Front. Electr. Electron. Eng. China 2008, 3(1): 1–9
DOI 10.1007/s11460-008-0001-y

RESEARCH ARTICLE

Yangqiu SONG, Jianguo LEE, Changshui ZHANG, Shiming XIANG

Semi-supervised Gaussian random field transduction and induction

Abstract This paper proposes a semi-supervised inductive algorithm adopting a Gaussian random field (GRF) and Gaussian process. We introduce the prior based on graph regularization. This regularization term measures the p-smoothness over the graph. A new conditional probability called the extended Bernoulli model (EBM) is also proposed. EBM generalizes the logistic regression to the semi-supervised case, and especially, it can naturally represent the margin. In the training phase, a novel solution is given to the discrete regularization framework defined on the graphs. For the new test data, we present the prediction formulation, and explain how the margin model affects the classification boundary. A hyper-parameter estimation method is also developed. Experimental results show that our method is competitive with the existing semi-supervised inductive and transductive methods.

Keywords Gaussian process, Gaussian random field, semi-supervised learning, graph based learning

1 Introduction

In the real world, many applications need a computer to recognize data from a partially labeled data set. For example, we can obtain large numbers of documents from the Internet, or acquire many images from a digital camera. However, it is costly to label all of them. Consequently, semi-supervised learning becomes an important topic to deal with the partially labeled problems [1]. While transductive methods only work on the observed labeled and unlabeled training data, inductive methods can handle the unseen data that are not in the training set. Therefore, semi-supervised induction has been attracting more attention recently.

A problem of semi-supervised learning is how to use a small number of labeled data and lots of unlabeled data to achieve a result as good as the supervised case. One efficient way is to use the graph or manifold regularization methods [2–5]. It is intuitive that every point should be similar to the points in its local neighborhood. Thus, the intrinsic structure of the data should be exploited by assuming that the data reside on the low dimensional manifold. Therefore, regularizing the learned functions to make them smooth on the manifold or on the graph constructed by the data is helpful. Although the successful transduction on graph (TOG) method [3,5] has realized the semi-supervised learning on data manifold, there are still two problems: 1) TOG is transductive and does not have any formulation to deal with the unseen data; 2) It regards the noise between labels and latent variables as Gaussian, which is not true for the classification problem.

In this paper, we propose a new algorithm that uses graph regularization-based Gaussian random field (GRF) to address the partially labeled data. To train a GRF, we estimate the latent variables from the posterior probability employing Laplace approximation. For the new test data, we present the approximated prediction formulation of the Gaussian process [6], and explain how the proposed model affects the final classification boundary.

This paper is organized as follows. In Sect. 2, we introduce the models used in our method and present the algorithm framework. Section 3 shows the experimental results on both test and real data sets. Finally, we conclude the paper in Sect. 4.

Received April 15, 2007; accepted August 13, 2007

Yangqiu SONG, Jianguo LEE, Changshui ZHANG, Shiming XIANG
State Key Laboratory on Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China
E-mail: songyq99@mail.tsinghua.edu.cn

2 Semi-supervised Gaussian process

We denote the input data point as a feature vector $\boldsymbol{x}_i$ ($i = 1, 2, \ldots, N$), and $\boldsymbol{X}_N \overset{\Delta}{=} \{\boldsymbol{x}_i\}_{i=1}^{N}$ is the observed data

set including both labeled and unlabeled data. For the semi-supervised problem, we attempt to extend the labels of the labeled data to the unlabeled data, whose labels are set to zero initially. The label is given by t_i ($i = 1, 2, \ldots, N$), and the label set is $\boldsymbol{t}_N = (t_1, t_2, \ldots, t_N)^{\mathrm{T}}$. We also denote the training data set as $D = \{\boldsymbol{x}_i, t_i\}_{i=1}^{N}$. The latent variable y_i is used to generate the process $\boldsymbol{x}_i \to y_i \to t_i$. Moreover, we impose some new noise functions on the process $y_i \to t_i$, which is more appropriate for the semi-supervised problem. $y_i = y(\boldsymbol{x}_i)$ is a function of $\boldsymbol{x}_i$, and $\boldsymbol{y}_N = (y_1, y_2, \ldots, y_N)^{\mathrm{T}}$ is the latent variable vector of input data. The hyper-parameters of the exponential weights are denoted as a vector Θ. For the inductive problem, we want to estimate the label t_{N+1} of a new point $\boldsymbol{x}_{N+1}$.

To train a GRF, we need to estimate the hyper-parameters, labels of unlabeled data and the latent variables. Therefore, the posterior $P(\boldsymbol{y}_N \mid \boldsymbol{t}_N, \Theta)$ of the latent variables is taken as the estimate of $\boldsymbol{y}_N$:

$$P(\boldsymbol{y}_N|\boldsymbol{t}_N,\Theta) = \frac{P(\boldsymbol{y}_N,\boldsymbol{t}_N|\Theta)}{P(\boldsymbol{t}_N)} = \frac{P(\boldsymbol{t}_N|\boldsymbol{y}_N)P(\boldsymbol{y}_N|\Theta)}{P(\boldsymbol{t}_N)}, \quad (1)$$

where $P(\boldsymbol{y}_N \mid \Theta)$ is the prior of the latent variables, and $P(\boldsymbol{t}_N \mid \boldsymbol{y}_N)$ is the likelihood. By taking the negative logarithm, we define the following objective function:

$$\Psi(\boldsymbol{y}_N) = -\lg P(\boldsymbol{t}_N|\boldsymbol{y}_N) - \lg P(\boldsymbol{y}_N|\Theta). \quad (2)$$

$P(\boldsymbol{t}_N)$ is omitted since it is a constant unrelated to $\boldsymbol{y}_N$. Note that there are two terms that affect the estimation in Eq. (2). Thus, we first present the definitions of both prior and the likelihood of the latent variables given in the hyper-parameters, following that the estimate procedure is shown.

2.1 Graph-based data-dependent prior

The graph-based methods consider a weighted undirected graph $G = (V, E)$ established on the data points in some feature space. The node V corresponds to the points, and weights on the edge E are functions of the pair-wise connected nodes. Here, we adopt the graph regularization based data-dependent prior, which is proposed in Ref. [4]. The regularization term restricts a function defined on the graph sufficiently smooth.

