LEARNING CLASSIFIERS ON A PARTIALLY LABELED DATA MANIFOLD

Qiuhua Liu, Xuejun Liao, and Lawrence Carin

Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0291, USA

ABSTRACT

We present an algorithm for learning parametric classifiers on a partially labeled data manifold, based on a graph representation of the manifold. The unlabeled data are utilized by basing classifier learning on neighborhoods, formed via Markov random walks. The proposed algorithm yields superior performance on three benchmark data sets and the margin of improvements over existing semi-supervised algorithms is significant.

Index Terms— semi-supervised learning, classifier, partially labeled data, graph, logistic regression

1. INTRODUCTION

Supervised learning has proven an effective technique for learning a classifier when there are sufficient examples of labeled data. In many applications (medical diagnosis, etc.), a generous provision of labeled data is usually not available, due to the highly demanding cost incurred by labeling a data point. Nevertheless, an unlabeled datum (vector of features) is comparatively much less expensive to acquire, leading to an ample set of data of which only a small subset are labeled. Recent years have witnessed a surge of interest in semi-supervised learning, which addresses the problem of how to efficiently utilize the partially labeled data in classifier designing.

To date, there have been a number of semi-supervised methods developed. The generative-model method, an early semi-supervised method, estimates the joint probability of data and labels via expectation-maximization (EM), treating the missing labels of unlabeled data as hidden variables [1]. Co-training [2], another early method, exploits two independent subvectors of features, using one to provide the label estimates for the other. The semi-supervised support vector machine (SVM) [3] represents a more recent method, which maximizes the margin between classes, taking into account both labeled and unlabeled data. The graph-based method [4, 5, 6], the main focus of current research in semi-supervised learning, exploits the assumption that strongly connected data points share the same label, and utilizes the spectral graph theory to quantify the between-data connectivity. For a more complete review of the literature, see [7].

Most graph-based algorithms operate in a transductive fashion, i.e., they directly learn the labels of the unlabeled data, instead of learning a classifier first and then using the classifier to infer the unseen labels (the inductive fashion). While transductive algorithms avoid the problem of model selection for a classifier, they lack a principled way of predicting the labels of new unlabeled data outside the training set. The work in [6] addresses this problem by constructing a graph-based prior distribution on the parameters of a classifier and learning the classifier by maximization of *a posterior* (MAP); the prior utilizes both labeled and unlabeled data, thus enforcing semi-supervised learning. Several drawbacks are inherent in the algorithm in [6]. First, the hyper-parameter balancing the importance of the prior relative to the data likelihood needs to be learned. Second, the label consistency of close data points cannot be accurately enforced by the prior if the convexity is desired in the MAP estimation. Third, the prior is data-dependent and hence is difficult to be extended to the multi-task case, in which one performs semi-supervised learning on multiple partially labeled sets simultaneously.

In this paper, we propose an algorithm for learning parametric classifiers on a partially labeled data manifold, by representing the manifold as a graph, where each vertex represents a data point and the weighted edge between two vertices manifests the immediate connectivity between the corresponding data points. We are motivated by the work in [4] and build the t-step connectivity between data points via a Markov random walk on the manifold. To account for heterogeneities in the data manifold, we let the random walk take different step-sizes at different data locations; each step-size dictates a Markov transition matrix and we select the step-size to assemble the transition matrix for the entire manifold. Our algorithm is an alternative to the algorithm in [6] and yet is not subject to the drawbacks there.

2. THE GRAPH REPRESENTATION OF A PARTIALLY LABELED DATA MANIFOLD

Let $G = (\mathcal{X}, \mathbf{W})$ be a graph, where $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ is the set of vertices and $\mathbf{W} = [x_{ij}]_{N\times N}$ is the affinity matrix with the (i, j)-th element w_{ij} indicating the strength of immediate connectivity between vertices $\mathbf{x}_i$ and $\mathbf{x}_j$. For the purpose of data classification, the vertex set $\mathbf{x}_i$ coincides with the set of data points (labeled or unlabeled), and w_{ij} is a quantitative measure of the closeness of data points $\mathbf{x}_i$ and $\mathbf{x}_j$. We are considering the semi-supervised setting, where only a subset of $\mathcal{X}$ are provided with class labels, leading to a partially labeled graph.

