A Framework for Modeling Positive Class Expansion with Single Snapshot

Yang Yu and Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210093, China
{yuy,zhouzh}@lamda.nju.edu.cn

Abstract. In many real-world data mining tasks, the coverage of the target concept may change as the time changes. For example,the coverage of "learned knowledge" of a student today may be different from his/er "learned knowledge" tomorrow, since the "learned knowledge" of the student is in expanding everyday. In order to learn a model capable of making accurate predictions, the evolution of the concept must be considered, and thus, a series of data sets collected at different time is needed. However, in many cases there is only a single data set instead of a series of data sets. In other words, only a *single snapshot* of the data along the time axis is available. In this paper, we show that for *positive class expansion*, i.e., the coverage of the target concept is in expanding as illustrated in the above "learned knowledge" example, we can learn an accurate model from the single snapshot data with the help of domain knowledge given by user. The effectiveness of the proposed framework is validated in experiments.

1 Introduction

In conventional machine learning and data mining research, it is assumed, explicitly or implicitly, that the test set is drawn from the same distribution of the training set and the target concept, i.e. *a posteriori* probability of class membership $p(y|\boldsymbol{x})$, is unchanged [4, 14]. However, in many real-world applications, we may encounter problems with varying target concept. The following are two examples.

Example 1: A manufacturer has released a new product to replace its old product. After an expensive mass-advertising for one week, the manufacturer got to know that a few customers have turned to the new product. In order to reduce the cost of advertizement, the manufacturer wants to identify target customers who have high chance to turn to the new product based on an analysis on customers who have already turned to the new product. Note that here, the number of customers turning to the new product is keeping on increasing.

Example 2: A Disease Control and Prevention Center receives several fatal cases of an unclear infectious disease. A model is urgently required to predict who are potential victims, based on periodical physical examination database, in order to perform effective quarantining actions. Note that here, the victims of the disease is keeping on increasing.

In these examples, a common difficulty is that the training data is collected at an earlier time point but the predictions are to be made later, meanwhile the coverage of the target concept at different time points may be quite different. For example, in the above Example 1, suppose there were only 10 customers who had already turned to the new product (i.e., 10 *positive examples*) at the time when the data were collected; but when the model is used to make prediction, 20 more customers have already turned to the new product. Thus, if we only consider the original 10 positive examples, maybe the model we build for predicting who will turn to the new product could not meet our demand.

A possible solution to the above problem is to wait for a long period to collect a series of training sets at a series of time points, such that the pattern of the evolution can be considered. However, this may be infeasible in most cases since waiting for a long period will cause, for example, great loss of benefits in the above Example 1 and loss of human lives in the above Example 2. Moreover, data used in data mining tasks are usually *observational* [7]. That is, data is usually given by other people and the data miner could not collect more data. So, the varying target concept problem has to be addressed when there is only a *single snapshot* data set instead of a series of data sets taken at different time points.

In this paper, we propose a framework to deal with the *positive class expansion* problem, which has been illustrated in the above examples, with a single snapshot data set. Our framework has two elements. The first element is the utilization of the observation that *the instances that are currently positive become positive ahead of the instances that are currently negative*, which leads to an AUC optimization problem. The second element is the incorporation of domain knowledge expressed by user preferences of pairs of instances. These two elements are unified as an optimization problem, which is solved by the SGBDota (Stochastic Gradient Boosting with Double Target) approach. The SGBDota approach achieves success in experiments, which validate the usefulness of our framework for dealing with positive class expansion problem with single snapshot.

The rest of the paper is organized as follows. Section 2 formalizes the problem. Section 3 reviews some related work. Section 4 proposes our framework. Section 5 reports on experiments. Finally, Section 6 concludes.

2 The Problem

Given a training set of i.i.d. instances $D = \{\boldsymbol{x}_i\}_{i=1}^n$ drawn from a distribution $\mathcal{D}$. Each instance is associated with a random variable y called class, which is determined by $p(y|\boldsymbol{x})$. In conventional classification problem, a learning algorithm outputs a function $\hat{f}(\cdot; D, p(y|\boldsymbol{x}))$ that minimizes the error of a loss function $\boldsymbol{L}$:

$$err_{\hat{f}(\cdot;D,p(y|\boldsymbol{x}))} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\boldsymbol{L}(\hat{f}(\boldsymbol{x}; D, p(y|\boldsymbol{x})), p(y|\boldsymbol{x})).$$