The prior probability is given by:

$$P(\boldsymbol{y}_N|\Theta) = \frac{1}{Z_r}\exp\left\{-\frac{S_p(\boldsymbol{y}_N)}{\mu}\right\}, \quad (3)$$

where Z_r is the normalization constant, and the smoothness function $S_p(\boldsymbol{y}_N)$ is

$$S_p(\boldsymbol{y}_N) = \frac{1}{p}\sum\nolimits_i \left(\sqrt{\sum_j W_{ij}\left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right)^2}\right)^p, \quad (4)$$

where $D_{ii} = \sum\limits_j W_{ij}$. W_{ij} is the weight function associated with the edges on the graph, which satisfies $W_{ij} \geqslant 0$ and $W_{ij} = W_{ji}$. It can be viewed as a symmetric similarity measure between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. A simple example of W_{ij} is $W_{ij} = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2)$. If different dimensions have different weights $W_{ij} = \exp\left(-\sum\limits_{k=1}^{d}\left(\boldsymbol{x}_i^k - \boldsymbol{x}_j^k\right)^2/2\sigma_k^2\right)$, we set the hyper-parameters as $\Theta = \{\sigma_1, \ldots, \sigma_k, \ldots, \sigma_d\}$. The function $S_p(\boldsymbol{y}_N)$ is based on the discrete analysis and geometry on the graphs, which can be seen as a discrete differential regularization operator [4]. This regularization term preserves the intrinsic geometry of the data and the local structure of the graph or manifold. It constrains the learned function from being changed too much from the neighboring points.

In the 2-smoothness case, the smoothness function has a compact form

$$S_2(\boldsymbol{y}_N) = \frac{1}{2}\boldsymbol{y}_N^{\mathrm{T}}(\boldsymbol{I} - \boldsymbol{S})\boldsymbol{y}_N, \quad (5)$$

where $\boldsymbol{S} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{W}\boldsymbol{D}^{-\frac{1}{2}}$, $\boldsymbol{D} = \mathrm{diag}(D_{11}, \ldots, D_{NN})$ and $(\boldsymbol{W})_{ij} = W_{ij}$. The matrix $\boldsymbol{\Delta} = \boldsymbol{I} - \boldsymbol{S}$ is called the normalized graph Laplacian in spectral graph theory [7]. Moreover, the gradient and the Hessian of $-\lg P(\boldsymbol{y}_N \mid \Theta)$ are (for arbitrary p-smoothness cases, see Appendix A):

$$\begin{cases} \boldsymbol{g}_N = \nabla_{\boldsymbol{y}_N}(-lgP(\boldsymbol{y}_N|\Theta)) = \boldsymbol{K}_N^{-1}\boldsymbol{y}_N \\ \boldsymbol{K}_N^{-1} = \nabla\nabla_{\boldsymbol{y}_N}(-lgP(\boldsymbol{y}_N|\Theta)) = \Delta = \boldsymbol{I} - \boldsymbol{S} \end{cases}. \quad (6)$$

The matrix $\boldsymbol{K}_N = \boldsymbol{\Delta}^{-1}$ is the covariance matrix of the prior probability, and it is also the inverse matrix of the normalized graph Laplacian. Therefore, the covariance between two points depends on all the other training data, including both labeled and unlabeled ones. In contrast, most traditional Gaussian processes adopt the covariance based on "local" distance information. This will make the covariance depend only on the self coordinates in the prior probability.

2.2 Extended Bernoulli model (EBM)

The Bernoulli model is also called the logistic regression model, which was developed to deal with the binary classification. To handle the unlabeled data, we present a new noise function on the process $y \to t$ called the extended Bernoulli model (EBM):

$$P(t_i|y_i) = \frac{1-\lambda}{1+\exp(-t_i y_i)} I_{\{t_i \neq 0\}} + \lambda I_{\{t_i = 0\}}, \qquad (7)$$

where I is the indicator function. $I_{\{t_i=0\}}$ means, if $t_i = 0$, $I = 1$, and if $t_i \neq 0$, $I = 0$. Here, for the semi-supervised problem, we set labels of the unlabeled data to zeros initially. Thus, if $t_i = 0$, the probability $P(t_i = 0 \mid y) \equiv \lambda$. The factor λ makes the function $P(t_i|y_i)$ with respect to t_i be a probability, which means $P(t_i = 1|y_i) + P(t_i = -1|y_i) + P(t_i = 0|y_i) \equiv 1$. As Fig. 1 shows, this model can be considered as a degenerated ordered category model (OCM) [8], where the variance of the probability $P(t_i = 0|y_i)$ is infinite. We define the margin as the range where $P(t_i = 0|y)$ is larger than $P(t_i = 1|y)$ and $P(t_i = -1|y)$. In the margin, the difference between $P(t_i = 1|y)$ and $P(t_i = -1|y)$ is smaller than the difference outside the margin. Therefore, the margin of EBM represents the more uncertain labels. The parameter λ controls this margin.

Moreover, Figures 1 (b) – (d) show the relationship between the prior and the posterior probability of latent variables. In the Gaussian process and GRF, we can assume that each latent variable is also conditionally Gaussian

$$P\left(y_i|\boldsymbol{y}_{N-\{i\}}, \boldsymbol{X}_N\right) = N(\mu_i, \sigma_i) \triangleq P(y_i), \qquad (8)$$

where μ_i and σ_i are related to the input points and labels.

As Figs. 1 (b) and (c) show, for $t_i = 1$ and $t_i = -1$, the mean and the variance of posterior $P(y_i|\, t_i = 1)$ and $P(y_i|\, t_i = -1)$ are related to the likelihood $P(t_i \mid y_i)$ and the prior $P(y_i)$. If μ_i is near zero, the posterior of the latent variable y_i is affected by the label t_i. It will have a