Though there are many alternative ways of defining w_{ij}, here we consider a widely used definition

$$w_{ij} = \exp(-0.5\,\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_i^2) \tag{1}$$

where $\|\cdot\|$ is the Euclidean norm. Following [4], we introduce a Markov transition matrix $\mathbf{A} = [a_{ij}]_{N\times N}$, which defines a Markov random walk. The (i,j)-th element

$$a_{ij} = (\textstyle\sum_{k=1}^{N} w_{ik})^{-1} w_{ij} \tag{2}$$

gives probability of walking from $\mathbf{x}_i$ to $\mathbf{x}_j$ by taking a single step. In general we are interested in a t-step random walk, the transition matrix of which is given by $\mathbf{A}$ raised to the power of t, i.e., $\mathbf{A}^t = [a_{ij}^{(t)}]_{N\times N}$. Letting $t = 0$ makes $\mathbf{A}^t$ degenerate to an identity matrix, in which case one can only stay at a single data point. Letting t equal to the number of vertices represents the worse case that the random walker must pass all vertices during a trip from one vertex to another. In general, a small (large) t under-utilizes (over-estimates) the unlabeled data. Other choices of t are discussed in [4].

In specifying the Markov transition matrix in (1) we have used a distinct σ_i for each data point $\mathbf{x}_i$. In the random walk, σ can be thought of as the step-size. Therefore location-dependent step-sizes allow one to account for possible heterogeneities in the data manifold — at locations where data are densely distributed a small step-size is enough, whereas at locations where data are sparsely distributed a large step-size is necessary to connect a data point to its nearest neighbor. A simple choice of the heterogeneous σ is to let σ_i to be a fraction of the shortest Euclidean distance between $\mathbf{x}_i$ and all other data points in $\mathcal{X}$. This ensures each data point is immediately connected to at least one neighbor.

3. NEIGHBORHOOD-BASED LEARNING

Any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ are said to be t-step neighbors, denoted as $\mathbf{x}_j \overset{t}{\sim} \mathbf{x}_i$, if $a_{ij}^{(t)} > 0$. Then $\mathcal{N}_t(\mathbf{x}_i) = \{\mathbf{x} : \mathbf{x} \overset{t}{\sim} \mathbf{x}_i\} \subseteq \mathcal{X}$, which represents the set of t-step neighbors of $\mathbf{x}_i$, is called the t-step neighborhood of $\mathbf{x}_i$. When $t = 0$, the neighborhood shrinks to a single data point, $\mathcal{N}_0(\mathbf{x}_i) = \{\mathbf{x}_i\}$. We define the probability of label y_i given the t-step neighborhood of $\mathbf{x}_i$

$$p(y_i|\mathcal{N}_t(\mathbf{x}_i), \boldsymbol{\theta}) = \textstyle\sum_{j=1}^{N} a_{ij}^{(t)}\, p(y_i|\mathbf{x}_j, \boldsymbol{\theta}) \tag{3}$$

where the magnitude of $a_{ij}^{(t)}$ automatically determines the contribution of $\mathbf{x}_j$ to the neighborhood, thus we are allowed to run the index j over the entire $\mathcal{X}$. The $p(y_i|\mathbf{x}_j, \boldsymbol{\theta})$ is the probability of label y_i given a single data point $\mathbf{x}_j$ (zero-step neighborhood) and it is represented by a standard probabilistic classifier parameterized by $\boldsymbol{\theta}$. In this paper we consider binary classification with $y \in \{-1, 1\}$, and choose the form of $p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$ as a logistic regression classifier

$$p(y_i|\mathbf{x}_j, \boldsymbol{\theta}) = \left[1 + \exp(-y_i \boldsymbol{\theta}^T \mathbf{x}_j)\right]^{-1} \tag{4}$$

where we assume a constant element 1 prefixed to each $\mathbf{x}$ (the prefixed $\mathbf{x}$ is still denoted as $\mathbf{x}$ for notational simplicity), thus the first element in $\boldsymbol{\theta}$ is a bias term.

