Published in final edited form as:
J Comput Graph Stat. 2014 April 3; 23(2): 383–402. doi:10.1080/10618600.2012.761139.

Two-Dimensional Solution Surface for Weighted Support Vector Machines

Seung Jun Shin [Ph.D. Candidate],
Department of Statistics, North Carolina State University, Raleigh, NC 27695

Yichao Wu [Associate Professor], and
Department of Statistics, North Carolina State University, Raleigh, NC 27695

Hao Helen Zhang [Associate Professor]
Department of Mathematics, University of Arizona, Tucson, AZ 85718

Seung Jun Shin: sshin@ncsu.edu; Yichao Wu: wu@stat.ncsu.edu; Hao Helen Zhang: hzhang@math.arizona.edu

Abstract

The support vector machine (SVM) is a popular learning method for binary classification. Standard SVMs treat all the data points equally, but in some practical problems it is more natural to assign different weights to observations from different classes. This leads to a broader class of learning, the so-called weighted SVMs (WSVMs), and one of their important applications is to estimate class probabilities besides learning the classification boundary. There are two parameters associated with the WSVM optimization problem: one is the regularization parameter and the other is the weight parameter. In this paper we first establish that the WSVM solutions are jointly piecewise-linear with respect to both the regularization and weight parameter. We then develop a state-of-the-art algorithm that can compute the entire trajectory of the WSVM solutions for every pair of the regularization parameter and the weight parameter, at a feasible computational cost. The derived two-dimensional solution surface provides theoretical insight on the behavior of the WSVM solutions. Numerically, the algorithm can greatly facilitate the implementation of the WSVM and automate the selection process of the optimal regularization parameter. We illustrate the new algorithm on various examples.

Keywords

binary classification; probability estimation; solution surface; support vector machine; weighted support vector machine

Correspondence to: Hao Helen Zhang, hzhang@math.arizona.edu.

Online Supplement: The online supplement materials contain the computer code and the instructions on how to build and install our newly developed R package *wsvmsurf*. All the files are available via a single zipped file “Supplement.rar”. After unzipping the file, one can retrieve one folder (“wsvmsurf”) and two text files (“example.R” and “readme.txt”). The folder “wsvmsurf” contains all the source files (which are properly structured) for building the R package “wsvmsurf”. The file “readme.txt” gives detailed instructions for installing the package. For windows machines, the package can be built by using *R-tools* available at www.r-project.org. For Linux machines, the installation can be done by using the R command *R CMD build wsvmsurf*. The file “example.R” contains the code to run two examples in the paper.

1 Introduction

Frequently encountered in real applications is binary classification, in which we are given a training set $\{(\mathbf{x}_i, y_i), i = 1, \cdots, n\}$ of size n and the goal is to learn a classification rule. Here $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$ denote a p-dimensional predictor and a binary response (or class label), respectively, for the ith example. The primary goal of the binary classification is to construct a classifier which can be used for class prediction of new objects with predictors given. Among many existing methods for binary classification, the support vector machine (SVM) (Vapnik; 1996) is one of the most well known classifiers. It has gained much popularity since its introduction. It originates with the simple idea of finding an optimal hyperplane to separate two classes. The hyperplane is optimal in the sense that the geometric margin between these two classes is maximized. Later it was shown by Wahba (1999) that the SVM can be fit in the general regularization framework by solving

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} H_1(y_i f(\mathbf{x}_i)) + \frac{\lambda}{2} J(f), \quad (1)$$

where $H_1(u) = \max(1 - u, 0)$ is the hinge loss function, $J(f)$ denotes the roughness penalty of a function $f(\cdot)$ in a function space $\mathcal{F}$, and the sign of $f(\mathbf{x})$ for a given predictor $\mathbf{x}$ will be used for class prediction. Here $\lambda > 0$ is a regularization parameter which balances data fitting measured by the hinge loss and model complexity measured by the roughness penalty. Lin (2002) shows that the Hinge loss is Fisher consistent. See Liu (2007) for a more detailed discussion on Fisher consistency of different loss functions. A common choice of the penalty is $J(f) = \|f\|^2_{\mathcal{F}}$. Other choices include the l_1-norm penalty (Zhu et al.; 2003; Wang and Shen; 2007) and the SCAD penalty (Zhang et al.; 2006). In general, we set $\mathcal{F}$ to be the Reproducing Kernel Hilbert Space (RKHS, Wahba; 1990) $\mathcal{H}_K$, generated by a non-negative definite kernel $K(\cdot,\cdot)$. By the Representer Theorem (Kimeldorf and Wahba; 1971), the optimizer of (1) has a finite dimensional representation given by

$$f(\mathbf{x}) = b + \sum_{i=1}^{n} \theta_i K(\mathbf{x}, \mathbf{x}_i). \quad (2)$$

Using (2), it can be shown that the roughness penalty of f in $\mathcal{H}_K$ is $\sum_{i=1}^{n}\sum_{j=1}^{n} \theta_i \theta_j K(\mathbf{x}_i, \mathbf{x}_j)$. Then the SVM estimates $f(\mathbf{x})$ by solving

$$\min_{b,\theta_1,\cdots,\theta_n} \sum_{i=1}^{n} H_1\left(y_i \left[b + \sum_{j=1}^{n} \theta_j K(\mathbf{x}_i, \mathbf{x}_j)\right]\right) + \frac{\lambda}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \theta_i \theta_j K(\mathbf{x}_i, \mathbf{x}_j). \quad (3)$$

For the SVM (3), Hastie et al. (2004) showed that the optimizers b, θ_1, $\cdots$, θ_n change piecewise-linearly when the regularization parameter λ changes and proposed an efficient solution path algorithm. From now on, we refer to this path as a λ-path.

Note that, in the standard SVM, each observation is treated equally no matter which class it belongs to. Yet this may not be always optimal. In some situations, it is desired to assign different weights to the observations from different classes; one such example is when one type of misclassification induces a larger cost than the other type of misclassification. Motivated by this, Lin et al. (2004) considered the weighted SVM (WSVM) by solving

$$\min_{b,\theta_1,\cdots,\theta_n} \sum_{i=1}^{n} \pi_i H_1 \left(y_i \left[b + \sum_{j=1}^{n} \theta_j K(\mathbf{x}_i, \mathbf{x}_j) \right] \right) + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \theta_i \theta_j K(\mathbf{x}_i, \mathbf{x}_j), \quad (4)$$

where the weight π_i's are given by

$$\pi_i \equiv \pi(y_i) = \begin{cases} 1-\pi & \text{if } y_i = +1 \\ \pi & \text{if } y_i = -1 \end{cases}$$

with $\pi \in (0, 1)$. Each point $(\mathbf{x}_i, y_i)$ is associated with one weight parameter π_i, the value of which depends on the label y_i. According to Lin et al. (2004), the WSVM classifier provides a consistent estimate of $\text{sign}(p(\mathbf{x}) - \pi)$ for any $\mathbf{x}$, where $p(\mathbf{x})$ is the conditional class probability $p(\mathbf{x}) = P(Y = 1 | \mathbf{X} = \mathbf{x})$. Using this fact, Wang et al. (2008) proposed the weighted SVM for probability estimation. The basic idea is to divide and conquer by converting a probability estimation problem into many classification sub-problems. Each classification sub-problem is assigned with a different weight parameter π, and these sub-problems are solved separately. Then the solutions are assembled together to construct the final probability estimator. A more detailed description is as follows. Consider a sequence of $0 < \pi^{(1)} < \cdots < \pi^{(M)} < 1$. For each $m = 1, \cdots, M$, solve the (4) associated with $\pi^{(m)}$ and denote the solution by $\hat{f}_m(\cdot)$. Finally, for any $\mathbf{x}$, construct the probability estimator as $\hat{p}(\mathbf{x}) = \{\pi^{(m+)} + \pi^{(m-)}\}/2$, where $m^+ = \text{argmax}_m \hat{f}_m(\mathbf{x}) > 0$ and $m^- = \text{argmin}_m \hat{f}_m(\mathbf{x}) < 0$. Advantages of the weighted SVM include flexibility and capability of handling large dimensional data.

One main concern about the probability estimation scheme proposed by Wang et al. (2008) is its computational cost. The cost comes from two sources: there are multiple sub-problems to solve, since the weight parameter π varies in (0, 1); each sub-problem is associated with one regularization parameter λ, which needs to be adaptively tuned in the range $(0,\infty)$. To facilitate the computation, the λ-path algorithm of the standard SVM (Hastie et al.; 2004) can be extended to the WSVM for any fixed $\pi \in [0, 1]$. In addition, Wang et al. (2008) developed the π-path algorithm for any fixed $\lambda > 0$. Both the λ-path and the π-path are piecewise-linear. However it is largely unknown how the WSVM solution $f_{\lambda,\pi}$ changes when both the regularization parameter λ and the weight parameter π vary together. The main purpose of our two-dimensional solution surface is to reduce the computation and tuning burden by automatically obtaining the solutions for all possible (π, λ) with an efficient algorithm.

One main motivation of this paper is to study the behaviors of the entire WSVM solutions and characterize them by a simple representation form through their relationship to π and λ.

