Solution Path for Semi-Supervised Classification with Manifold Regularization*

Gang Wang Tao Chen Dit-Yan Yeung Frederick H. Lochovsky
Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong, China

Abstract

With very low extra computational cost, the entire solution path can be computed for various learning algorithms like support vector classification (SVC) and support vector regression (SVR). In this paper, we extend this promising approach to semi-supervised learning algorithms. In particular, we consider finding the solution path for the Laplacian support vector machine (LapSVM) which is a semi-supervised classification model based on manifold regularization. One advantage of the this algorithm is that the coefficient path is piecewise linear with respect to the regularization parameter, hence its computational complexity is quadratic in the number of labeled examples.

1 Background

Traditional learning algorithms for classification are based on the supervised learning paradigm in which classifiers are trained on labeled examples only. However, in many applications, labeled examples are difficult or expensive to obtain since they need substantial labeling efforts from humans. On the other hand, large quantities of unlabeled examples are often readily available and are easy to obtain. Semi-supervised learning provides an appealing alternative by augmenting traditional supervised learning with a large amount of unlabeled data to build better classifiers. In so doing, we only need a small number of labeled examples for classifier training. In recent years, semi-supervised learning has aroused a great deal of research interest and has demonstrated impressive performance improvement in practice. Manifold regularization [2] is a family of graph-based learning algorithms based on the regularization theory. Besides the usual regularization term that controls the classifier complexity in the ambient space, this approach uses an additional regularization term that controls the classifier complexity in the intrinsic geometry of the data distribution. As a result, the underlying manifold structure is incorporated into the regularization framework. This data-dependent geometric regularization approach naturally integrates labeled and unlabeled data into a general-purpose learner.

Laplacian support vector machine (LapSVM) [2] is a specific realization of manifold regularization for semi-supervised classification problems. Suppose we are given a set of l labeled examples $\mathcal{D}_{\mathcal{L}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and a set of u unlabeled examples $\mathcal{D}_{\mathcal{U}} = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$. Here $\mathbf{x}_i$ and $\mathbf{x}_j$ are drawn from the input space $\mathcal{X}$ and $y_i \in \{-1, +1\}$ is the class label. In practice, u is always much larger than l. LapSVM seeks to solve the following optimization problem:

$$f^* = \arg \inf_{f \in \mathcal{H}_K} \left(\sum_{i=1}^{l} \xi_i + \lambda_A \|f\|_K^2 + \lambda_I \|f\|_I^2 \right) \quad (1)$$

$$\text{subject to} \quad \begin{cases} y_i f(\mathbf{x}_i) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad i = 1, 2, \ldots, l. \quad (2)$$

Unlike many regularization problems, the penalty part here consists of two regularization terms, where $\|f\|_K$ is the norm in the RKHS defined by kernel function K, and $\|f\|_I$ reflects the intrinsic structure of the data distribution. Specifically, $\|f\|_I$ forces the values of $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ to be close if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close to each other with respect to the intrinsic data distribution. λ_A and λ_I are the regularization parameters that control the degrees of regularization in the ambient space and the intrinsic geometry of the data distribution, respectively. The values of the regularization parameters λ_A and λ_I have to be specified in advance by the user. In practice, some default values are usually chosen for them even though these values are by no means optimal. Extensive exploration of the optimal parameter

*This research is supported by Competitive Earmarked Research Grant (CERG) 621706 from the Research Grants Council (RGC) of the Hong Kong Special Administrative Region, China.

values is seldom pursued since re-training the model many times under different parameter settings is computationally demanding.

Recently, a novel approach has emerged that seeks to explore the entire solution path for all parameter values without having to re-train the model multiple times. By estimating the generalization errors under different parameter values, the optimal parameter value can be found with a low extra computational cost. Efron et al. [3] developed the LARS algorithm which fits the coefficient path for the linear least square regression problem regularized with the L_1 norm. This is probably the first work that explores the correspondence between every regularization parameter value and the solution. An important finding is that the coefficient path is piecewise linear and hence it is efficient to explore the entire solution path by monitoring the breakpoints only. Inspired by this pioneering work, Zhu et al. [7] proposed an algorithm to compute the entire solution path for the L_1-norm SVC and Hastie et al. [4] proposed one for the standard L_2-norm SVC. More generally, Rosset and Zhu [5] showed that any model with an L_1 regularization and a quadratic, piecewise quadratic, piecewise linear, or linear loss function has a piecewise linear coefficient path. Besides for the regularization parameters, our recent work shows that this approach can also be used for finding the entire solution path for some other parameter [6].

