Learning from Labeled and Unlabeled Data Using Random Walks

Dengyong Zhou and Bernhard Schölkopf

Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tuebingen, Germany
{dengyong.zhou, bernhard.schoelkopf}@tuebingen.mpg.de

Abstract. We consider the general problem of learning from labeled and unlabeled data. Given a set of points, some of them are labeled, and the remaining points are unlabeled. The goal is to predict the labels of the unlabeled points. Any supervised learning algorithm can be applied to this problem, for instance, Support Vector Machines (SVMs). The problem of our interest is if we can implement a classifier which uses the unlabeled data information in some way and has higher accuracy than the classifiers which use the labeled data only. Recently we proposed a simple algorithm, which can substantially benefit from large amounts of unlabeled data and demonstrates clear superiority to supervised learning methods. Here we further investigate the algorithm using random walks and spectral graph theory, which shed light on the key steps in this algorithm.

1 Introduction

We consider the general problem of learning from labeled and unlabeled data. Given a set of points, some of them are labeled, and the remaining points are unlabeled. The task is to predict the labels of the unlabeled points. This is a setting which is applicable to many real-world problems. We generally need to also predict the labels of the testing points which are unseen before. However, in practice, we almost always can add the new points into the set of the unlabeled data.

Any learning algorithm can be applied to this problem, especially supervised learning methods, which train the classifiers with the labeled data and then use the trained classifiers to predict the labels of the unlabeled data. At present, one of the most popular supervised learning methods is the Support Vector Machine (SVM) [9]. The problem of interest here is if we can implement a classifier which uses the unlabeled data in some way and has higher accuracy than the classifiers which use the labeled data only [10].

Such a learning problem is often called semi-supervised. Since labeling often requires expensive human labor, whereas unlabeled data is far easier to obtain, semi-supervised learning is very useful in many real-world problems and has recently attracted a considerable amount of research. A typical application is web categorization, in which manually classified web pages are always a very

small part of the entire web, but the number of unlabeled examples can be almost as large as you want.

Recently we proposed a simple algorithm, which can substantially benefit from large amounts of unlabeled data and works much better than the supervised learning methods [11]. Here we further investigate the algorithm using random walks and spectral graph theory, which shed light on some key steps in this algorithm, especially normalization.

The paper is organized as follows. In Section 2 we describe our semi-supervised learning algorithm in details. In Section 3 the method is interpreted in the framework of lazy random walks. In Section 4 we define calculus on discrete objects and then build the regularization framework of the method upon the discrete calculus. In Section 5 we use a toy problem to highlight the key steps in the method, and also validate the method on a large-scale real-world dataset.

2 Algorithm

Given a point set $\mathcal{X} = \{x_1, \ldots, x_l, x_{l+1}, \ldots, x_n\} \subset \mathbb{R}^m$ and a label set $\mathcal{L} = \{-1, 1\}$, the first l points $x_i (i \leq l)$ are labeled as $y_i \in \mathcal{L}$ and the remaining points $x_u (l+1 \leq u \leq n)$ are unlabeled. Define a $n \times 1$ vector y with $y_i = 1$ or -1 if x_i is labeled as positive or negative, and 0 if x_i is unlabeled. We can view y as a real-valued function defined on $\mathcal{X}$, which assigns a value y_i to point x_i. The data is classified as follows:

1. Define a $n \times n$ affinity matrix W in which the elements are nonnegative, symmetric, and furthermore the diagonal elements are zeros.
2. Construct the matrix $S = D^{-1/2} W D^{-1/2}$ in which D is a diagonal matrix with its (i, i)-element equal to the sum of the i-th row of W.
3. Compute $f = (I - \alpha S)^{-1} y$, where I denotes the identity matrix and α is a parameter in $(0, 1)$, and assign a label $\mathrm{sgn}(f_i)$ to point x_i.

The affinity matrix can typically be defined by a Gaussian $W_{ij} = \exp(-\|x_i - x_j\|^2 / 2\sigma^2)$ except that $W_{ii} = 0$, where $\|\cdot\|$ represents Euclidean norm. We would like to emphasize the affinity matrix have not to be derived from a kernel [8]. For instance, construct a k-NN or ϵ-ball graph on data, and then define $W_{ij} = 1$ if points x_i and x_j are connected by an edge, and 0 otherwise. Note that in this case the requirement $W_{ii} = 0$ is satisfied automatically since there is no self-loop edge.

