JMLR: Workshop and Conference Proceedings 16 (2011) 71–84 Workshop on Active Learning and Experimental Design

Active Learning and Experimental Design with SVMs

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

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

Abstract

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

1. Introduction

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

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

This paper presents our methods, experiments and results for the Causality Active Learning Challenge.[1] Detailed settings of this competition can be found in Guyon et al. (2011). In this competition, we are second overall, and are the winner in one of the six

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

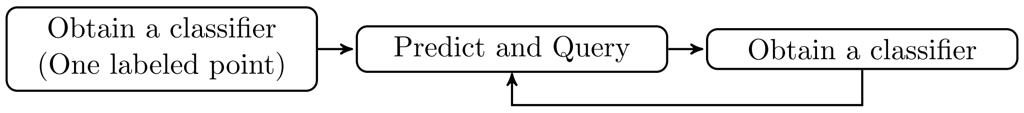


Figure 1: An active learning framework

data sets. This paper is an improved version of Tsai et al. (2010) by including some post-submission results.

2. Methods

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

2.1. The Framework

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

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

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

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

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

Figure 1 illustrates our framework.

2.2. Training with One Labeled Data Point

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

2.2.1. Making the Unknown Data Have the Same Label

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

2.2.2. One-class SVM

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

2.2.3. One-class SVM + SVR

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

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

Parameters p and m are selected by users.

2.2.4. Transductive Support Vector Machines

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

2.3. Querying Strategies

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

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

2.3.1. Some Simple Existing Methods

We consider some naïve or existing querying methods.

- **No Active Learning**
  In this method, we consider $S = \{1 \leq i \leq l\} \backslash Q$. That is, we query all the unlabeled training points at a time.
- **Random Query**
  Let P be a set generated by randomly drawing a point from $\{1, \ldots, l\}$ m times. Then we consider $S = P \backslash Q$. Since repetition is possible, the set P may have less than m points. Results of this approach may not be very stable due to the randomness.

- **Simple Query**
  Tong and Koller (2002) proposed a querying method for active learning with SVM. It suggests querying points which are close to the SVM hyperplane. Since the distance between a point and the hyperplane is $|\boldsymbol{w}^T\boldsymbol{x}_i|/\sqrt{\boldsymbol{w}^T\boldsymbol{w}}$, we identify points with the smallest m values in $\{|f_1|, \ldots, |f_l|\}$ as the set P. Then $S = P\backslash Q$.

2.3.2. A Decision Value Fitting Algorithm

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

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

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

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

2.3.3. Uniformly-discretized Decision Value Query

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

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

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

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

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

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

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

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

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

2.3.4. Arcsin-discretized Decision Value Query

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

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

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

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

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

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

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

3. Experiments

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

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

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

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

(a) Development data sets.

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

(b) Challenge data sets.

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

3.1. Classification Methods, Software, and Implementations

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

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

where

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

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

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

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

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

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

3.2. Results on the Development Sets

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

3.2.1. The First Labeled Point

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

2. Note that LIBLINEAR does not consider the bias term b in (2).

3. If we do not fix C^-, the cost for parameter selection may be too high.

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

(*a*) Linear classifiers for categorical data

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

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

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

data sets; the other is 0.1 for all data sets. A constant ratio is experimented because later we do not know the positive ratio of challenge data sets.