In this paper, we apply the solution path algorithm to LapSVM. Since the solution path is not piecewise linear with respect to either λ_A or λ_I, we transform the problem into a formulation with which a piecewise linear path can be obtained. We use a kernel defined in the intrinsic manifold space. This overcomes the difficulty of balancing between the ambient space and the geometric space, leading to further reduction of user efforts. After the graph Laplacian is constructed, the computational cost of this solution path algorithm depends only on the number of labeled examples, which is typically a small number compared with the total training set size. Hence, the optimal solution can be found very efficiently.

2 Problem Setup

A new representer theorem for LapSVM was proved in [2]. The minimizer of the optimization problem (1) admits an expansion

$$f^*(\mathbf{x}) = \sum_{i=1}^{l+u} \beta_i^* K(\mathbf{x}_i, \mathbf{x}) \tag{3}$$

in terms of both the labeled and unlabeled examples, where β_i^* are the optimal values of coefficients $\beta_i \in \mathbf{R}$ in the expansion. Note that the parameter dimensionality is $l+u$ which is usually a very large number.

A bias term is often added to the decision function. Let us denote $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{l+u})^T$ and $\mathbf{k}(\mathbf{x}) = (K(\mathbf{x}_1, \mathbf{x}), \ldots, K(\mathbf{x}_{l+u}, \mathbf{x}))^T$. Thus the decision function can be expressed as

$$f(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{k}(\mathbf{x}) + \beta_0. \tag{4}$$

Substituting (4) into (1), we can rewrite the primal optimization problem for LapSVM as

$$\min_{\boldsymbol{\beta}, \beta_0} \quad \sum_{i=1}^{l} \xi_i + \frac{\lambda}{2} \boldsymbol{\beta}^T (\rho \mathbf{K} + \mathbf{K}\mathbf{L}\mathbf{K}) \boldsymbol{\beta} \tag{5}$$

$$\text{subject to} \quad \begin{cases} y_i f(\mathbf{x}_i) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad i = 1, 2, \ldots, l \tag{6}$$

where λ and ρ replace λ_A and λ_I as the parameters, $\mathbf{K}$ is the $(l+u)\times(l+u)$ kernel matrix over all labeled and unlabeled data, and $\mathbf{L}$ is the graph Laplacian [1] given by $\mathbf{L} = \mathbf{D} - \mathbf{W}$ where W_{ij} in $\mathbf{W}$ are the edge weights in the adjacency graph and $\mathbf{D}$ is a diagonal matrix with diagonal entries $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$. Since computing $\mathbf{W}$ needs to examine every pair of labeled and unlabeled examples, its computational complexity is $O((l+u)^2)$.

LapSVM is particularly useful for problems in which the data exhibit strong manifold structure. In order that the regularization term for the intrinsic structure of the data distribution takes effect, we typically set λ_I to be much larger than λ_A. Equivalently, in (5), ρ is set to a small number.

By introducing Lagrange multipliers $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_l)^T$ and $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_l)^T$, we can obtain the Lagrangian $L_P(\boldsymbol{\beta}, \beta_0, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\zeta})$. Setting the corresponding derivatives to zero, we have

$$\frac{\partial L_P}{\partial \boldsymbol{\beta}}: \qquad \boldsymbol{\beta} = \frac{1}{\lambda} (\rho \mathbf{I} + \mathbf{L}\mathbf{K})^{-1} \mathbf{J}^T \mathbf{Y} \boldsymbol{\alpha}, \tag{7}$$

$$\frac{\partial L_P}{\partial \beta_0}: \qquad \sum_{i=1}^{l} y_i \alpha_i = 0, \tag{8}$$

