Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Efficient Computation and Model Selection in Semi-Supervised Learning ⋆

Gang Wang * **Shiyin Qin**** **and Pipei Huang****

* *Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China. (e-mail: wanggang@cse.ust.hk).*
** *School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. (e-mail: qsy@buaa.edu.cn, huangpipei@gmail.com)*

Abstract: Traditional learning algorithm uses only labeled data for training. However, labeled examples are often difficult or time consuming to obtain since they require substantial labeling efforts from humans. On the other hand, unlabeled data are often relatively easy to collect. Semi-supervised learning addresses this problem by using large quantities of unlabeled data with the labeled data to build better learning algorithms. In this paper, we propose a general approach augmenting traditional supervised learning into semi-supervised learning paradigm. A regularization framework which balances a tradeoff between loss and penalty is established. We investigate different implementations of loss function and suggest the methods which have the least computation expenses. The value of a hyperparameter, which determines the balance between loss and penalty, is crucial in model selection. Hence, we derive an algorithm that can fit the entire path of solutions for every value of hyperparameter. Its computational complexity is quadratic in the number of labeled examples only rather than the total number of labeled and unlabeled examples.

1. INTRODUCTION

Traditional learning algorithms for classification and regression are based on the supervised learning paradigm in which models 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 learning algorithm. In so doing, we only need a small number of labeled examples for training. In recent years, semi-supervised learning has aroused a great deal of research interest and has demonstrated impressive performance improvement in practice. Some semi-supervised learning methods proposed over the last few years include co-training, transductive SVM, EM with generative mixture model, and a variety of graph-based methods.

In this paper, we propose a general approach to formulate the semi-supervised learning problem. Specially, we use the dictionary method to construct the predicting function f as a linear expansion of basis functions, such as the following:

$$f(\mathbf{x}) = \sum_{i=1}^{q} \beta_i h_i(\mathbf{x}) + \beta_0, \tag{1}$$

⋆ This work was supported by hi-tech research and development (863) program of China (Grant No. 2006AA04Z207) and research fund for doctorial program of higher education of China (Grant No. 20060006018).

where $\mathcal{H} = \{h_1(\mathbf{x}), \ldots, h_q(\mathbf{x})\}$ is a dictionary of basis functions and $\boldsymbol{\beta} = (\beta_i)_{i=1}^{q}$ and β_0 are the coefficients of the function. In the semi-supervised learning paradigm, 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 $\mathcal{D}_{\mathcal{U}} = \{\mathbf{x}_j\}_{j=l+1}^{j=l+u}$. $\mathcal{D} = \mathcal{D}_{\mathcal{L}} \bigcup \mathcal{D}_{\mathcal{U}}$. Here $\mathbf{x}_i$ and $\mathbf{x}_j$ are drawn from the input space $\mathcal{X}$. The output y_i is a real number for regression, whereas it takes values $\{-1, +1\}$ for classification. In practice, u is always much larger than l. The problem of learning is to provide an estimator, i.e., a function $f : \mathcal{X} \to \mathcal{Y}$, based on the data set $\mathcal{D}$. The function f can be used to predict a value y given any value of $\mathbf{x} \in \mathcal{X}$. Since kernel methods [Schölkopf and Smola, 2002] have demonstrated great successes in solving many machine learning and pattern recognition problems, we use kernels to define the basis functions in the paper. Thus, the function $f(\mathbf{x})$ admits a representation of the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{l+u} \beta_i K(\mathbf{x}_i, \mathbf{x}) + \beta_0, \tag{2}$$

where $K(\cdot, \cdot)$ is a kernel function defined on $\mathcal{X} \times \mathcal{X}$. The decision function depends on both labeled and unlabeled data and each basis function $K(\mathbf{x}_i, \mathbf{x})$ is related to only one input example.

We consider the following optimization problem:

$$\min_{f} R = \sum_{i=1}^{l} L(y_i, f(\mathbf{x}_i)) + \frac{\lambda}{2} \sum_{i,j=1}^{u+l} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 w_{ij}, \tag{3}$$

which maintains a tradeoff between loss and penalty. $L(y, f(\mathbf{x}))$ is the loss function measuring the error when predicting y by $f(\mathbf{x})$. Its value depends on the labeled

data only. Similar to the regularization theory [Evgeniou et al., 2000], a large number of learning models can be derived subject to different choices of L. For the binary classification problem where $y \in \{1, -1\}$, we have:

- Hinge loss: $L = |1 - yf(\mathbf{x})|_+$;
- Exponential loss: $L = \exp(-yf(\mathbf{x}))$;
- Logistic loss: $L = \log(1 + \exp(-yf(\mathbf{x})))$.

