The Entire Solution Path for Support Vector Machine in Positive and Unlabeled Classification[1]

Yao Limin, Tang Jie, and Li Juanzi
Department of Computer Science, Tsinghua University
1-308, FIT, Tsinghua University, Beijing, China, 100084
{ylm, tangjie, ljz}@keg.cs.tsinghua.edu.cn

Abstract

Support Vector Machines (SVMs) is aimed at finding an optimal separating hyper-plane that maximally separates the two classes of training examples (more precisely, maximizes the margin between the two classes of examples). The hyper-plane, corresponding to a classifier, is obtained from the solution of a problem of quadratic programming that depends on a cost parameter. The choice of the cost parameter can be critical. However, in conventional implementations of SVMs it is usually supplied by the user or set as a default value. In this paper, we study how the cost parameter determines the hyper-plane. We especially focus on the case of classification using only positive and unlabeled data. We propose an algorithm that can fit the entire solution path by choosing the 'best' cost parameter while training SVM models. We compare the performance of the proposed algorithm with the conventional implementations that use default values as the cost parameter on two synthetic data sets and two real-world data sets. Experimental results show that the proposed algorithm can achieve better results when dealing with positive and unlabeled classification.

Keywords: Support Vector Machine; Cost Parameter; Positive and Unlabeled Classification.

---

[1] Received: 2008-3-16

Supported by the National Natural Science Foundation of China (No. 90604025, No. 60703059), Chinese Young Faculty Research Funding (20070003093)

Corresponding author: Yao Limin. Tel: 62788788-20; Email: ylm@keg.cs.tsinghua.edu.cn

Introduction

Support Vector Machines (SVMs), a new generation learning system based on recent advances in statistical learning theory, deliver state-of-the-art performance in real-world applications such as text categorization, hand-written character recognition, image classification, and bioinformatics [1] [2].

We recall the standard formulation of SVM. Given a set of training data $\{(x_1, y_1), \cdots, (x_n, y_n)\}$, in which x_i denotes an example (feature vector) and $y_i \in \{+1, -1\}$ denotes its classification label. In the linear case, the formulation of SVM is

$$\begin{aligned} &\min_{\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i \\ &s.t. \quad y_i(\beta_0 + \beta^T x_i) \geq 1 - \xi_i \end{aligned} \tag{1}$$

where the ξ_i are non-negative slack variables that allow points to be on the wrong side of their "soft margin" ($f(x) = \pm 1$), as well as the decision boundary. C is the cost parameter that controls the trade-off between the largest margin and the lowest number of errors. Intuition shows that in the separable case, with a sufficiently large C, the solution achieves the maximal margin separator and in the non-separable case, the solution achieves a largest margin with minimum errors (sum of the ξ_i).

The choice of the C can be critical [3]. The characteristics of the hyper-plane can vary largely using different values of C. This is especially true in the problem of positive and unlabeled classification (shortly PU classification) that is aimed at building classifiers with only positive and unlabeled examples, but no negative examples [4]. PU classification problem naturally arise in many real-world applications, for example, text classification, homepage finding, filtering spam from people's emails, image segmentation, and information extraction. In such cases, the

user only annotates part of the positive examples while let the others unlabeled, and hope that a *good* classifier can be learned.

Unfortunately, in conventional implementations of SVMs, the cost parameter C is usually supplied by the user or set as a default value. For example, SVM-light [5] uses the average norm of training examples as the default value. In practice, our empirical study shows that the quality of the trained SVM model may be very sensitive to C in PU classification. Thus, it is very important to discover an optimal value for C given a specific task of PU classification. This is exactly the problem addressed in this paper.

Recently, several approaches have been proposed to deal with PU classification using SVMs. For example, Liu et al. propose Biased SVM applying SVM to PU classification [6]. In Biased SVM, the authors propose to choose the best value for C by employing a cross validation method to verify the performance of resulting SVM models with the various values. Yu proposes an extension of the standard SVM approach called SVMC (Support Vector Mapping Convergence) for PU classification [7] [8]. SVMC basically exploited the natural "gap" between positive and negative examples in the feature space, which eventually corresponds to improve the generalization performance. However, SVMC suffered from under-sampling of positive documents, leading to overfit at some points and generalize poor results. See also Roc-SVM [4], S-EM [9], and PEBL [10]. Most of the approaches avoid the problem of the choice of C or use the empirical method (e.g. cross validation), which results in very high computational cost. It is also possible to discard the unlabeled data and learn only from the positive data. This was done in the one-class SVM [11], which tries to learn the support of the positive distribution. However, its performance is limited because it cannot take advantage of the unlabeled data.

The other type of related work is unbalance classification, where the task is to deal with very unbalanced numbers of positive and negative examples. Methods, for example two cost parameters are introduced for SVM to adjust the cost of false positives vs. false negatives [12], have been proposed. [13] also proposes a variant of the SVM — the SVM with uneven margins, tailored for text classification with the unbalanced problem. The unbalance classification is in nature different from the PU classification, as in the former problem labels of both positive and negative examples are annotated while in the later problem the unlabelled examples include part of positive examples and all negative examples.