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

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

3.2.2. Subsequent Queries

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

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

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

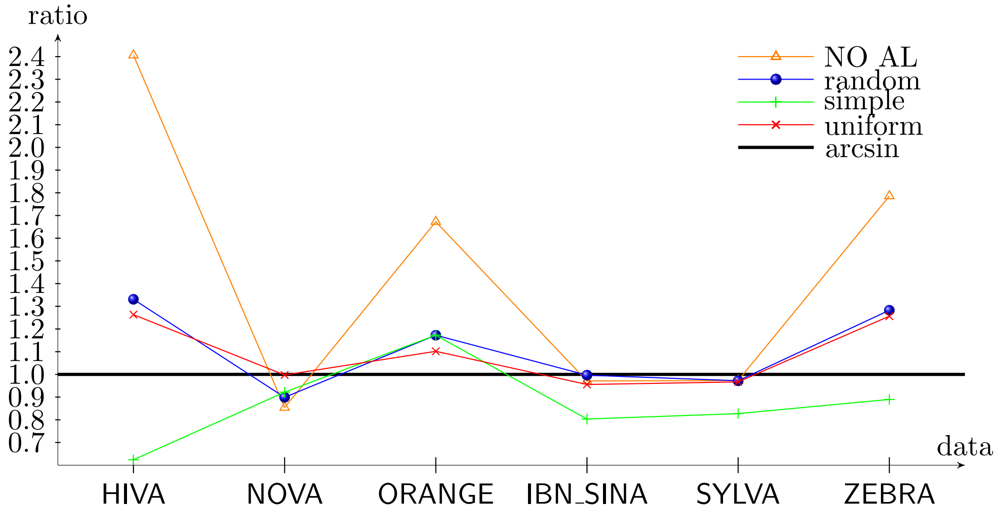


Figure 2: A comparison between arcsin-discretized and other querying methods. The ration means (ALC by any method)/(ALC by the arcsin method).

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

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

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

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

3.2.3. Stableness of TSVM

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

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

(a) Linear classifier

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

(b) Nonlinear classifier

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

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

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

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

3.3. Competition Results and Post Challenge Submissions

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

3.3.1. Methods for Challenge Data Sets

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

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

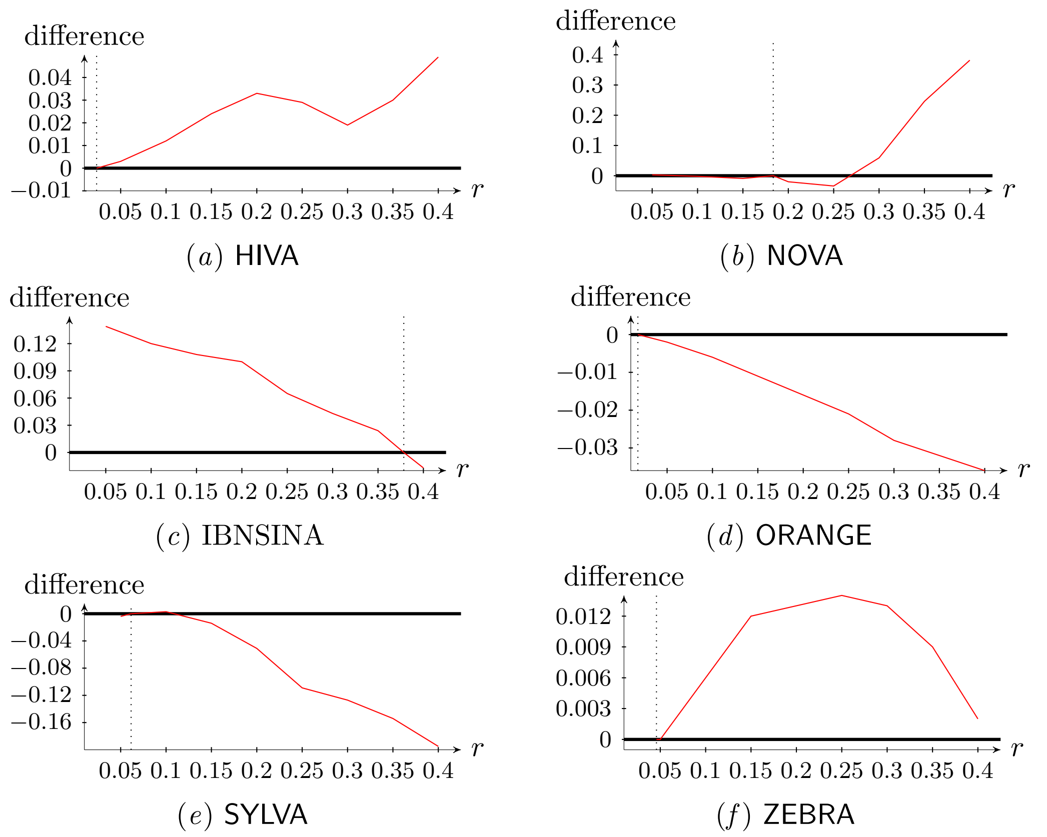


Figure 3: AUC value differences between using various r values and using the real positive ratio for TSVM. The dotted vertical line and the horizontal solid line respectively indicate the true positive ratio and the performance of using it.

