

# Robust Graph Hyperparameter Learning for Graph Based Semi-supervised Classification

Krikamol Muandet<sup>1</sup>, Sanparith Marukatat<sup>2</sup>, and Cholwich Nattee<sup>1</sup>

<sup>1</sup> School of Information, Computer and Communication Technology  
Sirindhorn International Institute of Technology  
Thammasat University, 131 Tiwanont Rd. Bangkadi  
Pathum Thani, Thailand 12000

<sup>2</sup> National Electronic and Computer Technology Center  
National Science and Technology Development Agency  
111 Thailand Science Park  
Pathum Thani, Thailand 12120

**Abstract.** Graph-based semi-supervised learning has attracted much attention in recent years. Many successful methods rely on graph structure to propagate labels from labeled data to unlabeled data. Although graph structure affects the performance of the system, only few works address its construction problem. In this work, the graph structure is constructed by adjusting the hyperparameter controlling the connection weights between nodes. The idea is to learn the hyperparameters for graphs that yields low leave-one-out prediction error on labeled data while retaining high stability of the prediction on unlabeled data. The problem is solved efficiently using the gradient-based optimization technique. Experimental results indicate that the proposed technique yields improvement of generalization error as well as stability of the labeling function.

**Keywords:** Machine Learning, Graph-Based Semi-Supervised Learning, Stability, Hyperparameter Learning, Model Selection.

## 1 Introduction

Graph-based semi-supervised learning techniques have attracted much attention in recent years. The general idea of these methods is based on building a graph whose nodes are data points connected with edges that encode similarities between them. The labels of unlabeled data are learned on this graph in such a way that nearby data points will have similar labels. Examples of graph-based semi-supervised algorithms include label propagation [1], graph mincuts [2], randomized mincuts [3], and regularization on graphs [4,5].

Although the graph structure is shown to have much influence in graph-based semi-supervised learning [5], few methods are proposed to deal with its construction problem. In [6] the hyperparameters of the graph are learned using the evidence maximization. Carriera-Perpinan and Zemel [7] propose to construct

an ensemble of graphs by combining multiple minimum spanning trees, formed from perturbed version of the data. Wang and Zhange [8] propose a method that is similar to locally linear embedding(LLE) to learn the graph weights by minimizing the reconstruction error of the data points from its neighbors. The most influential works are the algorithms introduced in [9] and [10]. In [9], the authors optimize the hyperparameters using average label entropy on unlabeled data points as a criterion, i.e., they select hyperparameters which yield the most confident labeling on unlabeled data. It is shown by experimental results in [10] that minimizing the entropy on unlabeled data is insufficient. Zhang and Lee instead find the parameters by minimizing leave-one-out prediction error on labeled data.

While there are several strategies to learn graph structure, propagation algorithms on the graph are quite similar. Indeed, most of graph-based semi-supervised techniques rely on an objective function which is a tradeoff between prediction error on labeled data and smoothness of label transition. In [5], the authors have shown that the smoothness term makes the label assignment a “stable” algorithm, thus bounds over the generalization error can be computed. This suggests that a good hyperparameter should not only minimize the leave-one-out prediction error as in [10], but also maximize the stability of the labeling function on the resulting graph.

In this work, we extend the techniques proposed in [9] and [10] by directly incorporating a regularization term which measures the stability of the labeling function on graphs. Involving this regularizer, the hyperparameters of a graph are learned using the gradient-based optimization technique. Using the stability measure, we can achieve a stable labeling function as a consequent of this graph learning method.

## 2 Graph-Based Semi-supervised Learning

Given a data set  $X$  consisting of  $l$  labeled examples  $x_i \in L = \{x_1, x_2, \dots, x_l\}$  and  $u$  unlabeled examples  $x_i \in U = \{x_{l+1}, x_{l+2}, \dots, x_n\}$ , where  $L$  and  $U$  denote sets of labeled and unlabeled examples, respectively. Let  $n = l + u$  be the total number of examples and  $m$  be the dimensionality of the input data. Associated with each labeled example is labels  $y_i \in Y$ . In this work, we consider only a binary classification problem in which  $Y = \{0, 1\}$ .

### 2.1 Graph-Based Semi-supervised Classification

A general task in graph-based semi-supervised classification is to assign labels to unlabeled nodes  $x_i \in U$  through a graph  $G$ . The nodes represent both labeled and unlabeled data points. The symmetric weight matrix  $W$  which encodes the similarities between nodes is typically computed as follow:

$$w_{ij} = \exp\left(-\sum_{d=1}^m (x_{i,d} - x_{j,d})^2 / \sigma_d^2\right), \quad (1)$$

where  $x_{i,d}$  is the  $d$ -th component of  $x_i \in \mathbb{R}^m$  and  $\sigma_d$  is the bandwidth hyper-parameters for the dimension  $d$ . The transition probability matrix between two nodes can be computed by  $P = D^{-1}W$ . The matrix  $D$  is a diagonal matrix whose  $D_{ii} = \sum_{j=1}^n w_{ij}$  is a degree of node  $i$ .

