Graph Based Multi-class Semi-supervised Learning Using Gaussian Process

Yangqiu Song, Changshui Zhang, and Jianguo Lee

State Key Laboratory of Intelligent Technology and Systems,
Department of Automation, Tsinghua University, Beijing, China, 100084
{songyq99, lijg01}@mails.tsinghua.edu.cn, zcs@mail.tsinghua.edu.cn

Abstract. This paper proposes a multi-class semi-supervised learning algorithm of the graph based method. We make use of the Bayesian framework of Gaussian process to solve this problem. We propose the prior based on the normalized graph Laplacian, and introduce a new likelihood based on softmax function model. Both the transductive and inductive problems are regarded as MAP (Maximum A Posterior) problems. Experimental results show that our method is competitive with the existing semi-supervised transductive and inductive methods.

1 Introduction

Graph based semi-supervised learning is an interesting problem in machine learning and pattern recognition fields [15,21]. The idea of using the unlabeled data for training will bring the geometric and manifold information to the algorithm, which may be effective to increase the recognition rate [1,2,8,18,19]. These graph based methods are either transductive [1,18,19] or inductive [2,8]. For multi-class classification, it is easy to use the one-against-one or the one-against-the-rest method to construct a classifier based on a set of binary classification algorithms. Most of the existing graph based methods solve this as a one-against-the-rest problem. This is reasonable because it is only required to change the form of the labels of the data points, and do not need to modify the algorithm framework.

An alternative method to solve the multi-class problem is using the softmax function [16]. Softmax function is naturally derives from the log-linear model, and is convenient to describe the probability of each class. Therefore, it is useful to model this conditional probability rather than impose a Gaussian noise in the Bayesian classification framework. Gaussian process is an efficient non-parametric method which uses this softmax function model [16]. Lawrence and Jordan [10] developed a semi-supervised learning algorithm using fast sparse Gaussian Process: Informative Vector Machine (IVM). It is not a graph based algorithm. Zhu and Ghahramani [20] develop a semi-supervised binary classification framework of Gaussian process which is based on Gaussian fields, and they make use of a 1-NN approximation extending to the unseen points.

In this paper, we propose a novel algorithm to solve the multi-class semi-supervised learning problem. Similar to Gaussian process [16], both the training

and prediction phases can be regarded as MAP (Maximum A Posterior) estimation problems. The difference is that, a prior based on normalized graph Laplacian [5] is used, and a new conditional probability generalized from softmax function is proposed. Using this kind of Gaussian process, we can solve both the transductive and inductive problems.

This paper is organized as follows. Section 2 will present the details of multi-class semi-supervised learning algorithm. Experimental results will be given in section 3. Conclusion is given in section 4.

2 Semi-supervised Gaussian Process

First, we introduce some notations. We denote the input data point as a feature vector $\mathbf{x}_n$ ($n = 1, 2, \dots, N$), and $\mathbf{X}_N \triangleq \{\mathbf{x}_i\}_{i=1}^N$ is the observed data set include both labeled and unlabeled data. The label of the input data point is given by $\mathbf{t}_i$. The label set of all the observed training data is $\mathbf{T}_N = (\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_N)^T$. For transductive problem, we need to re-estimate the label of the unlabeled data. For inductive problem, we want to estimate the label $\mathbf{t}_{N+1}$ of a new point $\mathbf{x}_{N+1}$. Instead of using the direct process $\mathbf{x} \rightarrow \mathbf{t}$, we adopt a latent variable $\mathbf{y}$ to generate a process as $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{t}$. We define the latent variable vector $\mathbf{y}_i = \mathbf{y}(\mathbf{x}_i)$ as functions of $\mathbf{x}_i$, and $\mathbf{Y}_N = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N)^T$ is the latent variable matrix of the input data. Then, some noisy function on the process $\mathbf{y} \rightarrow \mathbf{t}$ could be imposed. $P(\mathbf{Y}_N)$ is the prior, and the conditional probability is $P(\mathbf{T}_N|\mathbf{Y}_N)$.