We use subscripts to emphasize that the WSVM solution f is a function of λ and π and sometimes omit the subscripts when they are clear from the context. Another motivation for the need of the solution surface is to automate the selection of the regularization parameter and improve the efficiency of searching process. Although Wang et al. (2008)'s conditional class probability estimator performs well as demonstrated by their numerical examples, its performance depends heavily on λ. They proposed to tune λ by using a grid search in their numerical illustrations. Yet, it is well known that such a grid search can be computationally inefficient and, in addition, its performance depends on how fine the grid is. The above discussions motivate us to develop a two-dimensional solution surface (rather than a one-dimensional path) as a continuous function of both λ and π in the analogous way that one resolved the inefficiency of the grid search for selecting the regularization parameter λ of the SVM by computing the entire λ-regularization path (Hastie et al.; 2004). From now on, we refer to the new two-dimensional solution surface of the WSVM as the *WSVM solution surface*.

In order to show the difficulties in tuning the regularization parameter for the probability estimation (Wang et al.; 2008) and motivate our new tuning method based on the WSVM solution surface, we use a univariate toy example generated from a Gaussian mixture: x_i is randomly drawn from the standard normal distribution if $y_i = 1$ and from $N(1, 1)$ otherwise, with five points from each class. The linear kernel ($K(x_i, x_j) = x_i x_j$) is employed for the WSVM and its solution is then given by $f(x) = b + \beta x$ with $\beta = \sum_{i=1}^{n} \theta_i x_i$. In order to describe the behavior of $f(x)$, we plot $\lambda\beta$ based on the obtained WSVM solution surface (or path), since $\lambda\beta$, instead of β, is piecewise-linear in λ due to our parametrization. In Figure 1, the top two panels depict the solution paths of $\lambda\beta$ for the different values of λ fixed at 0.2, 0.4, 0.6, 0.8, 1.0 (left) and the corresponding estimates of $p\hat{}(\cdot)$ as a function of x (right); the bottom two panels plot the entire two-dimensional joint solution surface (left) and the corresponding probability estimate $p\hat{}(\cdot)$ as a function of λ as well as x (right). We note that, although all the five π-paths are piecewise-linear in π they have quite different shapes for different values of λ (see (a)). Thus the corresponding probability estimates can be quite different even for the same x (see (b)), suggesting the importance of selecting an optimal λ. By using the proposed WSVM solution surface, we can completely recover the WSVM solutions on the whole ($\lambda \times \pi$)-plane (see (c)), which enables us to produce the corresponding conditional probability estimators at a given x for every λ with very little computational expense (see (d)). We will shortly demonstrate that it is computationally efficient to extract marginal paths (λ-path or π-path) once the WSVM solution surface is obtained.

In this example, we use a grid of five equally-spaced λs which is very rough. In practice, it is typically not known a *priori* how fine the grid should be or what the appropriate range of the grid is. If the data are very large or complicated, the grid one choose may not be fine enough to capture the variation of the WSVM solution and will lose efficiency for the subsequent probability estimation. The proposed WSVM solution surface provides a complete portrait for the WSVM solution corresponding to any pair of λ and π and therefore naturally overcomes this kind of practical difficulties, in addition to the gain in computational efficiency.

In this article, we show that the WSVM solution is jointly piecewise-linear in both λ and π and propose an efficient algorithm to construct the entire solution surface on the ($\lambda \times \pi$)-plane by taking advantage of the established joint piecewise-linearity. As a straightforward application, an adaptive grid for tuning the regularization parameter of the probability estimation scheme (Wang et al.; 2008) is proposed. We finally remark that the WSVM solution surface has broad applications in addition to the probability estimation.

The rest of the article is organized as follows. In Section 2, the WSVM problem is formulated in details to develop the joint solution surface of the WSVM. In Section 3 we establish the joint piecewise-linearity of the WSVM solution on ($\lambda \times \pi$)-plane. An efficient algorithm is developed to compute the WSVM solution surface by taking advantage of its piecewise-linearity in Section 4; its computational complexity is explored in Section 5. The proposed WSVM solution surface algorithm is illustrated by the *kyphosis* data in Section 6. It is then applied to the probability estimation problems in several sets of real data in Section 7. Some concluding remarks are given in Section 8.

2 Problem Setup

By introducing nonnegative slack variables ξ_i, $i = 1, \cdots, n$ and using inequality constraints, the WSVM problem (4) can be equivalently rewritten as

$$\min_{b,\theta_1,\cdots,\theta_n} \sum_{i=1}^{n} \pi_i \xi_i + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \theta_i \theta_j K(\mathbf{x}_i, \mathbf{x}_j)$$
$$\text{subject to} \quad 1 - y_i \left[b + \sum_{j=1}^{n} \theta_j K(\mathbf{x}_i, \mathbf{x}_j) \right] \le \xi_i \quad \text{and} \quad \xi_i \ge 0, \quad i = 1, 2, \cdots, n.$$

The corresponding Lagrangian primal function is constructed as

$$L_P: \sum_{i=1}^{n} \pi_i \xi_i + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \theta_i \theta_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n} \alpha_i (1 - y_i f(\mathbf{x}_i)) - \sum_{i=1}^{n} \gamma_i \xi_i, \quad (5)$$

where $\alpha_i \ge 0$ and $\gamma_i \ge 0$ are the Lagrange multipliers. To derive the corresponding dual problem, we set the partial derivatives of L_P in (5) with respect to the primal variables to zero, which gives

$$\frac{\partial}{\partial \theta_i}: \quad \sum_{j=1}^{n} \theta_j K(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\lambda} \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j), \quad (6)$$

$$\frac{\partial}{\partial b}: \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad (7)$$

$$\frac{\partial}{\partial \xi_i}: \quad \alpha_i = \pi_i - \gamma_i, \quad (8)$$

along with the Krush-Kuhn-Turker (KKT) conditions

$$\alpha_i[1 - y_i f(x_i) - \xi_i] = 0, \quad (9)$$

$$\gamma_i \xi_i = 0. \quad (10)$$

Notice from (6) that the function (2) can be rewritten as

$$f(\mathbf{x}) = b + \frac{1}{\lambda}\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i). \quad (11)$$

By combining (6–10), the dual problem of the WSVM is given by

$$\begin{aligned} &\max_{\alpha_1,\cdots,\alpha_n} \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &\text{subject to} \quad \sum_{i=1}^{n} y_i \alpha_i = 0 \text{ and } 0 \le \alpha_i \le \pi_i, \quad \forall i = 1, \cdots, n. \end{aligned} \quad (12)$$

and this can be readily solved via the quadratic programming (QP) for any given λ and π. However, the two parameters λ and π may not be known in practice. A classical approach is to repeatedly solve the QP problem (12) for different pairs of (λ, π) in order to select the desired λ and π. This is very computationally intensive for large data set because the QP itself is a numerical method whose computational complexity increases polynomially in n.

For the standard SVM (equivalent to a special case of the WSVM with the weight parameter $\pi = 0.5$), Hastie et al. (2004) showed the piecewise-linearity of α_i in λ and developed an efficient algorithm to compute the entire piecewise-linear solution path. The same idea can be extended straightforwardly to the WSVM with any fixed weight parameter or any fixed individual weights. In addition, Wang et al. (2008) showed the WSVM solutions α_i are piecewise-linear in the weight parameter π while keeping the regularization parameter λ fixed. However it is largely unknown how the WSVM solution α_i changes with respect to the two parameters jointly. In this article, we show that the WSVM solutions α_is, as a function of both λ and π, form a continuous piecewise-linear solution surface on the ($\lambda \times \pi$)-plane and propose an efficient algorithm to compute the entire solution surface.

Similar to the idea of Hastie et al. (2004), we categorize all the examples, $i = 1, \cdots, n$ into the three disjoint sets as

- $\mathcal{E} = \{i : y_i f(\mathbf{x}_i) = 1, 0 \le \alpha_i \le \pi_i\}$ (elbow),
- $\mathcal{L} = \{i : y_i f(\mathbf{x}_i) < 1, \alpha_i = \pi_i\}$ (left),
- $\mathcal{R} = \{i : y_i f(\mathbf{x}_i) > 1, \alpha_i = 0\}$ (right).

It is easy to see that the above three sets are always defined uniquely by the conditions (6)–(10). The set names come from the particular shape of the *hinge loss* function (Hastie et al.; 2004). Note that $\{a_i, \forall i \in \mathcal{E}\}$ contains most of the information on how the WSVM solution changes on the $(\lambda \times \pi)$-plane, since the rest solutions $\{a_i, i \in \mathcal{L} \cup \mathcal{R}\}$ are trivially determined by the definition of the sets.

As λ and π change, the sets may change and, as long as this happens, we call it an *event*. All the solution surfaces for a_i, $i = 1, \cdots, n$ are continuous and hence no element in $\mathcal{L}$ can move directly to $\mathcal{R}$ or vice versa. Therefore there are only three possible events to be considered as follows. The first event defines when one of a_i, $i \in \mathcal{E}$ reaches π_i and the corresponding index i exits from $\mathcal{E}$ to $\mathcal{L}$ (*event 1*). Similarly, an a_i, $i \in \mathcal{E}$ can reach the other boundary 0 and the index moves form $\mathcal{E}$ to $\mathcal{R}$ (*event 2*). The last event happens when one element i of $\mathcal{L} \cup \mathcal{R}$ satisfies $y_i f(\mathbf{x}_i) = 1$ and enters to $\mathcal{E}$ (*event 3*).