$$\frac{\partial L_P}{\partial \xi_i}: \qquad \alpha_i = 1 - \zeta_i, \tag{9}$$

where $\mathbf{I}$ is the $(l+u)\times(l+u)$ identity matrix, $\mathbf{J} = [\mathbf{I}_{l\times l}, \mathbf{0}_{l\times u}]$ is an $l\times(l+u)$ matrix with $\mathbf{I}_{l\times l}$ being the $l\times l$ identity matrix, and $\mathbf{Y} = \mathrm{diag}(y_1, \ldots, y_l)$ is a diagonal matrix. Note that $\mathbf{L}\mathbf{K}$ is always not of full rank and hence is singular. However, adding the term $\rho\mathbf{I}$ to it can make it invertible. For notational simplicity, let $\mathbf{P} = (\rho\mathbf{I} + \mathbf{L}\mathbf{K})^{-1} \mathbf{J}^T \mathbf{Y}$ and hence $\boldsymbol{\beta} = \frac{1}{\lambda} \mathbf{P} \boldsymbol{\alpha}$.

Substituting (7)–(9) into $L_P(\boldsymbol{\beta}, \beta_0, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\zeta})$, we have

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2\lambda} \boldsymbol{\alpha}^T \mathbf{Y}\mathbf{J}\mathbf{K}(\rho\mathbf{I} + \mathbf{L}\mathbf{K})^{-1} \mathbf{J}^T \mathbf{Y} \boldsymbol{\alpha}, \tag{10}$$

eliminating all the primal variables. It follows from the Karush-Kuhn-Tucker (KKT) conditions that

$$\begin{aligned} y_i f(\mathbf{x}_i) > 1 \quad &\Rightarrow \quad \alpha_i = 0, \\ y_i f(\mathbf{x}_i) = 1 \quad &\Rightarrow \quad \alpha_i \in [0, 1], \\ y_i f(\mathbf{x}_i) < 1 \quad &\Rightarrow \quad \alpha_i = 1. \end{aligned}$$

Thus we arrive at the following dual optimization problem:

$$\max_{\boldsymbol{\alpha} \in \mathbf{R}^l} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2\lambda} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}, \tag{11}$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{l} y_i \alpha_i = 0 \\ 0 \le \alpha_i \le 1 \end{cases} \quad i = 1, 2, \ldots, l, \tag{12}$$

where $\mathbf{Q} = \mathbf{YJK}(\rho\mathbf{I} + \mathbf{LK})^{-1}\mathbf{J}^T\mathbf{Y}$ is a positive definite matrix with size $l \times l$. As we can see, the optimization problem for LapSVM is a convex optimization problem that can be solved using a standard SVM solver. The decision function is thus given by

$$f(\mathbf{x}) = \frac{1}{\lambda} \boldsymbol{\alpha}^T \mathbf{P}^T \mathbf{k}(\mathbf{x}) + \beta_0. \tag{13}$$

The dimensionality of $\boldsymbol{\alpha}$ in the dual problem is l, which is much smaller than the dimensionality of $\boldsymbol{\alpha}$ in the primal problem. $\mathbf{P}$ is a translating matrix that can be computed in advance. Through $\mathbf{P}$, we can map any $\boldsymbol{\alpha}$ value with dimensionality l to its corresponding parameter $\boldsymbol{\beta}$ with dimensionality $l+u$. Since the parameter $\boldsymbol{\alpha}$ is related to the labeled examples only, the solution path algorithm can focus on the labeled examples. The manifold information characterized by the unlabeled examples is incorporated into the translating matrix.

