SMOOTHNESS MAXIMIZATION VIA GRADIENT DESCENTS

Bin Zhao, Fei Wang, Changshui Zhang

State Key Laboratory of Intelligent Technologies and Systems,
Department of Automation, Tsinghua University,
Beijing 100084, P.R.China

ABSTRACT

The recent years have witnessed a surge of interest in graph based semi-supervised learning. However, despite its extensive research, there has been little work on graph construction. In this study, employing the idea of gradient descent, we propose a novel method called *Iterative Smoothness Maximization* (*ISM*), to learn an *optimal* graph automatically for a semi-supervised learning task. The main procedure of *ISM* is to minimize the upper bound of semi-supervised classification error through an iterative gradient descent approach. We also prove the convergence of *ISM* theoretically, and finally experimental results on two real-world data sets are provided to demonstrate the effectiveness of *ISM*.

Index Terms— Semi-Supervised Learning (SSL), Cluster Assumption, Gaussian Function, Gradient Descent

1. INTRODUCTION

In many practical applications of pattern classification and data mining, one often faces a lack of sufficient labeled data, since labeling often requires expensive human labor and much time. However, in many cases, large number of unlabeled data can be far easier to obtain.

Consequently, *Semi-Supervised Learning (SSL)* methods, which aim to learn from partially labeled data, are proposed[1]. In general, these methods can be categorized into two classes: *transductive learning*(*e.g.* [2]) and *inductive learning*(*e.g.* [3]). The goal of transductive learning is to estimate the labels of the given unlabeled data, whereas inductive learning tries to induce a decision function which has a low error rate on the whole sample space.

In recent years, *graph based semi-supervised learning* has become one of the most active research areas in *SSL* community [4]. The key to graph based *SSL* is the *cluster assumption* [5]. It states that (1) nearby points are likely to have the same label; (2) points on the same structure (such as a cluster or a submanifold) are likely to have the same label.

Based on the above assumptions, graph-based *SSL* uses a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ to describe the structure of a data set, where $\mathcal{V}$ is the node set corresponding to the labeled and unlabeled examples in the data set, and $\mathcal{E}$ is the edge set. In most of the traditional methods[2][5][6], each edge $e_{ij} \in \mathcal{E}$ is associated with a weight, usually computed by *Gaussian functions* which reflect the similarities between pairwise data points, *i.e.*

$$w_{ij} = exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)\right) \tag{1}$$

However, as pointed out by [1], although graph construction is at the heart of graph based *SSL*, it is still a problem that has not been well studied. More concretely, the variance σ in Eq.(1) can affect the final classification result significantly, which can be seen from the toy example shown in Fig.1, but according to [2], there has been no reliable approach that can determine an optimal σ automatically so far.

To address such a problem, we propose a gradient based method, *Iterative Smoothness Maximization* (*ISM*), which aims at learning both data labels and the optimal hyperparameter of the *Gaussian function* for constructing the graph. The *ISM* algorithm first establishes a cost function composed of two parts, *i.e.* the *smoothness* and the *fitness* of the data labels, to measure how good the classification result of the *Semi-Supervised Learning* task is. Then *ISM* will minimize this cost function by alternating the *smoothness maximization* step and the *graph reconstruction* step. We will prove theoretically the convergence of the *ISM* algorithm.

The rest of this paper is organized as follows. In section 2, we introduce some works related to this paper. The *ISM* algorithm is presented in detail in section 3. In section 4, we analyze the convergence property of the algorithm. Experimental results on two real-world data sets are provided in section 5, followed by the conclusions and future works in section 6.

2. NOTATIONS AND RELATED WORKS

In this section we will introduce some notations and briefly review some related works of this paper.

Given a point set $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_l, \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$ and a label set $\mathcal{L} = \{1, \cdots, c\}$, where the first l points in $\mathcal{X}$ are labeled as $t_i \in \mathcal{L}$, while the remaining points are unlabeled. Our goal is to predict the labels of the unlabeled points[1]. We

[1]In this paper we will concentrate on the transductive setting. One can easily extend our algorithm to inductive setting using the method introduced in [7].