In the scenario of varying target concept, the training set D is collected at a time point, where we call the training set as a *snapshot* and the time point as *training time*. After trained on the snapshot, a classifier is used to make predictions at a later time point, which we call *testing time*. From the training time to the testing time, $p(y|\boldsymbol{x})$ changes, while $\mathcal{D}$ keeps steady. The evaluation of $\hat{f}$ is therefore changed

$$err_{\hat{f}(\cdot;D,p(y|\boldsymbol{x}))} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\boldsymbol{L}(\hat{f}(\boldsymbol{x};D,p_{tr}(y|\boldsymbol{x})),p_{te}(y|\boldsymbol{x}))$$

where p_{tr} and p_{te} indicate the probability functions at training and testing time, respectively. The formalization of positive class expansion consists of two constraints:

$$\begin{array}{ll} 1) & f^*(\boldsymbol{x}) \in \{-1,+1\} \\ 2) & \forall x \sim \mathcal{D} : p_{te}(y=+1|\boldsymbol{x}) \geq p_{tr}(y=+1|\boldsymbol{x}), \end{array}$$

which means there are only two classes, positive and negative, and the positive class is in expanding.

3 Related Works

The positive class expansion problem appears to have some relationship with *PU-Learning* [12, 17], *concept drift* [9, 10], and *covariate shift* [8, 1]. But in fact it is very different from these tasks.

In PU-Learning, i.e., learning with positive and unlabeled data, it is required to discriminate positive instances from negative instances, while only positive instances are available in training data. A large part of works addressing PU-Learning follow two steps, e.g. [12, 17]. First, strong negative instances are discovered from the unlabeled data. Then a predictive model is built from the positive and identified negative instances.

In concept drift, an online learning environment is considered, where instances are coming sequentially batch by batch, and the target concept may change in the coming batch. A desired approach for concept drift problem is the one that correctly detects and fast adapts to the drift, e.g. [9, 10].

In covariate shift problem (or *sample selection bias* [13]) , training and test instances are drawn from different distributions, while the *a posteriori* probability is unchanged. Using the notations in the previous section, it minimizes the error:

$$err_{\hat{f}(\cdot;D,p(y|\boldsymbol{x}))} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{te}}\boldsymbol{L}\big(\hat{f}(\boldsymbol{x};D\sim\mathcal{D}_{tr},p(y|\boldsymbol{x})),p(y|\boldsymbol{x})\big)\ ,$$

where $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ are the distributions at training and testing time, respectively. Approaches (e.g. [8, 1]) addressing this problem try to correct the bias in the training instances, such that minimizing error on the training instances corresponds to minimizing error on the test instances.

The positive class expansion problem is apparently different from the above problems. In PU-Learning problem, most works make an assumption that the positive instances in the training set are representative of the positive class concept. But in our problem, the positive class is in expanding thus are not representative. We have noticed a recent work of PU-Learning considers different training and test distributions [11]. However, that work gears heavily to text mining by using specialized mechanisms, i.e., it tries to synthesize samples by using additional keywords. In concept drift, it expects a series of data sets with information for drift detection, but in our problem, there is no such information at all since only a single snapshot is available. In covariate shift, it assumes the *a posteriori* probability is unchanged from the training time to the testing time. On the contrary, the *a posteriori* changes in our problem.

The approach, we proposed to solve the optimization problem in our framework, is derived from Gradient Boosting [5, 6]. Gradient boosting is a greedy optimization approach that avoids solving complex equations by iteratively fitting residuals of the objective function. In order to minimize an arbitrary loss function L,

$$f^* = \arg\min\nolimits_f \mathbb{E}_{\boldsymbol{x}} L(f(\boldsymbol{x}), y_{\boldsymbol{x}}) ,$$

the approach builds an additive model $F(\boldsymbol{x}) = \sum_{t=0}^{T} \beta_t h(\boldsymbol{x}; \theta_t)$ as the solution to the minimization problem, where θ_t is the parameter of h, β_t and θ_t are decided by

$$(\beta_t, \theta_t) = \arg\min\nolimits_{(\beta,\theta)} L(F_{t-1} + \beta h(;\theta)) ,$$

where $F_t = F_{t-1} + \beta_t h(;\theta_t)$ and $F = F_T$. To avoid dealing with the complex loss function L, it first solves θ_t by fitting pseudo-residuals least-squarely according to

$$\theta_t = \arg\min_{\theta} \sum_{\boldsymbol{x} \in D} \left(h(\boldsymbol{x}; \theta) + \left. \frac{\partial L(f(\boldsymbol{x}))}{\partial f(\boldsymbol{x})} \right|_{f(\boldsymbol{x}) = F_{t-1}(\boldsymbol{x})} \right)^2 ,$$

and then it solves β_t according to

$$\beta_t = \arg\min\nolimits_\beta L(F_{t-1} + \beta h(;\theta_t)) .$$

4 The Proposed Framework

Since the positive class is in expanding, it is reasonable to assume that *the instances that are currently positive become positive ahead of the instances that are currently negative*. This assumption requires that all the positive instances should be ranked above the negative instances, which is exactly expressed by the AUC (*area under ROC curve*) [3] criterion. This assumption may imply the total information that we can obtain from the data set per se. Thus, we use *1 minus the AUC value* as the loss function to evaluate how the information provided by the training data is utilized:

$$L_{auc}(f) = 1 - \frac{1}{|D^+| \cdot |D^-|} \sum\nolimits_{\boldsymbol{x}^+ \in D^+} \sum\nolimits_{\boldsymbol{x}^- \in D^-} \boldsymbol{I}(f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-)) . \quad (1)$$

where D^+ and D^- are subsets of D that contains all of the positive and negative instances, respectively, and $\boldsymbol{I}(a)$ gets 1 if $a \geq 0$ and 0 otherwise.