3 Joint piecewise-linearity

In this section, we study the behavior of the WSVM solutions from a theoretical point of view. One major discovery we have made is that, a_i, $i \in \mathcal{E}$, hence all the a_i, $i = 1, \cdots, n$ changes in a jointly piecewise-linear manner when λ and π vary. The following theorem describes how a_i moves as λ and π change. For simplicity, we define $a_0 = \lambda b$.

Theorem 1

(Joint Piecewise-Linearity) Suppose we have a point (λ^ℓ, π^ℓ) in the $(\lambda \times \pi)$ plane. Let $\mathcal{E}^\ell$, $\mathcal{E}^\ell$, $\mathcal{L}^\ell$, $\boldsymbol{\alpha}^\ell = (\alpha_1^\ell, \cdots, \alpha_n^\ell)^T$, and α_0^ℓ denote the associated sets and solutions obtained at (λ^ℓ, π^ℓ), respectively. Now, we consider a subset $\mathcal{S}^\ell$ (of the $(\lambda \times \pi)$-plane) which contains (λ^ℓ, π^ℓ) such that no event happens within $\mathcal{S}^\ell$. In other words, for all $(\lambda, \pi) \in \mathcal{S}^\ell$, the associated three sets, $\mathcal{E}$, $\mathcal{L}$, and $\mathcal{R}$ remain the same as $\mathcal{E}^\ell$, $\mathcal{E}^\ell$, and $\mathcal{L}^\ell$, respectively. Then the solution a_i, $i \in \{0\} \cup \mathcal{E}^\ell$, denoted by $\boldsymbol{\alpha}_{0,\mathcal{E}}$ in a vector form moves in $\mathcal{S}^\ell$ as follows:

$$\boldsymbol{\alpha}_{0,\mathcal{E}} \equiv \boldsymbol{\alpha}_{0,\mathcal{E}}(\lambda, \pi) = \boldsymbol{\alpha}_{0,\mathcal{E}}^\ell + \mathbf{G}_\ell \boldsymbol{\Delta}, \quad \forall \quad (\lambda, \pi) \in \mathcal{S}^\ell, \quad (13)$$

where $\boldsymbol{\alpha}_{0,\mathcal{E}}^\ell = \{\alpha_i^\ell : i \in \{0\} \cup \mathcal{E}^\ell\}$ and $\boldsymbol{\Delta} = (\Delta_\lambda, \Delta_\pi)^T = (\lambda - \lambda^\ell, \pi - \pi^\ell)^T$. The gradient matrix, $\mathbf{G}_\ell$ is given by

$$\mathbf{G}_\ell = \mathbf{A}_\ell^{-1} \mathbf{B}_\ell = \begin{pmatrix} 0 & \mathbf{y}_\ell^T \\ \mathbf{y}_\ell & \mathbf{K}_\ell^* \end{pmatrix}^{-1} \begin{pmatrix} 0 & |\mathcal{L}^\ell| \\ \mathbf{1} & -\mathbf{k}_\ell^* \end{pmatrix}, \quad (14)$$

where $\mathbf{K}_\ell^* = \{y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) : for\ i, j, \in \mathcal{E}^\ell\}$; $\mathbf{k}_\ell^* = \{\sum_{j \in \mathcal{L}^\ell} y_i K(\mathbf{x}_i, \mathbf{x}_j) : i \in \mathcal{E}^\ell\}^T$; $\mathbf{y}_\ell = \{y_i : i \in \mathcal{E}^\ell\}^T$; $|A|$ denotes the cardinality of a set A; and $\mathbf{1}$ is the one vector of length $|\mathcal{E}^\ell|$.

Proof—We can rewrite (11) as:

$$f(\mathbf{x}) = \left[f(\mathbf{x}) - \frac{\lambda^\ell}{\lambda} f^\ell(\mathbf{x}) \right] + \frac{\lambda^\ell}{\lambda} f^\ell(\mathbf{x})$$
$$= \frac{1}{\lambda} \left[(\alpha_0 - \alpha_0^\ell) + \sum_{j \in \mathcal{E}^\ell} (\alpha_j - \alpha_j^\ell) y_j K(\mathbf{x}, \mathbf{x}_j) + (\pi - \pi_\ell) \sum_{j \in \mathcal{L}^\ell} K(\mathbf{x}, \mathbf{x}_j) \right] + \frac{\lambda^\ell}{\lambda} f^\ell(\mathbf{x}). \quad (15)$$

Moreover, in $\mathcal{S}^\ell$ we have $y_i f(\mathbf{x}_i) = y_i f^\ell(\mathbf{x}_i) = 1$, $i \in \mathcal{E}^\ell$, which leads to

$$\Delta_\lambda = \nu_0 y_i + \sum_{j \in \mathcal{E}^\ell} \nu_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) + \Delta_\pi \sum_{j \in \mathcal{L}^\ell} y_i K(\mathbf{x}_i, \mathbf{x}_j), \quad (16)$$

where $\nu_i = \alpha_i - \alpha_i^\ell$, $\forall i \in \{0\} \cup \mathcal{E}^\ell$. In addition, $\sum_{i=1}^n y_i \alpha_i = \sum_{i \in \mathcal{E}^\ell} y_i \alpha_i + \pi |\mathcal{L}^\ell|$ by definitions of the sets and the condition (7) gives

$$\sum_{i \in \mathcal{E}^\ell} y_i \nu_i = |\mathcal{L}^\ell| \Delta_\pi. \quad (17)$$

Finally, we have ($|\mathcal{E}^\ell|+1$) linear equations from (16), (17), which can be expressed in a matrix form as $\mathbf{A}_\ell \boldsymbol{\nu} = \mathbf{B}_\ell \boldsymbol{\Delta}$, where $\boldsymbol{\nu} = \{\nu i : i \in \{0\} \cup \mathcal{E}\}^T = \boldsymbol{\alpha}_{0,\mathcal{E}} - \boldsymbol{\alpha}_{0,\mathcal{E}^\ell}$. Finally the desired result follows by assuming $\mathbf{A}_\ell$ to be invertible.

We remark that $\mathbf{A}_\ell$ is rarely singular in practice, and related discussions can be found in Hastie et al. (2004). It is worthwhile to point out that the joint piecewise-linearity of the solution guarantees the marginal piecewise-linearity as presented in Corollary 2, but not vice versa. Therefore, *Theorem 1* implies the piecewise-linearity of the marginal solution paths as a function of either λ or π, which were separately explored by Hastie et al. (2004) and Wang et al. (2008).

Corollary 2

(Marginal Piecewise-Linearity) For a given π_0, the solution, $\boldsymbol{\alpha}_{0,\mathcal{E}}$ moves linearly in $\lambda \in \{\lambda : (\lambda, \pi_0) \in \mathcal{S}^\ell\}$ as follows.

$$\boldsymbol{\alpha}_{0,\mathcal{E}} = \boldsymbol{\alpha}^\ell_{0,\mathcal{E}} + \mathbf{g}_1^\ell \Delta_\lambda.$$

Similarly, $\boldsymbol{\alpha}_{0,\mathcal{E}}$ changes in $\pi \in \{\pi : (\lambda_0, \pi) \in \mathcal{S}^\ell\}$ for a given λ_0 as follows.

$$\boldsymbol{\alpha}_{0,\mathcal{E}} = \boldsymbol{\alpha}^\ell_{0,\mathcal{E}} + \mathbf{g}_2^\ell \Delta_\pi,$$

where $\mathbf{g}_1^\ell$ and $\mathbf{g}_2^\ell$ denote the first and second column vectors of $\mathbf{G}_\ell$ in (14), respectively.

The classification function $f(\mathbf{x})$ can be conveniently updated by plugging (13) into (15), which gives

$$f(\mathbf{x})=\frac{\lambda^{\ell}}{\lambda}\left[f^{\ell}(\mathbf{x})-h_1^{\ell}(\mathbf{x})\right]+h_1^{\ell}(\mathbf{x})+\frac{\pi-\pi^{\ell}}{\lambda}h_2^{\ell}(\mathbf{x}), \quad (18)$$

where

$$h_1^{\ell}(\mathbf{x})=g_{01}+\sum_{i\in\mathcal{E}^{\ell}} g_{i1}y_iK(\mathbf{x},\mathbf{x}_i),$$
$$h_2^{\ell}(\mathbf{x})=g_{02}+\sum_{i\in\mathcal{E}^{\ell}} g_{i2}y_iK(\mathbf{x},\mathbf{x}_i)+\sum_{i\in\mathcal{L}^{\ell}} K(\mathbf{x},\mathbf{x}_i),$$

and (g_{i1}, g_{i2}) denotes the row of $\mathbf{G}_\ell$ where $i \in \{0\} \cup \mathcal{E}^{\ell}$. We observe from (18) that $f(\mathbf{x})$ is not jointly piecewise-linear in (λ, π) while it is marginally piecewise-linear in λ^{-1} and π, respectively.

4 Solution Surface Algorithm

In this section, we propose an efficient algorithm to compute the entire solution surface of the WSVM on the $(\lambda \times \pi)$-plane by using the joint piecewise-linearity established in *Theorem* 1.