Let I_+ denote the set of indices of the points with $y_i = +1$ and l_+ the cardinality of I_+. Likewise, I_- and l_- are defined for the points with $y_i = -1$. We define the following sets of points:

$$\begin{aligned} \mathcal{E} &= \{i : y_i f(\mathbf{x}_i) = 1, 0 \le \alpha_i \le 1\} \\ \mathcal{L} &= \{i : y_i f(\mathbf{x}_i) < 1, \alpha_i = 1\} \\ \mathcal{R} &= \{i : y_i f(\mathbf{x}_i) > 1, \alpha_i = 0\} \end{aligned}$$

These three point sets refer to points lying at, inside and outside the margin, respectively. As we change the λ value, the margin will change and some events may occur during this process. An event is said to occur when a point enters or leaves the elbow, causing some point sets to change. We categorize these events as follows:

- A point enters the elbow:
  - from $\mathcal{L}$ to $\mathcal{E}$ with $\alpha_i = 1$
  - from $\mathcal{R}$ to $\mathcal{E}$ with $\alpha_i = 0$
- A point leaves the elbow:
  - from $\mathcal{E}$ to $\mathcal{L}$ with $\alpha_i = 1$
  - from $\mathcal{E}$ to $\mathcal{R}$ with $\alpha_i = 0$

For the points not at the elbow, i.e., in $\mathcal{R} \cup \mathcal{L}$, their α_i values remain fixed until an event occurs. Hence, it is sufficient to focus on the points at the elbow. As a point passes through $\mathcal{E}$, its α_i value will change from 0 to 1 or from 1 to 0.

3 Finding the Solution Path

3.1 Initialization

The solution path algorithm explores the correspondence between every λ value and the corresponding solution $\boldsymbol{\alpha}(\lambda)$. When λ tends to $+\infty$, the initialization becomes simpler. The objective degenerates to maximizing $\sum_{i=1}^{l} \alpha_i$ subject to the constraints (12). The initial values $\boldsymbol{\alpha}^0$ and β_0^0 depend on whether or not $l_+ = l_-$.

Lemma 1 *Suppose $l_+ = l_-$. For λ sufficiently large, all the $\alpha_i^0 = 1$ and $\beta_0^0 \in [-1, 1]$. The loss is $\sum_{i=1}^{l} \xi_i = l_+ + l_-$ for any β_0^0.*

In view of Lemma 1, the initial solution $\boldsymbol{\alpha}^0$ is $(1, \ldots, 1)^T$ and thus $\boldsymbol{\beta}^0 = \frac{1}{\lambda}\mathbf{P}\boldsymbol{\alpha}^0$. As λ decreases, in order to satisfy the constraints (12), all α_i's remain unchanged until one positive example $\mathbf{x}_{i_+}$ and one negative example $\mathbf{x}_{i_-}$ reach the elbow simultaneously. To find $\mathbf{x}_{i_+}$ and $\mathbf{x}_{i_-}$, note that $y_i f(\mathbf{x}_i) \le 1$ for $i = 1, \ldots, l$. Among all the positive examples, $\mathbf{x}_{i_+}$ is the first one that reaches the elbow. Similarly, among all the negative examples, $\mathbf{x}_{i_-}$ is the first one that reaches the elbow. Therefore

$$i_+ = \arg\max_{i \in I_+} f(\mathbf{x}_i) = \arg\max_{i \in I_+} \left((\boldsymbol{\alpha}^0)^T \mathbf{P}^T \mathbf{k}(\mathbf{x}_i) \right) \tag{14}$$

$$i_- = \arg\min_{i \in I_-} f(\mathbf{x}_i) = \arg\min_{i \in I_-} \left((\boldsymbol{\alpha}^0)^T \mathbf{P}^T \mathbf{k}(\mathbf{x}_i) \right). \tag{15}$$

When $\mathbf{x}_{i_+}$ and $\mathbf{x}_{i_-}$ both hit the elbow, the two equations $y_{i_+} f(\mathbf{x}_{i_+}) = 1$ and $y_{i_-} f(\mathbf{x}_{i_-}) = 1$ must hold. It then follows that the initial solutions λ^0 and β_0^0 are

$$\lambda^0 = \frac{(\boldsymbol{\alpha}^0)^T \mathbf{P}^T \left(\mathbf{k}(\mathbf{x}_{i_+}) - \mathbf{k}(\mathbf{x}_{i_-}) \right)}{2} \tag{16}$$

$$\beta_0^0 = \frac{-(\boldsymbol{\alpha}^0)^T \mathbf{P}^T \left(\mathbf{k}(\mathbf{x}_{i_+}) + \mathbf{k}(\mathbf{x}_{i_-}) \right)}{(\boldsymbol{\alpha}^0)^T \mathbf{P}^T \left(\mathbf{k}(\mathbf{x}_{i_+}) - \mathbf{k}(\mathbf{x}_{i_-}) \right)}. \tag{17}$$