Since the training data is not sufficient to build a good model in our problem, we need to incorporate domain knowledge from the user. It is not hard for the user to indicate pairwise preferences on some instances. For example, pairs of instances can be randomly drawn from the training set, and then the user is asked to judge which instance would become positive earlier in his/er opinion. Another possibility is to apply *a priori* rules to decide which instance would become positive earlier, such as *people of*

Mongoloid race may be easier to got SARS than people of Caucasoid race. Let $k(\cdot,\cdot)$ denotes the user's pairwise preferences, such that

$$k(\boldsymbol{x}_a,\boldsymbol{x}_b)=\begin{cases}+1, & \boldsymbol{x}_a \text{ is preferred}\\ -1, & \boldsymbol{x}_b \text{ is preferred}\\ 0, & \text{equal or undecided}\end{cases}.$$

where "$\boldsymbol{x}_a$ is preferred" means that the user thought that $\boldsymbol{x}_a$ would become positive earlier than $\boldsymbol{x}_b$. We then fit these preferences by the loss function:

$$L_{pref}(f)=1-\frac{1}{|D|^2}\sum\nolimits_{\boldsymbol{x}_a\in D}\sum\nolimits_{\boldsymbol{x}_b\in D}\boldsymbol{I}\big((f(\boldsymbol{x}_a)-f(\boldsymbol{x}_b))\cdot k(\boldsymbol{x}_a,\boldsymbol{x}_b)\big) \quad (2)$$

which imposes that the sign of $f(\boldsymbol{x}_a)-f(\boldsymbol{x}_b)$ should equal to the sign of $k(\boldsymbol{x}_a,\boldsymbol{x}_b)$.

Our final objective function combines Eq.1 and 2 by *a prior* weight λ:

$$\hat{f}=\arg\min\nolimits_f L_\lambda(f)=\arg\min\nolimits_f L_{auc}(f)+\lambda L_{pref}(f) \quad (3)$$

As mentioned before, we realize our framework based on Gradient Boosting [5, 6]. In the original Gradient Boosting algorithm, each instance $\boldsymbol{x}$ is associated to a target label $y_{\boldsymbol{x}}$. However, in Eq. 3, each instance $\boldsymbol{x}$ is associated to two targets, one is $y_{\boldsymbol{x}}$ while the other is determined by $k(\boldsymbol{x},\cdot)$. Therefore, we build an additive model F with two base learners in each iteration, h_1 and h_2, as:

$$F(\boldsymbol{x})=\sum\nolimits_{t=0}^{T}(\beta_{t,1}h_1(\boldsymbol{x};\theta_{t,1})+\beta_{t,2}h_2(\boldsymbol{x};\theta_{t,2}))\ ,$$

where h_1 and h_2 fit the residuals of L_{auc} and L_{pref}, respectively.

In the first step, we solve h_1 and h_2. In Eq.1 and Eq.2, $\boldsymbol{I}(\cdot)$ is non-differentiable. We use sigmoid function $(1+e^{-a})^{-1}$ to replace the identification function $\boldsymbol{I}(a)$. So the loss functions are rewritten as:

$$L_{auc}(f)=1-\frac{1}{|D^+|\cdot|D^-|}\sum\nolimits_{\boldsymbol{x}^+\in D^+}\sum\nolimits_{\boldsymbol{x}^-\in D^-}\left(1+e^{-(f(\boldsymbol{x}^+)-f(\boldsymbol{x}^-))}\right)^{-1} \quad (4)$$

$$L_{pref}(f)=1-\frac{1}{|D|^2}\sum\nolimits_{\boldsymbol{x}_a\in D}\sum\nolimits_{\boldsymbol{x}_b\in D}\left(1+e^{-(f(\boldsymbol{x}_a)-f(\boldsymbol{x}_b))\cdot k(\boldsymbol{x}_a,\boldsymbol{x}_b)}\right)^{-1} \quad (5)$$