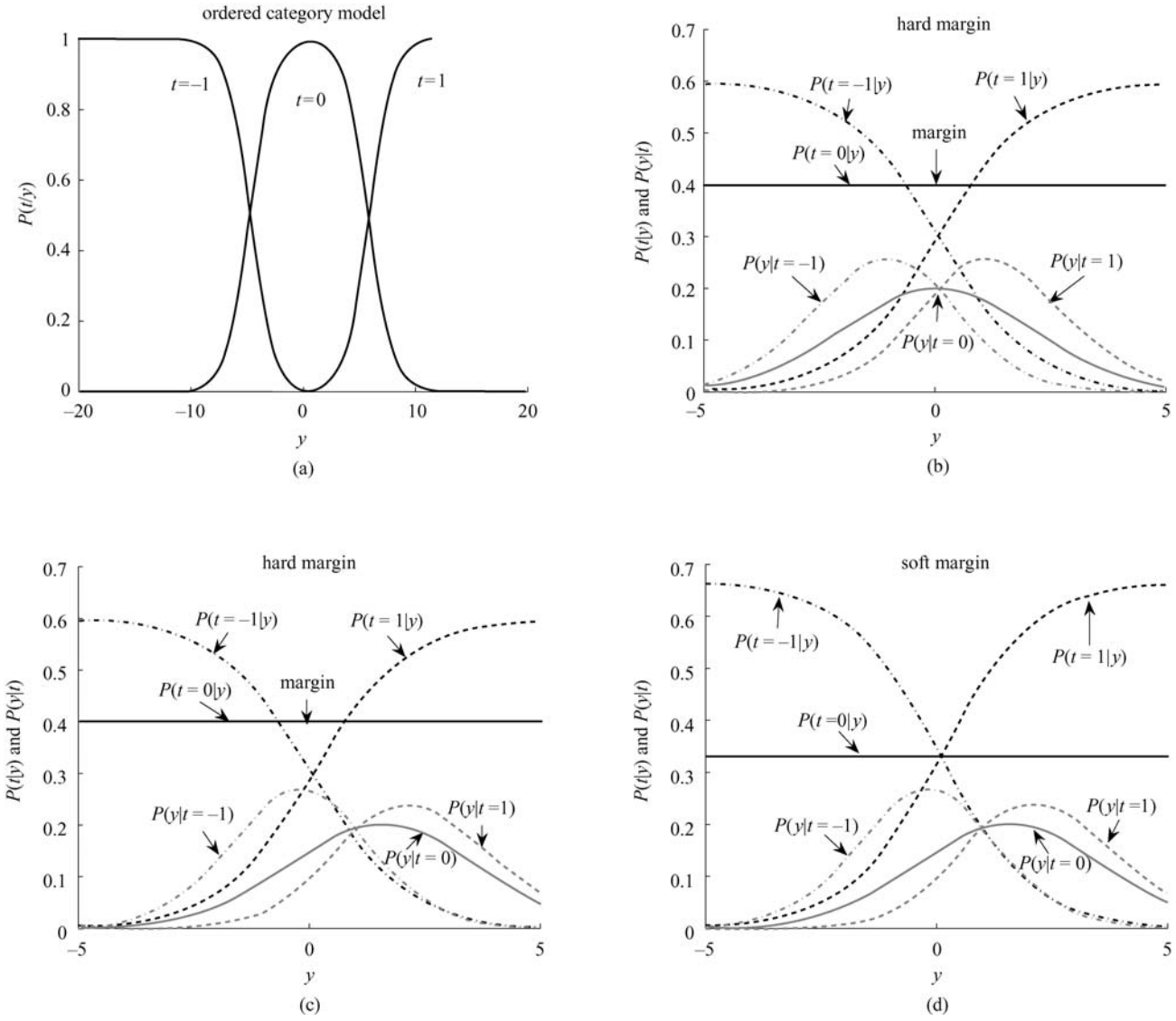

Fig. 1 Illustration of EBM: the relationship between the prior and the posterior probabilities. (a) OCM; (b) $\lambda = 0.4$, $\mu_y = 0$; (c) $\lambda = 0.4$, $\mu_y = 1.5$; (d) $\lambda = 1/3$, $\mu_y = 1.5$

positive estimated y_i when the label is 1, and negative y_i when the label is -1.

If the label is $t_i = 0$, due to Bayesian formulation, we have $P(t_i \mid y_i)\, P(y_i) = P(y_i \mid t_i)\, P(t_i)$. The probabilities $P(t_i)$ and $P(t_i \mid y_i)$ are both constant for $t_i = 0$, so the posterior probability $P(y_i \mid t_i = 0)$ only depends on the prior $P(y_i)$. If μ_i still approximates to zero, we will get a zero estimated y_i by maximizing the posterior probability. This is why we choose a graph regularization-based prior. As mentioned previously, each covariance between two points of this prior is related to all the training data. Therefore, if there is a small amount of labeled data in the training set, the μ_i of an unlabeled $\boldsymbol{x}_i$ will be influenced by the labeled data more than the one choosing the traditional prior. In consequence, μ_i is non-zero, leading to a non-zero estimated y_i (see Fig. 1 (c)).

Furthermore, by comparing Figs. 1 (c) and (d), we can see that, the margins do not affect the estimation of the latent variable. The estimated y_i remains the same despite different margin models being imposed on the process $y \to t$. However, any point whose latent variable y_i falls inside the margin will be labeled zero, which makes it remain unlabeled. This kind of points does not contribute to the prediction function (see in the prediction phase). Therefore, the classification boundary will be changed.

Due to the graphical model of GRF, we have $P(\boldsymbol{t}_N|\boldsymbol{y}_N) = \prod_{i=1}^{N} P(t_i|y_i)$. Then the gradient vector and the Hessian matrix of $-\lg P(\boldsymbol{t}_N \mid \boldsymbol{y}_N)$ are

$$\begin{aligned}\boldsymbol{\alpha}_N &= \nabla_{\boldsymbol{y}_N}\left(-\lg P(\boldsymbol{t}_N|\boldsymbol{y}_N)\right) = \left(-\frac{t_i}{1+\exp(t_i y_i)}\right)_i,\\ \boldsymbol{\Pi}_N &= \nabla\nabla_{\boldsymbol{y}_N}\left(-\lg P(\boldsymbol{t}_N|\boldsymbol{y}_N)\right)\\ &= \operatorname{diag}\left(\frac{t_i^2\exp(t_i y_i)}{(1+\exp(t_i y_i))^2}\right).\end{aligned} \tag{9}$$

Note that the parameter λ does not influence the gradient vector and the Hessian matrix, which is consistent with the analysis above. In the training phase, the unlabeled elements in the gradient vector $\boldsymbol{\alpha}_N$ and the Hessian matrix $\boldsymbol{\Pi}_N$ are zero.

