td>
<td>L2RLR</td>
<td>32</td>
<td>32</td>
<td>0.988</td>
</tr>
<tr>
<td>ORANGE</td>
<td>L1RLR</td>
<td>0.125</td>
<td>1</td>
<td>0.815</td>
</tr>
<tr>
<td>SYLVAexp</td>
<td>L2RL2</td>
<td>1</td>
<td>1</td>
<td>0.999</td>
</tr>
</tbody>
</table>

### Table VII

<table>
<thead>
<tr>
<th>Data set</th>
<th>classifier</th>
<th>C</th>
<th>γ</th>
<th>final AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBN_SINA</td>
<td>RBF</td>
<td>8</td>
<td>0.03125</td>
<td>0.991</td>
</tr>
<tr>
<td>SYLVA</td>
<td>RBF</td>
<td>32</td>
<td>0.0078125</td>
<td>0.999</td>
</tr>
<tr>
<td>ZEBRA</td>
<td>RBF</td>
<td>32</td>
<td>0.5</td>
<td>0.842</td>
</tr>
</tbody>
</table>

VII. Our Submission and Results on the Challenge Data Sets

In this section, we describe the methods used for the challenge data sets. Results of earlier experiments on development sets help us to properly choose these methods.

A. Methods for Challenge Data Sets

From the experiments on development data sets, we know that selecting proper methods for each data set is important. Because challenge sets are from the same domains as development sets, we try to obtain one-to-one mappings between them. Then methods suitable for a development set can be applied to the corresponding challenge set.

To do the mapping, we look for missing values, sparsity, and feature types in development and challenge data sets. We are able to identify the following relationships:

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

However, it is not clear what C and F are. We decide to use methods for SYLVA for these two sets. We expand the
categorical features in C and deal it using the method of SYLVAexp. We use the method for SYLVA to process F.

Once data mapping is determined, we port training/querying methods as well as SVM/LR parameters for development data sets to challenge data sets. See details in Table VIII. Take data set A as an example. Since we conjecture that it is from the same domain of IBN_SINA, we check Tables III and V to decide suitable methods. Note that we may not always select the best methods for development data. While doing the competition, we do not have time to summarize all the results and the selection may not be very systematic. In D, when only one instance is labeled, we use OSVM + SVR to predict, but we use the decision values by assigning all unlabeled instances $-1$ with L2-regularized LR (Section IV-A) to decide query instance. Also, the set $P$ in V-D and V-E is set as

$$P = \{j|j = \arg\min_{1 \leq i \leq l, \psi(f_i) \geq t} |\psi(f_i) - t|, \text{ where } t \in T\}$$

due to immaturity of our learning system. We use SVM/LR parameters obtained from cross validation on the full development data set. This is because CV on a few labeled points of the challenge set may not yield reliable results.

In Table VIII, the column “preprocess” indicates how we preprocess the data, where S means features are scaled to $[0, 1]$, E means categorical features are expanded, M means missing value indicators are added with missing values assigned to zero, and T means all missing values are assigned zero. The column “parameter ported from” shows the development set, from which we obtain SVM /LR parameters; see Tables VI and VII. The column “# of first queries” means the number of points to query after we conduct methods in the column “first point.” Similar to the situation for development data, we then query $s, 2s, 2^2s, 2^3s, \ldots$ points until all labels are obtained. For data set C, we have 2, 3, 256 as the numbers of first queries. This means we query 2, 3, 256, 512, 1024, $\ldots$ points.

In selecting our final submission method, we try to balance between stableness and high ALC. For the first point, we think one-class SVM and its variants are more stable from previous experiments. Hence we use one-class SVM for most data sets (A, C, D, and F). Although SVMlin is not stable, it sometimes gives very good AUC. We apply it for the first point on two data sets B and C. About querying methods, we do not consider random query (Section V-B) because it is not stable.

### Table VIII

<table>
<thead>
<tr>
<th>Data set</th>
<th>preprocess</th>
<th>first point query method</th>
<th>parameter ported from</th>
<th># of first queries</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>No</td>
<td>OSVM + SVR uniform + RBF</td>
<td>IBN_SINA</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>S,M</td>
<td>SVMlin real No AL L1LR</td>
<td>No</td>
<td>All</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
<td>SVMlin 0.11 uniform + L2LR2</td>
<td>SYLVAexp</td>
<td>2.5,256</td>
<td>11</td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>OSVM linear arsin + L2LR</td>
<td>No</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>S,T</td>
<td>OSVM RBF No AL RBF</td>
<td>ZEBRA</td>
<td>All</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>S</td>
<td>OSVM RBF uniform + RBF</td>
<td>No</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

### Table IX

<table>
<thead>
<tr>
<th>data</th>
<th>our results on challenge data sets</th>
<th>the best results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>first AUC</td>
<td>final AUC</td>
</tr>
<tr>
<td>A</td>
<td>0.439</td>
<td>0.928</td>
</tr>
<tr>
<td>B</td>
<td>0.643</td>
<td>0.733</td>
</tr>
<tr>
<td>C</td>
<td>0.428</td>
<td>0.779</td>
</tr>
<tr>
<td>D</td>
<td>0.433</td>
<td>0.970</td>
</tr>
<tr>
<td>E</td>
<td>0.726</td>
<td>0.857</td>
</tr>
<tr>
<td>F</td>
<td>0.534</td>
<td>0.997</td>
</tr>
</tbody>
</table>

### B. Competition Results

For each data set, we submit results by methods described earlier. Our rank on each data set is listed in Table VIII. Our overall rank is the $2^{nd}$.

The competition organizer has released the AUC and ALC of all contestants. We compare our final AUC and ALC with the highest AUC and ALC among all contestants in Table IX. Learning curves for all challenge data sets are in Figure 1. Our methods for situations when few training instances are labeled are not very stable. They may reach high initial AUC (0.643 in B or 0.726 in E) so that we rank $1^{st}$ and $2^{nd}$ on these two sets. For these two sets, our testing AUC using models trained on the fully labeled training set may not be the best; see Table IX. However, for other data sets, our initial AUC values are quite low. From the competition results, the winning team for each of the data sets has the first AUC value higher than 0.5.

### VIII. Discussion and Conclusions

By the definition of ALC, to achieve high ALC values we must perform well when there are few labeled training instances. To get high quality data, querying strategies are very important. From Tables IV and V, uniformly-discretized decision value query and arcsin-discretized decision value query are effective to find informative points. A and D are ranked $3^{rd}$ and $3^{rd}$ respectively while initial values are below 0.5. Also, in our querying methods, we only use the decision values to decide which instances to query. Thus we can quickly decide the query instances after training.

In conclusion, the querying methods and the used classifiers seem to be data dependent. We experiment with some methods including one-class SVM, SVR, and TSVM when there are few labeled training instances. These methods work reasonable well though there is no clear winner. About querying, we develop methods based on SVM decision values. In particular, a general framework allowing flexible ways of using decision
values is proposed. These methods can also be adopted to classifier with probability outputs. Results indicate that arcsin-discretized and uniformly-discretized querying methods are better than others.

REFERENCES