The residual of L_{auc} for each positive instance $x\in D^+$ is

$$\tilde{y}_{\boldsymbol{x}}=-\left.\frac{\partial L_{auc}(f)}{\partial f(\boldsymbol{x}^+)}\right|_{f(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})}\propto\sum_{\boldsymbol{x}^-\in D^-}\frac{e^{-(F_{t-1}(\boldsymbol{x})-F_{t-1}(\boldsymbol{x}^-))}}{\left(1+e^{-(F_{t-1}(\boldsymbol{x})-F_{t-1}(\boldsymbol{x}^-))}\right)^2} \quad (6)$$

and for each negative instance $x\in D^-$ is

$$\tilde{y}_{\boldsymbol{x}}\ =\ -\left.\frac{\partial L_{auc}(f)}{\partial f(\boldsymbol{x}^-)}\right|_{f(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})}\propto\ -\sum_{\boldsymbol{x}^+\in D^+}\frac{e^{-(F_{t-1}(\boldsymbol{x}^+)-F_{t-1}(\boldsymbol{x}))}}{\left(1+e^{-(F_{t-1}(\boldsymbol{x}^+)-F_{t-1}(\boldsymbol{x}))}\right)^2} \quad (7)$$

The residuals of L_{pref} is:

$$\tilde{k}_{\boldsymbol{x}} = - \left.\frac{\partial L_{pref}(f)}{\partial f(\boldsymbol{x})}\right|_{f(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})} \propto \sum_{\boldsymbol{x}_a \in D} \frac{-k(\boldsymbol{x}, \boldsymbol{x}_a) e^{-(F_{t-1}(\boldsymbol{x})-F_{t-1}(\boldsymbol{x}')) \cdot k(\boldsymbol{x},\boldsymbol{x}_a)}}{\left(1 + e^{-(F_{t-1}(\boldsymbol{x})-F_{t-1}(\boldsymbol{x}')) \cdot k(\boldsymbol{x},\boldsymbol{x}_a)}\right)^2} \tag{8}$$

Then, fitting to the residuals least-squarely, we have

$$\theta_{t,1} = \arg\min_{\theta} \sum\nolimits_{\boldsymbol{x} \in D} (\tilde{y}_{\boldsymbol{x}} - h(\boldsymbol{x};\theta))^2 \ , \quad \theta_{t,2} = \arg\min_{\theta} \sum\nolimits_{\boldsymbol{x} \in D} (\tilde{k}_{\boldsymbol{x}} - h(\boldsymbol{x};\theta))^2$$

Next, defining $\boldsymbol{\beta} \doteq (\beta_1, \beta_2)$, we solve $\beta_{t,1}$ and $\beta_{t,2}$ that minimize

$$(\beta_{t,1}, \beta_{t,2}) = \arg\min\nolimits_{\boldsymbol{\beta}} L_\lambda\big(F_{t-1} + \beta_1 h_1(;\theta_{t,1}) + \beta_2 h_2(;\theta_{t,2})\big)$$

To do the minimization, we solve

$$G(\boldsymbol{\beta}) \doteq \frac{\partial L_\lambda\big(F_{t-1} + \beta_1 h_1(;\theta_{t,1}) + \beta_2 h_2(;\theta_{t,2})\big)}{\partial \boldsymbol{\beta}} = 0$$

by the Newton-Raphson iteration with one step:

$$\left\{ \begin{array}{l} \dfrac{\partial G(\boldsymbol{\beta})_1}{\partial \beta_1}\delta_1 + \dfrac{\partial G(\boldsymbol{\beta})_1}{\partial \beta_2}\delta_2 + G(\boldsymbol{\beta})_1 = 0 \\ \dfrac{\partial G(\boldsymbol{\beta})_2}{\partial \beta_1}\delta_1 + \dfrac{\partial G(\boldsymbol{\beta})_2}{\partial \beta_2}\delta_2 + G(\boldsymbol{\beta})_2 = 0 \end{array} \right|_{\boldsymbol{\beta}=\boldsymbol{\beta}^0=(1,\lambda)} \tag{9}$$

where δ_1 and δ_2 are *corrections* of β_1 and β_2, respectively. Also note that $G(\boldsymbol{w})$ is a vector, $G(\boldsymbol{w})_1$ and $G(\boldsymbol{w})_2$ are its first and second elements, respectively, and $(1, \lambda)$ is set as the initial guess of β. After δ_1 and δ_2 have been solved, we have

$$\beta_1 = 1 - \delta_1 \ , \quad \beta_2 = \lambda - \delta_2 \ . \tag{10}$$

Before forming the approach, There are two important issues to be dealt with. One is that, when the indication function $\boldsymbol{I}(\cdot)$ is replaced by the sigmoid function, there have different behaviors between Eq.1 and Eq.4. Consider two instances $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ with $y_{\boldsymbol{x}_a} = +1$ and $y_{\boldsymbol{x}_b} = -1$. By Eq.1, f receives no punishment as long as $f(\boldsymbol{x}_a) > f(\boldsymbol{x}_b)$. However, by Eq.4, f always receives punishment even if $f(\boldsymbol{x}_a) > f(\boldsymbol{x}_b)$. When $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are equal or very close[1] according to the user's preference, the objective function will still pay much attention on ranking $\boldsymbol{x}_a$ before $\boldsymbol{x}_b$, which makes the built model over complex. We handle this issue by removing a part of negative instances that are closest to the positive instances according to the user's preference, when fitting the model for AUC residuals.