2.1 The Likelihood

For a C classes problem, the softmax function model [16] sets the label as $\mathbf{t}_i = [0, \dots, 0, \mathbf{t}_i^j = 1, 0, \dots, 0]$, if $\mathbf{x}_i$ is belonged to class j ($j = 1, 2, \dots, C$). We extend the labels to the unlabeled data, which is set to a zero vector $\mathbf{t}_i = \mathbf{0}$ initially. Thus, the conditional probability $P(\mathbf{t}_i|\mathbf{y}_i)$ ($i = 1, 2, \dots, N$) of a C classes problem is given by:

$$P(\mathbf{t}_i|\mathbf{y}_i) = \frac{1}{C+1} \prod_{j=1}^C \left(\frac{C \exp \mathbf{y}_i^j}{\sum_{j=1}^C \exp \mathbf{y}_i^j} \right)^{\mathbf{t}_i^j} \quad (1)$$

We call this model as an extended softmax function model (ESFM), and we have $\sum_{\mathbf{t}_i \in (C+1) \text{ classes}} P(\mathbf{t}_i|\mathbf{y}_i) \equiv 1$. The essence of ESFM is it modifies a C classes problem to be a $C+1$ classes problem. However, there has the difference between a traditional $C+1$ classes problem and our model. When $\mathbf{t}_i = \mathbf{0}$, we have $P(\mathbf{t}_i|\mathbf{y}_i) \equiv 1/(C+1)$. One of the softmax function $\frac{C}{C+1} (\exp \mathbf{y}_i^j / \sum_{j=1}^C \exp \mathbf{y}_i^j)$ will be determinately more than $1/(C+1)$. Therefore, this model will never classify a point to be unlabeled. By applying to a semi-supervised learning problem, this model will classify all the unlabeled data to the existent C classes. Then, the

Log-likelihood of the conditional probability is given by:

$$\begin{aligned} L &= -\log P(\mathbf{T}_N|\mathbf{Y}_N) = \sum_{i=1}^N L_i = \sum_{i=1}^N -\log P(\mathbf{t}_i|\mathbf{y}_i) \\ &= \sum_{i=1}^N -\left[\log \frac{1}{C+1} + \sum_{j=1}^C \mathbf{t}_i^j \left(\log C + \mathbf{y}_i^j - \log \sum_{m=1}^C \exp \mathbf{y}_i^m \right) \right] \end{aligned} \quad (2)$$

For computational simplicity, we rewrite the form of matrix $\mathbf{Y}_N$ and $\mathbf{T}_N$ as vectors: $(\mathbf{y}_1^1, \mathbf{y}_2^1, \dots, \mathbf{y}_N^1, \dots, \mathbf{y}_1^C, \mathbf{y}_2^C, \dots, \mathbf{y}_N^C)^T$ and $(\mathbf{t}_1^1, \mathbf{t}_2^1, \dots, \mathbf{t}_N^1, \dots, \mathbf{t}_1^C, \mathbf{t}_2^C, \dots, \mathbf{t}_N^C)^T$. Thus, by differentiating the logarithm of the likelihood probability we have:

$$\begin{aligned} a_i^j &= \sum_{m=1}^C \mathbf{t}_i^m \frac{\exp \mathbf{y}_i^j}{\sum_{m=1}^C \exp \mathbf{y}_i^m}, \quad b_i^j = \frac{\exp \mathbf{y}_i^j}{\sum_{m=1}^C \exp \mathbf{y}_i^m}, \quad c_i^j = a_i^j - \mathbf{t}_i^j \\ \mathbf{\Pi}_1 &= \text{diag} (a_1^1, a_2^1, \dots, a_N^1, \dots, a_1^C, a_2^C, \dots, a_N^C) \\ \mathbf{\Pi}_2 &= (\text{diag} (a_1^1, a_2^1, \dots, a_N^1), \dots, \text{diag} (a_1^C, a_2^C, \dots, a_N^C)) \\ \mathbf{\Pi}_3 &= (\text{diag} (b_1^1, b_2^1, \dots, b_N^1), \dots, \text{diag} (b_1^C, b_2^C, \dots, b_N^C)) \\ \boldsymbol{\alpha}_N &= \nabla_{\mathbf{Y}_N} (-\log P(\mathbf{T}_N|\mathbf{Y}_N)) = (c_1^1, c_2^1, \dots, c_N^1, \dots, c_1^C, c_2^C, \dots, c_N^C)^T \\ \mathbf{\Pi}_N &= \nabla \nabla_{\mathbf{Y}_N} (-\log P(\mathbf{T}_N|\mathbf{Y}_N)) = \mathbf{\Pi}_1 - \mathbf{\Pi}_2^T \mathbf{\Pi}_3 \end{aligned} \quad (3)$$

where $\boldsymbol{\alpha}_N$ and $\mathbf{\Pi}_N$ are the gradient vector and the Hessian matrix of $-\log P(\mathbf{T}_N|\mathbf{Y}_N)$ respectively.

2.2 The Prior

In Gaussian process, we take $P(\mathbf{Y}_N)$ as the prior. Many choices of the covariance functions for Gaussian process prior have been reviewed in [11], and covariance will affect the final classification significantly. We adopt the graph or manifold regularization based prior, which is also used for many other graph based methods [2,18]. In our framework, with the definition of $\mathbf{Y}_N$ above, we define:

$$P(\mathbf{Y}_N) = \frac{1}{Z} \exp\left\{-\frac{\mathbf{Y}_N^T \mathbf{K}_N^{-1} \mathbf{Y}_N}{2}\right\} \quad (4)$$

$\mathbf{K}_N$ is an $NC \times NC$ block diagonal matrix, which has the Kronecker product form:

$$\mathbf{K}_N^{-1} = \nabla \nabla_{\mathbf{Y}_N} (-\log P(\mathbf{Y}_N)) = \mathbf{I}_C \otimes \boldsymbol{\Delta} \quad (5)$$

where $\mathbf{I}_C$ is a $C \times C$ identity matrix, and the matrix:

$$\boldsymbol{\Delta} = \mathbf{I} - \mathbf{S} \quad (6)$$

is called normalized graph Laplacian¹ in spectral graph theory [5], and the prior $P(\mathbf{Y}_N)$ defines a Gaussian random field (GRF) [19] on the graph. $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$, $(\mathbf{W})_{ij} = W_{ij}$ and $\mathbf{D} = \text{diag}(D_{11}, \dots, D_{NN})$. $D_{ii} = \sum_j W_{ij}$, and W_{ij} is called

¹ We also introduce an extra regularization, since the normalized graph Laplacian is a semi definite positive matrix [5]. In particular, we set $\boldsymbol{\Delta} \leftarrow (\boldsymbol{\Delta} + \delta \mathbf{I})$, and fix the parameter in the experiments.

the weight function associated with the edges on graph, which satisfies: $W_{ij} > 0$ and $W_{ij} = W_{ji}$. It can be viewed as a symmetric similarity measure between $\mathbf{x}_i$ and $\mathbf{x}_j$.

The covariance matrix $\mathbf{K}_N$ is relative to the inverse matrix of the normalized graph Laplacian, so the covariance between two points is depend on all the other training data including both labeled and unlabeled [20]. In contrast, most of the traditional Gaussian processes adopt the Gram matrix based on “local” distance information to construct the covariance [11,20].