A general strategy is to find a real-valued function  $f$  on graph and then assign labels based on  $f$ . We want  $f$  to take values  $y_i$  on labeled data points and varies smoothly on unlabeled data points according to the weight matrix  $W$ . This criterion leads to the following choice of consistency of  $f$  on graphs [9]:

$$E(f) = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f(i) - f(j))^2 \quad (2)$$

where  $f(i)$  and  $f(j)$  are the function values at node  $i$  and  $j$ , respectively. To find the solution of the above equation, it is convenient to split the matrices  $W$  and  $P$  into four parts, starting with labeled data points followed by unlabeled data points:

$$W = \begin{pmatrix} W_{LL} & W_{LU} \\ W_{UL} & W_{UU} \end{pmatrix}, P = \begin{pmatrix} P_{LL} & P_{LU} \\ P_{UL} & P_{UU} \end{pmatrix} .$$

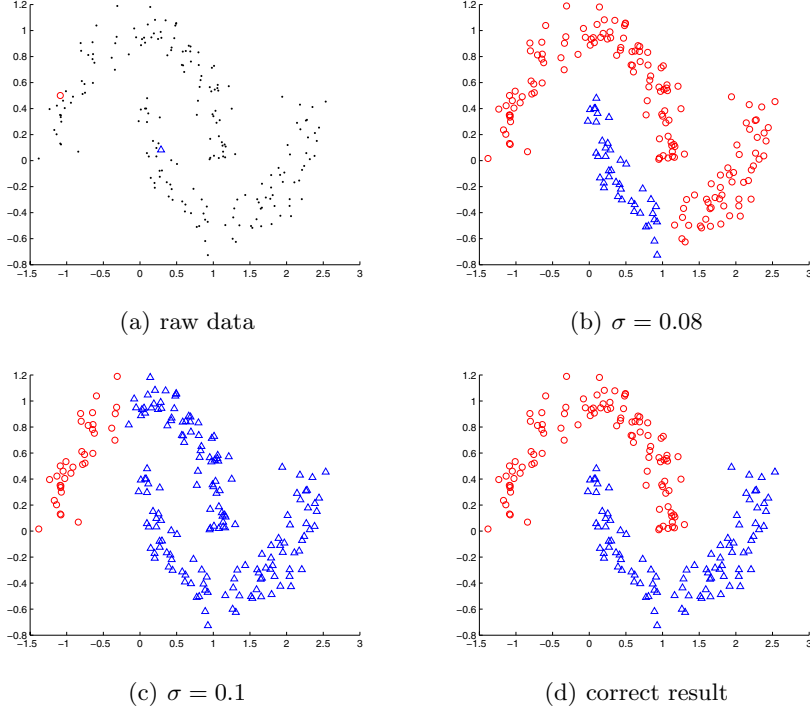
It is not difficult to show that the function that yields a minimum value of (2) is given by:

$$f_U = (D_{UU} - W_{UU})^{-1} W_{UL} f_L = (I - P_{UU})^{-1} P_{UL} f_L . \quad (3)$$

This function is called the *soft-label* function since its value does not directly specify the class to which the example belongs, but it gives a possibility of being in each class. The most obvious method to transform soft-label to hard-label is by thresholding, e.g., classify  $x$  as being in class 1 if  $f(x) > 0.5$ , and class 0 otherwise. This method generally works well when the classes are well-separated. However, this is not the case in many practical applications. In such cases, using simple thresholding may result in an unbalanced classification.

Another method to transform the soft-label to hard-label is *Class Mass Normalization*(CMN) introduced in [9]. The class distribution of the data is adjusted to match the class priors, that can be obtained from the labeled examples. For example, if the prior class proportion of class 1 and 0 is  $p$  and  $1 - p$ , then an unlabeled example  $x$  is classified as class 1 iff  $p \cdot (f(x) / \sum_i f(x_i)) > (1 - p) \cdot ((1 - f(x)) / \sum_i (1 - f(x_i)))$ . This method works well when we have sufficient labeled examples to determine the class prior that accurately represents the true class distribution.