To distinguish (3) and (4), we call the former a neighborhood-based classifier and the latter a regular classifier. The fundamental difference is, the regular classifier predicts y_i by $\mathbf{x}_i$ alone while the neighborhood-based classifier predicts y_i by $\mathbf{x}_i$ and the neighbors of $\mathbf{x}_i$. The neighborhood of $\mathbf{x}_i$ is formed by all $\mathbf{x}_j$'s that can be reached from $\mathbf{x}_i$ by t-step random walks, with each $\mathbf{x}_j$ contributing to the prediction of y_i in proportion to $a_{ij}^{(t)}$. The role of neighborhoods is then conspicuous — in order for $\mathbf{x}_i$ to be labeled y_i, each neighbor $\mathbf{x}_j$ must be labeled consistently with y_i, in the degree proportional to $a_{ij}^{(t)}$; in such a manner, y_i implicitly propagates over the neighborhood. By taking the neighborhoods into account, it is possible to learn a classifier with only a few labels present; the classifier thus learned is much less subject to over-fitting than when ignoring the neighborhoods.

The learning begins with the neighborhood-conditioned likelihood function

$$\begin{aligned} &p(\{y_i, i \in \mathcal{L}\}|\{\mathcal{N}_t(\mathbf{x}_i) : i \in \mathcal{L}\}, \boldsymbol{\theta}) = \textstyle\prod_{i\in\mathcal{L}} p(y_i|\mathcal{N}_t(\mathbf{x}_i), \boldsymbol{\theta}) \\ &= \textstyle\prod_{i\in\mathcal{L}} \sum_{j=1}^{N} a_{ij}^{(t)}\, p(y_i|\mathbf{x}_j, \boldsymbol{\theta}) \end{aligned} \tag{5}$$

where $\mathcal{L} \subseteq \{1, 2, \cdots, N\}$ denotes the set of indices of labeled data, and we assume the labels are conditionally independent. The likelihood function is the joint probability of observed labels given the t-step neighborhood of each corresponding data point. To enforce sparseness of $\boldsymbol{\theta}$ and promote generalization, we impose a normal prior on $\boldsymbol{\theta}$,

$$p(\boldsymbol{\theta}|\boldsymbol{\Lambda}) = (\det\boldsymbol{\Lambda})^{-1/2}(2\pi)^{-d/2}\exp(-\boldsymbol{\theta}^T\boldsymbol{\Lambda}\boldsymbol{\theta}/2) \tag{6}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, ...\lambda_i, ..., \lambda_d]$ are hyper-parameters, and d is the dimensionality of $\mathbf{x}$. Each hyper-parameter has an independent Gamma distribution, hence

$$p(\boldsymbol{\Lambda}|\alpha, \beta) \;=\; \textstyle\prod_{i=1}^{d} \dfrac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)} \lambda_i^{\alpha_i - 1}\exp(-\lambda_i\beta_i) \tag{7}$$

Marginalizing $\boldsymbol{\Lambda}$, we obtain the prior distribution conditional directly on α and β, $p(\boldsymbol{\theta}|\alpha,\beta) = \int p(\boldsymbol{\theta}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}|\alpha,\beta)\,d\boldsymbol{\Lambda}$, from which and (5) the posterior of $\boldsymbol{\theta}$ follows

$$\begin{aligned} &p(\boldsymbol{\theta}|\alpha, \beta, \{y_i, \mathcal{N}_t(\mathbf{x}_i) : i \in \mathcal{L}\}) \\ &= Z^{-1} \prod_{i\in\mathcal{L}} \sum_{j=1}^{N} a_{ij}^{(t)}\, p(y_i|\mathbf{x}_j, \boldsymbol{\theta}) \int p(\boldsymbol{\theta}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}|\alpha,\beta)\,d\boldsymbol{\Lambda} \end{aligned} \tag{8}$$

with Z a normalization constant. We are interested in the MAP estimate of $\boldsymbol{\theta}$, which maximizes (8) or, equivalently,

$$\ell(\boldsymbol{\theta}) \overset{def.}{=} \ln p(\boldsymbol{\theta}|\alpha, \beta, \{y_i, \mathcal{N}_t(\mathbf{x}_i) : i \in \mathcal{L}\}) + \ln Z$$
$$= \sum_{i\in\mathcal{L}} \ln \sum_{j=1}^{N} a_{ij}^{(t)}\, p(y_i|\mathbf{x}_j, \boldsymbol{\theta}) + \ln \int p(\boldsymbol{\theta}|\boldsymbol{\Lambda}) p(\boldsymbol{\Lambda}|\alpha, \beta)\, d\boldsymbol{\Lambda} \quad (9)$$

The $\boldsymbol{\theta}$ obtained by maximization of $\ell(\boldsymbol{\theta})$ generally is less subject to over-fitting for two reasons — the neighborhoods incorporated into the first term of $\ell(\boldsymbol{\theta})$ undermine over-fitting, as mentioned earlier; the second term of $\ell(\boldsymbol{\theta})$ enforces sparseness of $\boldsymbol{\theta}$, which again undermines over-fitting.