We next consider the initialization setting when the two classes are unbalanced, i.e., $l_+ \neq l_-$. Without loss of generality, we assume that $l_+ > l_-$. As λ tends to $+\infty$, $\boldsymbol{\beta}$ tends to the zero vector $\mathbf{0}$ due to (7). The optimal choice of β_0^0 is 1 and thus the loss is $\sum_{i=1}^{l} \xi_i = l_-$. The initial solution $\boldsymbol{\alpha}^0$ can be obtained by solving a quadratic programming (QP) problem.

Lemma 2 *Suppose $l_+ > l_-$. For λ sufficiently large, the initial solution $\boldsymbol{\alpha}^0$ can be obtained as*

$$\boldsymbol{\alpha}^0 = \arg\min_{\boldsymbol{\alpha}\in\mathbf{R}^l} \boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha}$$

$$\text{subject to} \quad \begin{cases} \alpha_i = 1 & \forall i \in I_- \\ \alpha_i \in [0,1] & \forall i \in I_+ \end{cases} \quad \text{and} \quad \sum_{i\in I_+} \alpha_i = l_-.$$

There are two possible cases in which $\mathbf{x}_{i_+}$ should be distinctively decided:

1. There are two or more elements in I_+ with $0 < \alpha_i^0 < 1$, or
2. α_i^0 is either 0 or 1 for all i in I_+.

For the first case, $\mathbf{x}_{i_+}$ is chosen such that $\alpha_{i_+}^0 \in (0,1)$ (at the elbow). For the second case, let I_+^1 denote the set of points in I_+ with $\alpha_i^0 = 1$. This set is comparable with I_+ in the balanced case. Therefore, $i_+ = \arg\max_{i\in I_+^1} (\boldsymbol{\alpha}^0)^T\mathbf{P}^T\mathbf{k}(\mathbf{x}_i)$. Since both $\mathbf{x}_{i_+}$ and $\mathbf{x}_{i_-}$ lie at the elbow, λ^0 and β_0^0 are identical in form to (16) and (17).

3.2 Solution Path

In this subsection, we consider the period between the lth event (with $\lambda = \lambda^l$) and the $(l+1)$th event (with $\lambda = \lambda^{l+1}$). The set $\mathcal{E}$ is stable during this period. Suppose $\mathcal{E}$ contains m indices which are represented as an m-tuple $(\mathcal{E}(1), \cdots, \mathcal{E}(m))$ such that $\mathcal{E}(i) < \mathcal{E}(j)$ for $i < j$, where m, which is typically a very small number $m \le l$, is the number of points at the elbows. We trace the solution path of α_i for each $i \in \mathcal{E}$. For the convenience of derivation, we define $\alpha_0 = \lambda\beta_0$. It follows that $f(\mathbf{x}) = \frac{1}{\lambda}(\boldsymbol{\alpha}^T\mathbf{P}^T\mathbf{k}(\mathbf{x}) + \alpha_0)$.

We first introduce some notations. Let $\mathbf{p}_i$ be the ith column of $\mathbf{P}$. Then $\mathbf{P}_\mathcal{E} = [\mathbf{p}_{\mathcal{E}(1)}, \mathbf{p}_{\mathcal{E}(2)}, \ldots, \mathbf{p}_{\mathcal{E}(m)}]$ is an $(l+u)\times m$ matrix. Moreover, $\mathbf{y}_\mathcal{E} = (y_{\mathcal{E}(1)}, \ldots, y_{\mathcal{E}(m)})^T$ is an $m \times 1$ vector and $\mathbf{K}_\mathcal{E} = \left[\mathbf{k}(\mathbf{x}_{\mathcal{E}_1}), \ldots, \mathbf{k}(\mathbf{x}_{\mathcal{E}_m})\right]^T$ is an $m \times (l+u)$ matrix. We have the following theorem.

