Knowl Inf Syst (2010) 25:211–227
DOI 10.1007/s10115-009-0238-7

REGULAR PAPER

A framework for modeling positive class expansion with single snapshot

Yang Yu · Zhi-Hua Zhou

Received: 1 October 2008 / Revised: 5 April 2009 / Accepted: 20 June 2009 /
Published online: 1 August 2009

Abstract In many real-world data mining tasks, the connotation of the target concept may change as time goes by. For example, the connotation of "learned knowledge" of a student today may be different from his/her "learned knowledge" tomorrow, since the "learned knowledge" of the student is 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. In many tasks, however, 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 formulate the Positive Class Expansion with single Snapshot (PCES) problem and discuss its difference with existing problem settings. To show that this new problem is addressable, we propose a framework which involves the incorporation of desirable biases based on user preferences. The resulting optimization problem is solved by the Stochastic Gradient Boosting with Double Target approach, which achieves encouraging performance on PCES problems in experiments.

Keywords Data mining · Machine learning · Concept expansion · PCES

1 Introduction

In conventional data mining and machine learning research, it was assumed explicitly or implicitly that, the test data set is drawn from the same distribution of the training data set, and the target concept, i.e., a posteriori probability of class membership $p(y|\boldsymbol{x})$, is unchanged [5,17]. With this assumption, a learner, which properly fits the target concept in the training data set, generalizes well on the test data set.

Y. Yu · Z.-H. Zhou (✉)
National Key Laboratory for Novel Software Technology,
Nanjing University, 210093 Nanjing, China
e-mail: zhouzh@nju.edu.cn

This assumption, however, is often violated in real-world applications. An example is the "potential customer prediction problem":

Example 1 A telecom company has successfully launched a 3G mobile telecommunications network. After a brief operating, the company would like to identify which customers are likely to switch to their 3G network from the current 2G network, by making use of existing customer usage and demographic data.

In this situation, there will be more and more 3G customers and ultimately, all users will be 3G customers when 2G network is abandoned. Note that the training data are collected at an earlier time point while predictions are about a later time point. During the period, many 2G customers will become 3G users and they are what need to be predicted. The difficulty of predicting future customers is that these customers, however, were labeled as 2G customers in the training data. If a conventional learning algorithm is applied to the training data, it would not discover these customers since they are 2G users according to the training data.

A possible solution to this problem is to collect a series of training sets at a series of time points, such that the pattern of the evolution can be considered. However, data used in data mining tasks are usually *observational* [8]. That is, data are usually given by other people and the data miner could not collect more data. Actually, in many tasks we are given only one data set instead of a series of training data sets. Thus, the above-mentioned 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 first formulate the Positive Class Expansion with single Snapshot (PCES) problem, and discuss its difference with existing problem settings. To show that PCES problems are solvable, we propose a framework involving two elements. The first element is the utilization of the observation that positive training instances become positive ahead of negative training instances. The second element is the incorporation of desirable biases based on user preferences. These two elements are unified into an optimization problem which is solved by the Stochastic Gradient Boosting with Double Target (SGBDota) approach. The SGBDota approach achieves encouraging performance in experiments, which validates the usefulness of our framework for dealing with the PCES problem.

The rest of the paper is organized as follows. Section 2 formulates the PCES problem. Section 3 discusses the difference between PCES and existing problem settings. Section 4 proposes a solution to the PCES problem. Section 5 reports on experiments. Finally, Sect. 6 concludes.

2 The PCES problem

In traditional learning setting, we are given a training set of i.i.d. instances $D = \{\boldsymbol{x}_i\}_{i=1}^n$ drawn from a distribution $\mathcal{D}$, where each instance is associated with a random variable y called class label which is determined by $p(y|\boldsymbol{x})$. A loss function is used to measure how close a function $f(\cdot; D, p(y|\boldsymbol{x}))$ is to the posterior probability function $p(y|\boldsymbol{x})$, that is,

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

A desired learning algorithm outputs a function $\hat{f}(\cdot; D, p(y|\boldsymbol{x}))$ which minimizes the error, that is,

$$\hat{f}(\cdot; D, p(y|\boldsymbol{x})) = \arg\min_f \mathrm{err}_{f(\cdot;D,p(y|\boldsymbol{x}))},$$

which means that the learned function should approach the underlying probability function $p(y|\boldsymbol{x})$ as much as possible over the instance space. Since it was assumed that the class labels of test instances are also determined by $p(y|\boldsymbol{x})$, such a function will generalize well on test data if the assumption holds.

In the PCES problem, the target concept is in expansion, i.e., the posterior probability function $p(y|\boldsymbol{x})$ changes along time axis. The training set D is collected at an earlier *training time*. We call the training set as a *snapshot* of the changing posterior probability function. After being trained on the snapshot, a classifier is obtained to make predictions at a later *testing time*. Denote $p_{\text{tr}}(y|\boldsymbol{x})$ and $p_{\text{te}}(y|\boldsymbol{x})$ as the posterior probability functions at the training and testing time, respectively.

It is obvious that the PCES problem cannot be handled in traditional learning setting, since the posterior probability function at testing time $p_{\text{te}}(y|\boldsymbol{x})$ is out of the setting, which leads to a poor generalization of the learned classifier to test data. For the PCES problem, the loss function used at the testing time is

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

which measures how close the learned function is to the posterior probability function at the testing time $p_{\text{te}}(y|\boldsymbol{x})$. So, a desired learning algorithm outputs a function $\hat{f}(\cdot;\, D,\, p(y|\boldsymbol{x}))$ which minimizes the error

$$\begin{aligned}&\hat{f}(\cdot;\, D,\, p(y|\boldsymbol{x})) = \arg\min_{f} \text{err}^{\text{PCES}}_{f(\cdot;D,p(y|\boldsymbol{x}))}\\ &\text{s.t.}\quad \forall \boldsymbol{x}\sim\mathcal{D}: p_{\text{te}}(y=y_t|\boldsymbol{x}) \geq p_{\text{tr}}(y=y_t|\boldsymbol{x}),\end{aligned}$$

where y_t is the target concept, and the constraint implies the expansion of the target class.