2.3 Training

To train a GRF, we make use of the Laplace approximation for the posterior probability $P(\boldsymbol{y}_N \mid \boldsymbol{t}_N, \Theta)$ with fixed hyper-parameters. By differentiating Eq. (2) with respect to $\boldsymbol{y}_N$, we have

$$\nabla\Psi = \boldsymbol{\alpha}_N + \boldsymbol{g}_N, \quad \nabla\nabla\Psi = \boldsymbol{\Pi}_N + \boldsymbol{K}_N^{-1}. \tag{10}$$

To find the expectation of the approximated Gaussian density Ψ in Eq. (2), the Newton-Raphson iteration is adopted

$$\boldsymbol{y}_N^{\text{new}} = \boldsymbol{y}_N - (\nabla\nabla\Psi)^{-1}\nabla\Psi. \tag{11}$$

Since $\nabla\nabla\Psi$ is always positive definite[1], Eq. (2) is a convex problem. When it converges to an optimal $\hat{\boldsymbol{y}}_N$, $\nabla\Psi$ turns to be a zero vector. The posterior probability $P(\boldsymbol{y}_N \mid \boldsymbol{t}_N, \Theta)$ is approximated as Gaussian, being centered at the estimated $\hat{\boldsymbol{y}}_N$. The inverse of the posterior covariance is $\nabla\nabla\Psi$.

The hyper-parameter is the standard deviation σ of the radial basis function (RBF) kernel. The estimation mainly follows Refs. [6,10], which is estimated by minimizing the negative logarithmic likelihood,

$$\begin{aligned}J(\Theta) &\equiv -\lg P(\boldsymbol{t}_N|\Theta)\\ &\approx \Psi(\boldsymbol{y}_N) + \frac{1}{2}\lg\det\left(\boldsymbol{K}_N^{-1}+\boldsymbol{\Pi}_N\right)\\ &= \sum_{i=1}^{N}\lg(1+\exp(t_i y_i)) + \frac{1}{2}\lg\det(\boldsymbol{K}_N\boldsymbol{\Pi}_N+\boldsymbol{I})\\ &\quad + \frac{1}{2}\boldsymbol{y}_N^{\mathrm{T}}\boldsymbol{K}_N^{-1}\boldsymbol{y}_N.\end{aligned} \tag{12}$$

We can update the hyper-parameter with gradient search (see Appendix B for details):

$$\begin{aligned}\frac{\partial J(\Theta)}{\partial\Theta} &= \boldsymbol{\alpha}_N^{\mathrm{T}}\frac{\partial\boldsymbol{y}_N}{\partial\Theta} + \frac{1}{2}\operatorname{tr}\left[(\boldsymbol{I}+\boldsymbol{K}_N\boldsymbol{\Pi}_N)^{-1}\frac{\partial\boldsymbol{K}_N\boldsymbol{\Pi}_N}{\partial\Theta}\right]\\ &\quad + \frac{1}{2}\left[2\left(\boldsymbol{K}_N^{-1}\boldsymbol{y}_N\right)^{\mathrm{T}}\frac{\partial\boldsymbol{y}_N}{\partial\Theta} + \boldsymbol{y}_N^{\mathrm{T}}\frac{\partial\boldsymbol{K}_N^{-1}}{\partial\Theta}\boldsymbol{y}_N\right].\end{aligned} \tag{13}$$

2.4 Prediction

For the new test data, we utilize the Gaussian process model to obtain the induction formulation. Provided the estimated hyper-parameters and labels, the prediction function of the Gaussian process is to compute the integral over the latent variable of a new point

$$P(t_{N+1}|\boldsymbol{D},\Theta) = \int P(t_{N+1}|y_{N+1})P(y_{N+1}|\boldsymbol{D},\Theta)\mathrm{d}y_{N+1}. \tag{14}$$

This probability can be obtained by integrating over the hyper-parameters; however, it can be costly when the data set is large. Thus, we employ the result of maximizing the marginal likelihood rather than integrating. The second term in Eq. (14) can be obtained by further integration as follows:

$$P(y_{N+1}|\boldsymbol{D},\Theta) = \int P(y_{N+1}|\boldsymbol{y}_N,\Theta)P(\boldsymbol{y}_N|\boldsymbol{t}_N,\Theta)\mathrm{d}\boldsymbol{y}_N. \tag{15}$$

It is not easy to calculate the high-dimensional integral. One way to solve this problem has been presented in the

[1] For the positive semi-definite case, we can add extra regularization as the jitter noise [9].

original Gaussian process classification's paper [6]. Since the approximated posterior $P(\boldsymbol{y}_N \mid \boldsymbol{t}_N, \Theta)$ is Gaussian, the estimated $\boldsymbol{y}_N$ in the training phase can be used instead of the integral to compute the probability $P(y_{N+1} \mid \boldsymbol{D}, \Theta)$. Since $P(y_{N+1}|\boldsymbol{y}_N,\Theta) = \frac{P(y_{N+1},\boldsymbol{y}_N|\Theta)}{P(\boldsymbol{y}_N|\Theta)}$, the objective function is:

$$\Psi(\boldsymbol{y}_N, y_{N+1}) = -\lg P(\boldsymbol{y}_{N+1}|\Theta). \tag{16}$$

We minimize Eq. (16) only with respect to $\boldsymbol{y}_{N+1}$ of a new point $\boldsymbol{x}_{N+1}$, which leads to

$$\hat{y}_{N+1} = \boldsymbol{k}^{\mathrm{T}} \boldsymbol{K}_N^{-1} \hat{\boldsymbol{y}}_N = \boldsymbol{k}^{\mathrm{T}} (\boldsymbol{I} - \boldsymbol{S}) \hat{\boldsymbol{y}}_N, \tag{17}$$

where $\boldsymbol{k} = (k_1, k_2, \ldots, k_N)^{\mathrm{T}}$ and $k_i = W_{N+1,i} = \exp(-\|\boldsymbol{x}_{N+1} - \boldsymbol{x}_i\|^2/2\sigma^2)$ is the covariance between a new given point and the ith training point, which is based on "local" information.