For the regression problem where y is a real number, the loss function can be defined as:

- Square loss: $L = (y - f(\mathbf{x}))^2$;
- ϵ-insensitive loss: $L = |y - f(\mathbf{x})|_\epsilon$;
- Huber loss: $L = \begin{cases} (y - f(\mathbf{x}))^2 & |y - f(\mathbf{x})| < 1 \\ 2(|y - f(\mathbf{x})| - 0.5) & |y - f(\mathbf{x})| \geq 1 \end{cases}$.

The second part in (3) is the penalty term which reflects the intrinsic structure of the data distribution. It 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. Thus, both labeled and unlabeled data are used in this part. This term is also called the graph Laplacian, which has been used in dimension reduction [Belkin and Niyogi, 2001] and manifold regularization [Belkin et al., 2004]. w_{ij} is the edge weight in the adjacency graph and more weight is given to the entry if its related two points are closer to each other. Generally, there are two variations, i.e., ϵ-neighborhood and k-nearest neighbors, to construct the adjacency graph. In the ϵ-neighborhood method, nodes i and j are connected by an edge if $\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq \epsilon$. In the k-nearest neighbors method, nodes i and j are connected by an edge if i is among the k nearest neighbors of j or j is among the k nearest neighbors of i.

Based on the optimization problem (3) with the decision function defined in (2), we have a general approach to formulate the semi-supervised learning problem for either classification or regression. From investigation on different implementations of loss function and their related computational complexity, we suggest to use the hinge loss for classification and the ϵ-insensitive loss for regression, since their computation expenses in the optimization depend only on the labeled examples. The unlabeled data can be preprocessed before the optimization is performed.

A hyperparameter λ, which balances loss and penalty, is crucial to determine the generalization ability of the model. If λ is set by a large value, the coefficients of the function f tends to be zero. On the other hand, if $\lambda = 0$, we only minimize the training error which always leads to the overfitting problem. Thus, the penalty term in (3) can be considered as the regularization, where λ plays a role in the capacity control. The value of λ has to be specified in advance by the user. In practice, some default values are usually chosen and the model is trained for multiple times. Extensive exploration of the optimal λ values is seldom pursued since re-training the model many times under different λ settings is computationally demanding. Recently, a novel approach has emerged that seeks to explore the entire path of solutions for all hyperparameter values without having to re-train the model multiple times [Rosset and Zhu, 2003, Efron et al., 2004]. By estimating the generalization errors under different hyperparameter values, the optimal hyperparameter value can be found with a low extra computational cost. In this paper, we apply the solution path algorithm to sequentially calculate all solutions corresponding to all λ values. The computational cost is much lower than that of the traditional approach which requires training the model multiple times.

2. SELECTION ON LOSS FUNCTIONS

For notation simplicity, we denote $\mathbf{y} = (y_1, \ldots, y_l)^T$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{l+u})^T$, $\mathbf{k}(\mathbf{x}) = (K(\mathbf{x}_1, \mathbf{x}), \ldots, K(\mathbf{x}_{l+u}, \mathbf{x}))^T$, $\mathbf{K} = [K(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^{l+u}$, $\mathbf{1}$ is a column vector with all entries being one, $\mathbf{I}$ is the identity matrix and $\mathbf{J} = [\mathbf{I}_{l\times l}, \mathbf{0}_{l\times u}]$ is an $l \times (l + u)$ matrix with $\mathbf{I}_{l\times l}$ being the $l \times l$ identity matrix. The decision function can be expressed as

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

When the square loss is used, the optimization problem becomes:

$$\min_{\boldsymbol{\beta},\beta_0} R_s = \|\mathbf{y} - \mathbf{JK}\boldsymbol{\beta} - \beta_0\mathbf{1}\|^2 + \frac{\lambda}{2}\boldsymbol{\beta}^T\mathbf{KLK}\boldsymbol{\beta}. \tag{5}$$