As mentioned previously, graph structures do influence the results of classification. Figure 1 illustrates examples of such a situation, which shows two-dimensional data set called **twomoon**. In Fig.1(a) two labeled examples, circle and triangle, are drawn along with unlabeled examples depicted as solid dots. As we can see in Fig.1(b) and (c), a small change in the value of  $\sigma$  used to construct graphs results in significantly different classification results. Moreover the optimal  $\sigma$ , which yields a correct result as shown in Fig.1(d), is difficult to find in practice. Consequently, we need an efficient graph learning algorithm to solve this problem as it plays vital role in graph-based semi-supervised classification.



**Fig. 1.** The results of classification using different values of  $\sigma$  to construct graphs; (a) the raw data with 2 labeled examples (b), the result when  $\sigma = 0.08$ , (c) the result when  $\sigma = 0.1$ , (d) the result of correct classification

## 2.2 Graph Learning Algorithms

As mentioned previously, this work extends two previous works namely Minimum Entropy (MinEnt) [9] and Leave-one-out Hyperparameter Learning (LOOHL) [10]. This section gives a brief review of these two techniques successively.

In MinEnt, the weight matrix is obtained by learning all  $\sigma_d$  in (1) from both labeled and unlabeled data. An *average label entropy*,  $H(f)$ , is used as a heuristic criterion for parameter learning, which is defined as

$$H(f) = \frac{1}{u} \sum_{i=l+1}^n H_i(f(i)) , \quad (4)$$

where  $H_i(f(i)) = -f(i) \log_2 f(i) - (1 - f(i)) \log_2 (1 - f(i))$  is the entropy of the soft-label at the individual unlabeled data point  $i$ . The concept of this algorithm is that a good weight  $W$  should result in a confident labeling, i.e., the average label entropy is low when soft-label at each unlabeled data point is close to the boundary of the labeling function.

Instead of considering label entropy at unlabeled data, the LOOHL learns the hyperparameters  $\sigma_d$  of the graph by minimizing the leave-one-out prediction error on all labeled examples. The objective function is:

$$Q = \sum_{t=1}^l h_t(f_{U,1}^t), \quad (5)$$

where  $h_t(x)$  is the cost function for labeled example  $x$  that penalizes the difference between the predicted soft-label and the true label of the instance. An efficient algorithm to gradient computation is proposed based on matrix inversion lemma and careful precomputation. Both MinEnt and LOOHL also mention a case of degenerative graphs in which  $\sigma_d$  grow either too large or too small. Thus a simple regularizer is used to prevent this problem. Moreover,  $P$  is replaced by  $\tilde{P} = \varepsilon \mathcal{U} + (1 - \varepsilon)P$ , where  $\mathcal{U}_{ij} = 1/n$ , to prevent the numerical problem.

### 3 Robust Graph Hyperparameter Learning

In this work, we propose a hyperparameter learning method which considers the leave-one-out prediction error on labeled examples and the stability cost of the labeling function on unlabeled examples. Using the result of (3), at each iteration  $t = 1, \dots, l$  of leave-one-out, the soft-label function can be determined by  $f_U^t = (I - \tilde{P}_{UU}^t)^{-1} \tilde{P}_{UL}^t f_L^t$  where  $f_L^t = (f_1, \dots, f_{t-1}, f_{t+1}, \dots, f_l)^T$ . The modified transition matrix is then defined as:

$$P_{UU}^t = \begin{pmatrix} p_{t,t} & p_{t,l+1} & \cdots & p_{t,n} \\ p_{l+1,t} & & & \\ \vdots & & P_{UU} & \\ p_{n,t} & & & \end{pmatrix}, P_{UL}^t = \begin{pmatrix} p_{t,1} & \cdots & p_{t,t-1} & p_{t,t+1} & \cdots & p_{t,l} \\ p_{l+1,1} & \cdots & p_{l+1,t-1} & p_{l+1,t+1} & \cdots & p_{l+1,l} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{n,1} & \cdots & p_{n,t-1} & p_{n,t+1} & \cdots & p_{n,l} \end{pmatrix}$$

The first row and column in  $P_{UU}^t$  and the missing column in  $P_{UL}^t$  correspond to a labeled example  $x_t$  used to evaluate prediction cost  $h_t(x_t)$ . Let  $e_k$  be a vector whose element is 1 at  $k$ -th position and 0 elsewhere. The soft label of this example is  $e_1^T f_U^t$ , the first element of  $f_U^t$ . Let  $f_{U,k}^t$  for  $1 \leq k \leq u+1$  define the soft-label of  $x_t$  together with other unlabeled examples. Then the objective function to be minimized can be written as:

$$R = \gamma \sum_{t=1}^l h_t(f_{U,1}^t) + (1 - \gamma) \sum_{k=2}^{u+1} s(f_{U,k}) \quad (6)$$

where  $\gamma$  balances accuracy and stability of the soft-label function. The  $f_{U,k} = (f_{U,k}^1, f_{U,k}^2, \dots, f_{U,k}^l)^T$  is the vector whose element is the soft-label of  $k$ -th unlabeled example at each  $t = 1, \dots, l$ . The cost function  $h_t(f_{U,1}^t)$  is defined as in (5). Possible choices for  $h_t(x_t)$ , if  $y_t = 1$ , are  $1 - x_t$ ,  $(1 - x_t)^2$ , and  $-\log x_t$ . If  $y_t = 0$ , the cost function can be  $x, x^a$ , and  $a^{x_t-1}$ , for example.