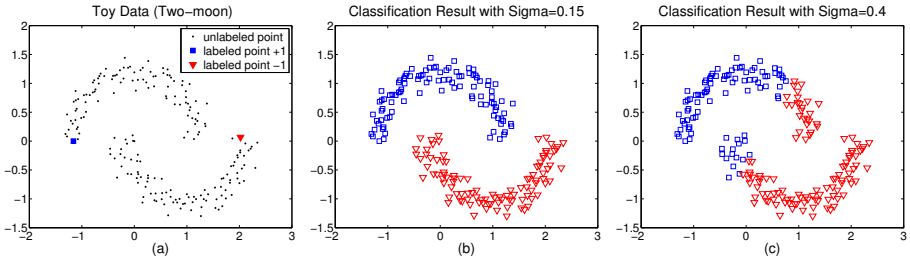

Fig. 1. Classification results on the two-moon pattern using the method in [2], a powerful transductive approach operating on graph with the edge weights computed by a Gaussian function. (a) toy data set with two labeled points; (b) classification results with $\sigma = 0.15$; (c) classification results with $\sigma = 0.4$. We can see that a small variation of σ will cause a dramatically different classification result.

denote the initial labels in data set by an $n \times c$ matrix T. For each labeled point $\mathbf{x}_i$, $T_{ij} = 1$ if $\mathbf{x}_i$ is labeled as $t_i = j$ and $T_{ij} = 0$ otherwise. For unlabeled points, the corresponding rows in T will be zero. The classification result on the data set $\mathcal{X}$ is also represented as an $n \times c$ matrix $F = [F_1^T, \ldots, F_n^T]^T$, which determines the label of $\mathbf{x}_i$ by $t_i = \arg\max_{1 \le j \le c} F_{ij}$. In *graph based semi-supervised learning*, we construct the $n \times n$ *weight matrix* W for graph $\mathcal{G}$ with its (i, j)-th entry $W_{ij} = w_{ij}$ computed by Eq.(1) , and $W_{ii} = 0$. The *degree matrix* D for graph $\mathcal{G}$ is defined as an $n \times n$ matrix with its (i, i)-entry equal to the sum of the i-th row of W.

Based on the above preliminaries, *Zhou et al* proposed the *Learning with Local and Global Consistency* (*LLGC*) algorithm to tackle the *SSL* problem by minimizing the following cost function[2]:

$$Q = \frac{1}{2}\left[\sum_{i,j=1}^{n} W_{ij} \left\| \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} \right\|^2 + \mu \sum_{i=1}^{n} \|F_i - T_i\|^2 \right] \tag{2}$$

The first term measures the *smoothness* of the data labels, and the second term restricts that a good classifying function should not change too much from the initial label assignment. The regularization parameter $\mu > 0$ adjusts the tradeoff between these two terms. Thus, the optimal classification function can be obtained by: $F^* = \arg\min_F Q$.

As we noted in section 1, one of the problems existing in these graph based methods is that the hyperparameter (*i.e.* σ in Eq.(1)) can affect the final classification results significantly. Therefore, many methods have been proposed to determine the optimal hyperparameter automatically, such as the *Local Scaling* [8] and *Minimum Spanning Tree* [6] methods. Although they can work well empirically, they are heuristic, and thus lack a theoretical foundation.

3. ITERATIVE SMOOTHNESS MAXIMIZATION

In this section we will introduce the main procedure of the *Iterative Smoothness Maximization* (*ISM*) algorithm.

3.1. Gradient Computing

We propose to learn the optimal σ through a gradient descent procedure. More concretely, employing the cost function proposed in Eq.(2), we can compute the gradient of $Q(F, \sigma)$ w.r.t. σ with F fixed as $F^* = \arg\min_F Q$.

$$\begin{aligned}
\frac{\partial Q(F,\sigma)}{\partial \sigma} &= \frac{\partial}{\partial \sigma}\left[\mu \sum_{j=1}^{n} ||F_j - T_j||^2 + \sum_{i,j=1}^{n} W_{ij}\left(\frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}}\right)^2\right] \\
&= \sum_{i,j=1}^{n} \frac{\partial}{\partial \sigma}\left[W_{ij}\left(\frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}}\right)^2\right] \\
&= \sum_{i,j=1}^{n} \left\{ \frac{\partial W_{ij}}{\partial \sigma}\left[\frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}}\right]^2 - W_{ij}\left[\frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}}\right] \right. \\
&\quad \left. \cdot \left[\frac{F_i}{\sqrt{D_{ii}^3}}\frac{\partial D_{ii}}{\partial \sigma} - \frac{F_j}{\sqrt{D_{jj}^3}}\frac{\partial D_{jj}}{\partial \sigma}\right] \right\}
\end{aligned} \tag{3}$$