The other issue is that some instances have either y values or preferences, but not both. We fit the model of AUC on instances where y values are available, fit the model for user preference on instances where preferences are available, and calculate the combination weights of the two models on the intersection of the instances.

[1] According to the user's preferences, we determine how close $\boldsymbol{x}_a$ is to $\boldsymbol{x}_b$ by counting how many instances that are either more preferred or less preferred than both of $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$.

Table 1. The SGBDota Approach

```
ALGORITHM ($D$, $T$, $h$, $\lambda$, $p$, $\nu$)
  $D$: training data;   $T$: number of iterations;   $h$: base learner;   $\lambda$: balance parameter
  $p$: proposition of negative instances to be removed;   $\nu$: shrinkage

1.  $D_{auc} \leftarrow \{\boldsymbol{x} \in D | y_{\boldsymbol{x}}$ is available $\}$
2.  $D_{pref} \leftarrow \{\boldsymbol{x} \in D |$ preference is available $\}$
3.  $D_{auc} \leftarrow D_{auc} - \{\boldsymbol{x} \in D^{-}_{auc} | \boldsymbol{x}$ is in the most preferred $p$ percent of $D^{-}_{auc}\}$
4.  $D_b \leftarrow D_{auc} \cap D_{pref}$
5.  $F_0(\boldsymbol{x}) \leftarrow 0$
6.  **for** $t \leftarrow$ **1 to** $T$
        *// calculate residuals*
7.      $\forall x \in D_{auc} : \tilde{y}_{\boldsymbol{x}} = - \left.\frac{\partial L_{auc}(f)}{\partial f(\boldsymbol{x})}\right|_{f(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})}$ by Eq.6 and 7
8.      $\forall x \in D_{pref} : \tilde{k}_{\boldsymbol{x}} = - \left.\frac{\partial L_{pref}(f)}{\partial f(\boldsymbol{x})}\right|_{f(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})}$ by Eq.8
        *// fit models*
9.      $\theta_{t,1} \leftarrow \arg\min_{\theta} \sum_{\boldsymbol{x} \in \text{Sample}(D_{auc})} (h_1(\boldsymbol{x};\theta) - \tilde{y}_{\boldsymbol{x}})^2$
10.     $\theta_{t,2} \leftarrow \arg\min_{\theta} \sum_{\boldsymbol{x} \in \text{Sample}(D_{pref})} (h_2(\boldsymbol{x};\theta) - \tilde{k}_{\boldsymbol{x}})^2$
        *// calculate combination weights*
11.     $(\beta_{t,1}, \beta_{t,2}) \leftarrow \arg\min_{(\beta_1,\beta_2)} L_{\lambda}(F_{t-1} + \beta_1 h_1(;\theta_{t,1}) + \beta_2 h_2(;\theta_{t,2}))$
        by Eq.9 and 10 on $D_b$
        *// update leaner*
12.     $F_t(.) \leftarrow F_{t-1}(.) + \nu(\beta_{t,1} h_1(;\theta_{t,1}) + \beta_{t,2} h_2(;\theta_{t,2}))$
13. **end for**
14. **return** $F_T(.)$
```

Table 1 presents the SGBDota (Stochastic Gradient Boosting with Double Target) algorithm. Note that in line 3, D_b might be small because it contains only the instances where y values and preferences are both available. But since D_b is only used to determine the 'step size' of the greedy search in line 11, it is unlikely to cause a great effect, especially when this effect will be further reduced by shrinkage parameter in line 12. In lines 9 and 10, the base learners are trained on a sample of the training data, but not on the training data directly, which is the essential part of Stochastic Gradient Boosting [6]. In line 12, the shrinkage is used to prevent from overfitting.

SGBDota have several parameters. λ can be set according to the user's confidence of the domain knowledge, ν could be set as 0.01 [6]. We eliminate p by a simple strategy: we run SGBDota with different p, and choose the value with which the preference is best fitted, i.e., the minimum L_{pref} that SGBDota reaches. Here L_{auc} is not involved in choosing of p because it contains no information about the expansion.

5 Experiments

5.1 Experimental Setting

In order to visualize the behavior of the proposed approach, we generate a 2-dimensional synthetic data set, by sampling 1,000 instances from four Gaussian models. In order to evaluate the performance of the proposed approach on real-world data sets, we derive four data sets from the UCI Machine Learning Repository [2].