In the second term of (6), the cost function  $s(f_{U,k})$  measures the variation of the soft-label over elements of  $f_{U,k}$ . Thus the sum of this quantity over all

unlabeled examples can be used to determine how stable the labeling function is. A possible choice of  $s(f_{U,k})$  is the variance function. Although this function captures the variation of soft-label, it does not take into account the confidence of labeling function. For example, if most of soft-label values lies near 0.5, we assign labels to unlabeled data with low confidence. Instead, the stability cost function used in this work is:

$$s(f_{U,k}) = -\bar{f}_{U,k} \log_2 \bar{f}_{U,k} - (1 - \bar{f}_{U,k}) \log_2 (1 - \bar{f}_{U,k}) \quad (7)$$

which is the entropy of an average of all elements of  $f_{U,k}$ . This function is chosen because it penalizes the high variation as well as the deviation from the function boundary. By minimizing this cost, labels are assigned with high confidence using the stable function.

To minimize  $R$ , we use the gradient-based optimization technique. The gradient is:

$$\begin{aligned} \partial R / \partial \sigma_d = & \gamma \left( \sum_{t=1}^l h'_t(f_{U,1}^t) e_1^T (\partial f_U^t / \partial \sigma_d) \right) \\ & + (1 - \gamma) \left( \sum_{k=2}^{u+1} s'(\bar{f}_{U,k}) \sum_{t=1}^l e_k^T (\partial f_U^t / \partial \sigma_d) \right) \end{aligned} \quad (8)$$

where  $\partial f_U^t / \partial \sigma_d = (I - \tilde{P}_{UU}^t)^{-1} (\partial \tilde{P}_{UU}^t / \partial \sigma_d \cdot f_U^t + \partial \tilde{P}_{UL}^t / \partial \sigma_d \cdot f_L^t)$  using matrix property  $dX^{-1} = -X^{-1}(dX)X^{-1}$ . The  $s'(\bar{f}_{U,k}) = \log_2((1 - \bar{f}_{U,k})/\bar{f}_{U,k})$ . Denoting  $(\beta^t)^T = (h'_t(f_{U,1}^t) e_1^T (I - \tilde{P}_{UU}^t)^{-1})$  and  $(\xi_k^t)^T = e_k^T (I - \tilde{P}_{UU}^t)^{-1}$  and noting that  $\tilde{P} = \varepsilon \mathcal{U} + (1 - \varepsilon)P$ . With careful re-arrangement and substitution, the gradient can be written as

$$\begin{aligned} \partial R / \partial \sigma_d = & \mu \sum_{t=1}^l (\beta^t)^T \left( \frac{\partial P_{UU}^t}{\partial \sigma_d} \cdot f_U^t + \frac{\partial P_{UL}^t}{\partial \sigma_d} \cdot f_L^t \right) \\ & + \mu' \left( \sum_{k=2}^{u+1} s'(\bar{f}_{U,k}) \sum_{t=1}^l (\xi_k^t)^T \left( \frac{\partial P_{UU}^t}{\partial \sigma_d} \cdot f_U^t + \frac{\partial P_{UL}^t}{\partial \sigma_d} \cdot f_L^t \right) \right) \end{aligned} \quad (9)$$

where  $\mu = \gamma(1 - \varepsilon)$  and  $\mu' = (1 - \gamma)(1 - \varepsilon)$ . From the relation that  $P(i, j) = w(i, j) / \sum_{j=1}^n w(i, j)$ , the partial derivative of transition probability  $P(i, j)$  can be computed by

$$\frac{\partial P(i, j)}{\partial \sigma_d} = (\theta_i)^{-1} \left( \frac{\partial w(i, j)}{\partial \sigma_d} - P(i, j) \cdot \sum_{j=1}^n \frac{\partial w(i, j)}{\partial \sigma_d} \right), \quad (10)$$