Pontil and Verri have studied properties of Support Vector Machines [14]. They have investigated the dependence of hyper-plane on the changes of the cost parameter. In [3] the authors argue that the choice of the SVM cost parameter can be critical. They propose an algorithm, called SvmPath, which can fit the entire path of SVM solutions for all possible values of C. The algorithm is based on the properties of so called piecewise-linear. [15] [16] also investigate the issue of the solution path for support vector regression to the cost parameter. [15] derives an algorithm to compute the entire solution path of the support vector regression. [16] proposes an algorithm for exploring the two-dimensional solution space defined by the regularization and cost parameters. See also [17] [18]. Our work is inspired by the work of [3]. The difference is that we focus on the problem of PU classification while [3] focuses on the classical classification.

This paper addresses the issue of fitting the entire solution path for SVMs in PU classification. We propose an algorithm, called PU-SvmPath, to do the choice of the cost parameter automatically while training SVM models for PU classification. We implemented the algorithm

and conducted experiments on synthetic data to evaluate the effectiveness of the proposed PU-SvmPath. We also applied PU-SvmPath to Bio-medical data classification and text classification. Experimental results show that the new algorithm is superior to the existing algorithms in PU classification.

This paper is organized as follows. In Section 2 we give the problem setting. In Section 3 we describe our approach and in Section 4 we explain our algorithm in details and in Section 5, we present the experimental results. We make concluding remarks in Section 6.

1. Problem Setting

In this paper, we consider PU classification. Here we first give the definition of the problem.

Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training data set, in which x_i denotes an example (a feature vector) and $y_i \in \{+1, -1\}$ denotes a classification label. Assume that the first k-1 examples are positive (labeled +1), the rest are unlabeled, which we considered as negative (-1) (Note: they might consist of unlabeled positive examples and real negative examples.). We denote by I_+ the set of indices corresponding to y_i=+1 points (positive examples), there being $n_+=|I_+|$ in total. Likewise for I_- and n_-, with $I = I_+ \cup I_-$. Our goal is to estimate a decision function $f(x) = \beta^T x + \beta_0$ (also called classifier). The noiseless case (no errors for positive examples but only for unlabeled examples) results in the following SVM formulation:

$$
\begin{aligned}
&\min_{\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=k}^{n}\xi_i \\
&s.t. \quad y_i(\beta_0 + \beta^T x_i) \geq 1 \qquad i \in I_+ \\
&\qquad y_i(\beta_0 + \beta^T x_i) \geq 1 - \xi_i \qquad i \in I_- \\
&\qquad \xi_i \geq 0 \qquad i \in I_-
\end{aligned}
\tag{2}
$$

Here, only unlabeled examples have the slack variables ξ_i indicating that errors only are allowed for unlabeled examples. To distinguish this formulation from the classical SVM, we call it as PU-SVM.

The objective function in the above formulation can be written in an alternatively equivalent form with the same constraints:

$$\min_{\beta} \sum_{i=k}^{n} \xi_i + \frac{\lambda}{2} \beta^T \beta \tag{3}$$

where the parameter λ corresponds to $1/C$ in (2). The formulation is also called as *Loss+Penalty* criterion [3]. In this paper, we will use the later formulation in explaining our algorithm and thus our goal of choosing the cost parameter C is cast as choosing the parameter λ (we call it as regularization parameter). For (3), we can construct the Lagrange primal function:

$$\sum_{i=k}^{n} \xi_i + \frac{\lambda}{2} \beta^T \beta + \sum_{i=1}^{n} \alpha_i (1 - y_i f(x_i)) - \sum_{i=k}^{n} \alpha_i \xi_i - \sum_{i=k}^{n} \gamma_i \xi_i \tag{4}$$

and set the derivatives to zero. We have:

$$\frac{\partial}{\partial \beta}: \qquad \beta = \frac{1}{\lambda} \sum_{i=1}^{n} \alpha_i y_i x_i, \qquad i \in I \tag{5}$$

$$\frac{\partial}{\partial \beta_0}: \qquad \sum_{i=1}^{n} \alpha_i y_i = 0, \qquad i \in I \tag{6}$$

$$\frac{\partial}{\partial \xi_i}: \qquad 1 - \alpha_i - \gamma_i = 0, \qquad i \in I_- \tag{7}$$

along with the KKT conditions:

$$\alpha_i (1 - y_i f(x_i)) = 0, \qquad i \in I_+ \tag{8}$$

$$\alpha_i (1 - y_i f(x_i) - \xi_i) = 0, \qquad i \in I_- \tag{9}$$

$$\gamma_i \xi_i = 0, \qquad i \in I_- \tag{10}$$

Substituting the formula (5) - (7) into (4), we obtain the Lagrange dual form:

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j x_i x_j y_i y_j \\ &s.t. \quad \alpha_i \geq 0, i \in I_+ \\ &\qquad 1 \geq \alpha_i \geq 0, i \in I_- \\ &\qquad \sum_{i=1}^{n} \alpha_i y = 0 \end{aligned} \tag{11}$$

We see that for positive examples we have constraints $\alpha_i \geq 0$ ($i \in I_+$), for unlabeled examples we have constraints $1 \geq \alpha_i \geq 0$ ($i \in I_-$). When $y_i f(x_i) = 1$, the point x_i is on the margin (called Support Vector). For the positive support vectors, we have $\alpha_i > 0$, while for the unlabeled support vectors, we have $1 > \alpha_i > 0$. When $y_i f(x_i) > 1$, the point x_i is outside the margin. In this case, both positive and unlabeled examples have $\alpha_i = 0$. When $y_i f(x_i) < 1$, the point x_i is inside the margin (since we are dealing with the noiseless case, only unlabeled examples can lie inside the margin). When this occurs, unlabeled examples have their $\alpha_i = 1$.