$\mathbf{L}$ is given by $\mathbf{L} = \mathbf{D} - \mathbf{W}$ [Belkin and Niyogi, 2001], where w_{ij} in $\mathbf{W}$ is the edge weight in the adjacency graph and $\mathbf{D}$ is a diagonal matrix with diagonal entries $D_{ii} = \sum_{j=1}^{l+u} w_{ij}$. The solution to the problem (5) can be obtained directly through setting the derivative of the objective function to zero. Thus, we have

$$\begin{bmatrix} \boldsymbol{\beta} \\ \beta_0 \end{bmatrix} = 2(2\bar{\mathbf{K}}^T\mathbf{J}^T\mathbf{J}\bar{\mathbf{K}} + \lambda\bar{\mathbf{K}}^T\mathbf{L}\bar{\mathbf{K}})^{-1}\bar{\mathbf{K}}^T\mathbf{J}^T\mathbf{y} \tag{6}$$

where $\bar{\mathbf{K}} = [\mathbf{K}, \mathbf{1}]$ is an $(u + l) \times (u + l + 1)$ matrix. The solution is obtained in a simple closed-form. However, we notice that there is a matrix inversion operation in the formula (6) with the computational complexity $O\left((u + l + 1)^3\right)$. For each possible λ value, such expensive matrix operation has to be performed once. Accordingly, it is computational prohibitive to explore a large number of candidate values of λ.

If we use the exponential loss to define the loss function, the problem is

$$\min_{\boldsymbol{\beta},\beta_0} R_e = \sum_{i=1}^{l} \exp(-y_i f(\mathbf{x}_i)) + \frac{\lambda}{2}\boldsymbol{\beta}^T\mathbf{KLK}\boldsymbol{\beta}. \tag{7}$$

The gradients of the function R_e with respect to $\boldsymbol{\beta}$ and β_0 are

$$\frac{\partial R}{\partial \boldsymbol{\beta}} = -\sum_{i=1}^{l} \exp(-y_i f(\mathbf{x}_i)) y_i \mathbf{k}(\mathbf{x}_i) + \lambda\mathbf{KLK}\boldsymbol{\beta} \tag{8}$$

$$\frac{\partial R}{\partial \beta_0} = -\sum_{i=1}^{l} \exp(-y_i f(\mathbf{x}_i)) y_i \tag{9}$$

In order to optimize the solution, the numeric method such as conjugate gradient or quasi-Newton can be used. Since the term $\mathbf{KLK}$ is fixed during the optimization procedure, its value can be calculated in the preprocessing step. Thus, it costs $O((u + l)^2)$ operations to compute the gradient (8). The overall complexity in optimizing the problem (7) depends on the number of iterations until optimization convergence. The numerical analysis [Burden and Faires, 2000] gives some theoretical results on the convergence rate. As we can see, the gradient computation depends on both labeled and unlabel examples, thus, the cost of optimizing the solution is expensive. For other

implementations of loss function such as the logistic loss and the huber loss where their optimizations are based on the gradient methods, their computational expenses are similar to that of using the exponential loss. Therefore, we have the same difficulty in optimizing solutions for many λ values as in the square loss.

The hinge loss and the ϵ-insensitive loss are piecewise linear functions. They are originally used to define the support vector classification (SVC) and support vector regression (SVR) [Vapnik, 1998] respectively. Since their primal optimization problems are not differentiable, the optimization is always performed in its corresponding dual problems. Due to the space limitations, we only consider the hinge loss in this paper [1] . When the hinge loss is used, the problem is

$$\min_{\boldsymbol{\beta},\beta_0} R_h = \sum_{i=1}^{l} \xi_i + \frac{\lambda}{2}\boldsymbol{\beta}^T \mathbf{K}\mathbf{L}\mathbf{K}\boldsymbol{\beta} \tag{10}$$

$$\text{subject to } \begin{cases} y_i f(\mathbf{x}_i) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad i = 1, 2, \ldots, l \tag{11}$$