Note that after minimizing Eq. (2), the gradient in Eq. (10) becomes $\nabla\Psi = \boldsymbol{\alpha}_N + \boldsymbol{K}_N^{-1}\hat{\boldsymbol{y}}_N = 0$. Thus, Eq. (17) is given by

$$\hat{\boldsymbol{y}}_{N+1} = -\boldsymbol{k}^{\mathrm{T}} \boldsymbol{\alpha}_N, \tag{18}$$

where $\boldsymbol{\alpha}_N = -\boldsymbol{K}_N^{-1}\hat{\boldsymbol{y}}_N = (\alpha_1, \alpha_2, \ldots, \alpha_N)^{\mathrm{T}}$. According to Eqs. (9) and (18), we can see that, if the latent variable of a point in the training set satisfies $y_i > 0$, and the point is outside the margin, it has a positive weight $-\alpha_i$ in the predicting function. Conversely, if a point outside the margin satisfies $y_i < 0$, the weight is negative. Moreover, α_i tends to be zero with very large y_i, and to approximate $\pm 1/2$ when y_i is nearly outside the margin. Finally, the points falling inside the margin will have the weights of zeros (according to Eq. (9)). Therefore, these points will not affect the classification result in the prediction phase.

The flowchart is given in the following.

1. Input:

Training set examples $\boldsymbol{D} = \{\boldsymbol{x}_i, t_i\}_{i=1}^N$.

2. Training phase:

a) Hyper-parameter estimation by Gradient search Eqs. (12) and (13).

b) Compute the Gradient $\nabla\Psi$.

c) Compute the Hessian $\nabla\nabla\Psi$.

d) Laplace approximation and the Newton-Raphson iteration:

$$\boldsymbol{y}_N{}^{\mathrm{new}} = \boldsymbol{y}_N - (\nabla\nabla\Psi)^{-1}\nabla\Psi.$$

3. Prediction:

$$\hat{y}_{N+1} = \boldsymbol{k}^{\mathrm{T}} \boldsymbol{K}_N{}^{-1} \hat{\boldsymbol{y}}_N = \boldsymbol{k}^{\mathrm{T}} (\boldsymbol{I} - \boldsymbol{S}) \hat{\boldsymbol{y}}_N .$$

4. Estimate the classification accuracy.

3 Experiments

3.1 Test data

We test the algorithm with the two-moon data. As Fig. 2 (a) shows, 200 input points with only two labeled examples are used to test our algorithm. The hyper-parameter estimation curves are shown in Fig. 2 (b). The objective value is plotted in (b) as a solid line, and the derivative is plotted as the dotted line. The classification results of the 2-smoothness case with different hard margins are shown in Figs. 2 (c) and (d). We can see that for large margins, lots of unlabeled points fall into the margin, and this will remove the geometric information provided by the training set. A proper small margin can efficiently remove the effect of the noise points, and this may lead to a more correct boundary of the predicting function. The results of more p-smoothness cases are shown in Fig. 3. As to $p = 0.5$, it is more sensitive to outliers.

3.2 Real world data

In this experiment, we test our algorithm on the United State Postal Service (USPS) digits data set. Each digit is represented by a 256 dimensional vector ranging from 0 to 1. We choose digits "1", "2", "3", "4", and there are 1269, 929, 824 and 852 examples for each class. We use "1" vs. "4", "2" vs. "3" and "1" "3" vs. "2" "4" as the tasks. All the tasks are split as seen (include labeled and unlabeled data) and unseen data sets. We adopt 2-smoothness function as the prior for all the real data experiments. The test accuracy plotted in the figures is an average of 50 random trials. The hyper-parameter estimation result is shown in Fig. 4 (a). We test our algorithm with different numbers of unlabeled points and the result is shown in Fig. 4 (b). We can see that it is better if more unlabeled data are added, since they bring more geometric information to the learning machine. We also test the algorithm with some fixed margins, which is shown in Fig. 4 (c). A small margin (i.e. $\lambda = 0.337$) will improve the test accuracy. Similar to the test problem, a proper margin will help remove the effect of the noise data, and a large margin will remove the geometric information.

For comparison, we test the following supervised methods: support vector machine (SVM) [11] and regularized least squares (RLS) [2]; transductive method – transduction on graphs (TOG) [3]; inductive method – Laplacian regularized least squares (LapRLS) [2]; and our method semi-supervised Gaussian process (SSGP). We split data into two types, seen and unseen, each covering 50%. For supervised methods, we only use the labeled points in the seen data to classify the unseen. For semi-supervised inductive methods, we use all the seen data to classify the unseen. As to TOG, each iteration has been run twice. First, we run it on the seen set, and following that the accuracy was evaluated on the seen set. Second, we use both the seen and unseen data to train another TOG algorithm, and evaluate the accuracy on the unseen set. We follow the settings of SVM and TOG in Ref. [3], where the width of RBF kernel for SVM is set to 5, and for TOG it is 1.25. The parameter C of SVM is set to 1 empirically. Moreover, the parameters of RLS and LapRLS, that is,