3 Lazy Random Walks

In this section we interpret the algorithm in terms of random walks inspired by [6]. We will see that this method simply classifies the points by comparing a specific distance measure between them and the labeled points of different classes.

Let $\Gamma = (V, E)$ denote a graph with a set V of n vertices indexed by number from 1 to n and an edge collection E. Assume the graph is undirected and

connected, and has no self-loops or multiple edges. A weight function $w : V \times V \to \mathbb{R}$ associated to the graph satisfies $w(i,j) = w(j,i)$, and $w(i,j) \geq 0$. Moreover, define $w(i,j) = 0$ if there is no edge between i and j. The degree d_i of vertex i is defined to be

$$d_i = \sum_{j \sim i} w(i,j), \tag{3.1}$$

where $j \sim i$ denotes the set of the points which are linked to point i.

Let D denote the diagonal matrix with the (i,i)-th entry having value d_i . Let W denote the matrix with the entries $W_{ij} = w(i,j)$. A lazy random walk on the graph is decided by the transition probability matrix $P = (1-\alpha)I + \alpha D^{-1}W$. Here α is a parameter in $(0,1)$ as before. This means, with the probability α, following one link which connects the vertex of the current position and is chosen with the probability proportional to the weight of the link, and with the probability $1-\alpha$, just staying at the current position.

There exists a unique stationary distribution $\pi = [\pi_1, \ldots, \pi_n]$ for the lazy random walk, i.e. a unique probability distribution satisfying the balance equation

$$\pi = \pi P. \tag{3.2}$$

Let $\mathbf{1}$ denote the $1 \times n$ vector with all entries equal to 1. Let vol Γ denote the volume of the graph, which is defined by the sum of vertex degrees. It is not hard to see that the stationary distribution of the random walk is

$$\pi = \mathbf{1}D/\text{vol } \Gamma. \tag{3.3}$$

Note that π does not depend on α.

Let X_t denote the position of the random walk at time t. Write $T_{ij} = \min\{t \geq 0 | X_t = x_j, X_0 = x_i, x_i \neq x_j\}$ for the *first hitting time* to x_j with the initial position x_i, and write $T_{ii} = \min\{t > 0 | X_t = x_i, X_0 = x_i\}$, which is called the *first return time* to x_i [1]. Let H_{ij} denote the expected number of steps required for a random walk to reach x_j with an initial position x_i, i.e. H_{ij} is the expectation of T_{ij}. H_{ij} is often called the *hitting time*. Let C_{ij} denote the expected number of steps for a random walk starting at x_i to reach x_j and then return, i.e. $C_{ij} = H_{ij} + H_{ji}$, which is often called the *commute time* between x_i and x_j. Clearly, C_{ij} is symmetrical, but H_{ij} may be not.

Let G denote the inverse of the matrix $D - \alpha W$. Then the commute time satisfies [6]:

$$C_{ij} \propto G_{ii} + G_{jj} - G_{ij} - G_{ji}, \text{ if } x_i \neq x_j, \tag{3.4}$$

and [1]

$$C_{ii} = 1/\pi_i. \tag{3.5}$$

The relation between G and C is similar to the inner product and the norm in Euclidean space. Let $\langle x_i,\ x_j \rangle$ denote the Euclidean inner product between x_i and x_j. Then the Euclidean norm of the vector $x_i - x_j$ satisfies

$$\|x_i - x_j\|^2 = \langle x_i - x_j,\ x_i - x_j \rangle = \langle x_i,\ x_i \rangle + \langle x_j,\ x_j \rangle - \langle x_i,\ x_j \rangle - \langle x_j,\ x_i \rangle.$$

In other words, we can think of G as a Gram matrix which specifies a kind of inner product on the dataset. The commute time is the corresponding norm derived from this inner product.