There are three classes in *postoperative*, i.e., *I*: patient sent to Intensive Care Unit, *A*: patient sent to general hospital floor, and *S*: patient prepared to go home. We use only *I* as the positive class at training time, and *I* plus *A* as the positive class at testing time.

segment contains seven classes of outdoor images, i.e., *brickface*, *sky*, *cement*, *window*, *path*, *foliage*, and *grass*. We set *grass* as the positive class at training time, and *grass+foliage+path* as the positive class at testing time.

veteran is the veteran's administration lung cancer trial data. We set instances with survival time less than 12 hours as positive at training time, and instances with survival time within 48 hours as positive at testing time.

pcb is the data recorded from the Mayo Clinic trial in primary biliary cirrhosis, of the liver conducted between 1974 and 1984. We set instances with living time within 365 days and 1460 days as positive at training and testing time, respectively.

On data sets *postoperative, veteran, segment* and *pcb*, we randomly split 2/3 of the instances to be training set, and the rest 1/3 instances are used for test. Note that the class labeling is different for training set and test set. Any learner is evaluated on the test set using AUC criterion. The split is repeated 20 times to obtain an average performance.

We try three assumptions of "domain knowledge" for experiment. The first one assumes that the positive class expands from dense positive area to spares positive area, the second one assumes the positive class expands from dense positive area to spares positive and spares negative area. For efficiency, the density is estimated by 200 times of random space splits and counting instances in the local region. The third one assumes the positive class expands along with the neighborhoods using linear neighborhoods propagation [15] (positive label is +1 and negative label is zero). We name the SGBDota with the three assumptions as SGBDota-1, SGBDota-2, and SGBDota-3. Nevertheless, it is expected to incorporate real domain knowledge in real world applications.

The default parameter setting of SGBDota is: $T = 50$, $\nu = 0.01$ approximately be the log-median of the suggested range $[0.005, 0.05]$ from [6], h is set to be a regression tree [16] for its efficiency, $\lambda = 1$, and p is searched from $\{1, 0.95, 0.9, 0.6, 0.3, 0\}$ on training set by the search strategy mentioned before, where $p = 1$ means only the negative instances with the lowest preference are kept undeleted for minimizing L_{auc}.

Approaches compared to SGBDota include Random Forests, PU-SVM [17], SGBAUC and Random. Random Forests includes 100 trees. PU-SVM uses RBF kernel. SGBAUC is AUC optimization by stochastic gradient boosting with shrinkage 0.01 and 50 iterations. Random is the random guess approach, which serves as a lower bound. All the other default parameters are taken from WEKA [16].

5.2 Comparison with Other Methods

First, we visualize the behavior of our approach by the synthetic data set. The data set is plotted in Fig.1(a), the instance space is $[0, 1] \times [0, 1]$. Four approaches, Random Forests, PU-SVM, SGBAUC, and SGBDota-1, are trained on the data set. Then, the ranking decision of each approach is probed by testing on every instance $\boldsymbol{x} = (x_1, x_2) \in \{0, 0.01, \ldots, 1\} \times \{0, 0.01, \ldots, 1\}$, from which bitmaps are constructed and displayed in Fig.1(b), (c), (d), and (e), using histogram equalization.

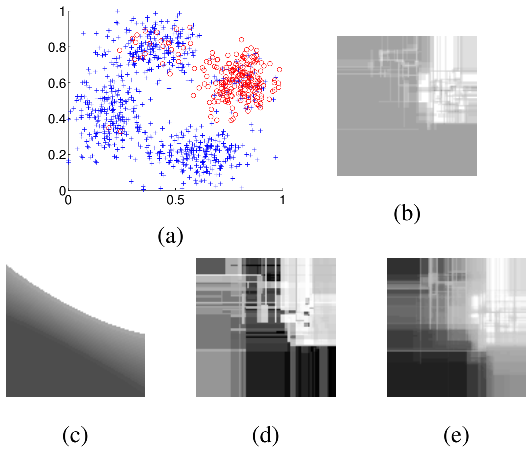

Fig. 1. (a) The synthetic data set, where red cycles indicate positive instances; (b)-(e) show the ranking decisions of Random Forests (b), PU-SVM (c), SGBAUC (d), and SGBDota-1 (e), respectively, where the more white the more positive.

Table 2. AUC values of SGBDota, Random Forests (RF), PU-SVM, SGBAUC, and Random. Each cell presents a *mean* ± *standard derivation*.

Data set	SGBDota-1	SGBDota-2	SGBDota-3	RF	PU-SVM	SGBAUC	Random
posto	.470±.131	.483±.111	.459±.132	.448±.076	.457±.107	.457±.084	.456±.148
segment	.821±.031	.822±.029	.744±.025	.750±.014	.753±.020	.744±.012	.506±.018
veteran	.658±.118	.650±.115	.544±.094	.637±.102	.627±.146	.658±.093	.522±.069
pbc	.721±.034	.726±.032	.638±.054	.710±.043	.709±.033	.665±.041	.503±.043

Table 3. Win/tie/loss counts of SGBDota against Random Forests (RF), PU-SVM, SGBAUC, and Random, by pairwise t-tests at 95% significance level.