4.1 Initialization

Denote index sets $I_+ = \{i : y_i = 1\}$ and $I_- = \{i : y_i = -1\}$. We initialize the algorithm at $\pi_0 = |I_+|/n$. Notice that the π_0 satisfies the so-called *balanced condition* that requires $\Sigma_{i\in I_+}\ \pi_i(\pi_0) = \Sigma_{i\in I_-}\ \pi_i(\pi_0)$. With $\pi = \pi_0$ it is easy to verify that, for a sufficiently large λ, $\alpha_i = \pi_0$ if $i \in I_+$ and $1 - \pi_0$ otherwise. Following the idea of Hastie et al. (2004), the initial values of λ and α_0 denoted by λ_0 and α_0^0, respectively are given by

$$\lambda_0=\frac{1}{2}\left[(1-\pi_0)\sum_{i\in I_+}(K_i^+-K_i^-)+\pi_0\sum_{i\in I_-}(K_i^+-K_i^-)\right] \text{ and}$$
$$\alpha_0^0=-\frac{1}{2}\left[(1-\pi_0)\sum_{i\in I_+}(K_i^+-K_i^-)-\pi_0\sum_{i\in I_-}(K_i^+-K_i^-)\right],$$

where $K_i^+=K(\mathbf{x}_i,\mathbf{x}_{i+}), K_i^-=K(\mathbf{x}_i,\mathbf{x}_{i-})$. The indices i_+ and i_- are defined as

$$i_+=\underset{i\in I_+}{\operatorname{argmax}}\{(1-\pi_0)\sum_{j\in I_+}K(\mathbf{x}_i,\mathbf{x}_j)-\pi_0\sum_{j\in I_-}K(\mathbf{x}_i,\mathbf{x}_j)\} \text{ and}$$
$$i_-=\underset{i\in I_-}{\operatorname{argmin}}\{(1-\pi_0)\sum_{j\in I_+}K(\mathbf{x}_i,\mathbf{x}_j)-\pi_0\sum_{j\in I_-}K(\mathbf{x}_i,\mathbf{x}_j)\}.$$

It is possible to initialize the algorithm at any π between 0 and 1 rather than π_0, however, we empirically observe that the initialized λ is the largest when $\pi = \pi_0$. Notice that the corresponding solution is trivial as $\alpha_i = \pi_i$ for all $i = 1, \cdots, n$ for any λ larger than λ_0. Therefore, the proposed algorithm focuses only on the non-trivial solutions obtained at $\varrho=$

$\{(\lambda, \pi) : 0 \le \lambda \le \lambda_0, 0 \le \pi \le 1\}$. Finally, the three sets initialized at (λ_0, π_0) denoted by $\mathcal{E}_0$, $\mathcal{L}_0$ and $\mathcal{R}_0$, respectively are given by

$$\mathscr{E}^0=\{i_+, i_-\}, \quad \mathscr{L}^0=\{1, \cdots, n\}\backslash\mathscr{E}^0, \quad \text{and} \quad \mathscr{R}^0=\varphi,$$

where ϕ denotes the empty set.

4.2 Update

Recall that, for any point (λ^ℓ, π^ℓ), no event occurs if $(\lambda, \pi) \in \mathcal{S}^\ell$ and the WSVM solution can be updated by applying Theorem 1 for any $(\lambda, \pi) \in \mathcal{S}^\ell$. Therefore, it is essential to know how to define $\mathcal{S}^\ell$ as large as possible for any (λ^ℓ, π^ℓ). We shall demonstrate next that the set $\mathcal{S}^\ell$ can be explicitly determined by some linear constraints.

Note that *event 1* happens when α_i reaches π_i for some $i \in \mathcal{E}^\ell$. Based on (13), we have the following inequality constraints to prevent *event 1* from happening:

$$g_{i1}\lambda+(g_{i2}+1)\pi \le t_i^\ell+1, \quad i \in \mathscr{E}_+^\ell \quad (9)$$

$$g_{i1}\lambda+(g_{i2}-1)\pi \le t_i^\ell, \quad i \in \mathscr{E}_-^\ell, \quad (20)$$

where $t_i^\ell=g_{i1}\lambda^\ell+g_{i2}\pi^\ell-\alpha_i^\ell, \mathscr{E}_+^\ell=\mathscr{E}^\ell \cap I_+$ and $\mathscr{E}_-^\ell=\mathscr{E}^\ell \cap I_-$. In a similar way, we have the following inequalities to prevent *event 2* :

$$g_{i1}\lambda+g_{i2}\pi \ge t_i^\ell, \quad i \in \mathscr{E}^\ell. \quad (21)$$

In order to prevent *event 3*, we have $y_i f(\mathbf{x}_i) < 1, \forall_i \in \mathcal{E}^\ell$ and $y_i f(\mathbf{x}_i) > 1, \forall_i \in \mathcal{L}^\ell$. Therefore, by noting (15), we have

$$(h_1^\ell(\mathbf{x}_i)-1)\lambda+h_2^\ell(\mathbf{x}_i)y_i\pi \le s_i^\ell, \quad i \in \mathscr{R}^\ell \quad (22)$$

$$(h_1^\ell(\mathbf{x}_i)-1)\lambda+h_2^\ell(\mathbf{x}_i)y_i\pi \ge s_i^\ell, \quad i \in \mathscr{L}^\ell, \quad (23)$$

where $s_i^\ell=y_i(h_1^\ell(\mathbf{x}_i)-f^\ell(\mathbf{x}_i))\lambda^\ell+h_2^\ell(\mathbf{x}_i)y_i\pi^\ell$. We remark that the equalities do not need to be strict since an *event* is instant transition. Recall that it is enough to consider the solution on $\mathcal{Q}$ and hence the additional constraints $0 \le \lambda \le \lambda_0$ and $0 \le \pi \le 1$ should be considered as well by default. In summary $\mathcal{S}^\ell$ can be defined by a subregion on the $(\lambda \times \pi)$-plane that satisfies all the constraints (19)–(23). Figure 2 illustrates a $\mathcal{S}^\ell$ generated from the initial point (λ_0, π_0) for the toy example in Section 1. We remark that $\mathcal{S}^\ell$ forms a *polygon* which can be uniquely expressed by its vertices, since the constraints are all linear.

We describe next how to determine the vertices of $\mathcal{S}^\ell$ in an efficient manner. First, compute all the pairwise intersection points of the boundaries of (19)–(23) then we have $\frac{n_C(n_C-1)}{2}$

intersection points, where n_c denotes the number of constraints (19)–(23), $n + |\mathcal{E}^\ell|$. The left penal of Figure 2 shows all the intersections of the boundaries of the obtained constraints. Then, we can define the vertices by identifying the intersections that satisfy all the constraints (19)–(23) as illustrated in (b) of Figure 2. We denote the vertices of the $\mathcal{S}^\ell$ as $\{v_1^\ell, \cdots, v_{n_v}^\ell\}$ where $v_r^\ell = (\lambda_r, \pi_r)$, $r = 1, \ldots, n_v$,.

There are a couple of issues we need to clarify here. First, based on our limited experience, we observe that n_v is small and typically does not exceed eight. Hence it is not efficient to compute all the $\frac{n_c(n_c-1)}{2}$ intersections due to the involved computational intensity. Note also that some constraints are dominated by others on $\mathcal{Q}$ and are thus automatically satisfied if other constraints hold. Consequently, we can save a huge amount of computational time by excluding those constraints which are dominated by others, especially when n is large. We also need to *order* the vertices for set-updates which are discussed next. Here the *ordering* means that a vertex v_r^ℓ, $r = 1, \cdots, n_v$ is adjacent to v_{r-1}^ℓ and v_{r+1}^ℓ, where we set $v_0^\ell = v_{n_v}^\ell$, $v_{n_v+1}^\ell = v_1^\ell$ (see (b) in Figure 2). The updating of $\alpha_1, \cdots, \alpha_n$ as well as α_0 at vertices of $\mathcal{S}^\ell$ provided is straightforward by *Theorem 1*.

Figure 3 illustrates how to extend the polygons on $\mathcal{Q}$ from current $\mathcal{S}^\ell$ in Figure 2-(b). Notice that the sides of $\mathcal{S}^\ell$ are determined by some boundaries of the constraints (19) – (23) and represent corresponding events. Each middle point of two adjacent vertices can be used as a new starting point, $(\lambda^{\ell+1}, \pi^{\ell+1})$ in *Theorem 1* to compute a new polygon $\mathcal{S}^{\ell+1}$. Computing middle points and corresponding solutions denoted by $m_r = (\lambda_r^{\ell+1}, \pi_r^{\ell+1})$ and $\overline{\boldsymbol{\alpha}}_r^{\ell+1} = \{(\alpha_0, \alpha_1, \cdots, \alpha_n) \text{ obtained at } m_r = (\lambda_r^{\ell+1}, \pi_r^{\ell+1})\}^T$, respectively where $r = 1, \cdots, n_v$ is trivial due to the piecewise-linearity of the solutions. The bar sign in $\overline{\boldsymbol{\alpha}}_r^{\ell+1}$ is used to emphasize the quantities are obtained at middle points, not vertices. At each middle point, the corresponding three sets denoted by $\mathcal{E}_r^{\ell+1}, \mathcal{R}_r^{\ell+1}$, and $\mathcal{L}_r^{\ell+1}$ respectively can be updated based on which line the middle point m_r is on. For example as shown in (a) of Figure 3, the three sets $\mathcal{E}_1^{\ell+1}, \mathcal{R}_1^{\ell+1}$, and $\mathcal{L}_1^{\ell+1}$ obtained at m_1 are updated as $\mathcal{E}_1^{\ell+1} = \mathcal{E}^\ell \backslash \{4\}, \mathcal{R}_1^{\ell+1} = \mathcal{R}^\ell \cup \{4\}$, and $\mathcal{L}_1^{\ell+1} = \mathcal{L}^\ell$. This is because $m_1 = (\lambda_1^{\ell+1}, \pi_1^{\ell+1})$ lies on the boundary which represents an *event* that the element 4 moves to the right set from the elbow set (see (a) in Figure 2). Now we have all the information required to generate a new polygon, $\mathcal{S}_1^{\ell+1}$, from the middle point, m_1 and $\mathcal{S}_1^{\ell+1}$ can be accordingly computed by treating m_1 as a new updating point (This is the reason why we use a superscript $\ell + 1$ to denote middle points). Figure 3 shows all the four newly-created polygons, $\mathcal{S}_r^1$, $r = 1, \cdots, 4$ respectively from the four middle points $m_1, \cdots, m_4$ of the sides of $\mathcal{S}^0$ in Figure 2-(b). Notice that the right vertical line represents $\lambda = \lambda_0$ and we do not have any interest beyond it. Finally, the proposed algorithm can be continued to extend the polygons being searched on $\mathcal{Q}$ and is terminated when the complete solution surface is recovered on $\mathcal{Q}$.