Note that H_{ij} is quite small whenever x_j is a node with a large stationary probability π_j. Thus we naturally consider to normalize H_{ij} by

$$\bar{H}_{ij} = \sqrt{\pi_i \pi_j} H_{ij}. \tag{3.6}$$

Accordingly the normalized commute time is

$$\bar{C}_{ij} = \bar{H}_{ji} + \bar{H}_{ij}. \tag{3.7}$$

Let $\bar{G}$ denote the inverse of the matrix $I - \alpha S$. Then the normalized commute time satisfies

$$\bar{C}_{ij} \propto \bar{G}_{ii} + \bar{G}_{jj} - \bar{G}_{ij} - \bar{G}_{ji}. \tag{3.8}$$

Noting the equality (3.5), we have

$$\bar{G}_{ij} = \frac{G_{ij}}{\sqrt{C_{ii} C_{jj}}}, \tag{3.9}$$

which is parallel to the normalized Euclidean product $\langle x_i, x_j \rangle / \|x_i\| \|x_i\|$ or cosine.

Let

$$p_+(x_i) = \sum_{\{j | y_j = 1\}} \bar{G}_{ij}, \text{ and } p_-(x_i) = \sum_{\{j | y_j = -1\}} \bar{G}_{ij}. \tag{3.10}$$

Then the classification given by $f = (I - \alpha S)^{-1} y$ is simply checking which of the two values $p_+(x_i)$ or $p_-(x_i)$ is larger, which is in turn comparing the normalized commute times to the labeled points of different classes.

If we just want to compare the non-normalized commute times to the different class labeled points, then the classification is given by $f = (D - \alpha W)^{-1} y$. Although the normalized commute time seems to be a more reasonable choice, there is still lack of the statistical evidence showing the superiority of the normalized commute time to the non-normalized one. However, we can construct a subtle toy problem (see Section 5) to essentially expose the necessity of normalization.

4 Regularization Framework

In this section we define *calculus* on graphs inspired by spectral graph theory [4] and [3]. A regularization framework for classification problems on graphs then can be naturally built upon the discrete calculus, and the algorithm derived from the framework is exactly the method presented in Section 2.

Let $\mathcal{F}$ denote the space of functions defined on the vertices of graph Γ, which assigns a value f_i to vertex i. We can view f as a $n \times 1$ vector. The *edge derivative* of f along the edge $e(i, j)$ at the vertex i is defined to be

$$\left.\frac{\partial f}{\partial e}\right|_i = \sqrt{w(i,j)} \left(\frac{1}{\sqrt{d_i}} f_i - \frac{1}{\sqrt{d_j}} f_j \right). \tag{4.1}$$

Clearly

$$\left.\frac{\partial f}{\partial e}\right|_i = -\left.\frac{\partial f}{\partial e}\right|_j . \tag{4.2}$$

The definition (4.1) in fact splits the function value at each point among the edges incident with it before computing the local changes of the function, and the value assigned to each edge is proportional to its weight. This statement can be clearer if we rewrite (4.1) as

$$\left.\frac{\partial f}{\partial e}\right|_i = \sqrt{\frac{w(i,j)}{d_i}} f_i - \sqrt{\frac{w(i,j)}{d_j}} f_j .$$

The *local variation* of function f at each vertex i is then defined by:

$$\left\|\nabla_i f\right\| = \sqrt{\sum_{e \vdash i} \left(\left.\frac{\partial f}{\partial e}\right|_i\right)^2}, \tag{4.3}$$

where $e \vdash i$ means the set of the edges incident with vertex i. The *smoothness* of function f is then naturally measured by the sum of the local variations at each point:

$$\mathcal{S}(f) = \frac{1}{2} \sum_i \left\|\nabla_i f\right\|^2 . \tag{4.4}$$

The graph *Laplacian* is defined to be [4]

$$\Delta = D^{-1/2}(D - W)D^{-1/2} = I - D^{-1/2} W D^{-1/2} = I - S, \tag{4.5}$$

where S is defined to be $S = D^{-1/2} W D^{-1/2}$. The Laplacian can be thought of as an operator defined on the function space:

$$\left.\Delta f\right|_i = \frac{1}{\sqrt{d_i}} \sum_{i \sim j} w(i,j) \left(\frac{1}{\sqrt{d_i}} f_i - \frac{1}{\sqrt{d_j}} f_j \right). \tag{4.6}$$

The smallest eigenvalue of the Laplacian is zero because the largest eigenvalue of S is 1. Hence the Laplacian is symmetric and positive semi-definite. Let $\mathbf{1}$ denote the constant function which assumes the value 1 on each vertex. We can view $\mathbf{1}$ as a column vector. Then $D^{-1/2}\mathbf{1}$ is the eigenvector corresponding to the smallest eigenvalue of Δ. Most importantly, we have the following equality

$$f^T \Delta f = \mathcal{S}(f), \tag{4.7}$$

which exposes the essential relation between the Laplacian and the gradient.