For convenience of discussion, in this paper we assume a binary classification situation, i.e., $y \in \{-1, +1\}$, and the positive class is the target concept. Thus, the optimization problem is

$$\begin{aligned}&\hat{f}(\cdot;\, D,\, p(y|\boldsymbol{x})) = \arg\min_{f} \text{err}^{\text{PCES}}_{f(\cdot;D,p(y|\boldsymbol{x}))}\\ &\text{s.t.}\quad y \in \{-1, +1\}\\ &\forall \boldsymbol{x}\sim\mathcal{D}: p_{\text{te}}(y=+1|\boldsymbol{x}) \geq p_{\text{tr}}(y=+1|\boldsymbol{x}).\end{aligned}$$

It is noticeable that PCES problem is ubiquitous. In addition to the Example 1 we mentioned in the previous section, here we give some more examples.

Example 2 A supermarket is preparing stock for a holiday sale. To prevent out of stock, hot selling items need to be predicted, by learning from previous selling record.

In this example, preventing out of stock is of interest, thus the items that were hot selling are labeled as positive and the rest are labeled as negative in the training data. However, there are always new fashions, hence some items labeled as negative in the training set may become positive (hot selling) at a later time point. Thus, the positive class is in expansion and this task is a PCES problem.

Example 3 A Disease Control and Prevention Center receives several fatal cases of an unclear infectious disease. In order to perform effective quarantining actions, a model is urgently required to predict potential victims, by learning from periodical physical examination database.

In this example, the initial patients are labeled as positive while other people are labeled as negative in training set. However, some negative instances will become positive because some people who were found healthy may become infected. Thus, the positive class is in expansion and this task is a PCES problem.

3 Related work

The PCES problem appears to have some relation with *PU-Learning* [14,22], *concept drift* [11,12], and *covariate shift* [1,9]. In this section we will discuss on their distinct differences.

In PU-Learning, i.e., learning with positive and unlabeled data, it is required to discriminate positive instances from negative instances, while only positive training instances are given. A large part of work addressing PU-Learning, e.g., [14,22], follows two steps. First, strong negative instances are discovered from the unlabeled data. Then a predictive model is built from the positive and identified negative instances. Most of the work makes an assumption that the positive instances in the training set are representative of the positive concept. In contrast to PU-Learning, in PCES problems the positive class is in expansion and thus the positive training instances are not representative. We have noticed a recent work of PU-Learning that considers different training and test distributions [13]. This work, however, gears heavily to text mining using text-specific mechanisms, thus cannot be applied to general cases.

Concept drift considers an online learning environment, where instances are coming sequentially batch by batch, and the target concept may change in the coming batch. Approaches for concept drift should be able to detect the drift correctly and adapt to it quickly, e.g., [11,12,19,27]. In contrast to PCES where only a single snapshot is available, concept drift approaches expect a series of data sets involving information for drift detection. In other words, concept drift approaches cannot work on a single snapshot data. We also note that, in the Markov chain framework, regime switching models [10], which are frequently used in economic analysis, try to handle the posterior probability changes. Regime switching, however, is in general a concept drift model, which requires a series of data with available information of changing trends, thus cannot handle the PCES problem where only a single snapshot is available.

In covariate shift (or *sample selection bias* [16]), 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

$$\mathrm{err}_{f(\cdot;D,p(y|\boldsymbol{x}))} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\mathrm{te}}} \boldsymbol{L}\left(f(\boldsymbol{x}; D \sim \mathcal{D}_{\mathrm{tr}}, p(y|\boldsymbol{x})), p(y|\boldsymbol{x})\right),$$

where $\mathcal{D}_{\mathrm{tr}}$ and $\mathcal{D}_{\mathrm{te}}$ are the distributions at training and testing time, respectively. Covariate shift approaches (e.g., [1,9]) try to correct the bias in the training instances, such that the minimization of training error will lead to the minimization of test error. In contrast to PCES problem where the posterior probability is in changing, covariate shift assumes that the posterior probability is fixed. Therefore, covariate shift approaches cannot be used to solve the PCES problem.

To the best of our knowledge, the first attempt to addressing the PCES problem was reported in Yu et al. [23], where an approach tackling the problem of Example 1 was proposed, named GetEnsemble. GetEnsemble models the positive class expansion by simulating an expanding process. Essentially, it first trains a decision tree from the training data. Then, it finds negative instances that are covered by positive leaves, i.e., leaves which contain more

positive instances than negative instances. Finally, it relabels these negative instances as positive ones. This process is repeated, and thus, more and more negative instances are relabeled as positive. These instances are predicted as the potential positive instances. It can be observed that GetEnsemble simply assumes a particular expanding scheme, and is designed for decision tree instead of a general learning approach. In this paper, we formally define the PCES problem and propose a framework that leads to a general learning approach that can handle various expanding schemes.

The approach we proposed to solve the optimization problem in our framework is derived from Gradient Boosting [6,7]. 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,

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

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

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

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

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

4 A solution to PCES

Imagine a time line that every instance is negative at the beginning and becomes positive at its turn-to-positive time. An ideal learner ranks all instances according to their turn-to-positive time. This motivates the idea that the learner should rank examples according to the available information of their turn-to-positive time.