4.3 Resolving Empty Elbow

Note that either *event 1* and or *event 2* leads to the possibility that $\mathcal{E}$ is empty, named *empty elbow*. This may cause a problem in the update described in Section 4.2, because *Theorem 1* cannot be applied when $\mathcal{E}$ is empty.

We suppose the *empty elbow* occurs at (λ^o, π^o) and use a superscript '*o*' to denote any quantity defined at (λ^o, π^o). In order to resolve the *empty elbow*, we first notice that the objective function (4) is differentiable with respect to b and a_i, $i = 1, \cdots, n$ whenever the the *elbow* set is empty since in this case there is no example satisfying $y_i f(\mathbf{x}_i) = 1$. Taking derivative of (4) with respect to b and a_i, we get two conditions to be satisfied under the *empty elbow* :i) $\pi^o = |\mathcal{L}_+^o|/|\mathcal{L}^o|$, where $\mathcal{L}_+^o = \mathcal{L}^o \cap I_+$ and ii) a_i are unique while a_0 is not. Moreover, a_0 can be any value in the following interval,

$$[a_L, a_U] \triangleq \left[\max_{i \in \mathcal{L}_-^o \cup \mathcal{R}_+^o} m_i^o, \quad \min_{i \in \mathcal{L}_+^o \cup \mathcal{R}_-^o} m_i^o \right] \quad (24)$$

where $m_i^o = y_i \lambda^o - \sum_{j=1}^n \alpha_j^o y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and $\mathcal{L}_-^o, \mathcal{R}_+^o$, and $\mathcal{R}_-^o$ are similarly defined as $\mathcal{L}_+^o$. Recall that a_0 is continuous and hence the *empty elbow* can be resolved only by a_0 touching one of the two boundaries a_L and a_U. Notice that the α_0^o is regarded as a starting point where the *empty elbow* begins and hence it must be one of a_L and a_U. Without loss of generality, we suppose $\alpha_0^o = a_L$ then the *empty elbow* should be resolved when it becomes the other boundary a_U. Let $i_L^o = \text{argmax}_{i \in \mathcal{L}_-^o \cup \mathcal{R}_+^o} m_i^o$ and $i_U^o = \text{argmin}_{i \in \mathcal{L}_+^o \cup \mathcal{R}_-^o} m_i^o$, then the $\mathcal{E}$ should be updated to $\{i_U^o\}$. The $\mathcal{L}$ and $\mathcal{R}$ are accordingly updated as well since the updated index i_U^o enters to the $\mathcal{E}$ from one of the two sets. In case of $\alpha_0^o = a_U$, we update in a similar way which leads to $\mathcal{E} = \{i_L^o\}$.

4.4 Pseudo Algorithm

Combining the previous several subsections, we now summarize our WSVM solution surface algorithm at Algorithm 1. We denote $\boldsymbol{a} = (a_0, a_1, \cdots, a_n)^T$ for simplicity. We note that the proposed algorithm computes the complete WSVM solution $\boldsymbol{a}$s for any $(\lambda, \pi) \in \mathcal{Q}$ without involving any numerical optimization. We have implemented the algorithm in R language and the wsvmsurf package is available from the authors upon request (and will be available on CRAN soon).

Table 1

Empirical computing time based on 100 independent repetitions: the machine we used equips Intel Core (TM) i3 550 @ 3.20GHZ CPU with 4GB memory.

n	Gaussian			Linear		
	time(s)	piece	time/piece	time(s)	piece	time/piece
10	0.53 (0.01)	164.3 (3.58)	0.00325	0.44 (0.01)	140.98 (2.70)	0.00311
30	5.78 (0.08)	2056.6 (27.16)	0.00281	4.36 (0.06)	1648.05 (21.24)	0.00265
50	19.08 (0.22)	6036.8 (55.29)	0.00316	14.38 (0.16)	4870.05 (52.40)	0.00295
100	108.00 (1.07)	25653.0 (193.50)	0.00421	80.72 (0.67)	20087.13 (148.05)	0.00402

5 Computational Complexity

The essential part of the proposed algorithm involves several steps. First, we solve the linear equation system (14) of size $1 + |\mathcal{E}^\ell|$, which involves $O((1 + |\mathcal{E}^\ell|)^3)$ computations. We empirically observe that $|\mathcal{E}^\ell|$, the number of support vector depends on n but is usually much less than n during the algorithm. Next, in order to determine the set $\mathcal{S}^\ell$, we compute constraints (19) – (23) and $h_1(\mathbf{x}_i)$ and $h_2(\mathbf{x}_i)$, which respectively requires $O(n|\mathcal{E}^\ell|)$ and $O(n^2)$ computations. Now we have $n_c = n + |\mathcal{E}^\ell|$ constraints and need to find their intersection points which satisfy all the constraints (19)–(23) simultaneously. This is the most computationally expensive step since it requires to evaluate $n_c(n_c - 1)/2$ intersection points and check n_c constraints for each point, which takes at least $O(n_c^3)$ computations in total.

Our limited numerical experience suggests that the final vertices of $\mathcal{S}^\ell$ is of size less than or equal to eight, which implies that most of the constraints are dominated by others. This leads us to suggest adding a refining step which removes those constraints dominated by others in the algorithm as follows. First, since all the constraints have a common linear form of $a\lambda + b\pi \geq c$ for different values of a, b and c for different constraints, we can classify the constraints into two types depending on the sign of b, positive or negative, assuming $b \neq 0$ for simplicity. See Figure 4-(a) for an illustration. If $b < 0$ for a constraint then the region represented by the constraint is below its boundary (dashed lines), otherwise it is above the boundary (dotted lines). For each type of constraints with either $b > 0$ or $b < 0$, we compute values of πs at $\lambda = 0$ and λ_0 (vertical lines), then we can exclude most of, but not all dominated boundaries by sorting the π values obtained. The refining step requires at most $O(n_c^{4/3})$ computations since it is based on the sorting algorithm, *Shellsort variant*. After the refining step, we have only $\tilde{n}_c$ constraints to consider for computing the $\mathcal{S}^\ell$ and $\tilde{n}_c$ is much smaller than n_c.

In order to demonstrate the benefit of the refining step, we consider the situations where the data sets from univariate Gaussian mixture for different sample size, n. Data are generated as $x_i \sim N(0, 1)$ if $i \in I_+$ and $x_i \sim N(1, 1)$ otherwise with $|I_+| = |I_-| = n/2$. Employing Gaussian kernel $K(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|_2^2/\sigma^2)$ with $\sigma = 1$, Figure 4-(b) illustrates how effective the refining step is. We observe that only about 20% of the constraints are used to compute $\mathcal{S}^\ell$ for relatively large n, suggesting a dramatic computational saving to evaluate $\mathcal{S}^\ell$ (only about $0.2^3 = .8\%$ computations is required, compared to the case when the refining step is not employed). The right panel (b) shows n_c^3 (black dashed-line) versus $\tilde{n}_c^3$ (red solid-line) for different n; the computational saving is substantial when n is large.

In order to obtain the entire solution surface we need to iterate the aforementioned updating steps. It is quite challenging to rigorously quantify the number of required iterations as a function of n, since it depends on not only the data but also the kernel function employed (and hence its parameter). We instead evaluate empirical computing time for different sample size n based on 100 independent repetitions (Table 1). The data are simulated from a bivariate Gaussian mixture with the same number of observations from each class (i.e., $|I_+| = |I_-| = n/2$). In particular, if $i \in I_+$ then $\mathbf{x}_i$ is from $N((0, 0)^T, \mathbf{I}_2)$ and otherwise $\mathbf{x}_i \sim N_2(2, 2)^T$, $\mathbf{I}_2$ where $\mathbf{I}_2$ is the two-dimensional identity matrix. In Table 1, we report the average

computing time, average number of pieces, and the computing time per piece for both the Gaussian kernel and the linear kernel. The kernel parameter σ of the Gaussian kernel is assumed to be fixed and set to be the median of pairwise distances between the two classes (Jaakkola et al.; 1999). Our limited numerical experience suggests that the computing time per a piece ($\mathcal{S}^t$) does not depend severely on n while the entire time increases proportional to n^2 since the number of polygons produced does. In Table 1, the numbers in parentheses are the corresponding standard errors.