For the classification problem on graphs, it is natural to define the cost function associated to a classification function f to be

$$\arg\min_{f \in \mathcal{F}} \left\{ \mathcal{S}(f) + \frac{\mu}{2} \left\| f - y \right\|^2 \right\}. \tag{4.8}$$

The first term in the bracket is called the *smoothness term* or *regularizer*, which requires the function to be as smooth as possible. The second term is called the *fitting term*, which requires the function to be as close to the initial label assignment as possible. The trade-off between these two competitive terms are captured by a positive parameter μ. It is not hard to show that the solution of (4.8) is

$$f = (1 - \alpha)(I - \alpha S)^{-1} y, \tag{4.9}$$

where $\alpha = 1/(1 + \mu)$. Clearly, it is equivalent to $f = (I - \alpha S)^{-1} y$.

Finally, we discuss the non-normalized variant of the definition of edge derivative:

$$\left.\frac{\partial f}{\partial e}\right|_i = \sqrt{w(i,j)}\big(f_i - f_j\big).$$

If we further define the graph Laplacian to be $L = D - W$, then the equality (4.7) still holds. Substituting the local variation based on the non-normalized edge derivative into the optimization problem (4.8), we then can obtain a different closed form solution $f = \mu(\mu I + L)^{-1} y$, which is quite close to the algorithm proposed by [2]. In Section 5, we will provide the experimental evidence to demonstrate the superiority of the algorithm based on the normalized edge derivative (4.1).

5 Experiments

5.1 Toy Problem

Shown in Figure 1(a) is the doll toy data, in which the density of the data varies substantially across different clusters. A similar toy dataset was used by [7] for clustering problems. The affinity matrix is defined by a Gaussian. The result given by the algorithm of $f = (D - \alpha W)^{-1} y$ derived from non-normalized commute time is shown in Figure 1(b). The result given by the algorithm $f = (\mu I + L)^{-1} y$ derived from non-normalized edge derivative is shown in Figure 1(c). Obviously, both methods fail to capture the coherent clusters aggregated by the data. The result given by the algorithm $f = (I - \alpha S)^{-1} y$, presented in Section 2, which can be derived from both normalized commute time and edge derivative is shown in Figure 1(d). This method sensibly classifies the dataset according with the global data distribution.

In addition, we use the toy problem to demonstrate the importance of zero diagonal in the first step of the standard algorithm. If we define the affinity matrix using a RBF kernel without removing the diagonal elements, the result is shown in Figure 1(e). The intuition behind setting the diagonal elements to zero is to avoid self-reinforcement.

Finally, we investigate the fitting term of the regularization framework using the toy problem. Note that we assign *a prior* label 0 to the unlabeled points in the fitting term. This is different from the regularization frameworks of supervised learning methods, in which the fitting term is only for the labeled points. If we remove the fitting on the unlabeled points, the result is given in Figure 1(f).