In PCES, since the positive class is in expansion, it is obvious that positive training instances become positive ahead of negative training instances. This observation suggests that all the positive instances should be ranked above the negative instances, which is exactly expressed by the widely used area under ROC curve (AUC) [3] criterion. Here, we use $1 - \text{AUC}$ as the loss function to evaluate how the priority of positive instances against negative instances has been captured, that is,

$$L_{\text{auc}}(f) = 1 - \frac{1}{|D^+| \cdot |D^-|} \sum_{\boldsymbol{x}^+ \in D^+} \sum_{\boldsymbol{x}^- \in D^-} \boldsymbol{I}(f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-)), \tag{1}$$

where D^+ and D^- are subsets of D that contain all positive and all negative instances, respectively, and $\boldsymbol{I}(a)$ gets 1 if $a \geq 0$ and 0 otherwise. Note here the pairwise rank-loss is used, which is equivalent to AUC.

L_{auc} considers only the goodness of the final ranking of positive and negative instances. Note that different expansion processes may lead to the same final ranking but very different rankings during the expansion period, and therefore, it is necessary to consider the expansion scheme for constructing a good model. Since the training data contain limited information about how the positive class expands, we propose to incorporate desirable bias which reflects the concerned expansion scheme based on user preferences, considering that all learning algorithms have inherent biases.

In many cases the user is able to indicate pairwise preferences on some instances. For example, pairs of instances can be randomly drawn from the training set, and then the user can be asked to judge which instance would become positive earlier from his/her viewpoint. Another possibility is to apply a priori rules to decide which instance would become positive earlier, such as Mongoloid people may be easier to get SARS than Caucasoid people. Let $k(\cdot,\cdot)$ denote the user's pairwise preferences in the way 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

$$L_{\text{pref}}(f) = 1 - \frac{1}{|D|^2} \sum_{\boldsymbol{x}_a \in D} \sum_{\boldsymbol{x}_b \in D} \boldsymbol{I}\left((f(\boldsymbol{x}_a) - f(\boldsymbol{x}_b)) \cdot k(\boldsymbol{x}_a, \boldsymbol{x}_b)\right), \tag{2}$$

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 is shown in Eq. 3, which is a combination of Eqs. 1 and 2.

$$\hat{f} = \arg\min_f L_\lambda(f) = \arg\min_f L_{\text{auc}}(f) + \lambda L_{\text{pref}}(f). \tag{3}$$

We realize the framework based on Gradient Boosting [6,7], which is a variant of AdaBoost [21,26]. In the original Gradient Boosting algorithm, each instance $\boldsymbol{x}$ is associated with a target label $y_{\boldsymbol{x}}$. In Eq. 3, however, each instance $\boldsymbol{x}$ is associated with 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

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

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 Eqs. 1 and 2, $\boldsymbol{I}(\cdot)$ is non-differentiable. We use sigmoid function $(1+\mathrm{e}^{-a})^{-1}$ to replace the identification function $\boldsymbol{I}(a)$. So the loss functions are rewritten as

$$L_{\text{auc}}(f) = 1 - \frac{1}{|D^+| \cdot |D^-|} \sum_{\boldsymbol{x}^+ \in D^+} \sum_{\boldsymbol{x}^- \in D^-} \left(1 + \mathrm{e}^{-(f(\boldsymbol{x}^+) - f(\boldsymbol{x}^-))}\right)^{-1}, \tag{4}$$

$$L_{\text{pref}}(f) = 1 - \frac{1}{|D|^2} \sum_{\boldsymbol{x}_a \in D} \sum_{\boldsymbol{x}_b \in D} \left(1 + \mathrm{e}^{-(f(\boldsymbol{x}_a) - f(\boldsymbol{x}_b)) \cdot k(\boldsymbol{x}_a, \boldsymbol{x}_b)}\right)^{-1}. \tag{5}$$

The residual [6,7] of L_{auc} for each positive instance $x \in D^+$ is

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

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

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

The residuals of L_{pref} is

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

Then, fitting to the residuals in a least-square way, we have

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

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

$$(\beta_{t,1}, \beta_{t,2}) = \arg\min_{\boldsymbol{\beta}} L_\lambda \left(F_{t-1} + \beta_1 h_1(;\theta_{t,1}) + \beta_2 h_2(;\theta_{t,2})\right).$$

For the minimization, we solve

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

by the Newton–Raphson iteration with one step, that is,

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

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

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

Two important issues need our consideration. The first is that, when the indicator function $\boldsymbol{I}(\cdot)$ is replaced by the sigmoid function, there are significant differences between Eqs. 1 and 4. As an example, let us 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 so far as $f(\boldsymbol{x}_a) > f(\boldsymbol{x}_b)$; by Eq. 4, however, f always receives a punishment even when $f(\boldsymbol{x}_a) > f(\boldsymbol{x}_b)$. When $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are equal or very close [1] according to the user's preferences, the learner will still pay much attention on ranking $\boldsymbol{x}_a$

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

Table 1 The SGBDota approach

ALGORITHM (D, T, h, λ, p, ν)

D: training data

T: number of iterations

h: base learner

λ: balance parameter

p: proportion of negative instances to be removed

ν: shrinkage

1. $D_{\text{auc}} \leftarrow \{\boldsymbol{x} \in D | y_{\boldsymbol{x}}$ is available $\}$
2. $D_{\text{pref}} \leftarrow \{\boldsymbol{x} \in D |$ preference is available $\}$
3. $D_{\text{auc}} \leftarrow D_{\text{auc}} - \{\boldsymbol{x} \in D_{\text{auc}}^{-} | \boldsymbol{x}$ is in the most preferred p percent of $D_{\text{auc}}^{-}\}$
4. $D_b \leftarrow D_{\text{auc}} \cap D_{\text{pref}}$
5. $F_0(\boldsymbol{x}) \leftarrow 0$
6. **for** $t \leftarrow$ **1 to** T