- E is ZEBRA because they have both numerical feature type.

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

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

1. When there is only one labeled point, choose a better method between "All negative" and TSVM $r = 0.1$ using Table 2.
2. For $i = 0, 1, 2, \ldots$
    - Query labels by the following methods, and use the corresponding solver in Tables 4(*a*) and 4(*b*) to train and predict.

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

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

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

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

  – See Table 3. If active learning seems to work (e.g., IBN_SINA,NOVA, and SYLVA), do arcsin-discretized query with $s = 16$. That is, query $2^i s$ points.
  – Otherwise, query all the remaining labels.
- If all training instances are labeled, stop.

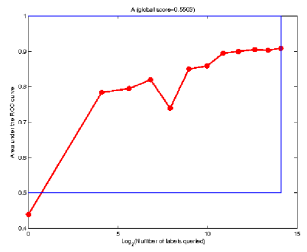
Table 6 also lists methods applied by using the above procedure. They are slightly different from those used in the competition.

3.3.2. Results

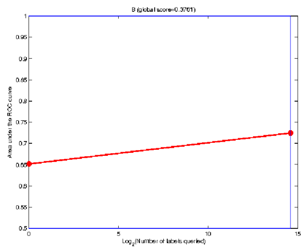
We compare our AUC and ALC value with the highest AUC and ALC among all contestants in Table 7. Here AUC means the final AUC using the whole training set. We also list our rank for each problem. Clearly, our post-challenge submissions give slightly better results than our competition submissions. The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

4. Discussion and Conclusions

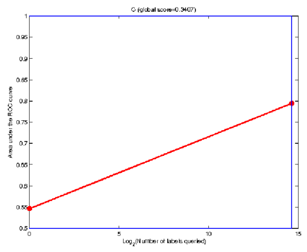
In this work, we consider the active learning framework shown in Figure 1. Since the evaluation criteria is ALC with log-scaled x-axis, the performance at the first iteration is very important. We investigate various methods to obtain a good classifier when there is only one labeled point. We also compare all these methods on the six data sets. Results indicate that the best method seems to be data dependent, but semi-supervised methods, e.g., transductive SVM, can generate comparatively good results. Then for subsequent



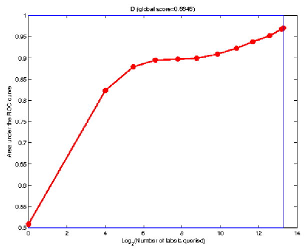
(a) A



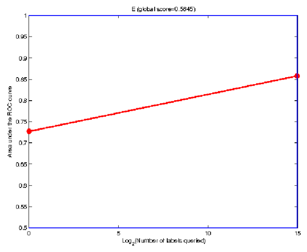
(b) B



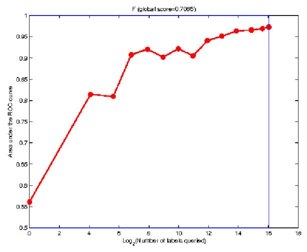
(c) C



(d) D



(e) E



(f) F

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

queries, we develop a general algorithm to select a set of points. The algorithm uses only decision values from classifiers, so little extra cost is needed. Users can choose a suitable mapping functions μ in Section 2.3.2 for their data sets. Unfortunately, we have neither theoretical justification for the performance of our methods, nor do we know how severely the results may be effected using different mapping functions. These issues are possible future works. However, while none of the methods is the best, we observe that querying methods based on arcsin-discretized and uniformly-discretized decision values are better than others in our experiments, and the uniformly-discretized approach generates pretty stable results.

References

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

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

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

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

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

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

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

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

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

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

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

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

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