Finally, one may desire to extract the marginal path, either a λ-path or a π-path, from the WSVM solution surface. This can be easily done after obtaining all the polygons and the corresponding WSVM solutions. It is straightforward to interpolate the WSVM solution for any pair (λ, π) using the WSVM solutions corresponding to vertices of all those polygons produced from the WSVM solution surface. Let 'd' be the total number of vertices. As aforementioned, d is proportional to n^2. Then it requires only $O(d)$ computations to obtain the marginal solution paths, since the vertices of each polygon are properly ordered already.

6 Illustration

In this section, we illustrate how the proposed WSVM solution surface algorithm works by using the kyphosis data (Chambers and Hastie; 1992) available in R package rpart. The data contain the status (absence or presence of a kyphosis) of $n = 81$ children who had a corrective spinal surgery. The class variable y_i is coded −1 for absence and +1 for presence of a kyphosis after the operation. Three predictors are recorded for each child (i.e., $p = 3$): age of a patient (in month), the number of vertebrae involved, and the number of the first vertebra operated on. The Gaussian kernel is used with kernel parameter σ set to be 0.01. The entire solution surface consists of 7,502 polygons (or pieces) and the total computing time is 34.90 seconds. In Figure 5, the panel (a) plots all the vertices of the polygons produced, (denoted by red dots), and the panel (b) depicts the entire surface of α_{18}. There are 81 α_i, $i = 1, \cdots, 81$-surfaces totally, one corresponding to each of the ith observation. For the purpose of illustration, we only show one of them for $i = 18$. The solution surfaces for the rest 80 α_is can be visualized in a similar way. Once we have the entire two-dimensional solution surfaces, the marginal solution paths can be readily obtained for a fixed value of either λ or π. The two panels (c) and (d) represent the marginal regularization paths with a fixed $\pi = 0.2$ and the marginal paths with a fixed $\lambda = 0.5$, extracted from the WSVM solution surface, respectively. Notice that there are 81 observations in the data and hence each piecewise-linear path in (c) and (d) represents a marginal solution path of a α_i where $i = 1, \cdots, 81$.

7 Applications to Probability Estimation

In this section, we revisit the motivating application of the proposed joint solution surface algorithm to the probability estimation (Wang et al.; 2008) introduced in Section 1. In Wang et al. (2008), the regularization parameter λ is selected by a grid search over the interval $[10^{-3}, 10^3]$ with ten equally-spaced points in each sub-interval $(10^j, 10^{j+1}]$ where $j = -3, \cdots, 2$. In practice, it can be difficult to choose an appropriate range for parameter search or to determine the right level of grid coarseness. A general rule is: if the WSVM solution varies

rapidly for different λ, we need a fine gird; otherwise, a coarse grid will be more appropriate in terms of computational efficiency. However, without a *priori* information available, a subjective choice of the range could be either too wide or too narrow, and the grid could be too fine or too rough. The proposed WSVM solution surface algorithm provides a natural and informative guidance to tackle these issues. We propose to use all the distinct λs obtained from the WSVM solution surface algorithm as the grid, namely the set of all λs corresponding to all vertices of those polygons obtained by the WSVM solution surface. We like to point out that the proposed grid of λ is adaptive to the solution variation, in the sense that we have a finer grid if the solution surface is complicated and a rough grid otherwise.

We compare these two grid search methods using two microarray datasets available at LIBSVM Data website (http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html). For both of the datasets, the sample sizes are much smaller then the number of predictors. The Gaussian kernel is employed to train the WSVM and the associated kernel parameter σ is set to be the median distances between the two classes. In order to evaluate performance of the probability estimator using the two different grids, the fixed grid employed in Wang et al. (2008) and the proposed adaptive grid, the *cross entropy error* (CRE) is used. The CRE over a set of data $\{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, n\}$ is defined as follows.

$$CRE = -\frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}(1+y_i)\log\hat{p}(\mathbf{x}_i) + \frac{1}{2}(1+y_i)\log\{1-\hat{p}(\mathbf{x}_i)\}\right]. \quad (25)$$

Table 2 contains summarized information of the well-known microarray data sets used, Duke cancer and Colon cancer data. One merit of the proposed probability estimator is that it is not only distribution-free method but also free from the dimension of predictors *p*, since it exploits the kernel trick. For tuning λ, we employ the cross-validation (leave-one-out and 10-fold) instead of using independent tuning set since the data sets have relatively small sample sizes. Table 3 shows the test CREs for the microarray examples. We point out that clear improvements of the proposed adaptive grid are observed for the microarray examples compared to the fixed grid. This is because the joint solution surfaces of the microarray examples are complicated and the fixed grid fails to reflect this complicated pattern of the solutions.

8 Concluding Remarks

In this article, we first establish that the WSVM solution is jointly piecewise-linear in λ and π and then develop an efficient algorithm to compute the entire WSVM solution surface on the ($\lambda \times \pi$)-plane by taking advantage of the joint piecewise-linearity. To demonstrate its practical value, we propose an adaptive tuning scheme for the probability estimation method based on the WSVM solution surface. Illustrations using real data sets show that the proposed adaptive tuning grid from the WSVM solution surface improves the performance of the probability estimator especially when the solution surfaces are complicated. We remark that the proposed solution surface provides the entire information of the WSVM solution for arbitrary pair of λ and π and therefore potentially contains wide applicability in various statistical applications.

The piecewise-linearity of the marginal λ-path or regularization paths has been well-studied in the literature. See Rosset and Zhu (2007) for a unified analysis under various models, where a general characterization of the loss and penalty functions to give piecewise-linear coefficient paths is derived. We would like to point out that it is possible to extend the notion of joint piecewise-linearity to more general problems beyond the weighted SVM. For example, we can show that the kernel quantile regression solution also enjoys the joint piecewise-linear property in terms of its regularization parameter and the quantile parameter. However, it is not yet clear what are general sufficient conditions for the joint piecewise-linearity to hold. We find that the sufficient conditions for the marginal (one-dimensional) piecewise-linearity, given in Rosset and Zhu (2007), generally do not imply the joint (two-dimensional) piecewise-linearity. For example, support vector regression (Vapnik; 1996) has piecewise-linear marginal paths, with respect to the regularization parameter or the intensity parameter, but its solution does not have the joint piecewise-linearity property. Based on our experience, the two-dimensional solution surface is much more complex than one-dimensional marginal paths, mainly due to the intricate roles played by two parameters and their relationship. This is an important topic and worth further investigation.

Acknowledgments

The authors would like to thank Professor Richard A. Levine, the associate editor, and two reviewers for their constructive comments and suggestions that led to significant improvement of the article. This work is partially supported by NIH/NCI grants R01 CA-149569 (Shin and Wu), R01 CA-085848 (Zhang), and NSF Grant DMS-0645293 (Zhang). The content is solely the responsibility of the authors and does not necessarily represent the official views of NIH or NCI.

References

Alon U, Barkai N, Notterman D, Gish K, Ybarra S, Mack D, Levine A. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Cell Biology. 1999; 96:6745–6750.

Chambers, J.; Hastie, T. Statistical Models in S. Wadsworth and Brooks/Cole; Pacific Grove, CA: 1992.

Hastie T, Rosset R, Tibshirani R, Zhou J. The entrie regularization path for the supprot vector machine. Journal of Machine Learning Reasearch. 2004; 5:1931–1415.

Jaakkola, T.; Diekhans, M.; Haussler, D. Using the fisher kernel method to detect remote protein homologies. Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology; Heidelberg, Germany: AAAI; 1999. p. 149-158.

Kimeldorf G, Wahba G. Some results on tchebycheffian spline functions. Journal of Mathematical Analysis and Applications. 1971; 33:82–95.

Lin Y. Support vector machines and the bayes rule in classification. Data Mining and Knowledge Discovery. 2002; 6:259–275.

Lin Y, Lee Y, Wahba G. Support vector machines for classification in nonstandard situations. Machine Learning. 2004; 46:191–202.

Liu, Y. Fisher consistency of multicategory support vector machines. Proceedings of Eleventh International Conference on Artificial Intelligence and Statistics; Heidelberg, Germany: AAAI; 2007. p. 289-296.

Rosset S, Zhu J. Piecewise linear regularized solution paths. Annals of Statistics. 2007; 35 (3):1012–1030.

Vapnik, V. The Natual of Statistical Learning. Springer-Verlag; 1996.

Wahba, G. Spline models for observational data; CBMS-NSF Regional Conference Series in Applied Mathematics; SIAM; 1990.

Wahba, G. Support vector machines, reproducing kernel hilbert spaces, and randomized gacv. In: Schölkopf, B.; Burges, CJC.; Smola, AJ., editors. Advances in Kernel Methods: Support Vector Learning. MIT Press; 1999. p. 125-143.