   // calculate residuals

7. $\forall x \in D_{\text{auc}} : \tilde{y}_{\boldsymbol{x}} = -\left.\frac{\partial L_{\text{auc}}(f)}{\partial f(\boldsymbol{x})}\right|_{f(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})}$ by Eqs. 6 and 7
8. $\forall x \in D_{\text{pref}} : \tilde{k}_{\boldsymbol{x}} = -\left.\frac{\partial L_{\text{pref}}(f)}{\partial f(\boldsymbol{x})}\right|_{f(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})}$ by Eq. 8

   // train base models

9. $\theta_{t,1} \leftarrow \arg\min_{\theta} \sum_{\boldsymbol{x} \in \text{Sample}(D_{\text{auc}})} (h_1(\boldsymbol{x}; \theta) - \tilde{y}_{\boldsymbol{x}})^2$
10. $\theta_{t,2} \leftarrow \arg\min_{\theta} \sum_{\boldsymbol{x} \in \text{Sample}(D_{\text{pref}})} (h_2(\boldsymbol{x}; \theta) - \tilde{k}_{\boldsymbol{x}})^2$

    // 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 Eqs. 9 and 10 on D_b

    // update learner

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** $F_T(.)$

before $\boldsymbol{x}_b$ due to the objective function, which makes the resulting model overly complicated. To deal with this issue, among other possible ways, we remove some negative instances that are the closest to the positive instances according to the user preferences, when fitting the model for AUC residuals.

The other issue is that there are possibly some instances which have either y values or preferences, but not both. To deal with this issue, we fit a model for AUC on instances where y values are available, and fit another model for user preferences on instances where user preferences are available, and then calculate the combination weights of the two models on instances which have both y values and user preferences.

Table 1 presents the SGBDota algorithm. Note that in line 4, D_b might be small because it contains only the instances where y values and preferences are both available. Since D_b is

only used to determine the 'step size' of the greedy search in line 11, it is unlikely to cause a large influence, especially when the influence will be reduced further by shrinkage parameter in line 12. In lines 9 and 10, the base learners are trained on a sample of the training data instead of the training data directly, which is a essential part of Stochastic Gradient Boosting [7]. In line 12, the shrinkage is used to reduce the chance of overfitting.

For parameters of SGBDota, λ can be set according to the user's confidence about the preferences, ν can be set as 0.01 [7]. We choose p with a simple strategy, that is, we run SGBDota with different p and choose the value with which the preferences are best fitted, i.e., with the minimum L_{pref} that SGBDota reaches. Here L_{auc} is not involved in choosing p because it contains little information about the expansion scheme.

5 Experiments

5.1 Settings

In order to visualize the behavior of the proposed approach, we generate a two-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 tasks, we derive four data sets from the UCI Machine Learning Repository [2], i.e., *postoperative*, *segment*, *veteran* and *pcb*.

postoperative contains three classes of patients, i.e., *I*: patients sent to Intensive Care Unit, *A*: patients sent to general hospital floor, and *S*: patients 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 h as positive at training time, and instances with survival time within 48 h 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 1,460 days as positive at training and testing time, respectively.

Figure 1 presents a two-dimensional visualization of the four UCI data sets, using the first and the second principal components of each data set. It can be observed from Fig. 1 that, the positive class in each training set, which is represented by '+' markers, has a different distribution to the positive class in the corresponding test set, which is represented by '*' markers.

On each of the four UCI data sets, we randomly pick two-thirds of the data to compose the training set, and use the remaining one-third of the data for testing. Note that the class labeling is different for training set and test set. Learners will be evaluated on the test set. The split is repeated for 20 times to obtain an average performance.

We try three preferences in the experiments. The first one assumes that the positive class expands from dense positive area to sparse positive area. The second one assumes that the positive class expands from dense positive area to sparse positive and sparse negative area. For efficiency, the density of an instance is estimated by 200 times of random space splits and counting instances located in the local region of the instance, which is implemented using complete-random trees [15]. The third one assumes that the positive class expands along