where  $\theta_i = \sum_j w(i, j)$  and  $\partial w(i, j) / \partial \sigma_d = 2w(i, j)(x_{i,d} - x_{j,d})^2 / \sigma_d^3$  by (1). Followed from [10], the computational complexity of the algorithm is very expensive. A possible way to reduce the cost is by applying matrix inversion lemma in computation of  $(I - \tilde{P}_{UU}^t)^{-1}$  for each iteration  $t$ . Moreover, the computational

cost can be further reduced by using an efficient precomputation. Splitting the gradient computation into two parts,  $\partial R/\partial\sigma_d = \mu \cdot \partial H/\partial\sigma_d + \mu' \cdot \partial S/\partial\sigma_d$  and substituting (10) into (9), each gradient term can be derived separately as:

$$\begin{aligned} \frac{\partial H}{\partial\sigma_d} &= \sum_{t=1}^l \sum_{i=1}^{u+1} \frac{\beta_i^t}{\theta_i^t} \left( \sum_{j=1}^{u+1} \frac{\partial w_{UU}^t(i, j)}{\partial\sigma_d} f_{U, j}^t + \sum_{j=1}^{l-1} \frac{\partial w_{UL}^t(i, j)}{\partial\sigma_d} f_{L, j}^t \right. \\ &\quad \left. - \sum_{k=1}^n \frac{\partial w^t(i, k)}{\partial\sigma_d} \left( \sum_{j=1}^{u+1} P_{UU}^t(i, j) \cdot f_{U, j}^t + \sum_{j=1}^{l-1} P_{UL}^t(i, j) \cdot f_{L, j}^t \right) \right) \\ &= \sum_{i=1}^n \sum_{j=1}^n \alpha_{ij} \frac{\partial w(i, j)}{\partial\sigma_d} \end{aligned} \quad (11)$$

$$\begin{aligned} \frac{\partial S}{\partial\sigma_d} &= \sum_{k=2}^{u+1} s'(\bar{f}_{U, k}) \sum_{t=1}^l \sum_{i=1}^{u+1} \frac{\xi_{k, i}^t}{\theta_i^t} \left( \sum_{j=1}^{u+1} \frac{\partial w_{UU}^t(i, j)}{\partial\sigma_d} f_{U, j}^t + \sum_{j=1}^{l-1} \frac{\partial w_{UL}^t(i, j)}{\partial\sigma_d} f_{L, j}^t \right. \\ &\quad \left. - \sum_{p=1}^n \frac{\partial w^t(i, p)}{\partial\sigma_d} \left( \sum_{j=1}^{u+1} P_{UU}^t(i, j) \cdot f_{U, j}^t + \sum_{j=1}^{l-1} P_{UL}^t(i, j) \cdot f_{L, j}^t \right) \right) \\ &= \sum_{k=2}^{u+1} s'(\bar{f}_{U, k}) \sum_{i=1}^n \sum_{j=1}^n \rho_{ij}^k \frac{\partial w(i, j)}{\partial\sigma_d} . \end{aligned} \quad (12)$$

Let  $\delta(\cdot)$  be Kronecker delta. The  $\rho_{ij}^k$  can be computed by the following formula:

$$\rho_{ij}^k = \begin{cases} (\theta_i)^{-1} \sum_{t=1}^l \xi_{k, i-l+1}^t (f_{U, j-l+1}^t - \sum_{k=l+1}^n p_{ik} f_{U, k-l+1}^t) & \text{if } i > l, j > l, \\ -p_{it} f_{U, 1}^t - \sum_{k=1: k \neq t}^l p_{ik} f_k & \\ (\theta_i)^{-1} \sum_{t=1}^l \xi_{k, i-l+1}^t (f_{U, 1}^t \cdot \delta(j=t) + f_j \cdot \delta(j \neq t)) & \text{if } i > l, j \leq l, \\ -p_{it} f_{U, 1}^t - \sum_{k=l+1}^n p_{ik} f_{U, k-l+1}^t - \sum_{k=1: k \neq t}^l p_{ik} f_k & \\ (\theta_i)^{-1} \xi_{k, 1}^i (f_{U, j-l+1}^i - \sum_{k=l+1}^n p_{ik} f_{U, k-l+1}^i - \sum_{k=1}^l p_{ik} f_k) & \text{if } i \leq l, j > l, \\ (\theta_i)^{-1} \xi_{k, 1}^i (f_j - \sum_{k=l+1}^n p_{ik} f_{U, k-l+1}^i - \sum_{k=1}^l p_{ik} f_k) & \text{if } i \leq l, j \leq l. \end{cases} \quad (13)$$

To compute  $\alpha_{ij}$ , substitute  $\xi_{k, p}^t$  by  $\beta_p^t$  without subscript  $k$ . Since computing the gradient of stability term requires much computation, a possible method to reduce this computational cost is to use only a subset of unlabeled examples to compute the gradient. We call this algorithm **Robust Graph Hyperparameter Learning (RobustHL)**. The graph constructed by this technique is called *robust graph*. Algorithm 1 shows the efficient gradient computation of RobustHL.