By introducing Lagrange multipliers $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_l)^T$ and $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_l)^T$ for the constraints (11), we can obtain the Lagrangian $R_h(\boldsymbol{\beta}, \beta_0, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\zeta})$. Setting the corresponding derivatives to zero, we have

$$\frac{\partial R_h}{\partial \boldsymbol{\beta}} : \boldsymbol{\beta} = \frac{1}{\lambda}(\mathbf{L}\mathbf{K})^{-1}\mathbf{J}^T\mathbf{Y}\boldsymbol{\alpha}, \tag{12}$$

$$\frac{\partial R_h}{\partial \beta_0} : \sum_{i=1}^{l} y_i \alpha_i = 0, \tag{13}$$

$$\frac{\partial R_h}{\partial \xi_i} : \alpha_i = 1 - \zeta_i, \tag{14}$$

where $\mathbf{Y} = \text{diag}(y_1, \ldots, y_l)$ is a diagonal matrix. For notational simplicity, we let $\mathbf{P} = (\mathbf{L}\mathbf{K})^{-1}\mathbf{J}^T\mathbf{Y}$ and simplify (12) to $\boldsymbol{\beta} = \frac{1}{\lambda}\mathbf{P}\boldsymbol{\alpha}$.

Substituting (12)–(14) into $R_h(\boldsymbol{\beta}, \beta_0, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\zeta})$, we have

$$R_d(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2\lambda}\boldsymbol{\alpha}^T\mathbf{Y}\mathbf{J}\mathbf{L}^{-1}\mathbf{J}^T\mathbf{Y}\boldsymbol{\alpha}, \tag{15}$$

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

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

Thus we arrive at the following dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{l} \alpha_i - \frac{1}{2\lambda}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha}, \tag{16}$$

$$\text{subject to } \begin{cases} \sum_{i=1}^{l} y_i \alpha_i = 0 \\ 0 \leq \alpha_i \leq 1 \end{cases} \quad i = 1, 2, \ldots, l, \tag{17}$$

where $\mathbf{Q} = \mathbf{Y}\mathbf{J}\mathbf{L}^{-1}\mathbf{J}^T\mathbf{Y}$ is a matrix with size $l \times l$. As we can see, the optimization problem (16) is the standard SVM formulation where any SVM solver can be used to optimize the solution. The decision function is thus given by

$$f(\mathbf{x}) = \frac{1}{\lambda}\boldsymbol{\alpha}^T\mathbf{P}^T\mathbf{k}(\mathbf{x}) + \beta_0. \tag{18}$$

The dimensionality of the parameter $\boldsymbol{\alpha}$ in the dual problem is l, which is much smaller than the dimensionality of the parameter $\boldsymbol{\alpha}$ in the primal problem. The information from the unlabeled data are contained in the two matrices $\mathbf{P}$ and $\mathbf{Q}$. Both of them can be computed in advance before the optimization is performed. Thus, if more unlabeled examples are provided, it will only increase the computational cost in preprocessing, but not affect the complexity in the optimization procedure. $\mathbf{P}$ is a translating matrix which maps the parameter $\boldsymbol{\alpha}$ with dimensionality l to its corresponding $\boldsymbol{\beta}$ with dimensionality $l + u$. After the solution $\hat{\boldsymbol{\alpha}}$ is optimized, the corresponding solution of $\hat{\boldsymbol{\beta}}$ can be calculated directly. The complexity in solving the problem (16) is $O(l^3)$, therefore, it allows us to explore more possible λ values than using other loss functions.

3. SOLUTION PATH ALGORITHM FOR MODEL SELECTION

The basic idea underlying solution path algorithms comes from continuation methods, which compute the current solution based on an already obtained one. Specifically, we can interpret a solution path algorithm as follows: given a hyperparameter value μ and its corresponding solution $\hat{f}_\mu$, [2] a solution path algorithm seeks to update the solution from $\hat{f}_\mu$ to $\hat{f}_{\mu+s}$ in an efficient way as μ changes to $\mu + s$ by a small value s. The updating formula is often expressed as $\hat{f}_{\mu+s} = \hat{f}_\mu + u(\mu, s)$.