Our goal now is to solve the equation (11) so as to find the *entire solution path* for all possible $\lambda \geq 0$.

2. Our Approach: The Entire Generalization Path for SVM

We propose an algorithm for solving the equation (11). Our basic idea is as follows. Same as that in [3], we start with λ large and decrease it toward zero, keeping track of all the 'events' that occur along the way. Different from [3], as the initial value of λ, we propose to carefully select a 'large' one, with which we can construct an initial hyper-plane with all positive examples correctly classified. Then as λ decreases, $\|\beta\|$ increases (See formula (5)), and hence the width of the margin decreases. As this width decreases, points move from being inside to outside the margin (in our problem, only unlabeled examples inside the margin at the beginning). Their corresponding α_i change from $\alpha_i = 1$ when they are inside the margin ($y_i f(x_i) < 1$) to $\alpha_i = 0$ when they are outside the margin ($y_i f(x_i) > 1$). By continuity, those points must linger on the margin ($y_i f(x_i) = 1$) while their α_i decrease from 1 to 0. In this process we always keep all the positive examples correctly classified. Positive points can only leave the margin to outside the margin,

with their corresponding $\alpha_i>0$ changing to $\alpha_i=0$ or join the margin from the outside with $\alpha_i=0$ changing to $\alpha_i>0$.

In our approach, each point x_i belongs to one of the following three sets. (For convenience, we denote α_i of positive examples as $\alpha_i^p=\{\alpha_i|i\in I_+\}$ and likewise, $\alpha_i^n=\{\alpha_i|i\in I_-\}$.)

- M=$\{i: y_i f(x_i) = 1, \alpha_i^p>0, 1>\alpha_i^n>0\}$, M=$M^p \cup M^n$ for Margin, where M^p denotes positive points on the margin ($f(x)=1$) and M^n denotes unlabeled ones on the margin ($f(x)= -1$).
- L=$\{i: y_i f(x_i) < 1, \alpha_i^n =1\}$, L for inside the margin, only unlabeled points.
- R=$\{i: y_i f(x_i) > 1, \alpha_i=0\}$, R for outside the margin.

Points in various sets may have various events (one event can be viewed as an action of leaving one set to enter into another set). By tracking all the events in iterations, we can solve the entire generalization path and select the best solution. One of the key issues here is how to define the events for efficiently learning. We will give the detailed definitions of the events in Section 4.3.

3. Algorithm: PU-SvmPath

We have implemented our algorithm PU-SvmPath by extending SvmPath [3], which is used for finding entire generalization path for conventional Support Vector Machine. In this paper, we only consider the noiseless case of positive and unlabeled classification for facilitating the description. However, the proposed algorithm can be extended to the noisy case also (allowing noise in the labeled positive examples). The related issues are what we are currently researching, and will be reported elsewhere.

3.1. Outline

The input of our algorithm is a training set which consists of positive and unlabeled examples. The objective is to find a PU-SVM model (including β and λ in (3) or α and λ in Lagrange form implicit in (11)).

Algorithm: PU-SvmPath
Step 1. Initialization: determine the initial values of α and λ, and thus β and β_0;
1. $\alpha_i^n=1,\ \forall i \in I_-$;
2. Finding the values of α_i^p by solving; (see Section 4.2)
   $\min_\alpha \left\|\beta^*(\alpha)\right\|^2$
3. Calculating λ_0, β and β_0 using the initial values of α;
Step 2. Finding entire generalization path; (see Section 4.3)
4. do{
5. Building all possible events based on the current values of the parameters;
6. For each event, calculating the new λ by supposing it occurs;
7. Selecting the largest $\lambda<\lambda_l$ from all the events as λ_{l+1};
8. Updating α with λ_{l+1} by the fact that α are piecewise-linear in λ_{l+1};
9. Updating the sets M, L, and R;
10. }while (terminal conditions are not satisfied).

Figure 1. The Algorithm of PU-SvmPath

In our algorithm: PU-SvmPath, we propose solving the entire regularization path in the following steps:

(1) Initialization. It determines the initial values for β, α and λ. With the initial values, we can establish the initial state of the three point sets defined above. When determining the initial values, we need consider the fact that n_- is much larger than n_+ and the constraint that all positive examples should be correctly classified. Moreover, we need to satisfy the constraints (6). We then employ a quadratic programming algorithm to obtain the initial configuration.

(2) Finding entire generalization path. The algorithm runs iteratively. In each iteration, we try to find a new λ_{l+1}, based on the current value λ_l (l denotes the l-th iteration). It searches in the possible event space for an event which has the largest $\lambda<\lambda_l$. Then it establishes the λ_{l+1} and

hence updates α with λ_{l+1} according to the fact that α are piecewise-linear in the regularization parameter λ_{l+1}.

(3) Termination. The algorithm runs until some terminal conditions are satisfied.

Figure 1 summarizes the proposed algorithm. In the rest of the section, we will explain the three steps in details.

3.2. Initialization

In initialization, the task is to find the initial values for β_0, α, and λ. We denote the initial λ as λ_0. In order to find an initial value for λ, we first consider all unlabeled example inside the margin (note: the larger the λ, the larger the margin). Thus for all unlabeled example ($i \in I_-$), all the $\xi_i > 0$, $\gamma_i = 0$, and hence $\alpha_i = 1$. In this case, when β=0, the optimum choice for β_0 is 1, and the loss is $\sum_{i=1}^{n_-} \xi_i = 2n_-$. However, we are required that (6) holds.