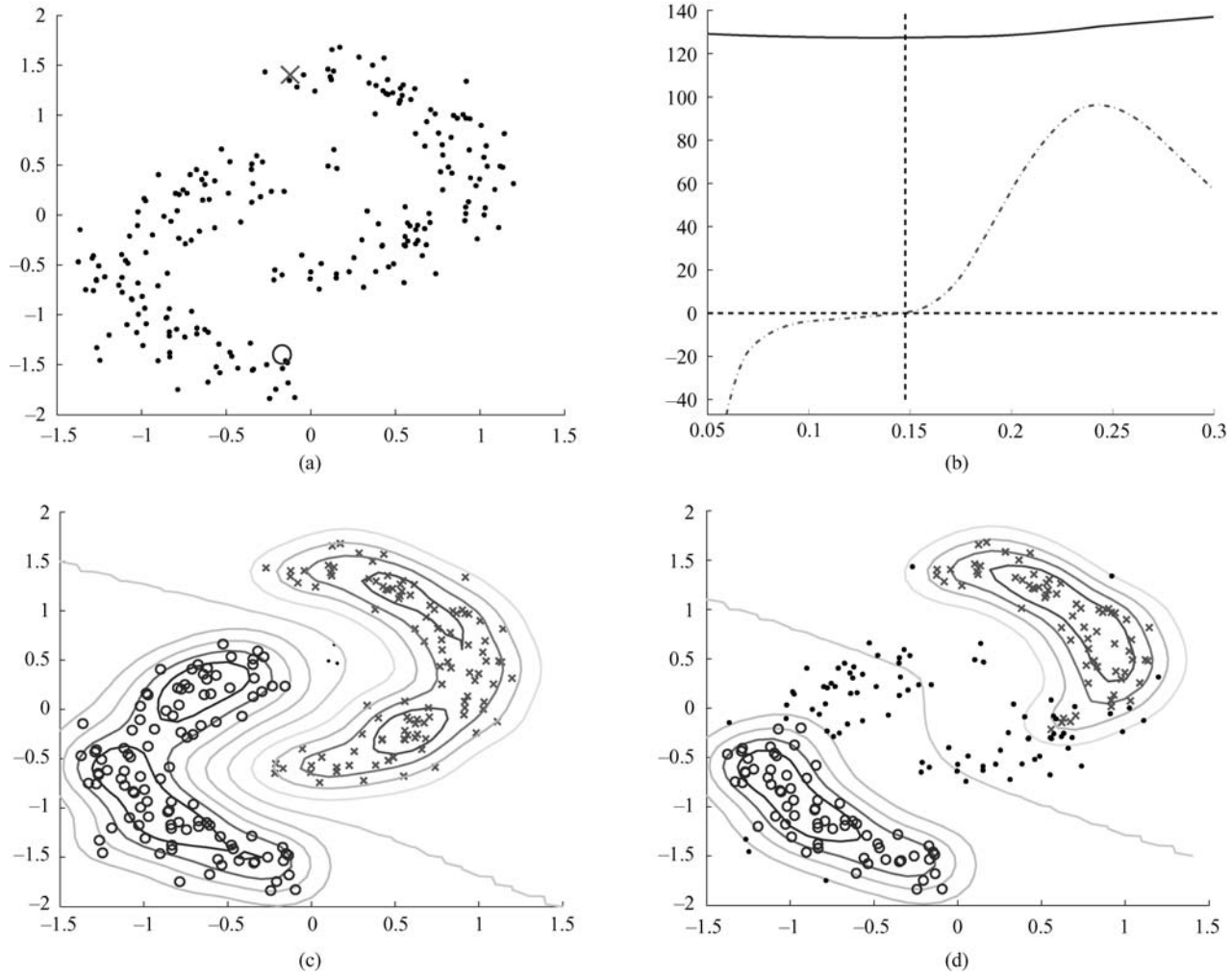

Fig. 2 Test data and the results of different margins.
(a) Training data; (b) parameter estimation; (c) hard margin: $\lambda = 0.37$; (d) hard margin: $\lambda = 0.39$
Note: Two points are labeled in the training set. The contour in each figure describes the estimated $\boldsymbol{y}_{N+1}$ of the unseen points.

the width of the RBF kernel are 1.25, $\gamma_A l = 0.005$ and $\gamma_I l/(u+l)^2 = 0.045$ [2]. We use a soft margin model where $\lambda = 1/3$ in SSGP to compare with the other algorithms. The average accuracy rates are shown in Figs. 4 (d) – (f). The best is the transductive method on the test unseen set, since it uses both the seen and unseen information. The training accuracy on the seen set for TOG is similar to our algorithm using a soft margin model. For the inductive problem, our method significantly outperforms SVM and RLS, and also LapRLS under the setting in Ref. [2].

4 Conclusions

In this paper, we present an algorithm that uses the Gaussian random field for semi-supervised learning. To introduce the manifold or the geometric information of the unlabeled data, we adopt the graph regularization-based prior for the arbitrary p-smoothness case. The initial labels of unlabeled data in our algorithm can be updated when we run alternating optimization iterations. Besides, employing the Gaussian process formulation, we can predict the new test data. Experimental results on the real data sets show that a proper margin can remove the effect of the noise in the training data and improve the test accuracy. Furthermore, the results of soft margin model for both transduction and induction are competitive or even better than the existing state-of-the-art algorithms.

In the future, we plan to do some further researches on how to find a more appropriate Bayesian margin. We also think that fast learning is another interesting problem, which would help solve the computational problem of the arbitrary p-smoothness case.

Appendix A: Arbitrary p-smoothness case

By defining the function $f_i \equiv \|\nabla y_i\|^2 = \sum_j W_{ij}\left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right)^2$, the gradient of the negative logarithm of Eq. (3) is:

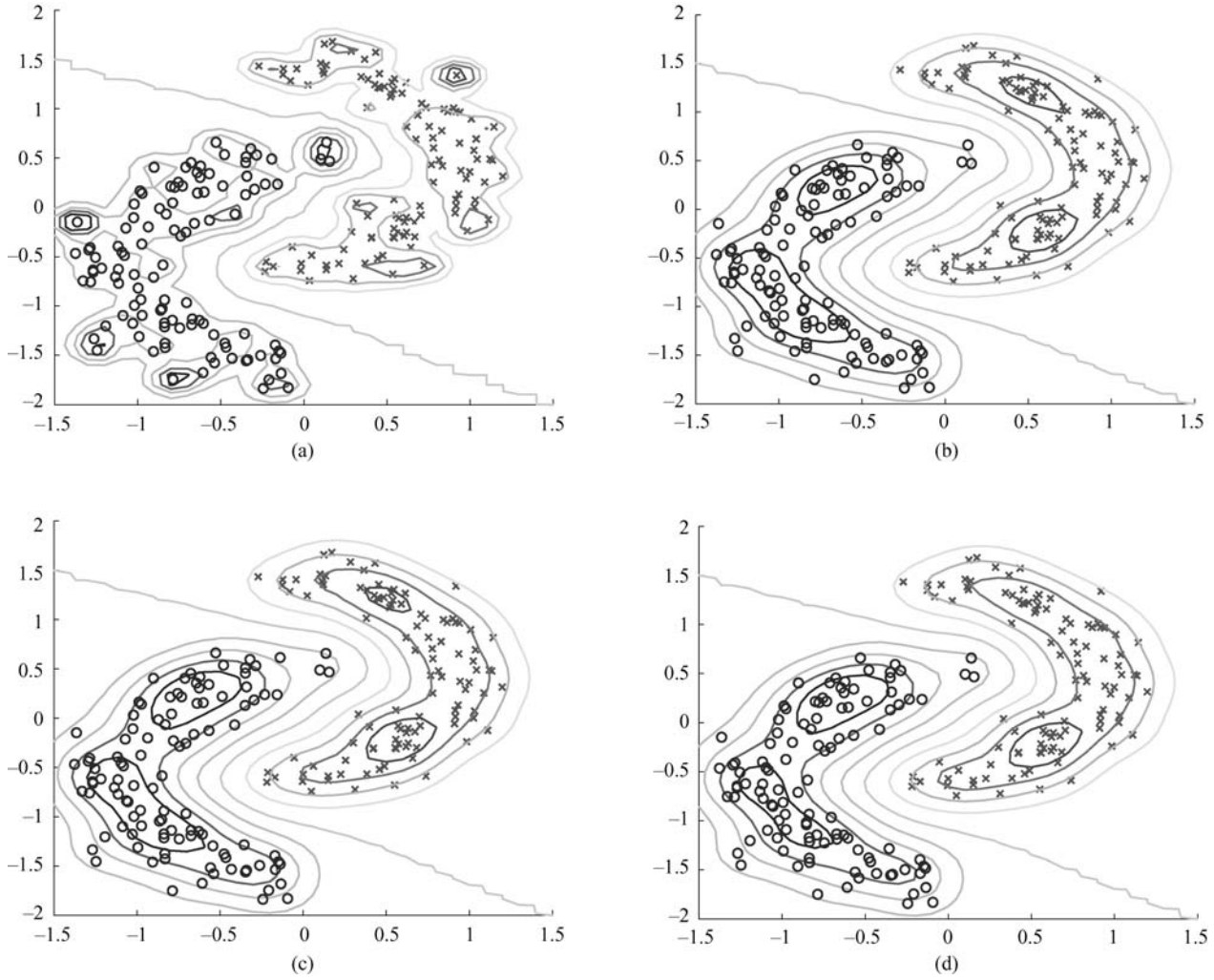

Fig. 3 Test data and the results of different p-smoothness cases.
(a) $p = 0.5$, $\sigma = 0.1$; (b) $p = 1$, $\sigma = 0.2$; (c) $p = 1.5$, $\sigma = 0.2$; (d) $p = 2$, $\sigma = 0.1$

$$\boldsymbol{g}_N = \nabla \boldsymbol{y}_N(-\lg P(\boldsymbol{y}_N)),$$

where :

$$(\boldsymbol{g}_N)_i = \sum_j \left(f_i^{\frac{p}{2}-1} + f_j^{\frac{p}{2}-1}\right) \frac{W_{ij}}{\sqrt{D_{ii}}} \left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right), \qquad \text{(A1)}$$

and the Hessian matrix is

$$\boldsymbol{K}_N^{-1} = \nabla\nabla \boldsymbol{y}_N(-\lg P(\boldsymbol{y}_N)),$$

where:

$$\begin{aligned}(\boldsymbol{K}_N^{-1})_{ii} = \sum_j \frac{W_{ij}}{D_{ii}} \Bigg\{ & f_i^{\frac{p}{2}-1} + f_j^{\frac{p}{2}-1} \\ & + (p-2)\left[f_j^{\frac{p}{2}-2} W_{ij} \left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right)^2 \right. \\ & \left. + \sum_k f_i^{\frac{p}{2}-2} W_{ik} \left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right)\left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_k}{\sqrt{D_{kk}}}\right) \right] \Bigg\},\end{aligned}$$

$$\begin{aligned}(\boldsymbol{K}_N^{-1})_{ij} = & -\frac{W_{ij}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \left(f_i^{\frac{p}{2}-1} + f_j^{\frac{p}{2}-1}\right) \\ & + (p-2) \sum_k \Bigg\{ f_i^{\frac{p}{2}-2} \frac{W_{ik} W_{ij}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_k}{\sqrt{D_{kk}}}\right) \\ & \left(\frac{y_j}{\sqrt{D_{jj}}} - \frac{y_i}{\sqrt{D_{ii}}}\right) + f_k^{\frac{p}{2}-2} \frac{W_{ik} W_{jk}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \\ & \left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_k}{\sqrt{D_{kk}}}\right)\left(\frac{y_j}{\sqrt{D_{jj}}} - \frac{y_k}{\sqrt{D_{kk}}}\right) \\ & + f_j^{\frac{p}{2}-2} \frac{W_{ij} W_{jk}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right) \\ & \left(\frac{y_j}{\sqrt{D_{jj}}} - \frac{y_k}{\sqrt{D_{kk}}}\right) \Bigg\}. \qquad \text{(A2)}\end{aligned}$$