Since this approach has much lower computational demand without the need for training the model multiple times, we can afford to estimate the generalization errors for a much larger set of hyperparameter values in searching for the optimal choice. Solution path is also called regularization path if the path following algorithm is subject to the regularization hyperparameter. Efron et al. [2004] proposed an algorithm called the least angle regression (LARS) algorithm. It can be used to trace the regularization path for linear least square regression regularized with the L_1 norm. An important finding is that the path of the solutions is piecewise linear and hence it is efficient to explore the entire path by monitoring the breakpoints between the linear segments only. Hastie et al. [2004] proposed an algorithm to compute the regularization path for the standard L_2-norm SVC and Zhu et al. [2003] proposed one for the L_1-norm SVC. They are again based on the property that the paths are piecewise linear with respect to the regularization hyperparameter. Rosset [2004] and Wang et al. [2007] proposed path-following algorithms for approximating nonlinear solution paths. In this paper, we apply the solution path algorithm to semi-supervised learning. Specially, we explore the path of solutions for all possible λ values in the problem (16). Since this problem can be transformed to the standard SVM formulation, the

[1] The formulation properties and the optimality conditions in the ϵ-insensitive loss are very similar to that in the hinge loss.

[2] As the hyperparameter value μ changes, the solution estimator $\hat{f}$ will change accordingly. Thus the estimator can be considered as a function of μ. We use $\hat{f}_\mu$ to indicate the dependence on μ.

solution path algorithm proposed by [Hastie et al., 2004] can be applied to the problem (16) with some modifications.

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 for labeled points: [3]

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

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.

3.1 Initialization

The path can start from the solution for any initial value of λ, since the values of $\boldsymbol{\alpha}$ fully determine the sets $\mathcal{L}$, $\mathcal{E}$ and $\mathcal{R}$. However, finding the solution requires solving a quadratic programming (QP) problem. The problem becomes simpler when λ tends to $+\infty$, . The objective degenerates to maximizing $\sum_{i=1}^{l} \alpha_i$ subject to the constraints (17). 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 (11), 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

$$i_+ = \arg\max_{i \in I_+} f(\mathbf{x}_i) = \arg\max_{i \in I_+} \left((\boldsymbol{\alpha}^0)^T \mathbf{P}^T \mathbf{k}(\mathbf{x}_i)\right) \quad (19)$$
$$i_- = \arg\min_{i \in I_-} f(\mathbf{x}_i) = \arg\min_{i \in I_-} \left((\boldsymbol{\alpha}^0)^T \mathbf{P}^T \mathbf{k}(\mathbf{x}_i)\right). \quad (20)$$

[3] $\mathcal{E}$, $\mathcal{L}$ and $\mathcal{R}$ are referred to as the *Elbow*, *Left of the elbow* and *Right of the elbow*, respectively, in Hastie et al. [2004].

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

$$\lambda^0 = \frac{(\boldsymbol{\alpha}^0)^T \mathbf{P}^T \big(\mathbf{k}(\mathbf{x}_{i_+}) - \mathbf{k}(\mathbf{x}_{i_-})\big)}{2} \quad (21)$$
$$\beta_0^0 = \frac{-(\boldsymbol{\alpha}^0)^T \mathbf{P}^T \big(\mathbf{k}(\mathbf{x}_{i_+}) + \mathbf{k}(\mathbf{x}_{i_-})\big)}{(\boldsymbol{\alpha}^0)^T \mathbf{P}^T \big(\mathbf{k}(\mathbf{x}_{i_+}) - \mathbf{k}(\mathbf{x}_{i_-})\big)}. \quad (22)$$

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 (12). 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 QP problem.

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

$$\boldsymbol{\alpha}^0 = \arg\min_{\boldsymbol{\alpha}} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}$$
$$\text{subject to } \begin{cases} \alpha_i = 1 & \forall i \in I_- \\ \alpha_i \in [0, 1] & \forall i \in I_+ \end{cases} \text{ and } \sum_{i \in I_+} \alpha_i = l_-.$$

Moreover, the initial solution $\boldsymbol{\beta}^0$ is given by $\boldsymbol{\beta}^0 = \frac{1}{\lambda}\mathbf{P}\boldsymbol{\alpha}^0$.

In order to find the initial value λ^0, we employ the same strategy as in the balanced case, i.e., finding two points $\mathbf{x}_{i_+}$ and $\mathbf{x}_{i_-}$ such that they both hit the elbow before $\boldsymbol{\alpha}^0$ begins to change. Also, as in the balanced case, $\mathbf{x}_{i_-}$ is decided by (20). There are two possible cases in which $\mathbf{x}_{i_+}$ should be distinctively decided:

(1) 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 (21) and (22).