We formalize the initialization problem as that of optimization. Here we use Lagrange primal form (11) of the PU-SVM as the objective function.

At the start point, all unlabeled examples have their α_i = 1, along with the KKT condition $\sum_{i=1}^{n} \alpha_i y_i = 0$, the first part of the objective function results in $\sum_{i=1}^{n} \alpha_i = 2n_-$, which is a constant. Let $\beta^* = \beta\lambda$. From (5), we have

$$\beta^*(\alpha) = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{12}$$

We can then obtain a new objective function from (11) with constraints as follows:

$$\begin{aligned} &\min_{\alpha} \left\| \beta^*(\alpha) \right\|^2 \\ &s.t. \qquad \alpha_i \in [0,1], i \in I_+ \\ &\qquad\quad \alpha_i = 1, i \in I_- \\ &\qquad\quad \sum_{i \in I_+} \alpha_i = n_- \end{aligned} \tag{13}$$

Solving this problem, we can get the values of α_i ($i \in I_+$). We now establish the "starting point" λ_0 and β_0. We can first calculate the β^* by (12). As mentioned above, all α_i^p ($i \in I_+$) are either 0 or $\alpha_i^p > 0$. Suppose $\alpha_i^p > 0$ (say on the margin). Let $i_- = \text{argmin}_{i \in I_-} \beta^T x_i$. Then the points i_+ and i_- are on the positive margin ($\beta^T x_{i+} + \beta_0 = 1$) and the negative margin ($\beta^T x_{i-} + \beta_0 = -1$) respectively. From the following equations:

$$\frac{\beta^{*T} x_{i_+}}{\lambda} + \beta_0 = 1 \quad \frac{\beta^{*T} x_{i_-}}{\lambda} + \beta_0 = -1 \tag{14}$$

We have

$$\lambda_0 = \frac{\beta^{*T} x_{i_+} - \beta^{*T} x_{i_-}}{2} \quad \beta_0 = -\frac{\beta^{*T} x_{i_+} + \beta^{*T} x_{i_-}}{\beta^{*T} x_{i_+} - \beta^{*T} x_{i_-}} \tag{15}$$

3.3. Finding λ_{l+1}

As mentioned above, the points fall into three sets: M, L, and R. For each set of points, there are several possible events. We define the events as follows:

a. The initial event, which means that two or more points enter the margin. The event happens at the beginning of the algorithm (initialization) or when the set *M* is empty.

b. A point leaves from *R* to enter *M*, with its value of α_i initially 0.

c. A point $i \in I_-$ leaves from *L* to enter *M*, with its value of α_i initially 1.

d. One or more points in *M* leave to join *R*.

e. One or more points $i \in I_-$ in *M* enter *L*.

Whichever the case, for continuity reason, the sets will stay stable until an event occurs. For a point in *R*, only one event can occur, i.e. event *b*; two events can occur on a point in *M*, i.e. event *e* and event *d*; one event (event *c*) can occur on a point in *L* (only unlabeled points). At the beginning of the algorithm or when the set *M* is empty, the event *a* will occur.

When staying in a stable situation, i.e. α changes linearly with λ, as implicit in equation 21, we can imagine the possible events of all the points. For each event, we calculate the new value λ and hence the new α. We select the event that has the largest $\lambda<\lambda_l$ and use its λ as the λ_{l+1}. Then we can update α. The process continues until some terminal conditions are satisfied.

The key point then is how to compute the new value of λ_{l+1} for an event. Considering a point passing from R through M to L, its α will change from 0 to 1, vice versa (points in I_+ are constrained to stay in M or R only). When the point x_i is on the margin, we have $y_i f(x_i) = 1$. In this way, we can establish a path for each α_i.

We adopt the method proposed in [3] to compute the λ_{l+1} for an event. We now explain the method in details. We use the subscript l to represent the l-th event occurred. Suppose $|M_l|=m$, and let α_i^l, β_0^l, and λ_l be the values of these parameters at the point of entry. Likely f^l is the function at this point. For convenience we define $\alpha_0=\lambda\beta_0$, and hence $\alpha_0^l=\lambda\beta_0^l$. Then we have

$$f(x) = \frac{1}{\lambda}\left(\sum_{i=1}^{n} y_j \alpha_j (x \bullet x_j) + \alpha_0\right) \tag{16}$$

For $\lambda_l > \lambda > \lambda_{l+1}$, we can write

$$\begin{aligned} f(x) &= [f(x) - \frac{\lambda_l}{\lambda} f^l(x)] + \frac{\lambda_l}{\lambda} f^l(x) \\ &= \frac{1}{\lambda}[\sum_{j \in M_l} (\alpha_j - \alpha_j^l) y_j (x \bullet x_j) + (\alpha_0 - \alpha_0^l) + \lambda_l f^l(x)] \end{aligned} \tag{17}$$

The second line follows because all the unlabeled points in L_l have their $\alpha_i=1$, and those in R_l have their $\alpha_i=0$, for this range of λ. Since each of the m points $x_i \in M_l$ are to stay on the margin, we have $y_i f(x_i) = 1$. According to (17), we have:

$$y_i f(x_i) = \frac{1}{\lambda}[\sum_{j \in M_l} (\alpha_j - \alpha_j^l) y_j y_i (x_i \bullet x_j) + y_i(\alpha_0 - \alpha_0^l) + \lambda_l] = 1 \tag{18}$$