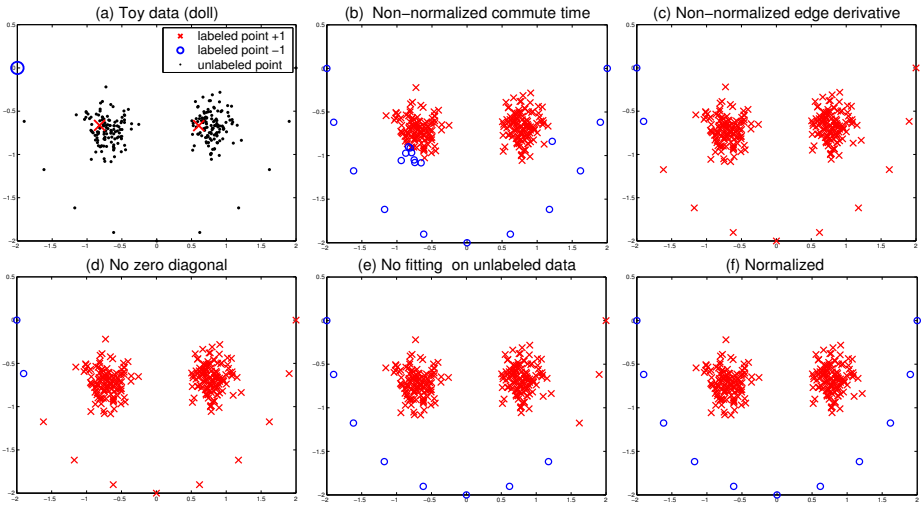

Fig. 1. Classification on the doll toy data. Both methods in (b) and (c) without normalization fail to classify the points according with the coherent clusters. The importance of zero diagonal and the fitting on the unlabeled data are demonstrated respectively in (d) and (e). The result from the standard algorithm is shown in (f).

The intuition behind the fitting on the unlabeled points is to make the algorithm more stable.

5.2 Digit Recognition

we addressed a classification task using the USPS dataset containing 9298 handwritten digits. Each digit is a 16x16 image, represented as a 256 dimensional vector with entries in the range from -1 to 1.

We used k-NN [5] and one-vs-rest SVMs [8] as baselines. Since there is no reliable approach for model selection if only very few labeled points are available, we chose the respective optimal parameters of these methods. The k in k-NN was set to 1. The width of the RBF kernel for the SVM was set to 5. The affinity matrix used in our method was derived from a RBF kernel with its width equal to 1.25. In addition, the parameter α was set to 0.95.

The test errors for different methods with the number of labeled points increasing from 10 to 100 are summarized in the left panel of Figure 2, in which each error point is averaged over 100 random trials, and samples are chosen so that they contain at least one labeled point for each class. The results shows clear superiority of our algorithm (marked as *random walk*) over the supervised learning methods k-NN and SVMs. The right panel of Figure 2 shows how the parameter α influences the performances of the method, in which the number of labeled points is fixed at 50. Obviously, this method is not sensitive to the value of α.

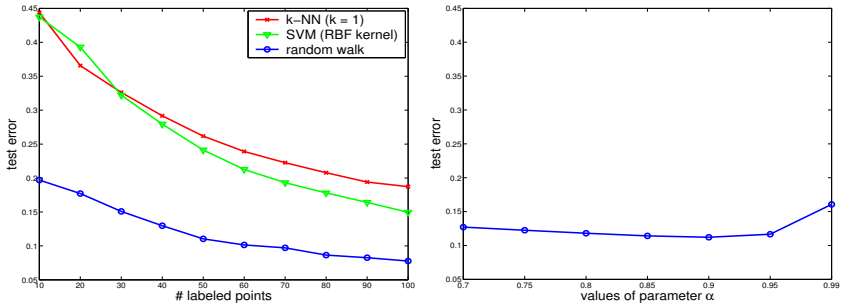

Fig. 2. Digit recognition with USPS handwritten 16x16 digits dataset for a total of 9298. The left panel shows test errors for different algorithms with the number of labeled points increasing from 10 to 100. The right panel shows how the different choices of the parameter α influence the performance of our method (with 50 labeled points).

Acknowledgments We would like to thank Arthur Gretton for helpful discussions on normalized commute time in random walks.

References

1. D. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs*. In Preparation, http://stat-www.berkeley.edu/users/aldous/RWG/book.html.
2. M. Belkin, I. Matveeva, and P. Niyogi. Regression and regularization on large graphs. Technical report, University of Chicago, 2003.
3. T. Chan and J. Shen. Variational restoration of non-flat image features: Models and algorithms. *SIAM Journal of Applied Mathematics*, 61(4):1338–1361, 2000.
4. F. Chung. *Spectral Graph Theory*. Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, 1997.
5. P. A. Devijver and J. Kittier. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, London, 1982.
6. J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.
7. A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.
8. B. Schölkopf and A. J. Smola. *Learning with kernels*. MIT Press, Cambridge, MA, 2002.
9. V. N. Vapnik. *Statistical learning theory*. Wiley, NY, 1998.
10. T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.
11. D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.