3.2 Path Following Algorithm

Let us 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 $\mathbf{K}_\mathcal{E} = \big[\mathbf{k}(\mathbf{x}_{\mathcal{E}_1}), \ldots, \mathbf{k}(\mathbf{x}_{\mathcal{E}_m})\big]^T$ 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 $\lambda^{l+1} < \lambda < \lambda^l$, we have the following results:

(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

$$\begin{pmatrix} \alpha_0 \\ \alpha_{\mathcal{E}(1)} \\ \cdot \\ \cdot \\ \cdot \\ \alpha_{\mathcal{E}(m)} \end{pmatrix} = \begin{pmatrix} \alpha_0^l \\ \alpha_{\mathcal{E}(1)}^l \\ \cdot \\ \cdot \\ \cdot \\ \alpha_{\mathcal{E}(m)}^l \end{pmatrix} + (\lambda - \lambda^l)\mathbf{A}_{\mathcal{E}}^{-1}\mathbf{y}^a, \quad (23)$$

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 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 (23). Plugging the updating rule (23) into the regression function (18), we can calculated 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 after the initialization step 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 along the solution path, a set of linear equations is solved with $O(m^3)$ time complexity where m is typically quite small. Moreover, scanning through the labeled examples to evaluate the next move has $O(l^2)$ time complexity. From the experimental results, it often requires $O(l)$ iterations to explore the entire path. Therefore, the overall time complexity of the solution path algorithm is $O(l^3 + l \times m^3)$, which is similar to solve the optimization problem (16) once.

4. EXPERIMENTS

In our experiments, 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 with respect to λ. In order to simplify the initialization step, we select the same number of labeled points from each class. To evaluate the classification performance for different solutions along the path, out-of-sample classification accuracy is measured based on the separate test 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 use the k-nearest neighbors method to compute the graph Laplacian where k is set to 6.

Figure 1 shows the dataset generated from two moons, which totally contains 200 points. When there is only one positive and one negative labeled points, the decision function remains the same along the entire solution path since no event occurs until λ decreases to 0. We randomly label one positive and one negative points, hence, the solution can be obtained right after the initialization. The corresponding decision function has well separated the points between two classes.

We randomly select 70%, 50% and 30% of the points from the two moons data for training while keeping the rest for testing. The solution path algorithm is then executed and the classification accuracies on the test set are shown in Figure 2(a)–(c). In Figure 2(a), since 70% of the points are used for training, the manifold structure is preserved well by the training data. 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. The points from the same class may not belong to the same manifold. 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. 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 different parts leading to very low classification accuracy. As λ decreases, the solutions for different λ are obtained. We can choose a good decision function that generalizes well based on the testing data. When only 30% of the points are used for training, as shown in Figure 2(c), the two moons structure degenerates to a number of small clusters. This case may be more similar to the real data we use in practice. For this case, having more labeled points is essential for identifying the class labels of these clusters. We can see some breakpoints which correspond to the breakpoints on the piecewise linear solution path. When more labeled points are added into the training set, the solution path will contain more breakpoints along the solution path. Since the solution

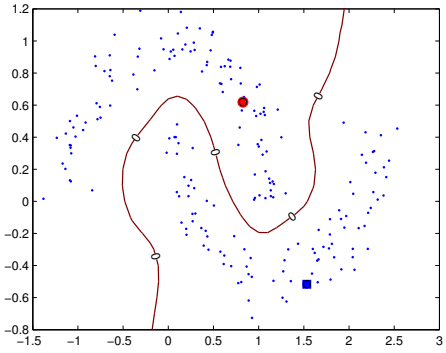

Fig. 1. Two moons dataset generated from two centroid moons. The decision boundary is the solution when only one positive and one negative points are randomly labeled.