Writing $\delta_j=\alpha_j^l - \alpha_j$, from (18) we have

$$\sum_{j \in M_l} \delta_j y_i y_j (x_i \bullet x_j) + y_i \delta_0 = \lambda_l - \lambda, \forall i \in M_l \tag{19}$$

Furthermore, since at all times we are required that (6) holds, we have that

$$\sum_{j \in M_l} \delta_j y_j = 0 \tag{20}$$

Equation (19) and (20) constitute m+1 linear equations with m+1 unknown δ_j. We can obtain $\delta_j=(\lambda_l - \lambda)b_j$, $j \in 0 \cup M_l$, where b_j is a variable obtained by solving the equations. Hence:

$$\alpha_j = \alpha_j^l - (\lambda_l - \lambda)b_j, j \in 0 \cup M_l \tag{21}$$

The equation (21) means that for $\lambda_l > \lambda > \lambda_{l+1}$, the α_j will change linearly in λ.

We can also write (17) as

$$f(x) = \frac{\lambda_l}{\lambda}[f^l(x) - h^l(x)] + h^l(x) \tag{22}$$

where

$$h^l(x) = \sum_{j \in M_l} y_j b_j (x \bullet x_j) + b_0 \tag{23}$$

Thus the function changes in an inverse manner in λ for $\lambda_l > \lambda > \lambda_{l+1}$.

We now obtain an important property that α_j change *linearly* in λ between two events (called *piecewise-linearly*), which enables us easily establish λ for each event from (21) and (22):

- when one of the point x_i leave M to enter L or R, we have its α_i=1 or α_i=0. According to (21), we can compute its λ by

$$\lambda = \frac{(1 - \alpha_j^l)}{b_j} + \lambda_l \quad \text{or} \quad \lambda = \frac{(0 - \alpha_j^l)}{b_j} + \lambda_l \tag{24}$$

- when one of points from L or R to enter M, we have $y_i f(x_i)$=1. Substituting it into (22), we can obtain its λ by

$$\lambda = \frac{\lambda_l}{y_i - h^l(x)}[f^l(x) - h^l(x)] \tag{25}$$

In this way, we obtain λ for each possible event and thus are able to select the largest $\lambda < \lambda_l$ as the λ_{l+1}. We then make use of the property that α_j change piecewise-linearly in λ to obtain all α.

3.4. Termination

In positive and unlabeled learning, λ runs all the way down to zero. For this to happen without f blowing up in (22), we must have f^l - h^l=0. So that the boundary and margins remain fixed at a point where $\sum_i \xi_i$ is as small as possible, and the margin is as wide as possible subject to this constraint.

3.5. Kernels

The proposed algorithm can be easily extended to the more general kernel form. In the kernel case, we can replace the inner product $(x \bullet x_j)$ in (16) by a kernel function K

$$f(x) = \frac{1}{\lambda}(\sum_{i=1}^{n} y_j \alpha_j K(x \bullet x_j)) + \beta_0 \quad (26)$$

This general kernel case makes our algorithm support non-linear classification problem.

4. Experiments

4.1. Datasets and Experiment Setup

4.1.1. Datasets

We evaluated our proposed method on two synthetic data sets and two real-world data sets.

We first constructed two synthetic data sets: PU_toy1 and PU_toy2. Both of them are two-dimensional. PU_toy1 consists of 100 positive examples and 100 negative examples, which were generated using a function of two-multivariate norm distributions with mean $[2,2]^T$ and covariance [2,0; 0,2] for positive examples and mean $[-2, -2]^T$ and covariance [2,0; 0,2] for negative examples. PU_toy2 consists of 200 positive examples and 200 negative examples, which were also generated using a function of two-multivariate norm distributions with mean $[2, -3]^T$ and covariance [2,0; 0,2] for positive points and mean $[-3, 2]^T$ and covariance [2,0; 0,2] for negative points.

We also tried to carry out experiments on real-world data sets: a Bio-medical data set: Types of Diffuse Large B-cell Lymphoma (DLBCL)[2] and a Text Classification data set: 20newsgroups[3]. The former data describes the distinct types of diffuse large B-cell lymphoma (DLBCL), the most common subtype of non-Hodgkin's lymphoma, using gene expression data. There are 47 examples, 24 of them are from "germinal centre B-like" group while 23 are "activated B-like" group. Each example is described by 4026 genes [19]. For the text classification data, we chose eight categories from 20newsgroups[2] to create a data set. The eight categories are: *ms-windows.misc*, *graphics*, *pc.hardware*, *misc.forsale*, *hockey*, *christian*, *sci.crypt*, *rec.autos*. Each category contains 100 documents.

4.1.2. Experiment Setup

In all experiments, we conducted evaluation in terms of F1-measures, which is defined as $F1 = 2PR / (P + R)$. Here P and R respectively represent precision and recall.

We made comparison with existing methods: SvmPath [3], SVM-light [5], Bias-SVM [6], and One-class SVM [11]. SvmPath is proposed for fitting the entire regularization path for the classical SVM, not for PU classification. We made comparison with SvmPath to indicate the necessity of proposed method. SVM-light is also designed for classical SVM and it needs the user to provide values for the cost parameter. We use the default values as the parameters in SVM-light. For SvmPath and SVM-light, we view the unlabeled examples as negative examples. Bias-SVM targets at PU classification. It takes into consideration of both labeled and unlabeled examples. It needs the user to provide two cost parameters respectively for labeled positive examples and unlabeled examples. In [6] the authors propose employing cross-validation to

[2] http://sdmc.lit.org.sg/GEDatasets/Datasets.html#DLBCL

select the best cost parameters. It obviously results in high computational cost. In our experiments, we use a typical method by considering the numbers of positive and unlabeled examples to determine the values for the cost parameters. One-class SVM is appropriate for *one-class* classification (it tries to learn a classifier from the positive data). Table 1 indicates the methods we used to set the values for the cost parameters in the experiments.