	RF	PU-SVM	SGBAUC	Random
SGBDota-1	1/3/0	1/3/0	2/2/0	3/1/0
SGBDota-2	2/2/0	2/2/0	2/2/0	3/1/0
SGBDota-3	0/2/2	0/2/2	0/2/2	0/2/2

In Fig.1(a), since the expansion is from regions of high positive density to low positive density, we expect that there are two expanding paths, one is from the right cluster to the top cluster and then to the left cluster, the other is from the right cluster to the bottom cluster. From Fig.1(b), it is observed that Random Forests does not find the expanding path from the right cluster to the bottom cluster. From Fig.1(c), PU-SVM does not find the expanding path from the right cluster to the top cluster. From Fig.1(d). SGBAUC ignores the path from the left cluster to the bottom cluster. Finally, SGBDota concerns all the expanding paths, as shown in Fig.1(e).

Results of comparisons between SGBDota and the other methods are presented in Table 2. Since each approach is tested 20 times on each data set, for two approaches in comparison, we employ a pairwise two-tail t-test with significance level at 0.05 to test if they have significant differences. A win/loss is counted SGBDota is significantly better/worse than the comparing approach on a data set, otherwise a tie is counted. Table3 lists the counts. It can be found that SGBDota-1 and SGBDota-2 never lose, and are better than all the other approaches on *segment*. SGBDota-2 is moreover better than all the other approaches on *pbc*. This indicates that, with effective domain knowledge, our approach can exceed the-stat-of-the-arts learning approaches.

5.3 Influence of Parameters

We study the influence of the parameter p. The performances of SGBDota with $p = \{1, 0.95, 0.9, 0.6, 0.3, 0\}$, are depicted in Fig.2, where the performance of Random Forests, PU-SVM, SGBAUC, and Random are also plotted.

On *segment* and *pbc*, there is a large flat area where SGBDota-1 and SGBDota-2 have the best performance. On *postoperative* and *veteran*, SGBDota-1 and SGBDota-2 have less chance to be the best, which may be because the domain knowledge we used here is not strong. We also note that *postoperative* is a hard data set, where Random Forests is worse than Random, and SGBAUC can only achieve equal performance with Random, but SGBDota-1 and SGBDota-2 have a significant chance to exceed Random.

Then, we study how the parameter λ affects the performance of SGBDota. Here we use SGBDota-1 as a representative of the SGBDota approach. We test it by varying p

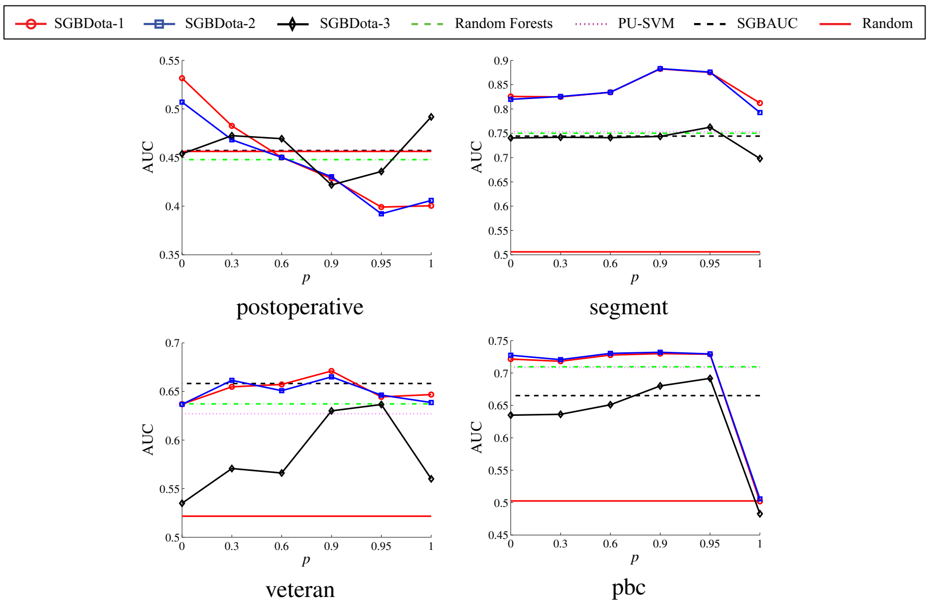

Fig. 2. The influence of the parameter p on the performance of SGBDota.