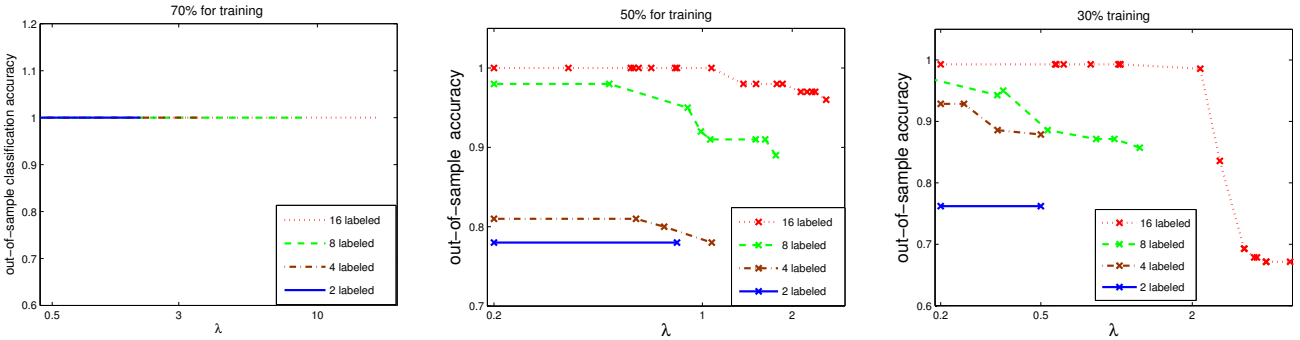

Fig. 2. Out-of-sample classification accuracy along the solution path for different numbers of labeled points. (a) 70% points for training; (b) 50% points for training; (c) 30% points for training. The horizontal axis is in log scale.

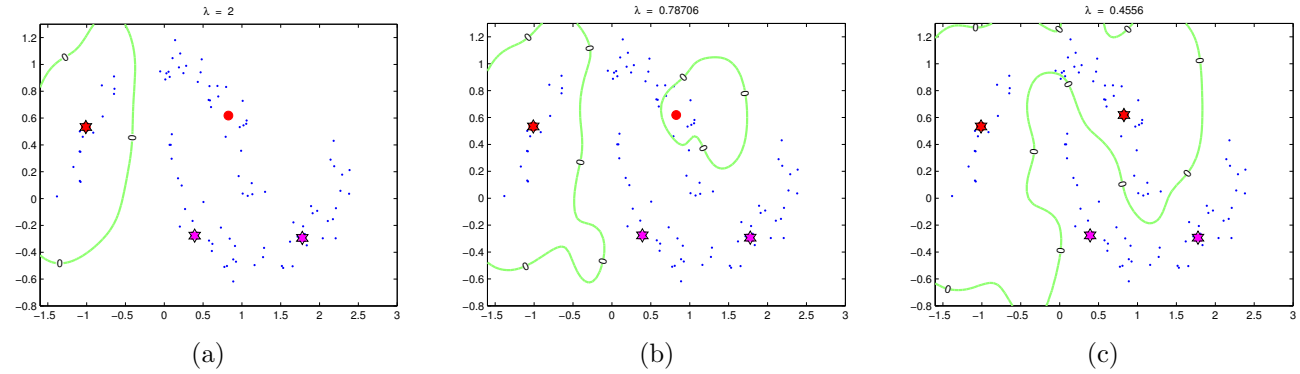

Fig. 3. When 4 examples are labeled and the others are unlabeled, different decision functions are shown for different λ values. (a) $\lambda = 2$; (b) $\lambda = 0.79$; (c) $\lambda = 0.46$.

path algorithm explores the solutions for all λ values, the solution which generalizes best on the test set can be identified easily.

5. CONCLUSIONS

In this paper, we propose a general approach to formulate the semi-supervised learning problem and investigate different implementations of the loss function. We prefer to use the hinge loss in the classification problem, since the optimization in its dual problem has the complexity $O(l^3)$, which is only related to labeled examples. Furthermore, we apply the solution path algorithm to sequentially explore all solutions for all possible λ values when the hinge loss is used. The complexity to explore an entire range of λ values is similar to training the model once, thus, it gives an efficient method to choose the optimal hyperparameter value. The solution path for the ϵ-insensitive loss also extends in piecewise linear manner, thus it can overcome the difficulty in model selection. Due to the space limitation, we do not illustrate the derivation of this loss function and more experimental results.

REFERENCES

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.

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.

R.L. Burden and J. D. Faires. *Numerical Analysis (7th Edition)*. Brooks Cole, 2000.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

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.

S. Rosset. Following curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems 17 (NIPS-04)*, 2004.

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

B. Schölkopf and A.J. Smola. *Learning with kernels*. MIT Press, 2002.

V. N. Vapnik. *Statistical learning theory*. John Wiley & Sons, New York, 1998.

G. Wang, D.Y. Yeung, and F. Lochovsky. A kernel path algorithm for support vector machines. In *Proceedings of the 24th International Conference on Machine Learning (ICML-07)*, 2007.

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.