## 4 Experimental Results

In the experiment, we compare the prediction accuracy of four model selection methods, namely 5-fold Cross Validation, MinEnt, LOOHL, and RobustHL.

---

**Algorithm 1.** An efficient gradient update algorithm for RobustHL

---

```

initialize gradient  $g \leftarrow (0, \dots, 0)^T \in \mathbb{R}^m$ 
compute  $\alpha_{ij}$ ,  $\rho_{ij}^k$ , and  $s'(\bar{f}_{U,k})$ 
for  $i, j = 1, \dots, n$  do
  for each dimension  $d = 1, \dots, m$  do
     $g_d \leftarrow g_d + (\alpha_{ij} + \sum_{k=2}^{u+1} s'(\bar{f}_{U,k}) \cdot \rho_{ij}^k) \cdot \partial w(i, j) / \partial \sigma_d$ 
  end for
end for

```

---

These techniques are used to learn the hyperparameters of the graph. Then the accuracy of label assignment is computed. Instead of comparing the performance of our technique with other semi-supervised techniques, we pay more attention to the comparison of different model selection techniques. Furthermore, characteristics of the robust graph learned using the proposed stability measure are investigated. The benchmark used in the experiment consists of six data sets as shown in Table 1. The benchmark is well-known and is widely used to assess the power of various semi-supervised learning techniques. They can be categorized into two groups. The first group includes `g241c`, `g241d`, and `Digit1` which are artificially created in order for certain assumptions to hold. The data sets in second group are derived from real data to indicate the performance of algorithms in real world applications. It includes `USPS`, `COIL2`, and `BCI`. The detailed explanation of each data set can be found in [11].

In the first experiment, we assess the performance of algorithms when there are 10 labeled examples for each data set. To see the effect of soft-label transformation, both thresholding and CMN are applied in each algorithm. To make the results from the experiment robust and independent of properties of the chosen data points, the algorithms are tested on randomly selected 12 subsets of labeled examples and report the average accuracy as well as its confidence interval.

Throughout the experiment, the smoothing factor  $\varepsilon$  is fixed at 0.001 for all model selection methods. The parameter  $\gamma$  is selected from  $\{0.2, 0.4, 0.6, 0.8\}$ . For 5-fold CV, the parameter  $\sigma_d$  is selected from  $\{\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8\}$  which is the same for all dimensions. The final objective function of other three methods is:

$$C_1 \times Loss + C_2 \times \sum_d (1/\sigma_d - 1/\tilde{\sigma}_d)^2 / m ,$$

where *Loss* is the original cost function of MinEnt, LOOHL, and RobustHL. The second term in the above equation is introduced to regularize the value of  $\sigma_d$  by preventing the value from approaching either 0 or  $\infty$ . For each ratio  $C_1 : C_2$  in  $\{10^{-2}, 10^{-1}, 1, 10, 100\}$ , the  $\tilde{\sigma}_d$  is taken from  $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$  to avoid the local minima. All input vectors are normalized to have length 1. The result of this experiment is shown in Table 2.

In the second experiment, the data set is split into training set and test set. In training set, the number of labeled examples varies across 10%, 30%, 50%, 70%, and 90% of this set. Then the labels are assigned to unlabeled examples in both training set and test set. For each particular set of labeled examples, the

**Table 1.** Basic properties of benchmark data sets

Data Set	Dimension	Points	Comment
g241c	241	1500	artificial
g241d	241	1500	artificial
Digit1	241	1500	artificial
USPS	241	1500	imbalance
COIL <sub>2</sub>	241	1500	
BCI	117	400	