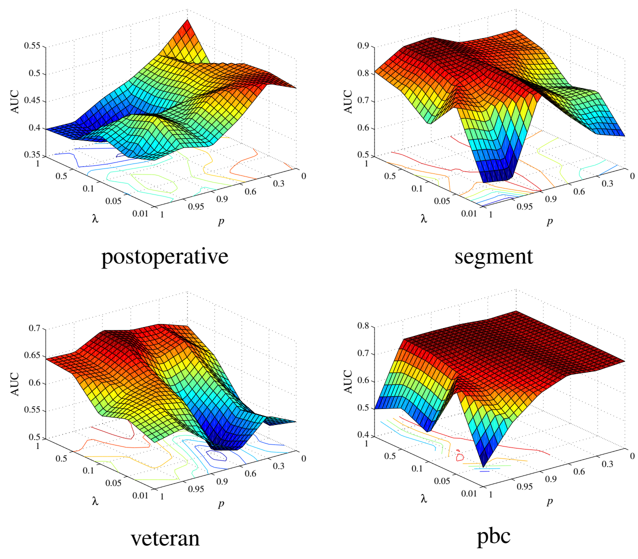

Fig. 3. The influence of the parameters λ and p on the performance of SGBDota.

in $\{1, 0.95, 0.9, 0.6, 0.3, 0\}$ and λ in $\{0.01, 0.05, 0.1, 0.5, 1\}$. Fig.3 plots the test results of SGBDota on the four UCI data sets.

From Fig.3, it can be observed that $\lambda = 1$ is better than smaller value, which is reasonable because if the used domain knowledge is useful, it should be heavily weighted, otherwise the knowledge is probably fake. While it is hard to choose a good fixed value of p, because when the domain knowledge is strong, p needs to be large to reduce the side-effect of optimizing L_{auc}, otherwise p should be small. Therefore, in practical use, it is convenient to let $\lambda = 1$, and set p according to user's confidence about the domain knowledge or leaving p be determined by the search strategy.

6 Conclusion

In this paper, we propose a framework to deal with positive class expansion problem with single snapshot. Our framework includes two elements, i.e., the utilization of the observation that *the instances that are currently positive become positive ahead of the instances that are currently negative*, and the incorporation of domain knowledge expressed as user preferences of pairs of instances. We formulate the problem as an optimization problem, and propose the SGBDota (Stochastic Gradient Boosting with Double Target) approach which achieves success in experiments.

In our future work we will try to apply our approach to some real-world tasks which suffer from the positive class expansion problem. We will also try to extend the proposed framework to other varying target concept problems, such as the positive class shrinking problem.

Acknowledgement

This research was supported by the National Science Foundation of China (60635030, 60721002), the National High Technology Research and Development Program of China (2007-AA01Z169), and the Foundation for the Author of National Excellent Doctoral Dissertation of China (200343).

References

1. Bickel, S., Brüeckner, M., Scheffer, T.: Discriminative learning for differing training and test distributions. In: Proceedings of the 24th International Conference on Machine Learning, Corvallis, OR (2007) 81–88
2. Blake, C.L., Keogh, E., Merz, C.J.: UCI repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, CA (1998)
3. Bradley A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30(7)** (1997) 1145–1159
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd ed. John Wiley and Sons, New York (2001)
5. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29(5)** (2001) 1189–1232
6. Friedman, J.H.: Stochastic gradient boosting. Computational Statistics & Data Analysis **38(4)** (2002) 367–378
7. Hand, D., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press, Cambridge, MA (2001)
8. Huang, J., Smola, A.J., Gretton, A., Borgwardt, K.M., Schölkopf, B.: Correcting sample selection bias by unlabeled data. In Schölkopf, B., Platt, J., Hoffman, T., eds.: Advances in Neural Information Processing Systems 19. MIT Press, Cambridge, MA (2007) 601–608
9. Klinkenberg, R., Joachims, T.: Detecting concept drift with support vector machines. In: Proceedings of the 17th International Conference on Machine Learning, Standord, CA (2000) 487–494
10. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: A new ensemble method for tracking concept drift. In: Proceedings of the 3rd IEEE International Conference on Data Mining, Los Alamitos, CA (2003) 123–130
11. Li, X., Liu, B.: Learning from positive and unlabeled examples with different data distributions. In: Proceedings of the 16th European Conference on Machine Learning, Porto, Portugal (2005) 218–229
12. Liu, B., Lee, W.S., Yu, P.S., Li, X.: Partially supervised classification of text documents. In: Proceedings of the 19th International Conference on Machine Learning, Sydney, Australia (2002) 387–394
13. Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of Statistical Planning and Inference **90(2)** (2000) 227–244
14. Vapnik, V.: Statistical Learning Theory. John Wiley and Sons, New York (1998)
15. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. In: Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA (2006) 985–992
16. Witten, I. H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed. Morgan Kaufmann, San Francisco (2005)
17. Yu, H., Han, J., Chang, K.C.C.: PEBL: Positive example based learning for web page classification using svm. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Alberta, Canada (2002) 239–248