Theorem 1 *Suppose the solutions to $\{\alpha_i\}$ and α_0 are $\{\alpha_i^l\}$ and α_0^l when $\lambda = \lambda^l$. Then when $\lambda^{l+1} < \lambda < \lambda^l$,*

1. *If $i \in \mathcal{L}\cup\mathcal{R}$, $\alpha_i = \alpha_i^l$ is fixed at 0 or 1 which is independent of λ.*
2. *The solutions to $\{\alpha_{\mathcal{E}(i)}\}$ and α_0 are given by*

$$\begin{pmatrix} \alpha_0 \\ \alpha_{\mathcal{E}(1)} \\ \cdot \\ \cdot \\ \cdot \\ \alpha_{\mathcal{E}(m)} \end{pmatrix} = \begin{pmatrix} \alpha_0^l \\ \alpha_{\mathcal{E}(1)}^l \\ \cdot \\ \cdot \\ \cdot \\ \alpha_{\mathcal{E}(m)}^l \end{pmatrix} + (\lambda - \lambda^l)\mathbf{A}_\mathcal{E}^{-1}\mathbf{y}^a, \tag{18}$$

where $\mathbf{A}_\mathcal{E} = \begin{bmatrix} 0 & \mathbf{y}_\mathcal{E}^T \\ \mathbf{1} & \mathbf{K}_\mathcal{E}\mathbf{P}_\mathcal{E} \end{bmatrix}, \mathbf{y}^a = \begin{bmatrix} 0 \\ \mathbf{y}_\mathcal{E} \end{bmatrix}.$

We do not give the proof in this paper due to the tight page limit, and the derivation is similar to that in [4]. From Theorem 1, we can see that the solutions to $\{\alpha_{\mathcal{E}(i)}\}$ and α_0 are linear in λ while those to others remain unchanged. As λ decreases, the algorithm monitors the occurrence of any of the following events:

- One of the $\alpha_{\mathcal{E}(i)}$ for $i = 1, \ldots, m$ reaches 0 or 1.
- A point $i \notin \mathcal{E}$ hits the elbow, i.e., $y_i f(\mathbf{x}_i) = 1$.

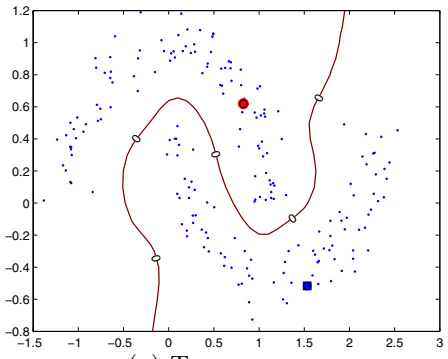
The λ values for the first type of events can be calculated directly from (18). Plugging the updating rule (18) into the regression function (13), we can calculate the λ values in which the second type of events occur. Hence, by monitoring the occurrence of these events, we choose the largest $\lambda < \lambda^l$ for which an event occurs. This λ value is a breakpoint and is denoted by λ^{l+1}. We then update the point sets and continue until λ tends to zero.

Through this process, we can explore the entire solution path by updating the parameters iteratively. As λ changes from a large value towards 0, most labeled examples pass through the elbow from inside the margin to outside. It is possible that a point passes through the elbow multiple times. In each update, a set of linear equations is solved with $O(m^3)$ where m is typically quite small. Moreover, scanning through the labeled examples to evaluate the next move has $O(l)$ time complexity. Since it needs $O(l)$ iterations to explore the entire path, the overall time complexity is $O(l^2 + l\times m^3)$. Remember that the complexity for the Laplacian matrix is $O((l+u)^2)$, and the computation of $\mathbf{P}$ requires the inversion of a matrix with size $(l+u)\times(l+u)$, which costs $O((l+u)^3)$. Since u is usually much larger than l, the solution path algorithm only takes a very small portion of the total computation for LapSVM.

4 Experiments

The behavior of the solution path algorithm can best be illustrated using video, and we have prepared some illustrative examples as video at `http://www.cse.ust.hk/~wanggang/sol_path/manifold_path.html`.