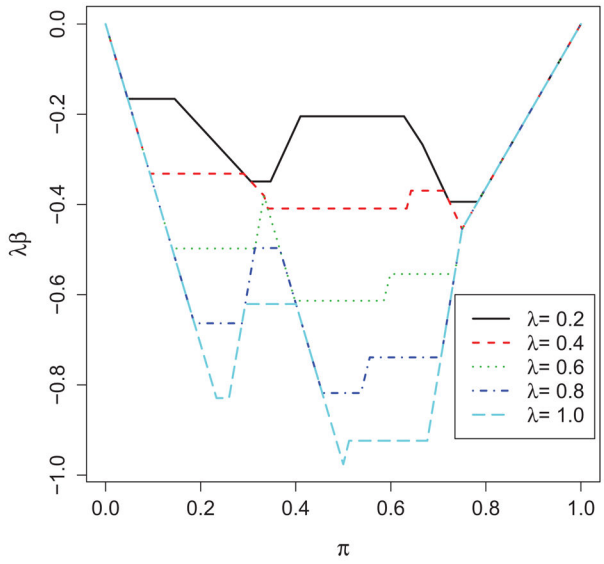
Wang J, Shen X, Liu Y. Probability estimation for large-margin classifier. Biometrika. 2008; 95 :149–167.

Wang L, Shen X. On l1-norm multi-class support vector machines: methodology and theory. Journal of the American Statistical Association. 2007; 102:595–602.

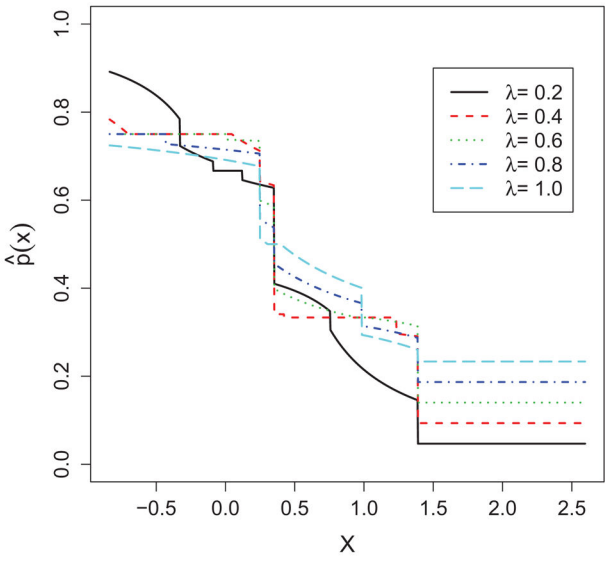
West M, Blanchette C, Dressman H, Huang E, Ishida S, Spang R, Zuzan H Jr, JAO, Marks J, Nevins J. Predicting the clinical status of human breast cancer by using gene expression profiles. Proceedings of the National Academy of Sciences. 2001; 98:11462–11467.

Zhang HH, Ahn J, Lin X, Park C. Gene selection using support vector machines with nonconvex penalty. Bioinformatics. 2006; 22(1):88–95. [PubMed: 16249260]

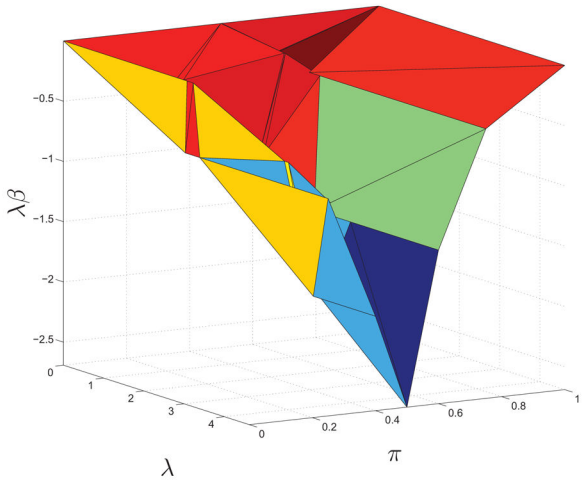
Zhu J, Rosset S, Hastie T, Tibshirani R. 1-norm support vector machines. Advances in Neural Information Processing Systems. 2003; 16



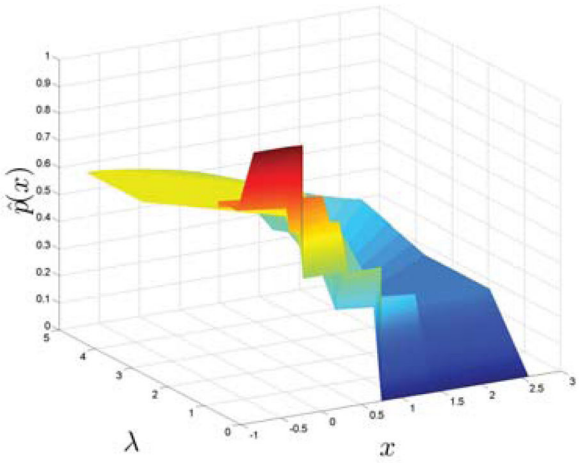
(a) π-paths of $\lambda\beta$ for different λs.



(b) $\hat{p}(x)$ for different λs.



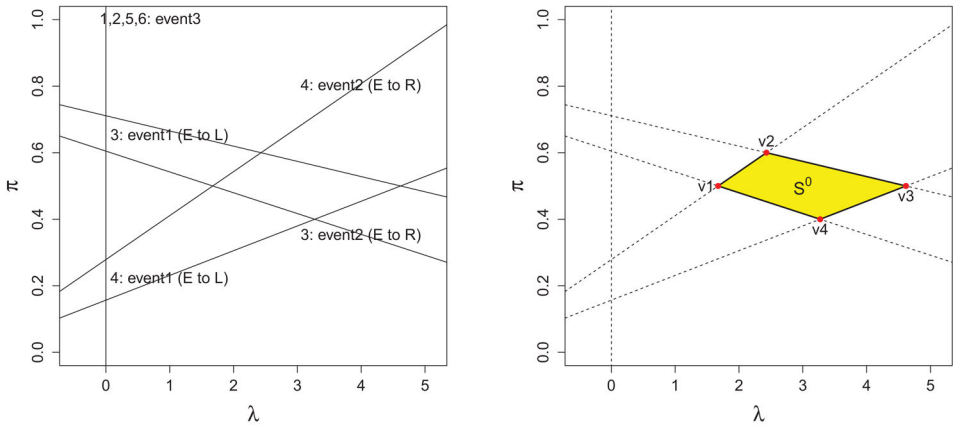
(c) WSVM solution surface of $\lambda\beta$.



(d) Surface of $\hat{p}(x)$ on $(x \times \lambda)$-plane.

Figure 1.

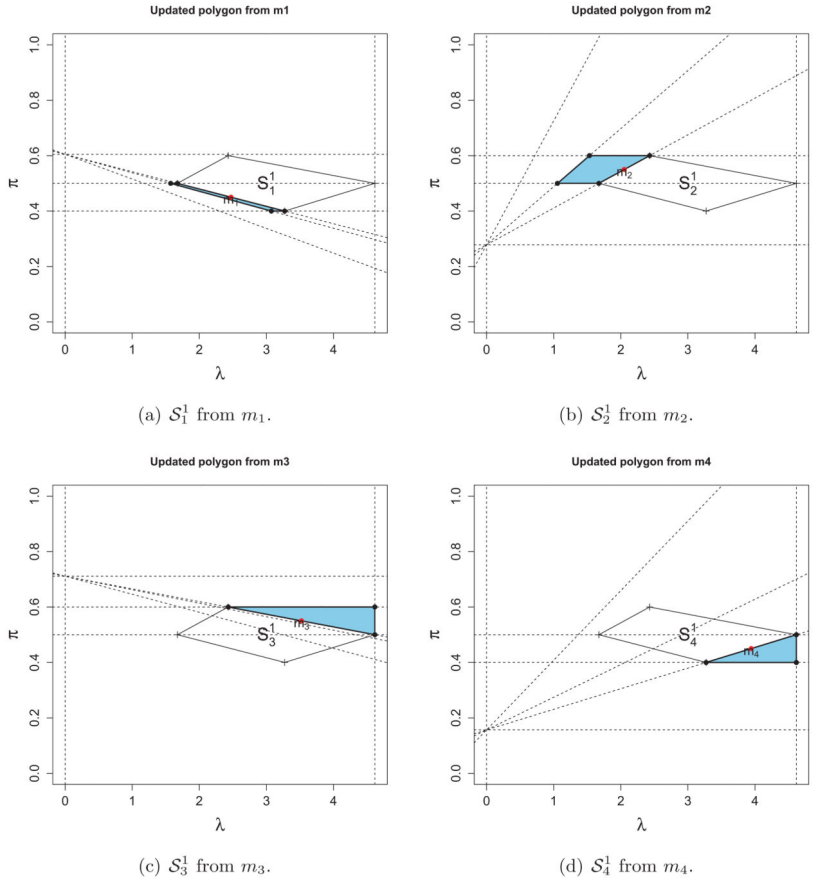
Simulated toy example: the top two panels depict the solution $\lambda\beta$ and the estimate $\hat{p}(x)$ given by five marginal π-solution paths, with λ values fixed at 0.2, 0.4, 0.6, 0.8, 1.0. The bottom two panels plot the joint solution surface of $\lambda\beta$ and the corresponding surface of $\hat{p}(x)$. Notice that the horizontal axis of (b) is x. Similarly the surface in (d) lies on the $(x \times \lambda)$ plane.



(a) Boundaries of the constraints. (b) $\mathcal{S}^0$ with 4 vertices $(v_1^0, \cdots, v_4^0)$.