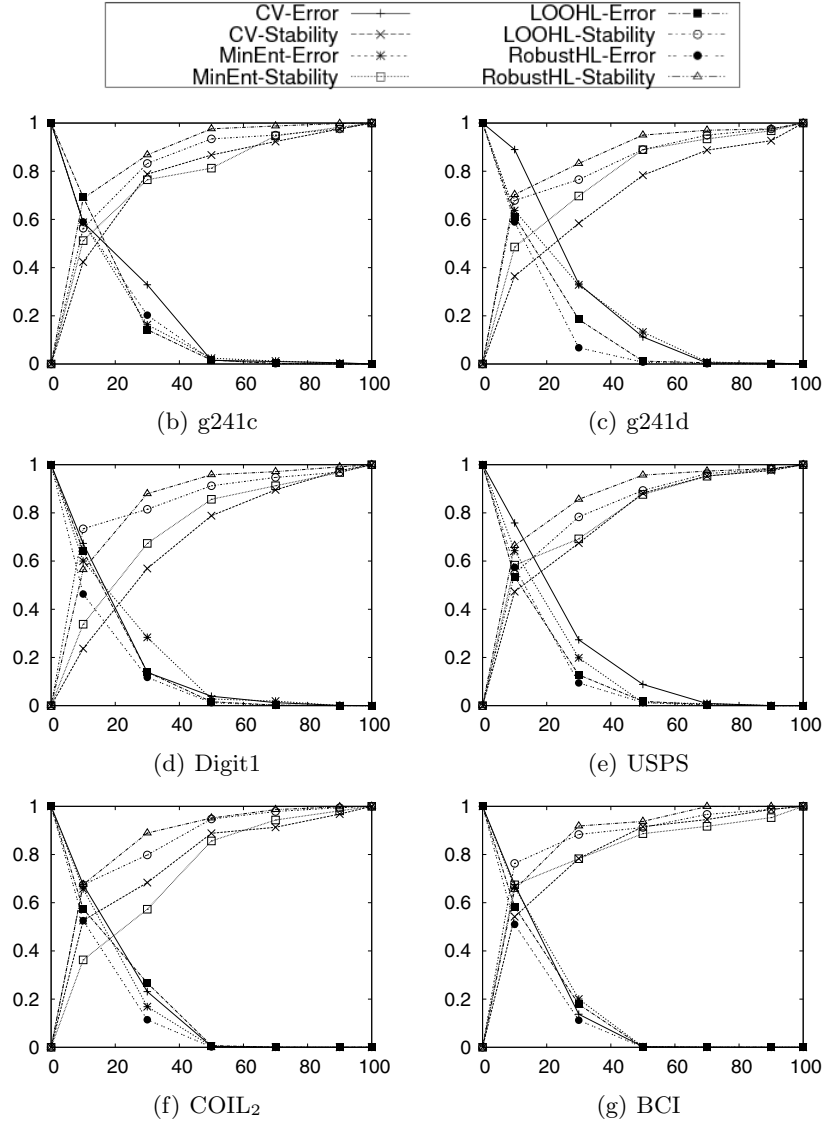
**Table 2.** Classification accuracy(%) and confidence interval with 10 labeled examples

Method	g241c	g241d	Digit1	USPS	COIL <sub>2</sub>	BCI
CV+Thresh	50.90±3.0	51.64±4.5	51.31±5.2	80.01±.05	55.78±5.4	50.28±1.1
CV+CMN	52.14±4.2	52.61±5.3	57.04±13.7	80.02±.04	58.11±6.9	49.94±.75
MinEnt+Thresh	55.32±6.1	56.18±6.8	59.91±4.2	75.24±7.7	56.87±9.0	54.74±4.4
MinEnt+CMN	57.28±7.7	58.15±7.0	63.81±6.5	<b>82.86±4.6</b>	58.94±10.3	58.08±5.9
LOOHL+Thresh	76.27±4.0	57.04±5.0	63.39±3.0	80.27±5.2	68.47±5.9	69.10±5.4
LOOHL+CMN	73.28±4.8	<b>59.18±6.9</b>	63.9±7.7	81.60±4.1	69.74±7.8	69.40±4.9
RobustHL+Thresh	<b>76.50±2.5</b>	56.44±3.1	62.55±2.1	78.22±4.5	67.32±5.0	71.21±3.5
RobustHL+CMN	72.51±2.8	58.18±4.8	<b>65.03±7.1</b>	81.25±3.4	<b>71.25±5.6</b>	<b>72.12±2.8</b>

classification error and stability on test set are calculated. In this experiment, the same model selection methods as previous experiment are used. Figure 2 shows the plot of classification error and stability of four model selection methods when the number of labeled examples in each data set increases.

From the result in Table 2, we can make the following conclusions.

1. MinEnt, LOOHL, and RobustHL graph learning methods perform better than typical CV in all data sets except MinEnt+Thresh and RobustHL+Thresh in USPS. In this data set, CV+Thresh achieves higher accuracy than these two methods.
2. LOOHL and RobustHL in general achieve higher accuracy than MinEnt in all data sets except USPS. In this data set, MinEnt+CMN achieves higher accuracy than both LOOHL and RobustHL.
3. For LOOHL and RobustHL, both methods have competitive results. The accuracy achieved by these two techniques differs no more than 2%. Indeed, this is reasonable because the objective function of LOOHL and RobustHL is similar. The only difference between these methods is the stability measure introduced in RobustHL.
4. As a result of stability measure introduced in RobustHL, the confidence interval of RobustHL is tighter than that of LOOHL. This result is supported by the work of Belkin et al. [5] that the stability also affects the confidence of the classification. This means the training accuracy of RobustHL should be closer to the true accuracy than LOOHL.