Table 1. The Methods for Setting the Cost Parameters

Methods	**Cost Parameter**	
SVM-light	C=1/avg^2	
Bias-SVM	C1=log(n-)/(avgp^2*log(n+))	C2=1/(avgn^2)

In the table, C, $C1$, and $C2$ denote the cost parameters. *avg*, *avgp*, *avgn* respectively represent the average norm of all examples, positive examples, and negative examples. n_+ and n_- are the numbers of the positive and negative examples (Note that in PU classification, we take unlabeled examples as negative).

4.2. Experimental Results

4.2.1. Results on the Synthetic Dataset

We conducted the experiments as follows. We split each of the data set into two sub sets with the same size, one for training and one for test. In the training data set, γ percent of the positive points were randomly selected as labeled positive examples and the rest of the positive examples and negative examples were viewed as unlabeled examples. We ranged γ from 20% to 80% (0.2-0.8) to create a wide range of test cases. Table 2 shows experimental results on the synthetic data sets. In the table, SvmPath, SVM-light, Bias-SVM, and One-class SVM respectively represent the methods introduced above. PU-SvmPath denotes our method. PU_toy1-20% denotes that we

[3] http://people.csail.mit.edu/jrennie/20Newsgroups/

use 20% of the positive examples in PU_toy1 as labeled positive examples and the others as unlabeled examples (including unlabeled positive examples and negative examples). Likewise for the others.

We see from the table that the proposed PU-SvmPath can significantly outperform the other methods in most of the test cases, especially with few labeled positive examples (20% and 40%). When γ (the ratio of labeled positive examples) increases to 80%, all of the methods can obtain good results.

Table 2. Average F1-scores on Synthetic Datasets (%)

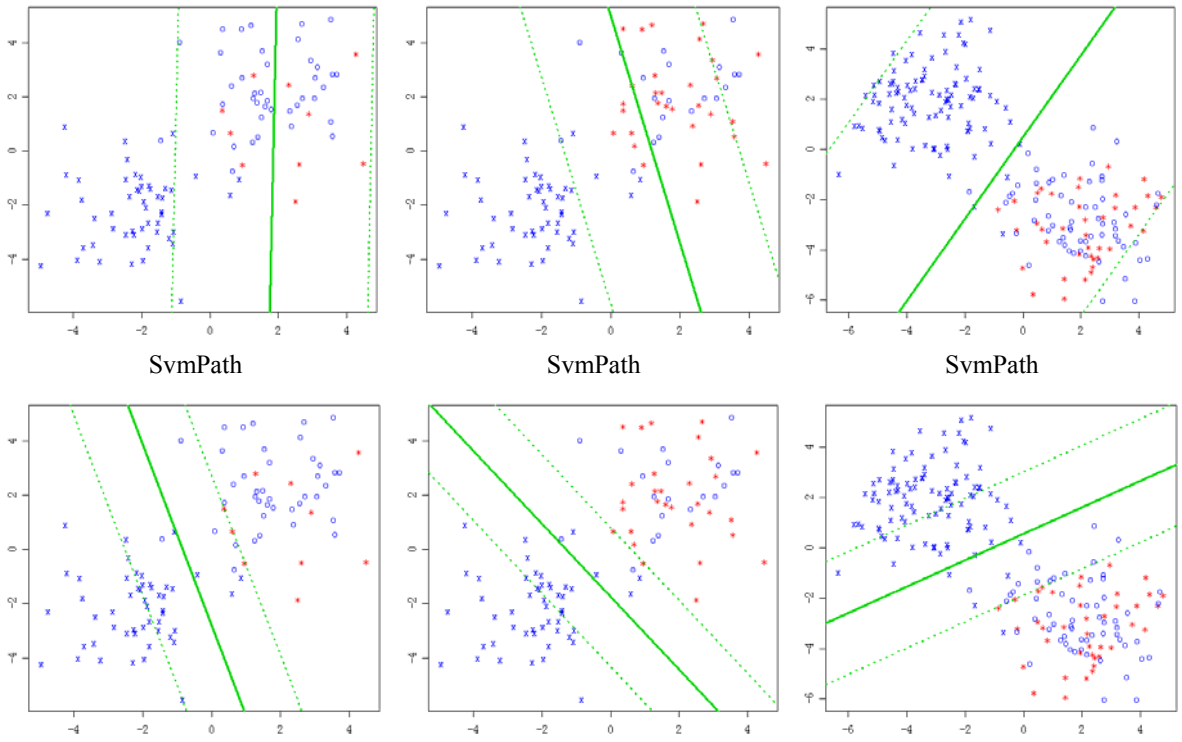
Test Case	SvmPath	PU-SvmPath	SVM-light	Bias-SVM	One-class SVM
PU_toy1-20%	35.48	**94.23**	0.00	3.85	41.27
PU_toy1-40%	82.35	**95.24**	0.00	60.27	61.11
PU_toy1-60%	85.06	**96.15**	88.89	95.83	48.48
PU_toy1-80%	96.91	92.59	96.97	96.97	48.48
PU_toy2-20%	95.00	**98.52**	0.00	0.00	49.62
PU_toy2-40%	99.00	98.52	0.00	0.00	62.07
PU_toy2-60%	92.47	**97.56**	95.29	95.83	63.95
PU_toy2-80%	99.50	97.09	99.50	99.50	63.01