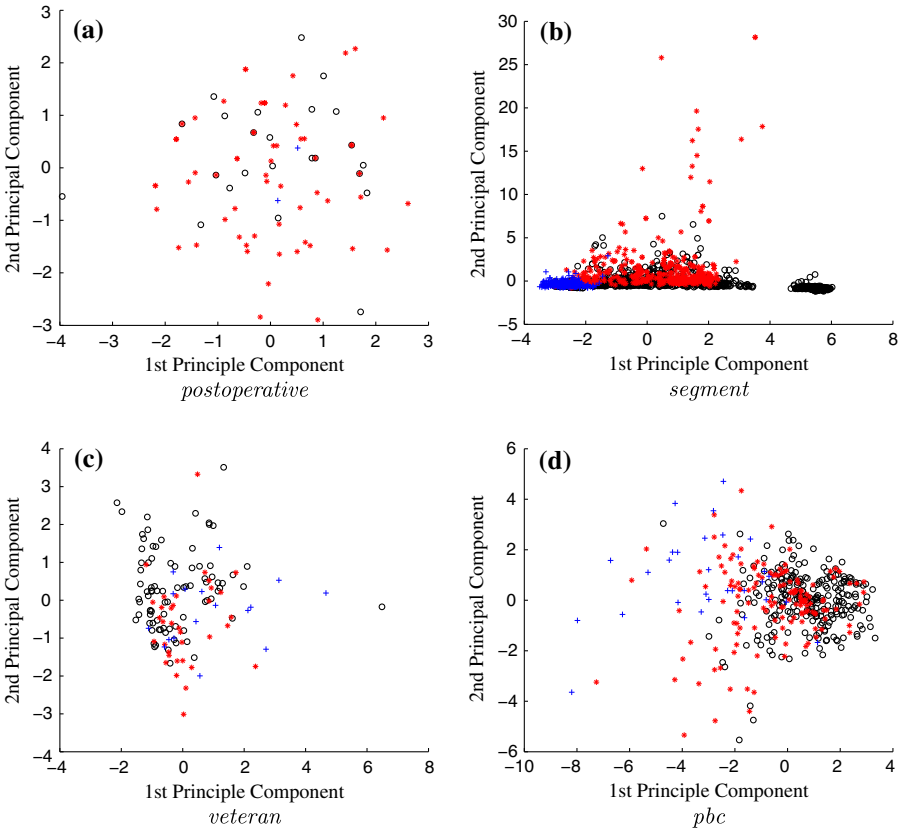

Fig. 1 Visualization of the four UCI data sets. Each data set is projected onto a two-dimensional plan using its first and second principal components. Instances marked by '+' are positive at both training time and testing time; those marked by '*' are positive at testing time but not at the training time; those marked by 'o' are negative at both training time and testing time.

with the neighborhoods using *linear neighborhoods propagation* [18] (positive label is +1 and negative label is 0). Note that the linear neighborhoods propagation is less reasonable than the first two preferences. The performance with different preferences will be examined in the experiments. We name the SGBDota with the three preferences as SGBDota-1, SGBDota-2 and SGBDota-3, respectively. Note that other preferences can also be used with SGBDota.

The default parameter setting of SGBDota is as follows. $T = 50$, $\nu = 0.01$ is approximately the log-median of the suggested range [0.005, 0.05] according to [7], h is realized as a regression tree [20] 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 that only the negative instances with the lowest preference are kept for minimizing L_{auc}.

We compare SGBDota with *GetEnsemble* [23], *PU-SVM* [22], *Random Forests* [4], *SGBAUC* and *Random*. PU-SVM uses RBF kernel. Random Forests, which is recognized as one of the most successful tree ensemble methods [25], includes 100 trees. SGBAUC is a degenerated version of SGBGota, which performs AUC optimization by stochastic gradient

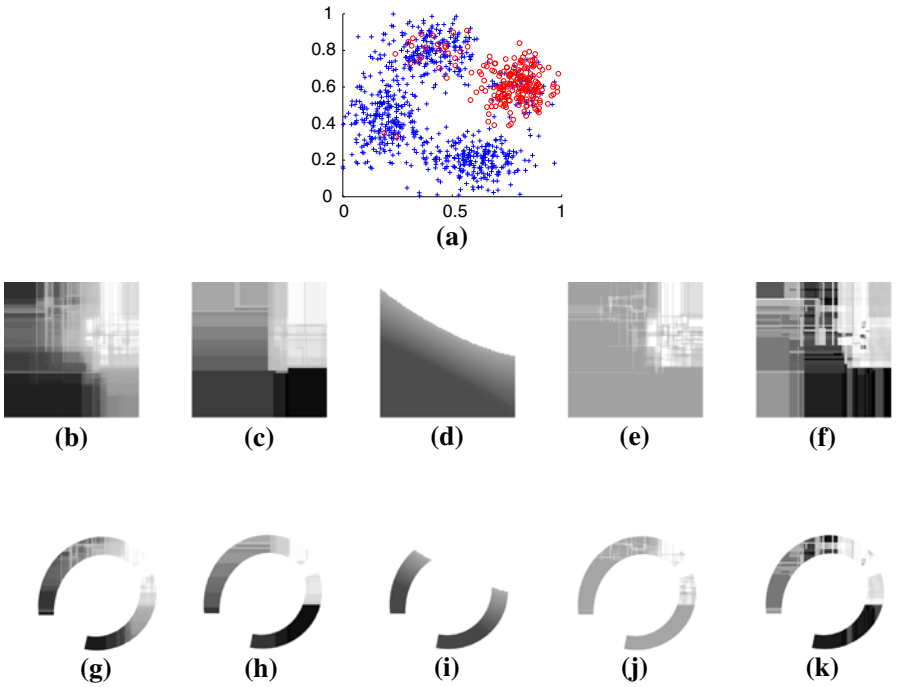

Fig. 2 **a** The synthetic data set, where *cycles* indicate positive instances; **b–f** show the ranking decisions of **b** SGBDota-1, **c** GetEnsemble, **d** PU-SVM, **e** SGBAUC, and **f** Random Forests, respectively, where the more light the more positive. For a clear perception, the two expanding paths in **b–f** are highlighted in **g–k**, respectively. The more smooth the paths growing from light to dark, the better the performance

boosting with shrinkage 0.01 and 50 iterations. Random makes prediction by random guess, which serves as a baseline. All the other parameters are set to default values in WEKA [20].

5.2 Results

First, we visualize the performance of the proposed approach on the synthetic data set. The data set is plotted in Fig. 2a, where the instance space is $[0, 1] \times [0, 1]$ and four clusters of instances are generated by four Gaussian models. We call the clusters as the top, the left, the bottom and the right cluster, respectively. Instances in the right cluster are mostly positive, and the remaining clusters are mainly negative. Five approaches, SGBDota-1, GetEnsemble, PU-SVM, Random Forests and SGBAUC are trained in the training 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 a bitmap per approach is constructed and displayed in Fig. 2b–f, using histogram equalization.