In our experiments, we consider one synthetic and one real datasets. We partition each dataset into a training set and a test set. In the training set, we randomly select a small number of points as labeled examples and keep the rest unlabeled. We then compute the graph Laplacian and explore the solution path based on the training set. We select the same number of labeled points from each class to simplify the initialization step. To evaluate the classification performance for different solutions along the path, out-of-sample classification accuracy is measured based on the separate test



(a) Two moons

Figure 1. Two moons dataset with strong manifold structure. The decision boundary in the figure is the LapSVM solution when only one positive and one negative points are labeled.

set. In the experiments, all the points are normalized between $[-1, 1]$ before the process, and the Guassian RBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma)$ is used with $\sigma = 0.1$. We set $\rho = 0.001$ and use the k-nearest neighbors method to compute the graph Laplacian where k is set to 6.

4.1 Two Moons Dataset

Figure 1 shows the two moons dataset which has strong manifold structure. The dataset contains 200 points. When only one positive and one negative points are selected as labeled examples, the decision function remains the same along the entire solution path since no event occurs until λ decreases to 0.

We randomly select 70%, 50% and 30% of the points from the two moons dataset for training while keeping the rest for testing. In Figure 2(a), since 70% of the points are used for training, the manifold structure is preserved well. As a result, only two labeled points are sufficient to give a good decision boundary achieving 100% classification accuracy. Labeling more points is not necessary and hence the four curves corresponding to different numbers of labeled points overlap completely. In Figure 2(b) when 50% of the points are used for training, the manifold structure becomes weaker. Thus, having only two labeled training points cannot achieve very high classification accuracy. Better classification accuracy can be obtained when more training points are labeled. We also notice that the decision function changes dramatically for different λ values. Figure 3 shows a typical example. When $\lambda = 2$, the decision function splits the points from one class into two parts leading to very low classification accuracy. As λ decreases, the optimal solutions for different λ are obtained. When only 30% of the points are used for training, as shown in Figure 2(c), the manifold structure degenerates to a number of small clusters. This case may be more similar to the real datasets we use in practice. For this case, having more labeled points is essential for identifying the class labels of these clusters. Since the solution path algorithm explores the solutions for all λ values, the solution which generalizes best on the test set can be identified easily.

4.2 USPS Dataset

In this experiment, we consider the problem of classifying handwritten digits. The dataset we use is the USPS handwritten digit dataset, where each digit is represented as an 8-level gray-scale image of size 16×16. There are 1100 images for each digit. For each digit dataset, we partition it into a training set of 600 images and a test set of 500 images. The images for each digit form a manifold in the image space. However, since the images for each digit possess large variations due to rotation, thickness and style, the corresponding manifold structure may not be continuous. We consider three two-class classification problems, i.e., digits 2 vs. 3, 5 vs. 6, and 8 vs. 9.

Different labeled point sets will induce different solution paths especially when the number of labeled points is very small. To reduce the effect due to random sampling, we perform multiple runs on different labeled point sets for each given number of labeled points. The best accuracies obtained on the test set along the entire path are then used to compute the average classification accuracy for the corresponding number of labeled points. Table 1 summarizes the results. When the number of labeled points is small, the accuracies obtained from different solution paths vary significantly, leading to relatively large standard deviation. This seems to indicate that the images from one digit class do not form a single compact manifold but possibly multiple clusters. Thus, as more points are labeled, the labels for the clusters can be identified correctly. We notice that the standard deviations are quite small when the size of the labeled set is 16 or 32. The accuracies obtained from the optimal solutions for different solution paths are stable enough. Therefore, the Laplacian SVM can learn a good classifier when a small labeled set get the help from unlabeled points.

5 Conclusion

We have proposed an efficient algorithm for computing the solution path for Laplacian SVM. The solution path is piecewise linear with respect to a regularization parameter, and the computational requirement depends on the number of labeled examples only. The optimal solution can be found very efficiently.