2.3 Transduction and Induction

In Gaussian process, both the training phase and the prediction phase can be regarded as MAP (Maximum A Posterior) estimation problems [16]. In the training phase, we estimate the latent variables $\mathbf{Y}_N$ from the training data. By computing the mode of posterior probability $P(\mathbf{Y}_N|\mathbf{T}_N)$ as the estimate of $\mathbf{Y}_N$, which is the negative logarithm of $P(\mathbf{Y}_N|\mathbf{T}_N) = P(\mathbf{Y}_N, \mathbf{T}_N)/P(\mathbf{T}_N)$, we define:

$$\Psi(\mathbf{Y}_N) = -\log P(\mathbf{T}_N|\mathbf{Y}_N) - \log P(\mathbf{Y}_N) \quad (7)$$

$P(\mathbf{T}_N)$ is omitted for it is a constant unrelated to $\mathbf{Y}_N$. $P(\mathbf{Y}_N)$ is the prior, which is based on graph regularization. $P(\mathbf{T}_N|\mathbf{Y}_N)$ is the new conditional probability: extended softmax function model (ESFM). Therefore, Laplace approximation method [16] can be used for estimating $\mathbf{Y}_N$ from the posterior. To find the minimum of Ψ in equation (7), the Newton-Raphson iteration [16] is adopted:

$$\mathbf{Y}_N^{new} = \mathbf{Y}_N - (\nabla\nabla\Psi)^{-1}\nabla\Psi \quad (8)$$

Where:

$$\nabla\Psi = \boldsymbol{\alpha}_N + \mathbf{K}_N^{-1}\mathbf{Y}_N, \quad \nabla\nabla\Psi = \boldsymbol{\Pi}_N + \mathbf{K}_N^{-1} \quad (9)$$

Since $\nabla\nabla\Psi$ is always positive definite, (7) is a convex problem. When it converges to an optimal $\hat{\mathbf{Y}}_N$, $\nabla\Psi$ will be zero vector. The posterior probability $P(\mathbf{Y}_N|\mathbf{T}_N)$ can be approximated as Gaussian, being centered at the estimated $\hat{\mathbf{Y}}_N$. The covariance of the posterior is $\nabla\nabla\Psi$. After the Laplace approximation, the latent variable vector $\hat{\mathbf{Y}}_N$ will depend on all the training data, and the corresponding $\hat{\mathbf{T}}_N$ could be updated.

In the prediction phase, the objective function is:

$$\Psi(\mathbf{Y}_N, \mathbf{y}_{N+1}) = -\log P(\mathbf{T}_{N+1}|\mathbf{Y}_{N+1}) - \log P(\mathbf{Y}_{N+1}) \quad (10)$$

which is minimized only with respect to $\mathbf{y}_{N+1}$. This leads to:

$$\hat{\mathbf{y}}_{N+1} = \mathbf{K}_{N+1}^T \mathbf{K}_N^{-1} \hat{\mathbf{Y}}_N \quad (11)$$

where $\mathbf{K}_{N+1} = \mathbf{I}_C \otimes \mathbf{k}$, and $\mathbf{k}_i = W_{N+1,i} = \exp(-\|\mathbf{x}_{N+1} - \mathbf{x}_i\|^2/2\sigma^2)$ is the covariance of a new given point and the i th training point. Note that, in the training phase, we have: $\nabla\Psi = \hat{\boldsymbol{\alpha}}_N + \mathbf{K}_N^{-1}\hat{\mathbf{Y}}_N = 0$. Thus, equation (11) is given by:

$$\hat{\mathbf{y}}_{N+1} = -\mathbf{K}_{N+1}^T \hat{\boldsymbol{\alpha}}_N \quad (12)$$

We substitute the estimated $\hat{\mathbf{Y}}_N$ and $\hat{\mathbf{T}}_N$ to equation (3) to realize the prediction. The form of predictive function (12) has a relationship with the RKHS [14]. Other semi-supervised inductive methods (such as [2]) also show how to find an appropriate RKHS using the Representer Theorem.