Figure 2.

The illustration of defining $\mathcal{S}^0$ generated from the initial point (λ_0, π_0) for the toy example in Section 1: Each line in the left penal (a) represents a constraint boundary where an *event* occurs as labeled. For example, the upper right label represents the event: the fourth example moves from $\mathcal{E}$ to $\mathcal{R}$ on the boundary. The set $\mathcal{S}^0$ obtained from the boundaries in (a) is depicted in (b).

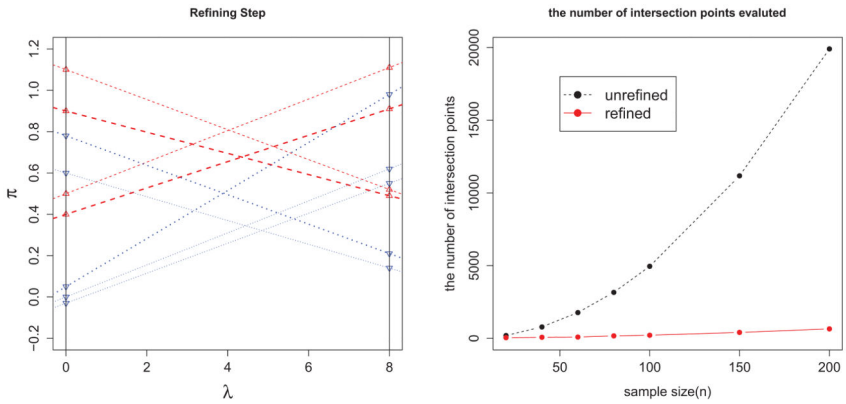


(a) $\mathcal{S}_1^1$ from m_1. (b) $\mathcal{S}_2^1$ from m_2.

(c) $\mathcal{S}_3^1$ from m_3. (d) $\mathcal{S}_4^1$ from m_4.

Figure 3.

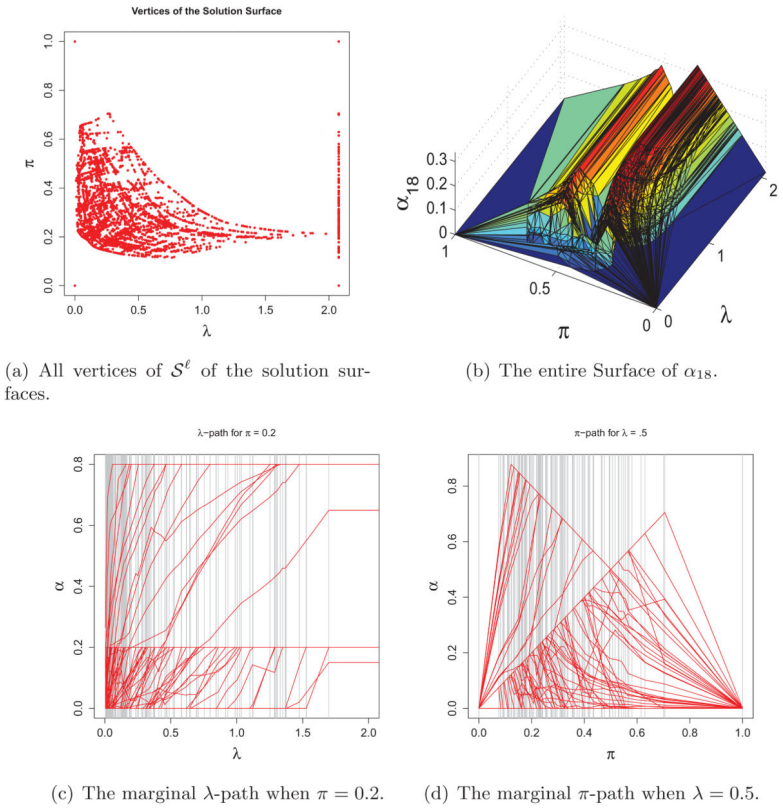
The updated polygons $\mathscr{S}_r^1$, $r = 1, \cdots, 4$ from the middle points $m_1, \cdots, m_4$ of the sides of $\mathcal{S}^0$ in Figure 2(b): Dotted lines in each subfigure depict the boundaries of constrains for obtaining the polygon.



(a) Refining Step. (b) n_c^3 and $\tilde{n}_c^3$ vs. n.

Figure 4.

Illustration of the refining step and its effect for simulated example: The left penal (a) shows all the boundaries of constraints. Dashed (red) lines represents boundaries of constraints with $b < 0$ and only two constraints (bold) are used to defined $\mathcal{S}^t$ and the other two excluded are chosen by comparing π values at $\lambda = 0$ and λ_0. Dotted (blue) lines are the ones with $b > 0$ and we can exclude dominated constraints in a similar manner. The right panel (b) shows dramatic saving in computation after use of the refining step by comparing the numbers of intersection points of unrefined constraints (n_c) and refined ones ($\tilde{n}_c$) as functions of sample size, n.



(a) All vertices of $\mathcal{S}^{\ell}$ of the solution surfaces.

(b) The entire Surface of α_{18}.

(c) The marginal λ-path when $\pi = 0.2$.

(d) The marginal π-path when $\lambda = 0.5$.

Figure 5.
Real data illustration on *kyphosis* data.

Table 2

Sources for the microarray data used for illustrations: The numbers of predictors (p) are all 7219 and much larger than the sample size (n).

Name	train(n)	test($\tilde{n}$)	p	Source
Duke Cancer	38	4	7219	West et al. (2001)
Colon Cancer	40	22	2000	Alon et al. (1999)

Table 3

Test CREs for Microarray data sets ($p > n$).

data	tuning	fixed	adaptive	improve
Duke	LOCV	0.4991	0.4794	3.94%
	10-fold	0.4846	0.4729	2.42%
Colon	LOCV	0.3058	0.2511	17.89%
	10-fold	0.3058	0.2646	13.45%

Algorithm 1

WSVM solution surface of the WSVM

Input: A training data set $(\mathbf{x}_i, y_i)$, $i = 1, \cdots, n$ and a non-negative definite kernel function $K(\mathbf{x}, \mathbf{x}')$.

Output : The entire (piecewise-linear) solution surface on $\mathcal{Q}$ of a_i, $i = 0, \cdots, n$, the optimizer of (4).

1 1. Initialize $\text{Input}^{(1)} \leftarrow \left\{\text{in}_1^{(1)}\right\} = \{\lambda^0, \pi^0, \boldsymbol{\alpha}^0, \mathcal{E}^0, \mathcal{L}^0, \mathcal{R}^0\}^T$ from Section 4.1 and $k \leftarrow 1$.

2 **While** $\text{Input}^{(k)} = \left\{\text{in}_1^{(k)}, \cdots, \text{in}_{J_k}^{(k)}\right\}$ is not empty (i.e, $J_k \neq 0$),

2-1 **For** $j = 1, \cdots, J_k$

i. Set λ^ℓ, π^ℓ, $\boldsymbol{\alpha}^\ell$, $\mathcal{E}^\ell$, $\mathcal{E}^\ell$, and $\mathcal{L}^\ell$ from $\text{in}_j^{(k)}$.

ii. If $\mathcal{E}^\ell$ is empty, resolve the *empty elbow* as described in Section 4.3, otherwise skip this step and go directly to the following step iii.

iii. Define $\mathcal{S}^\ell$ by identifying vertices $v_1^\ell, \cdots, v_{n_v}^\ell$ using (19)–(23) and update $\boldsymbol{\alpha}_r^\ell$ at v_r^ℓ, $r = 1, \cdots, n_v$ from (13).

Set $\text{out}_j \leftarrow \{\text{out}_{j,1}, \cdots, \text{out}_{j,n_v}\}$, where $\text{out}_{j,r} \leftarrow \{v_r^\ell, \boldsymbol{\alpha}_r^\ell\}$ for $r = 1, \cdots, n_v$.

iv. Compute the middle points $m_1 \cdots m_{n_v}$ and the corresponding $\bar{\boldsymbol{a}}_r$ at m_r, $r = 1, \cdots, n_v$.

Update $\mathcal{E}_r^{\ell+1}$, $\mathcal{L}_r^{\ell+1}$ and $\mathcal{R}_r^{\ell+1}$ at $m_r = (\lambda_r^{\ell+1}, \pi_m^{\ell+1})$, $r = 1, \cdots, n_v$.

Set $\text{in}_j \leftarrow \{\text{in}_{j,1}, \cdots, \text{in}_{j,n_v}\}$, where $\text{in}_{j,r} \leftarrow \{\lambda_r^{\ell+1}, \pi_r^{\ell+1}, \boldsymbol{\alpha}_r^{\ell+1}, \mathcal{E}_r^{\ell+1}, \mathcal{L}_r^{\ell+1}, \mathcal{R}_r^{\ell+1}\}^T$ for $r = 1, \cdots, n_v$.

end

2-2 Set $\text{Output}^{(k)} \leftarrow \{\text{out}_1, \cdots, \text{out}_J\}$.

2-3 Set $\text{Input}^{(k+1)} \leftarrow \{\text{in}_1, \cdots, \text{in}_J\}$.

2-4 Refine $\text{Input}^{(k+1)}$ by deleting any element which revisits an old middle points.

end

3 Set Output $\leftarrow \{\text{Output}^{(1)} : \cdots : \text{Output}^{(k)}\}$