Using the *Gaussian function* to measure similarity, we get

$$\frac{\partial W_{ij}}{\partial \sigma} = \frac{\partial \exp(-\frac{d_{ij}^2}{2\sigma^2})}{\partial \sigma} = \frac{d_{ij}^2 \exp(-\frac{d_{ij}^2}{2\sigma^2})}{\sigma^3} \tag{4}$$

$$\frac{\partial D_{ii}}{\partial \sigma} = \frac{\partial \sum_j W_{ij}}{\partial \sigma} = \sum_j \frac{\partial W_{ij}}{\partial \sigma} = \sum_j \frac{d_{ij}^2 \exp(-\frac{d_{ij}^2}{2\sigma^2})}{\sigma^3} \tag{5}$$

3.2. Learning Rate Selection in Gradient Descent

In gradient descent, the hyperparameter is updated as $\sigma_{new} = \sigma_{old} - \eta \frac{\partial Q}{\partial \sigma}|_{\sigma=\sigma_{old}}$; learning rate η affects the performance of gradient descent severely. In *ISM*, η is selected dynamically to accelerate convergence of the algorithm.

If we fix F, the cost function Q only varies with σ. Therefore, we plot the cost function curve in Fig.2, in which the current value of σ is denoted by point A. In the case where

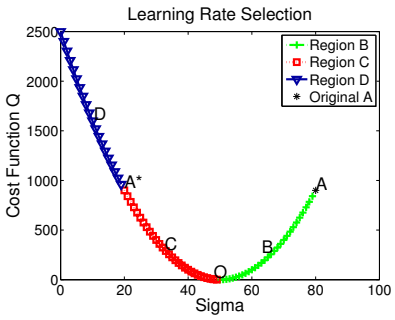

Fig. 2. Possible regions σ might appear during iteration, where B represents the region between A and the minimum point, O. Regions C and D are separated by point A* satisfying $Q(\sigma_{A*}) = Q(\sigma_A)$.

the cost function has multiple local minima, this figure could be considered as a local region of the cost function curve.

After updating σ with gradient descent, its new value $\hat{\sigma}_1$ might appear in three regions : B, C or D. In *ISM*, we hope the cost function decreases monotonically to guarantee the convergence of the algorithm. Also, to simplify our method, we hope the value of σ avoids oscillation. Hence, B is the ideal region for $\hat{\sigma}_1$, and $\hat{\sigma}_1$ should be near the minimum point O. If $\hat{\sigma}_1$ appears in region C or D, then the learning rate is too large and we set η to equal η_s, the value of which is small enough to guarantee that with η_s, the new variance σ appears in region B. On the other hand, if $\hat{\sigma}_1$ appears in region B, we increase the learning rate by doubling it and calculate $\hat{\sigma}_2$ with the new η. If $\hat{\sigma}_2$ appears in region C or D, we resume the original learning rate and output $\hat{\sigma}_1$ as the variance in this iteration; otherwise, we output $\hat{\sigma}_2$.

3.3. Implementation of Iterative Smoothness Maximization

The implementation details of the *ISM* algorithm is as following:

1. Initialization. $\sigma = \sigma_0$, total iteration steps N_0, initial learning rate η_0 and small learning rate η_s[2].
2. Calculate the optimal F. $\frac{\partial Q}{\partial F} = 0 \Rightarrow F_{n+1} = (1-\alpha)(I - \alpha S(\sigma_n))^{-1}T$ where $S = D^{-1/2}WD^{-1/2}$ and $\alpha = 1/(1+\mu)$.[3][2]
3. Update σ with gradient descent and adjust learning rate η. $\hat{\sigma_1} = \sigma_n - \eta \frac{\partial Q}{\partial \sigma}|_{\sigma=\sigma_n}$
   (a) If $Q(F_{n+1}, \hat{\sigma_1}) < Q(F_{n+1}, \sigma_n)$ and $\mathrm{sgn}\left(\frac{\partial Q}{\partial \sigma}|_{\sigma=\hat{\sigma_1}}\right) = \mathrm{sgn}\left(\frac{\partial Q}{\partial \sigma}|_{\sigma=\sigma_n}\right)$, then $\eta = 2\eta$, $\hat{\sigma_2} = \sigma_n - \eta \frac{\partial Q}{\partial \sigma}|_{\sigma=\sigma_n}$.
      i. If $Q(F_{n+1}, \hat{\sigma_2}) < Q(F_{n+1}, \sigma_n)$ and $\mathrm{sgn}\left(\frac{\partial Q}{\partial \sigma}|_{\sigma=\hat{\sigma_2}}\right) = \mathrm{sgn}\left(\frac{\partial Q}{\partial \sigma}|_{\sigma=\sigma_n}\right)$, then $\sigma_{n+1} = \hat{\sigma_2}$
      ii. Else $\sigma_{n+1} = \hat{\sigma_1}$, $\eta = \eta/2$
   (b) Else $\eta = \eta_s$, $\sigma_{n+1} = \sigma_n$
4. If $n > N_0$, quit iteration and output classification result; else, go to step 2.

[2]In order to guarantee the convergence of the algorithm, η_s is set to be close to 0.

[3]The parameter α used in our method is simply fixed at 0.99.[2]

4. CONVERGENCE STUDY OF ITERATIVE SMOOTHNESS MAXIMIZATION

Since the algorithm proposed here is iterative, it is crucial to study its convergence property. Without loss of generality, we only consider binary classification here, which can be easily extended to the multi-class case. The cost function could be written as $Q(f, \sigma) = f^T(I - S)f + \mu(f - t)^T(f - t)$.

$$\frac{\partial Q}{\partial f} = 0 \Leftrightarrow f^* = (1-\alpha)(I - \alpha S)^{-1}t \tag{6}$$

While the Hessian matrix is as follows:

$$H = \frac{\partial^2 Q}{\partial f^2} = I - S + \mu I \tag{7}$$

$\forall x \in R^n, x \neq 0$,

$$\begin{aligned} x^T H x &= x^T(I - S)x + \mu x^T x \\ &= \sum_{i,j=1}^{n} W_{ij}\left(\frac{1}{\sqrt{D_{ii}}}x_i - \frac{1}{\sqrt{D_{jj}}}x_j\right)^2 + \mu\sum_{i=1}^{n} x_i^2 > 0 \end{aligned} \tag{8}$$

Hence, the Hessian matrix is positive definite, moreover, f^* is the unique zero point of $\frac{\partial Q}{\partial f}$. Therefore, f^* is the global minimum of Q with σ fixed. The property of f^* being the global minimum of Q leads to the following inequality: $Q(f_{n+1}, \sigma_n) < Q(f_n, \sigma_n)$, where $f_{n+1} = f^*$. In step 3 of *ISM*, σ_{n+1} is calculated as $\hat{\sigma}_{n+1} = \sigma_n - \eta \frac{\partial Q}{\partial \sigma_n}$, with the learning rate in gradient descent selected to guarantee that $Q(f_{n+1}, \sigma_{n+1}) < Q(f_{n+1}, \sigma_n)$. Thus, in every iteration of *ISM*, the following inequality is guaranteed:

$$Q(f_{n+1}, \sigma_{n+1}) < Q(f_{n+1}, \sigma_n) < Q(f_n, \sigma_n) \tag{9}$$

which implies that the cost function $Q(\sigma, f)$ decreases monotonically. Moreover, since $Q(\sigma, f) > 0$, Q is lower bounded by zero. Hence, the algorithm is guaranteed to converge. Actually, our method converges after 29 steps of iteration on Two-moon data set with $\sigma_0 = 1$.

5. EXPERIMENTS

We will validate our method on two real world data sets in this section.