The training phase has the same framework as the transductive learning algorithm [18]. [18] imposes a Gaussian noise model and uses the one-against-the-rest way to deal with the multi-class problem. We generalize the softmax function to directly add the unlabeled data to the formulation. Our ESFM will reduce to the one-against-the-rest problem if the off-diagonal blocks of matrix $\mathbf{\Pi}_N$ are zeros. This means $\mathbf{y}_i^j$ and $\mathbf{y}_i^k$ ($k \neq j$) are decoupled. $\mathbf{y}_i^j$ is larger than the other $\mathbf{y}_i^k$ ($k \neq j$) if a point $\mathbf{x}_i$ belongs to class j , and $\mathbf{y}_i^j$ is smaller when $\mathbf{x}_i$ is other classes. Although we have modeled the probability of a point belonging to each class, the training phase scales to $O(C^3N^3)$ computational complexity. In [16], the authors point that this could be reduce to $O(CN^3)$ by using Woodbury formula. Further more, the inverse of matrix could be approximated by using Nyström method [17]. Thus, the training phase will reduce to $O(CNM^2)$ computational complexity, where M is the number of a small subset of the training points. On the contrary, the prediction phase has decoupled. We could calculate $\mathbf{y}_i^j$ for each class respectively. The prediction phase is $O(CNN_{test})$ computational complexity.

2.4 Hyper-parameter Estimation

This section mainly follows [16,20]. The hyper-parameter is the standard deviation σ of the RBF kernel. We estimate the hyper-parameter which minimizes the negative logarithmic likelihood:

$$\begin{aligned} J(\sigma) &= \log(-P(\mathbf{T}_N|\sigma)) \\ &\approx \sum_{i=1}^N \sum_{j=1}^C \mathbf{t}_i^j (\log \sum_{m=1}^C \exp \mathbf{y}_i^m - \mathbf{y}_i^j) + \frac{1}{2} \log |\mathbf{K}_N \mathbf{\Pi}_N + \mathbf{I}| + \frac{1}{2} \mathbf{Y}_N^T \mathbf{K}_N^{-1} \mathbf{Y}_N \end{aligned} \quad (13)$$

The derivation of the objective function (13) is:

$$\begin{aligned} \frac{\partial J(\sigma)}{\partial \sigma} &= \alpha_N^T \frac{\partial \mathbf{Y}_N}{\partial \sigma} + \frac{1}{2} \text{tr} \left[(\mathbf{I} + \mathbf{K}_N \mathbf{\Pi}_N)^{-1} \frac{\partial \mathbf{K}_N \mathbf{\Pi}_N}{\partial \sigma} \right] \\ &\quad + \frac{1}{2} \left[2 (\mathbf{K}_N^{-1} \mathbf{Y}_N)^T \frac{\partial \mathbf{Y}_N}{\partial \sigma} + \mathbf{Y}_N^T \frac{\partial \mathbf{K}_N^{-1}}{\partial \sigma} \mathbf{Y}_N \right] \end{aligned} \quad (14)$$

3 Experiments

3.1 Toy Data

We test the multi-class problem with a toy data. The training data set contains 7 classes. There are three Gaussian, two moon and two round shapes in the figure. As Fig.1 (a) shows, each of the Gaussian distribution has 25 points, and each

Fig. 1. A multi-class toy problem. The object value is plotted in (b) as blue line, and the derivation is the red dot line.

class of moon and round shape data has 50 points. Only 9 points are labeled. The hyper-parameter estimation result is shown in Fig.1 (b). Fig.1 (c) shows the mesh result of the sum of estimated function $\sum_j \exp y_j$ on the 2-D space. We can see that, the corresponding estimated $\exp y_j$ be very large if the nearby has training points. The labeled points can also affect the estimated result. See the round shape in figure. The round shape in Fig.1 (a) is not closed, so the estimated function is smaller when the point is far (measured by geodesic distance) from the labeled point on the manifold. Finally, the classification result of the test set are shown in Fig.1 (d).