In Fig. 2a, 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, while the other is from the right cluster to the bottom cluster. From Fig. 2c–f, it can be observed that GetEnsemble ignores the path from the right cluster to the bottom cluster; PU-SVM does not find the expanding path from the right cluster to the top cluster; Random Forests tries to distinguish positive instances from negative instances instead of finding the expanding paths; and SGBAUC produces a quite irregular ranking,

Table 2 AUC values (mean ± STD) of SGBDota, Random Forests (RF), PU-SVM, SGBAUC, GetEnsemble and Random

Approach	*posto*	*segment*	*veteran*	*pbc*
SGBDota-1	0.470 ± 0.131	0.821 ± 0.031	0.658 ± 0.118	0.721 ± 0.034
SGBDota-2	0.483 ± 0.111	0.822 ± 0.029	0.650 ± 0.115	0.726 ± 0.032
SGBDota-3	0.459 ± 0.132	0.744 ± 0.025	0.544 ± 0.094	0.638 ± 0.054
GetEnsemble	0.464 ± 0.083	0.757 ± 0.030	0.663 ± 0.090	0.684 ± 0.033
SGBAUC	0.457 ± 0.084	0.744 ± 0.012	0.658 ± 0.093	0.665 ± 0.041
PU-SVM	0.457 ± 0.107	0.753 ± 0.020	0.627 ± 0.146	0.709 ± 0.033
RF	0.448 ± 0.076	0.750 ± 0.014	0.637 ± 0.102	0.710 ± 0.043
Random	0.456 ± 0.148	0.506 ± 0.018	0.522 ± 0.069	0.503 ± 0.043

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

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

which might be caused by the fact that optimizing AUC alone would meet too many local optimum to reach a good result. In contrast to these approaches, SGBDota concerns all the expanding paths, as shown in Fig. 2b. To have a clear visual perception, the two expanding paths are cut out from the images and presented in Fig. 2g–k. The more smooth the paths growing from light to dark, the better.

Table 2 presents the comparison of SGBDota with other methods on the four real-world data sets. We employ pairwise t tests with 95% significance level to judge whether there are significant differences or not, and the results are summarized in Table 3, where a win/loss is counted if SGBDota is significantly better/worse than the compared method on a data set, and otherwise a tie is counted.

As shown in Table 3, SGBDota-1 and SGBDota-2 are never significantly worse than the compared methods, and are significantly better than all the other approaches on *segment*. Moreover, SGBDota-2 is better than all other approaches on *segment* and *pbc*. The data sets *postoperative* and *veteran* have relatively fewer and sparser instances; this might explain why SGBDota does not have a superior performance. The above observations suggest that, by incorporating proper preferences, SGBDota can exceed stat-of-the-art approaches. On the other hand, SGBDota-3 is with worse performance, which probably owes to the fact that the assumption of linear neighborhood propagation in the preference is not always reasonable. This suggests that we must be careful when incorporating user preferences, because incorporating misleading preferences will lead to a poor performance.

5.3 Influence of parameters

First, we study the influence of the parameter p, which controls the percentage of negative instances to be ignored to avoid an over-complicated model. The performances of

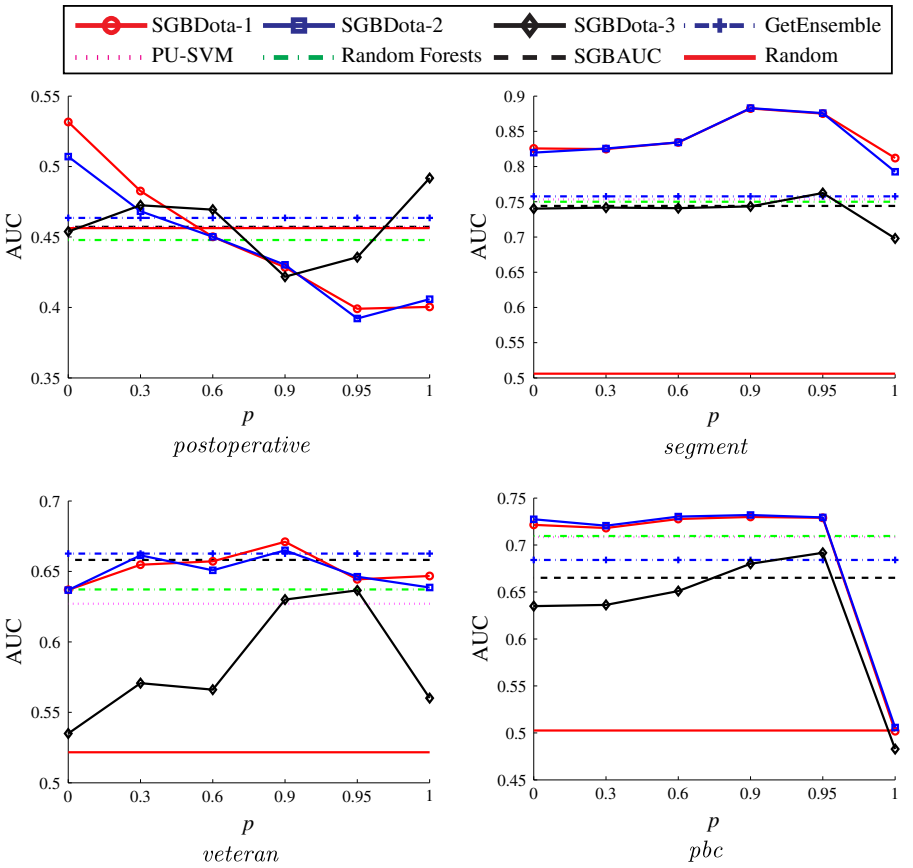

Fig. 3 The influence of the parameter p on the performance of SGBDota

SGBDota with $p = \{1, 0.95, 0.9, 0.6, 0.3, 0\}$ are plotted in Fig. 3, where the performances of GetEnsemble, PU-SVM, Random Forests, SGBAUC and Random are also shown.

On *segment* and *pbc*, there is a large flat region where SGBDota-1 and SGBDota-2 reach the best performance. On *postoperative* and *veteran*, SGBDota-1 and SGBDota-2 have a less chance to be the best, which may be caused by that the preferences are not very appropriate. We also note that *postoperative* is a difficult data set, where Random Forests is even worse than Random, and SGBAUC and GetEnsemble are only comparable to Random, but SGBDota-1 and SGBDota-2 have a chance to exceed Random significantly.