**Fig. 2.** The classification error and stability of the labeling function obtained from different model selection techniques against increasing number of labeled examples(%)

It is interesting to mention the following points about the stability of each model selection technique shown in Fig. 2.

1. In some data sets, the MinEnt method has lower stability than CV, which generally has lowest stability. As a result, it implies that the average label entropy may not be a good criterion for learning a robust graph. In [10], the

authors also state that MinEnt may fail to learn a good graph. This means the the average label entropy may be insufficient in graph learning.

2. In general, LOOHL method has higher stability than CV and MinEnt. Mentioned in [5] and [12], the stability of the learner is defined in term of leave-one-out scenario. As a result, the leave-one-out model selection scheme usually results in a stable function. Therefore, it is not surprising that the LOOHL can achieve such high stability.
3. For RobustHL, introducing explicit regularized term of stability in addition to the leave-one-out prediction loss improves the stability of labeling function. As can be seen in Fig. 2, the stability on labeling function of RobustHL is the highest one. Therefore, we conclude that using the stability measure on unlabeled examples as a criterion in graph learning improves the stability of the labeling function, that consequently yields an improvement in classification accuracy.

In general, the labels of unlabeled examples can be predicted with higher confidence when more label information is available. The predicted label of unlabeled examples are determined by their soft-label using simple thresholding or CMN. However, this soft-label may dramatically change and result in a different prediction when we add more labeled examples.

The stability of the soft-label on unlabeled examples can be used as a criterion to determine the sufficient number of labeled examples. As can be seen in Fig. 2, the stability of soft-label on unlabeled examples increases as more labeled examples are added. However, at particular point, it will increase very slowly, i.e., adding more labeled examples will no longer affect the soft-label of the unlabeled examples. In other words, the prediction result is unlikely to change. At this point, we can stop providing label information since doing so will no longer improve the accuracy of the classification.

According to the experimental results, RobustHL generally achieves higher stability and lower classification error than other three techniques. This means that the stability of the labeling function of RobustHL can be used as the criteria to determine the sufficient number of labeled examples very effectively.

## 5 Conclusion

In this work, the new hyperparameter learning in graph-based semi-supervised classification is proposed. The parameters is learned by minimizing leave-one-out prediction error on labeled examples while retaining high stability on unlabeled examples. The problem can be solved efficiently using gradient-based optimization. Encouraging results show that the proposed technique has high performance compared to existing model selection techniques in graph-based semi-supervised learning.

**Acknowledgments.** This work is supported in part by the Young Scientist and Technologist Programme or YSTP (SIIT-NSTDA:S1Y48/F-002) of the National

Science and Technology Development Agency, Thailand. Special thank to Xinhua Zhang for providing the useful codes for testing.

## References

1. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University (2002)
2. Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. In: Brodley, C.E., Danyluk, A.P., Brodley, C.E., Danyluk, A.P. (eds.) ICML, pp. 19–26. Morgan Kaufmann, San Francisco (2001)
3. Blum, A., La, J., Rwebangira, M., Reddy, R.: Semi-supervised learning using randomized mincuts (2004)
4. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Thrun, S., Saul, L.K., Schölkopf, B. (eds.) NIPS 2003, vol. 16, pp. 321–328. MIT Press, Cambridge (2004)
5. Belkin, M., Matveeva, I., Niyogi, P.: Regularization and semi-supervised learning on large graphs. In: Shawe-Taylor, J., Singer, Y. (eds.) COLT 2004. LNCS, vol. 3120, pp. 624–638. Springer, Heidelberg (2004)
6. Kapoor, A., Qi, Y.A., Ahn, H., Picard, R.: Hyperparameter and kernel learning for graph based semi-supervised classification. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems 18, pp. 627–634. MIT Press, Cambridge (2006)
7. Carreira-Perpiñán, M.Á., Zemel, R.S.: Proximity graphs for clustering and manifold learning. In: NIPS (2004)
8. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. In: ICML 2006 (2006)
9. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 20th International Conference on Machine Learning (ICML 2003) (2003)
10. Zhang, X., Lee, W.S.: Hyperparameter learning for graph based semi-supervised learning algorithms. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 19, pp. 1585–1592. MIT Press, Cambridge (2007)
11. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge (2006)
12. Bousquet, O., Elisseeff, A.: Algorithmic stability and generalization performance. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) NIPS, pp. 196–202. MIT Press, Cambridge (2000)