3.2 Real Data

In this experiment, we test some state-of-the-art algorithms and ours on the real data sets. Some of the images are resized, and all the vectors in these data sets are normalized to the range from 0 to 1. The data sets are (More details of the settings are shown in Table 1):

1. USPS Data [9]: We choose digits “1” - “4”, and there are 1269, 929, 824, and 852 examples for each class.
2. 20-Newsgroups Data [18]: There are four topics: “autos”, “motorcycles”, “baseball” and “hockey”. The documents have been normalized in 3970 TFIDF vectors in 8014 dimensional space.
3. COIL-20 Data [12]: We pick the first 5 of the 20 objects to test the algorithms. Each object has 72 images, which were taken at pose intervals of 5 degrees.

4. ORL Face Data [13]: There are 10 different images of each of 40 distinct subjects in the ORL data set. We choose the 15th-30th subjects.
5. Yale Face Data [6]: This data set contains 165 images of 15 individuals.
6. UMIST Face Data [7]: This set consists of 564 images of 20 people. Each covering a range of poses from profile to frontal views. The first 10 subjects are selected.
7. UCI Image Segmentation data [3]: The instances of this set were drawn randomly from a database of 7 outdoor images. We choose 4 classes, and each class has totally 330 samples including both training and test data.
8. UCI Optical Recognition of Handwritten Digits [3]: We also choose the digits "1"-"4", and there are 571, 557, 572 and 568 examples for each class.

We test several algorithms for comparison: the supervised methods SVM (Support Vector Machine) [4] and RLS (Regularized Least Squares) [2]; transductive method TOG (Transduction On Graphs) [18]; the inductive method LapRLS (Laplacian Regularized Least Squares) [2]; and our method SSGP (Semi-Supervised Gaussian Process). SVM uses the one-against-one scheme, while RLS, LapRLS and TOG use the one-against-the-rest scheme. Each test accuracy of the results is an average of 50 random trials. For each trial, we randomly choose a subset of the data. The selected data are dealt as a splitting seen (include labeled and unlabeled data) and unseen data sets. For supervised methods, we only use the labeled points in the seen data for training. For semi-supervised inductive methods, we use all the seen data to classify the unseen. For TOG, we run two times for each iteration. First, we run it on the seen set, and evaluate the accuracy again on the seen set. Second, we use both the seen and unseen to train another TOG algorithm, and evaluate the accuracy on the unseen set. Moreover, we use the weight $W_{ij} = \exp(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2)$ to construct the graph. Unlike the other data sets, to deal with the 20-Newsgroups data, we construct a 10-NN weighted graph instead of a fully connected graph. The weight on the graph is changed to be $W_{ij} = \exp(-\frac{1}{2\sigma^2} (1 - \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|}))$. The empirically selected parameters are also shown in Table 1.

Fig.2 shows the results. We can see that, for data distributed on a manifold (Fig.2 (a) (c) (f) (g) (h)), the test accuracy of TOG is the best. This is because it

Table 1. Experimental Settings and Empirically Selected Parameters

Data Set	Class	Dim	N_{subset}	Seen	Unseen	σ_{Graph}	σ_{SVM}	C_{SVM}
USPS	4	256	1000	50%	50%	1.25	5	1
20-Newsgroups	4	8014	1000	50%	50%	0.15	10	1
COIL-20	5	4096	360	80%	20%	1.25	10	1
ORL	15	2576	150	80%	20%	1	10	1
YALE	15	4002	165	70%	30%	2.5	12	3
UMIST	15	2576	265	60%	40%	1	6	5
UCI Image	4	19	800	50%	50%	0.1	1.5	3
UCI Digits	4	64	800	50%	50%	0.15	1.5	1

Fig. 2. Real Data

uses both the seen and unseen data, and the unseen data provide more geometric information. For the data do not show explicit manifold information (Fig.2 (b) (d) (e)), the transductive methods on seen set give the best result. The results of SSGP on the seen set is competitive with TOG on the seen set. Moreover, SSGP is also competitive with the semi-supervised inductive method LapRLS. For most data sets, SSGP and LapRLS do better than the supervised methods SVM and RLS, since semi-supervised methods use the information provided by the unlabeled data. However, see the Yale data set for example, semi-supervised methods only do litter better than the supervised methods, even TOG training based on the whole set could not give much better result.