Next, we study how the parameter λ affects the performance of SGBDota. This parameter controls the tradeoff between the AUC criterion and the preferences. The larger the λ value, the more dominative the user preferences. Here, we use SGBDota-2 and SGBDota-3 as representatives of the SGBDota approach, which were shown as incorporating the best and the worst preferences in the above section, respectively. We test them by varying p in $\{1, 0.95, 0.9, 0.6, 0.3, 0\}$ and λ in $\{0.01, 0.05, 0.1, 0.5, 1\}$. Figures 4 and 5 plot the test results of SGBDota-2 and SGBDota-3 on the four UCI data sets, respectively.

From Figs. 4 and 5 we can find that, $\lambda = 1$ is a good choice in all cases except for SGBDota-3 on *postoperative*. This suggests that the AUC criterion and user preferences are

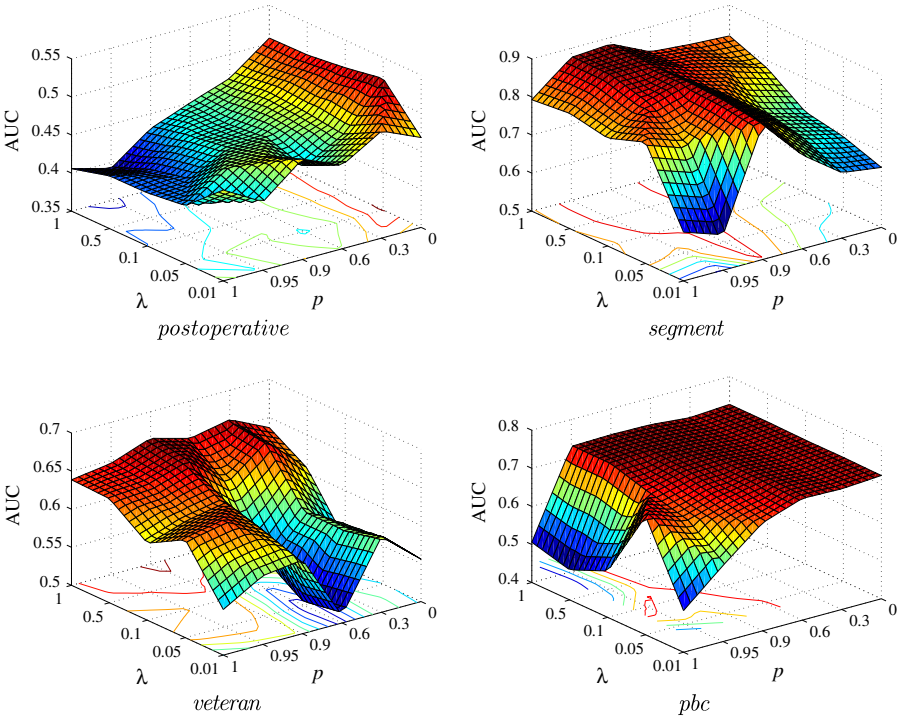

Fig. 4 The influence of the parameters λ and p on the performance of SGBDota-2

comparably important in most cases. Therefore, in practice it is convenient to let $\lambda = 1$, and set p according to the user's confidence about the preferences or simply leave p be determined by the search.

6 Conclusion

This paper extends our preliminary study which defines the PCES problem [24], In PCES, the connotation of target class concept is in expansion from training time to testing time, while only a data set collected at the training time is available. PCES is different from existing problem settings, and effective solution to PCES can be beneficial to many real-world applications.

To show that PCES problems are solvable, in this paper we propose a framework which involves two elements, i.e., the utilization of the training data based on the observation that positive training instances became positive ahead of negative training instances, and the incorporation of appropriate biases based on user preferences. We formulate the problem as an optimization problem, and propose the SGBDota approach, which achieves encouraging performance in experiments.

It will be interesting to study PCES problems in more real-world applications. We will also try to extend the proposed framework to variants of PCES, such as problems where the positive class is in shrinking.