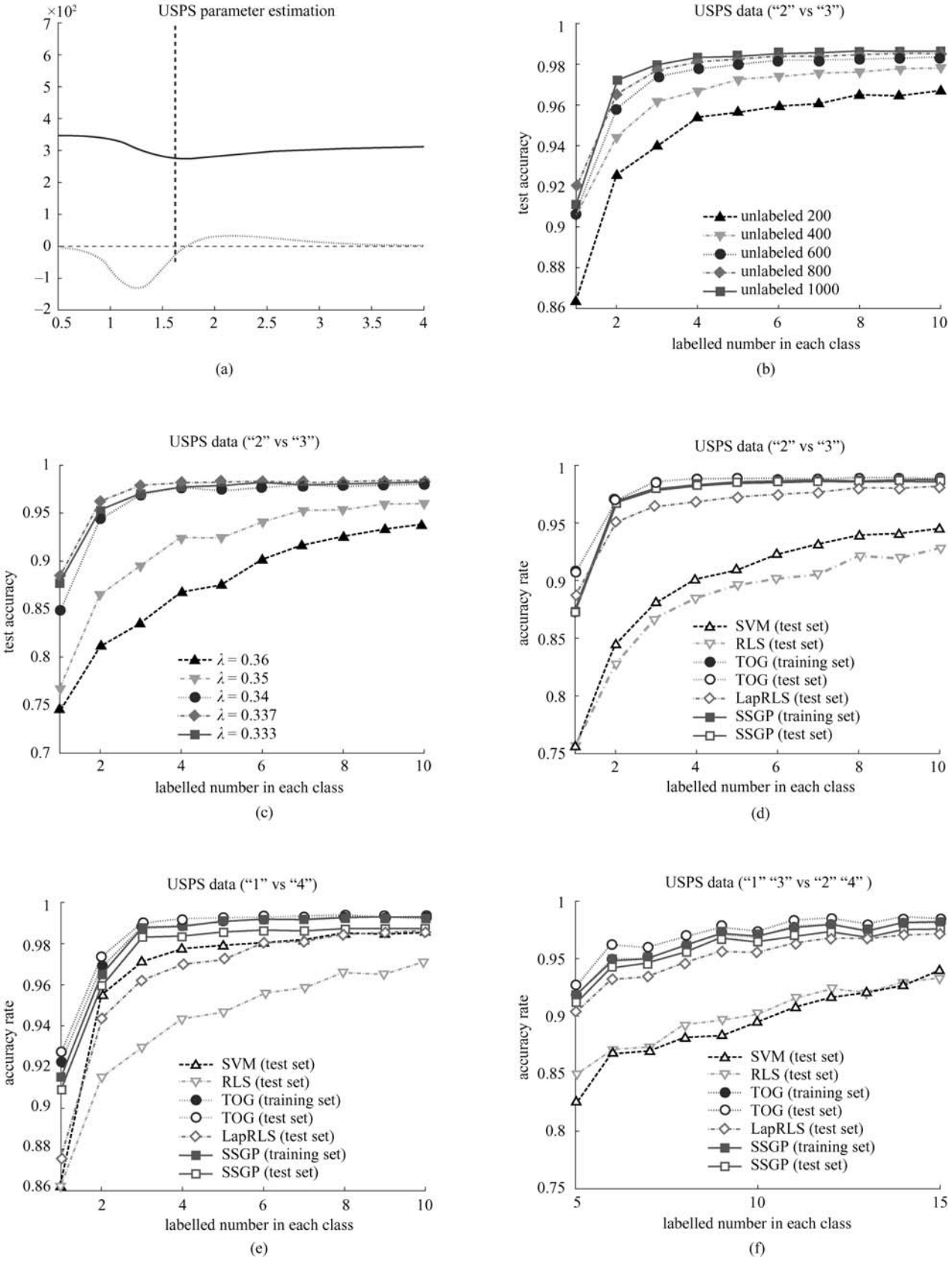

Fig. 4 USPS data. (a) Parameter estimation; (b) different training numbers; (c) different margins; (d) "2" vs. "3"; (e) "1" vs. "4"; (f) "1" "3" vs. "2" "4"

Appendix B: Derivation of hyper-parameter estimation

We estimate the hyper-parameter $\Theta = \sigma$ by gradient search, which minimizes the negative logarithmic likelihood in Eq. (12). According to Eqs. (6) and (10), by using the fact $\nabla\Psi = 0$, and taking derivatives on both sides, we have

$$\frac{\partial \boldsymbol{y}_N}{\partial \sigma} = (\boldsymbol{I} + \boldsymbol{K}_N \boldsymbol{\Pi}_N)^{-1} \frac{\partial \boldsymbol{K}_N}{\partial \sigma} (-\boldsymbol{\alpha}_N). \tag{B1}$$

And pursuant to Eq. (5) we have

$$\frac{\partial \boldsymbol{K}_N^{-1}}{\partial \sigma} = -\boldsymbol{D}^{-\frac{1}{2}} \frac{\partial \boldsymbol{W}}{\partial \sigma} \boldsymbol{D}^{-\frac{1}{2}} + \boldsymbol{D}^{-1} \left(\frac{\partial \sqrt{\boldsymbol{D}}}{\partial \sigma} \boldsymbol{W} \sqrt{\boldsymbol{D}} + \sqrt{\boldsymbol{D}} \boldsymbol{W} \frac{\partial \sqrt{\boldsymbol{D}}}{\partial \sigma} \right) \boldsymbol{D}^{-1}, \tag{B2}$$

where

$$\frac{\partial \boldsymbol{W}_{ij}}{\partial \sigma} = \boldsymbol{W}_{ij} \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2 / \sigma^3,$$

$$\frac{\partial \sqrt{\boldsymbol{D}_{ii}}}{\partial \sigma} = \frac{1}{2\sqrt{\boldsymbol{D}_{ii}}} \sum_k \boldsymbol{W}_{ik} \left\| \boldsymbol{x}_i - \boldsymbol{x}_k \right\|^2 / \sigma^3. \tag{B3}$$

Apart from that, differences of $\boldsymbol{K}_N$ and $\boldsymbol{K}_N \boldsymbol{\Pi}_N$ are given by

$$\frac{\partial \boldsymbol{K}_N}{\partial \sigma} = -\boldsymbol{K}_N \frac{\partial \boldsymbol{K}_N^{-1}}{\partial \sigma} \boldsymbol{K}_N \tag{B4}$$

and

$$\frac{\partial \boldsymbol{K}_N \boldsymbol{\Pi}_N}{\partial \sigma} = \frac{\partial \boldsymbol{K}_N}{\partial \sigma} \boldsymbol{\Pi}_N + \boldsymbol{K}_N \frac{\partial \boldsymbol{\Pi}_N}{\partial \sigma}, \tag{B5}$$

where

$$\frac{\partial \boldsymbol{\Pi}_N}{\partial \sigma} = \frac{\partial \boldsymbol{\Pi}_N}{\partial \boldsymbol{y}_N} \frac{\partial \boldsymbol{y}_N}{\partial \sigma} = \operatorname{diag}\left(\frac{t_i^3 \exp(t_i y_i)(1 - \exp(t_i y_i))}{(1 + \exp(t_i y_i))^3} \right) \operatorname{diag}\left(\frac{\partial \boldsymbol{y}_N}{\partial \sigma} \right). \tag{B6}$$

Acknowledgements This work was supported by the Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList).

References

1. Zhu X. Semi-supervised learning literature survey. Technical Report, Computer Sciences, University of Wisconsin-Madison, 2005
2. Belkin M, Niyogi P, Sindhwani V. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research, 2006, 7: 2399–2434
3. Zhou D, Bousquet O, Lal T N, et al. Learning with local and global consistency. In: Proceedings of Advances in Neural Information Processing Systems, 2004, 16: 321–328
4. Zhou D, Schölkopf B. Regularization on discrete spaces. In: Proceedings of the 27th DAGM Symposium, 2005, 361–368
5. Zhu X, Ghahramani Z, Lafferty J. Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the 20th International Conference of Machine Learning, 2003, 912–919
6. Williams C, Barber D. Bayesian classification with Gaussian processes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998, 20(12): 1342–1351
7. Chung F. Spectral Graph Theory. No. 92 in Tegional Conference Series in Mathematics. American Mathematical Society, 1997
8. Lawrence N D, Jordan M I. Semi-supervised learning via Gaussian processes. In: Proceedings of Advances in Neural Information Processing Systems, 2004, 17: 753–760
9. Neal R M. Monte-carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report, Department of Statistics, University of Toronto, 1997
10. Zhu X, Ghahramani Z. Semi-supervised learning: from Gaussian fields to Gaussian processes. Technical Report, Carnegie Mellon University, 2003
11. Chang C, Lin C. Libsvm: A Library for Support Vector Machines, 2001