4 Conclusion

This paper proposes a novel graph based multi-class semi-supervised algorithm. It can work on both seen and unseen data. The accuracy rate is competitive with the existing transducitve and inductive methods. If the data present explicit

structure of a manifold, the graph based semi-supervised learning algorithms work efficiently for both transductive and inductive problems. In the future, we would like to do some research on the methods to speed up the algorithm.

References

1. Belkin, M. and Niyogi, P.: Using manifold structure for partially labeled classification. *NIPS* Vol.15, pp. 929-936, (2003).
2. Belkin, M., Niyogi, P. and Sindhvani, V.: On Manifold Regularization. *AI and Statistics*, pp. 17-24, (2005).
3. Blake, C. L. and Merz, C. J.: UCI Repository of Machine Learning Databases <http://www.ics.uci.edu/mllearn/MLRepository.html>.
4. Chang C., Lin C.: LIBSVM: A Library for Support Vector Machines. (2001).
5. Chung, F.: *Spectral Graph Theory*. No. 92 in Tegional Conference Series in Mathematics. American Mathematical Society (1997).
6. Georgiades A., Belhumeur P. and Kriegman D.: From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose. *IEEE Trans on PAMI*, Vol. 23(6), pp. 643-660, (2001).
7. Graham D. and Allinson N.: Characterizing Virtual Eigensignatures for General Purpose Face Recognition. In *Face Recognition: From Theory to Applications*, Vol. 163, pp. 446-456, (1998).
8. Delalleau, O. Bengio, Y. and Roux, N. L.: Efficient Non-Parametric Function Induction in Semi-Supervised Learning. *AI and Statistics*, pp. 96-103, (2005).
9. Hull, J.: A database for handwritten text recognition research. *IEEE Trans. on PAMI*, Vol. 16(5), pp. 550-554, (1994).
10. Lawrence, N. D. and Jordan, M. I.: Semi-supervised learning via Gaussian processes. *NIPS*, Vol. 17, pp. 753-760, (2005).
11. Mackay D.: Introduction to Gaussian processes. *Technical Report*, (1997).
12. Nene S. A., Nayar S. K. and Murase H.: Columbia Object Image Library (COIL-20), *Technical Report*, (1996).
13. ORL Face Database. <http://www.uk.research.att.com/facedatabase.html>.
14. Seeger M.: Relationships between Gaussian processes, Support Vector machines and Smoothing Splines. *Technical Report*, (1999).
15. Seeger, M.: Learning with Labeled and Unlabeled Data. *Technical Report*, (2000).
16. Williams, C. K. I. and Barber, D.: Bayesian Classification with Gaussian Processes. *IEEE Trans. on PAMI*, Vol. 20(12), pp. 1342-1351, (1998).
17. Williams C. and Seeger M.: Using the Nyström Method to Speed Up Kernel Machines. *NIPS*, Vol. 13, pp. 682-688, (2001).
18. Zhou, D., Bousquet, O., Lal, T. N., Weston, J. and Schölkopf, B.: Learning with Local and Global Consistency. *NIPS*, Vol. 16, pp. 321-328, (2003).
19. Zhu, X., Ghahramani, Z. and LaKerty, J.: Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. *ICML*, Vol. 20, pp. 912-919, (2003).
20. Zhu, X. and Ghahramani, Z.: Semi-Supervised Learning: From Gaussian Fields to Gaussian Processes. *Technical Report*, (2003).
21. Zhu, X.: Semi-Supervised Learning Literature Survey. *Technical Report*, (2005).