5.1. Digit Recognition

In this experiment, we will focus on classifying handwritten digits. We use images of digits 1, 2, 3 and 4 from USPS[4] handwritten 16×16 digits data set and there are 1005, 731, 658 and 652 samples in each class, with a total of 3046. We employ *Nearest Neighbor* classifier and *SVM*[9] as baselines. For comparison, we also provide the classification results of

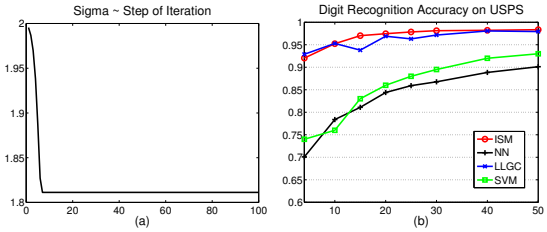

Fig. 3. Classification results on USPS data. (a) Changes of σ during iteration (b) Recognition accuracies, with the horizontal axis representing the number of randomly labeled samples.

LLGC method[2] in which the affinity matrix is constructed by a *Gaussian function* with variance 1.25, which is tuned with grid search. The number of labeled samples increases from 4 to 50 and the test errors averaged over 50 random trials are summarized in Fig.3(b), from which we can clearly see the advantages of *ISM* and *LLGC*. And obviously the hyperparameter in *LLGC* resulted from grid search is suboptimal.

5.2. Object Recognition

In this section, we will address the task of object recognition using the COIL-20[5] data set, a database of gray-scale images of 20 objects. For each object there are totally 72 images of size 128×128. Similar as digit recognition, we employ *Nearest Neighbor* classifier and *SVM* as baselines. For comparison, we also test the recognition accuracy of *LLGC* method in which the affinity matrix is constructed by a Gaussian function with variance 40, which is also tuned with grid search. The number of labeled samples per class increases from 2 to 18 and the test errors averaged over 50 random trials are summarized in Fig.4(b), from which we can clearly see the advantages of *ISM* and *LLGC*, while in *ISM* the hyperparameter σ is determined automatically.

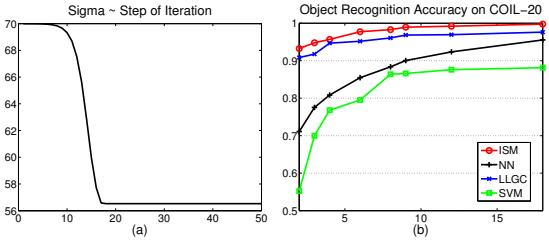

Fig. 4. Object recognition results on COIL-20 data. (a) Changes of σ during iteration (b) Recognition accuracies, with the horizontal axis representing the number of randomly labeled samples per class.

[4]http://www.kernel-machines.org/data.html

[5]http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php

6. CONCLUSIONS AND FUTURE WORKS

Employing the idea of gradient descent, we propose the *Iterative Smoothness Maximization* algorithm in this paper, to learn an optimal graph automatically for a semi-supervised learning task. Moreover, the algorithm is guaranteed to converge. Experimental results on both toy and real-world data sets show the effectiveness of *ISM* for parameter selection when only few labeled samples are provided. Although we focus on learning hyperparameter σ in the *Gaussian function*, our method can be applied to other kernel functions as well.

Besides the significant advantages of *ISM*, there are still certain aspects that we can research more to improve the efficiency of our method. This includes employing more advanced optimization algorithms to speed up convergence and extending *ISM* to allow the use of different σ along different directions.

7. ACKNOWLEDGEMENT

This work is supported by the project (60675009) of the National Natural Science Foundation of China.

8. REFERENCES

[1] X. Zhu, "Semi-supervied learning literature survey," *Computer Sciences Technical Report, 1530, University of Wisconsin-Madison*, 2006.

[2] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf, "Learning with local and global consistency," *Advances in Neural Information Processing Systems*, 2003.

[3] M. Belkin, P. Niyogi, and V. Sindhwani, "On manifold regularization," *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[4] O. Chapelle, B. Scholkopf, and A. Zien, *Semisupervised Learning*, MIT Press: Cambridge, MA, 2006.

[5] O. Chapelle, J. Weston, and B. Scholkopf, "Cluster kernels for semi-supervised learning," *Advances in Neural Information Processing Systems 15*, 2003.

[6] X. Zhu, *Semi-Supervised Learning with Graphs*, Doctoral thesis, Carnegie Mellon University, May 2005.

[7] O. Delalleu, Y. Bengio, and N. Le Roux, "Non-parametric function induction in semi-supervised learning," *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[8] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," *Advances in Neural Information Processing Systems*, 2004.

[9] B. Scholkopf and A. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2006.