The method of using SVM-light with one default cost parameter can only work well in the cases that have enough labeled positive examples and cannot come up with any results when there are only few labeled positive examples, for example, in the cases of PU_toy1-20%, PU_toy1-40%, PU_toy2-20%, and PU_toy2-40%. We also note that when there are sufficient labeled data, say γ=0.8, the SVM-light can achieve the best performance (96.97% on PU_toy1 and 99.50% on PU_toy2).

The Bias-SVM can achieve better results than the SVM-light. However, the performances of the resulting models are sensitive to the data sets. For example, Bias-SVM can obtain 60.27% (F1 score) on PU_toy1 with γ as 0.4, but cannot yield any results on the other data set PU_toy2 with

the same γ. The performances are also sensitive to the values of the cost parameters. With different values, the performances might vary largely.

One-class SVM can learn a classifier with only positive examples, however it cannot take advantage of the unlabeled data. Its performance is poorer than PU-SvmPath. It is also inferior to SvmPath, SVM-light, and Bias-SVM. When γ is small (0.2 and 0.3), it outperforms SVM-light and Bias-SVM.



SvmPath | SvmPath | SvmPath

PU-SvmPath (51 steps) (a) PU_toy1-20% | PU-SvmPath (61 steps) (b) PU_toy1-60% | PU-SvmPath (135 steps) (c) PU_toy2-40%

Figure 2. Hyper-planes generated by SvmPath and PU-SvmPath

SvmPath can fit the entire regularization path so that it can find a 'best' value for the cost parameter. However, SvmPath is proposed for classical classification, not for PU classification. From table 2, we can see that in most of the test cases, the proposed PU-SvmPath significantly outperforms SvmPath. We made detailed analysis for comparing the two algorithms. Figure 2

shows the hyper-planes learned by SvmPath and PU-SvmPath in three test cases: PU_toy1-20%, PU_toy1-60%, and PU_toy2-40%. In the figures, "*", "o", and "x" indicates labeled positive examples, unlabeled positive examples, and unlabeled negative examples respectively. The upper three figures are hyper-planes generated by SvmPath and the below three figures are those generated by PU-SvmPath.

We see that in the three test cases, PU-SvmPath can construct more accurate hyper-planes (see Figure 2(a)) and more regular hyper-planes (see Figure 2(b) and Figure 2(c)) than SvmPath.

The major problem of SvmPath in the PU classification is that it treats all the unlabeled examples as negative ones, thus results in a very unbalance classification task (with only a few positive examples while a large number of negative examples). This leads the hyper-lanes constructed by SvmPath to move toward to the positive examples (see Figure 2 (a) and (b)).

4.2.2. Results on the Biomedical Dataset

The SVM is popular in situations where the number of features exceeds the number of examples. The Bio-medical data set (DLBCL) is just in such case, where it has 4026 features while only 47 examples. Here one typically fits a linear classifier. We argue that the proposed PU-SvmPath can play an important role for these kinds of data.

In DLBCL, we have two categories with each containing half of the examples. The task is to classify an example into one of the two categories (equally positive and negative classes). With the two categories, we then have two test cases with one category as positive and the other as negative. For each test case, we split the data set into two sub sets with the same size, one for training and one for test. In the training data set, γ percent of the positive points were randomly

selected as labeled positive examples and the rest of the positive examples and negative examples were viewed as unlabeled examples. For γ, we only selected as 50% and 75% (0.5 and 0.75), because the number of the examples is limited. Table 3 shows experimental results on Biomedical dataset. Gene data size is small. The features are much more than the data size. It is a difficult classification problem.

Table 3. Average F1-scores on Biomedical Dataset (%)

γ	SvmPath	PU-SvmPath	SVM-Light	Bias-SVM	One-class SVM
0.50	43.48	**47.06**	0.00	15.38	0.00
0.75	41.67	**73.68**	28.57	58.82	0.00

The results show that PU-SvmPath can significantly outperform SvmPath (+3.58% when γ as 0.5 and +32.01% when γ as 0.75) as well as significantly outperform SVM-light. SVM-light cannot learn a good classifier using the default cost parameter. It confirms the necessity of the choice of the cost parameter. PU-SvmPath outperforms Bias-SVM as well. One-class SVM cannot result in any results on this data.

4.2.3. Results on the Text Classification Dataset

We illustrate our algorithm on another real-world data: 20newsgroup. The data set consists of eight categories with each containing 100 documents. Then the task is to classify a document into one of the eight categories. We adopt the "one class versus all others" approach, i.e., take one class as positive and the other classes as negative. Then we have eight document classification tasks. For each document, we employ tokenization, stop-words filtering, and stemming. For each classification task, we split the data set into two sub sets with the same size, one for training and one for test. In the training data set, same as that in the above experiments, γ

percent of the positive points were randomly selected as labeled positive examples and the rest of the positive documents and negative documents were viewed as unlabeled examples. We ranged γ from 10% to 90% (0.1-0.9) and create 10 test cases.