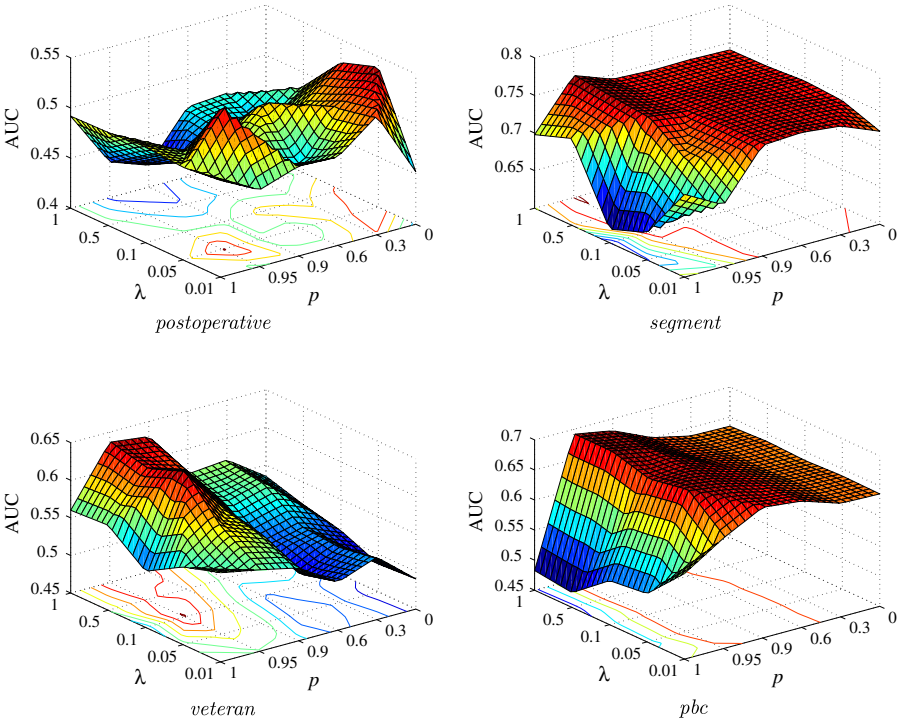

Fig. 5 The influence of the parameters λ and p on the performance of SGBDota-3

Acknowledgments The authors wish to thank Sheng-Jun Huang and Wei-Wei Tu for proof reading the article, and the anonymous reviewers for helpful comments. This research was supported by the National Science Foundation of China (60635030, 60721002), the National High Technology Research and Development Program of China (2007AA01Z169), the Foundation for the Author of National Excellent Doctoral Dissertation of China (200343), the Jiangsu Science Foundation (BK2008018) and the Jiangsu 333 High-Level Talent Cultivation Program.

References

1. Bickel S, Brüeckner M, Scheffer T (2007) Discriminative learning for differing training and test distributions. In: Proceedings of the 24th international conference on machine learning, Corvallis, OR, pp 81–88
2. Blake CL, Keogh E, Merz CJ (1998) UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, CA. http://www.ics.uci.edu/~mlearn/MLRepository.html
3. Bradley AP (1997) The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159
4. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
5. Duda RO, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley, New York
6. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29(5):1189–1232
7. Friedman JH (2002) Stochastic gradient boosting. Comput Stat Data Anal 38(4):367–378
8. Hand D, Mannila H, Smyth P (2001) Principles of data mining. MIT Press, Cambridge

9. Huang J, Smola AJ, Gretton A, Borgwardt KM, Schölkopf B (2007) 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, pp 601–608
10. Kim C-J, Nelson CR (1999) State-space models with regime switching. MIT Press, Cambridge
11. Klinkenberg R, Joachims T (2000) Detecting concept drift with support vector machines. In: Proceedings of the 17th international conference on machine learning, Standord, CA, pp 487–494
12. Kolter JZ, Maloof MA (2003) Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Proceedings of the 3rd IEEE international conference on data mining, Melbourne, FL, pp 123–130
13. Li X, Liu B (2005) Learning from positive and unlabeled examples with different data distributions. In: Proceedings of the 16th European conference on machine learning, Porto, Portugal, pp 218–229
14. Liu B, Lee WS, Yu PS, Li X (2002) Partially supervised classification of text documents. In: Proceedings of the 19th international conference on machine learning, Sydney, Australia, pp 387–394
15. Liu FT, Ting KM, Yu Y, Zhou Z-H (2008) Spectrum of variable-random trees. J Artif Intell Res 32:355–384
16. Shimodaira H (2000) Improving predictive inference under covariate shift by weighting the log-likelihood function. J Stat Plan Inference 90(2):227–244
17. Vapnik V (1998) Statistical learning theory. Wiley, New York
18. Wang F, Zhang C (2006) Label propagation through linear neighborhoods. In: Proceedings of the 23rd international conference on machine learning, Pittsburgh, PA, pp 985–992
19. Wang K, Zhang J, Shen F, Shi L (2008) Adaptive learning of dynamic Bayesian networks with changing structures by detecting geometric structures of time series. Knowl Inf Syst 17(1):121–133
20. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco CA
21. Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS, Zhou Z-H, Steinbach M, Hand DJ, Steinberg D (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
22. Yu H, Han J, Chang KC-C (2002) 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, pp 239–248
23. Yu Y, Zhanc D-C, Liu X-Y, Li M, Zhou Z-H (2007) Predicting future customers via ensembling gradually expanded trees. Int J Data Warehous Min 3(2):12–21
24. Yu Y, Zhou Z-H (2008) A framework for modeling positive class expansion with single snapshot. In: Proceedings of the 12th Pacific-Asia conference on knowledge discovery and data mining, Osaka, Japan, pp 429–440
25. Zhou Z-H (2008) Ensemble learning. In: Li SZ (ed) Encyclopedia of biometrics. Springer, Berlin
26. Zhou Z-H, Yu Y (2009) AdaBoost. In: Wu X, Kumar V (eds) The top ten algorithms in data mining. Chapman & Hall, Boca Raton
27. Zhu X, Wu X, Yang Y (2006) Effective classification of noisy data streams with attribute-oriented dynamic classifier selection. Knowl Inf Syst 9(3):339–363

Author Biographies

Yang Yu received his B.Sc. degree in computer science from Nanjing University, China, in 2004. Currently he is a Ph.D. candidate in the Department of Computer Science and Technology at Nanjing University, and is a member of the LAMDA Group. His main research interests include machine learning and evolutionary computation, especially in ensemble learning, theoretical aspects of evolutionary algorithms, and the combination of the two.

Zhi-Hua Zhou is currently Professor in the Department of Computer Science and Technology and director of the LAMDA group at Nanjing University. His main research interests include machine learning, data mining, pattern recognition and information retrieval. He is associate editor-in-chief of *Chinese Science Bulletin*, associate editor of *IEEE Transactions on Knowledge and Data Engineering*, and on the editorial boards of *Artificial Intelligence in Medicine*, *Intelligent Data Analysis*, *Journal of Computer Science and Technology*, *Science in China*, etc. He was an associate editor of *Knowledge and Information Systems* (2003–2008). He is a steering committee member of PAKDD and PRICAI, and has served as program chair/co-chair of PAKDD'07 and PRICAI'08, and vice chair or area chair of ICDM'06, ICDM'08, SDM'09, CIKM'09, etc.