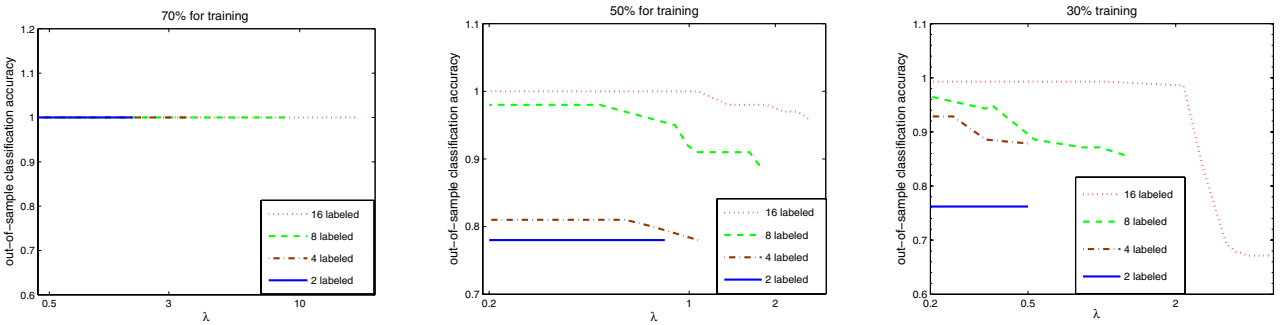


Figure 2. Out-of-sample classification accuracy along the entire solution path. (a) 70% points for training; (b) 50% points for training; (c) 30% points for training. The horizontal axis is in log scale.

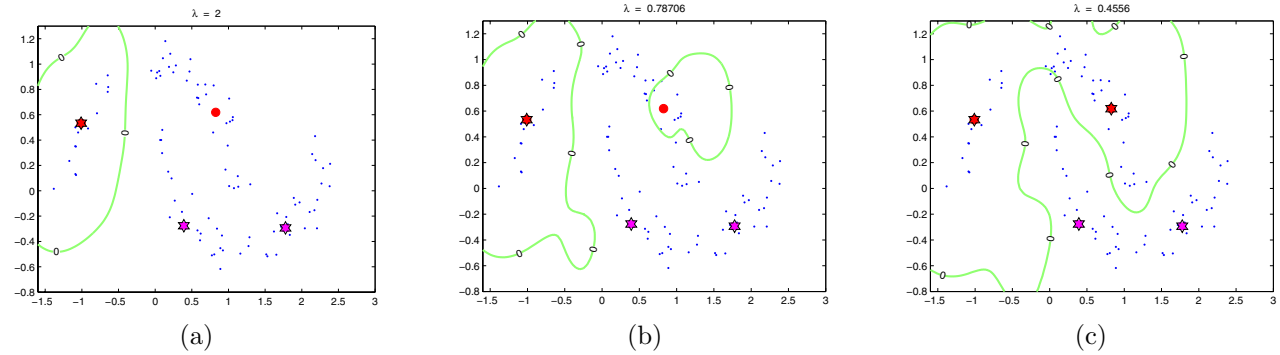


Figure 3. 50% of the points are chosen for training while four points are labeled and the others are unlabeled. The decision functions are shown for different λ values. (a) $\lambda = 2$; (b) $\lambda = 0.8$; (c) $\lambda = 0.2$.

%	2 labeled	4 labeled	8 labeled	16 labeled	32 labeled
2 vs. 3	58.4 (15.6)	79.1 (12.6)	95.7 (3.4)	97.9 (0.6)	98.4 (0.4)
5 vs. 6	61.7 (15.5)	83.9 (10.8)	95.3 (3.6)	98.6 (1.0)	99.0 (0.3)
8 vs. 9	63.4 (16.7)	73.8 (13.5)	88.5 (7.9)	95.7 (3.4)	97.1 (0.3)

Table 1. Average classification accuracies for the optimal solutions with different numbers of labeled points. The corresponding standard deviations are shown inside brackets.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14 (NIPS-01)*, 2001.

[2] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from examples (tr-2004-06). Technical report, Department of Computer Science, University of Chicago, 2004.

[3] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. Technical report, Stanford University, 2002.

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

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

[6] G. Wang, D. Yeung, and F. Lochovsky. Two-dimensional solution path for support vector regression. In *Proceedings of the 23th International Conference on Machine Learning (ICML-06).*, 2006.

[7] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*, 2003.