Table 4 shows the experimental results on 20newsgroup. The results show that in most of the test cases, PU-SvmPath can significantly outperform SvmPath and SVM-light as well as One-class SVM. However, we need note that PU-SvmPath is only comparable with Bias-SVM when γ is small (from 0.1 to 0.5) and is inferior to Bias-SVM when γ increases. This is because the 20newsgroups data has some noisy examples, which indicates that the γ percent labeled positive example consists of noisy examples. So far PU-SvmPath can only handle the noiseless case. The results also indicate that using the methods of two cost parameters can better describe the unbalance situation between the positive and negative examples in the training data.

Table 4. Average F1-scores on 20newsgroup (%)

γ	SvmPath	PU-SvmPath	SVM-Light	Bias-SVM	One-class SVM
0.1	13.72	11.10	3.92	11.09	14.27
0.2	29.88	21.58	4.46	21.85	21.95
0.3	40.33	35.83	3.46	36.60	24.06
0.4	39.55	45.34	7.60	50.43	24.62
0.5	56.15	55.22	13.02	60.42	24.45
0.6	57.87	60.17	24.63	71.00	24.81
0.7	67.88	69.72	43.35	80.51	24.66
0.8	68.47	78.20	65.06	87.79	25.05
0.9	73.54	85.45	76.81	93.22	24.63

4.3. Discussion

Our work is inspired from Hastie's piecewise solution path for SVMs. As [18] points out, many models share piecewise-linear relationships between coefficient path and the cost parameter C or 1/λ. Positive and Unlabeled data is a special example of SVM model. This bias model also has the piecewise-linear property. However, the difference lies in that we should keep all positive examples correct. The hyper-planes obtained by different algorithms show that the

constraint that positive examples are outside the margin does affect the final SVM solution. Applying the piecewise-linear algorithm to real data still needs more investigation. We will detect more on the different solutions obtained by PU_SvmPath and the solutions of traditional SVM model on the text classification dataset.

5. Conclusion

In this paper, we investigated the issue of fitting the entire solution path for SVM in positive and unlabeled classification. We proposed an algorithm which can determine the cost parameter automatically while training the SVM model. Experimental results on synthetic data and real-world data show that our approach can outperform the existing methods.

As future work, we tried to extend the proposed algorithm to the noisy case, where the positive examples can be noisy (e.g. mistakenly labeled). We intend to use two parameters C^+ and C^- to control the errors of positive and negative examples respectively. One of the key points is to investigate the properties of the two parameters with the Lagrange multiplier $\boldsymbol{\alpha}$.

6. References

[1] Cortes C and Vapnik V. 1995. Support-vector networks. Machine Learning, Vol. 20, 1995, pp273-297.

[2] Hastie T, Tibshirani R, and Friedman J. 2001. The elements of statistical learning; data mining, inference and prediction. Springer Verlag, New York, 2001.

[3] Hastie T, Rosset S, Tibshirani R and Zhu J. 2004. The entire regularization path for the support vector machine. Journal of Machine Learning Research, (5):1391-1415, 2004.

[4] Li X and Liu B. 2003. Learning to classify text using positive and unlabeled data. In Proc. of IJCAI'2003

[5] Joachims T. 1999. Making large scale SVM learning practical. In Book: Practical Advances in Kernel Methods - Support Vector Learning, B. Scholkopf, C.J.C. Burges, and A.J. Smola, editors. MIT Press.

[6] Liu B, Dai Y, Li X, Lee W, and Yu P. 2003. Building text classifiers using positive and unlabeled examples. In Proc. of ICDM'03.

[7] Yu H. 2003. SVMC: Single-class classification with support vector machines, In Proc. of IJCAI'2003

[8] Yu H, Zhai C, and Han J. 2003. Text classification from Positive and Unlabeled Documents. In Proc. of CIKM'2003, pp232-239

[9] Liu B, Lee W, Yu P, and Li X. 2002. Partially supervised classification of text documents. In Proc. of ICML'2002, pp387-394.

[10] Yu H, Han J., and Chang K C. 2002. PEBL: positive example based learning for Web page classification using SVM. In Proc. of ACM SIGKDD (KDD'2002), ACM Press, New York, 2002, pp239-248.

[11] Manevitz L, and Yousef M. 2001. One-class SVMs for document classification. Journal of Machine Learning Research, 2. pp139-154

[12] Morik K, Brockhausen P, and Joachims T. 1999. Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring. In Proc. of ICML'1999, pp.268-277

[13] Li Y and Shawe-Taylor J. 2003. The SVM with uneven margins and Chinese document categorization. In Proceedings of PACLIC'2003, pp216-227.

[14] Pontil M and Verri A. 1998. Properties of support vector machines. Neural Comput., 10(4):955-974,1998

[15] Gunter L and Zhu J. 2005. Computing the solution path for the regularized support vector regression. In Proc. of NIPS'05.

[16] Wang G, Yeung D Y, Lochovsky F H. 2006. Two-dimensional solution path for support vector regression. In Proc. of ICML'2006, Pittsburge, PA, U.S.A. pp993-1000

[17] Cauwenberghs G and Poggio T. 2001. Incremental and decremental support vector machine learning. In Proc. of NIPS'2001. Cambridge, MA, 2001.

[18] Rosset S and Zhu J. 2003. Piecewise linear regularized solution paths, Technical report, Stanford University.

[19] Alizadeh A A, Eisen M B, *et al*. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature 2000, 403(6769):503-11