4. THE LEARNING ALGORITHM

We maximize (9) by expectation-maximization (EM). For any $\{\delta_{ij} : \delta_{ij} \geq 0, \sum_{j=1}^{N} \delta_{ij} = 1\}$ and $\{q(\boldsymbol{\Lambda}) : \int q(\boldsymbol{\Lambda}) d\boldsymbol{\Lambda} = 1\}$, we apply Jensen's inequality to the righthand side of (9) to obtain the lower bound

$$\ell(\boldsymbol{\theta}) \geq Q(\boldsymbol{\theta}|\delta, q) \overset{def.}{=} \sum_{i\in\mathcal{L}} \sum_{j=1}^{N} \delta_{ij} \ln \frac{a_{ij}^{(t)} p(y_i|\mathbf{x}_j, \boldsymbol{\theta})}{\delta_{ik}}$$
$$+ \int q(\boldsymbol{\Lambda}) \ln \frac{p(\boldsymbol{\theta}|\boldsymbol{\Lambda}) p(\boldsymbol{\Lambda}|\alpha, \beta)}{q(\boldsymbol{\Lambda})} d\boldsymbol{\Lambda} \quad (10)$$

where the equality holds when

$$\delta_{ij} = \frac{p(y_i|\mathbf{x}_j, \boldsymbol{\theta}) a_{ij}^{(t)}}{\sum_{k=1}^{N} p(y_i|\mathbf{x}_k, \boldsymbol{\theta}) a_{ik}^{(t)}}, \quad q(\boldsymbol{\Lambda}) = \frac{p(w, \boldsymbol{\Lambda}|\alpha, \beta)}{\int p(w, \boldsymbol{\Lambda}|\alpha, \beta) d\boldsymbol{\Lambda}} \quad (11)$$

where $p(w, \boldsymbol{\Lambda}|\alpha, \beta) = \int p(\boldsymbol{\theta}|\boldsymbol{\Lambda}) p(\boldsymbol{\Lambda}|\alpha, \beta) d\boldsymbol{\Lambda}$. The EM algorithm consists of iterating the following two steps.

1. E-step: computing $\{\delta_{ij}\}$ and $q(\boldsymbol{\Lambda})$ using (11);
2. M-step: compute the re-estimate of $\boldsymbol{\theta}$ as $\boldsymbol{\theta} = \arg\max_{\widehat{\boldsymbol{\theta}}} Q(\widehat{\boldsymbol{\theta}}|\delta, q)$

The convergence is monitored by checking $\ell(\boldsymbol{\theta})$, which is guaranteed to monotonically increase over the EM iterations.

Two points are noticeable regarding the technical details. First, since (7) is conjugate to (6), $q(\boldsymbol{\Lambda})$ is of the same form as (7) with updated hyper-parameters,

$$q(\boldsymbol{\Lambda}) = \prod_{i=1}^{d} \frac{(\beta_i + 0.5\, \theta_i^2)^{\alpha_i + 0.5}}{\Gamma(\alpha_i + 0.5)} \lambda_i^{\alpha_i - 0.5} e^{-\lambda_i(\beta_i + 0.5\theta_i^2)} \quad (12)$$

and the integral $\int p(\boldsymbol{\theta}|\boldsymbol{\Lambda}) p(\boldsymbol{\Lambda}|\alpha, \beta) d\boldsymbol{\Lambda}$ is equal to

$$\frac{1}{(2\pi)^{d/2}} \prod_{i=1}^{d} \frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)} \frac{\Gamma(\alpha_i + \frac{1}{2})}{\left(\beta_i + \frac{1}{2}\theta_i^2\right)^{\alpha_i + \frac{1}{2}}} \quad (13)$$

a useful result in evaluating $\ell(\boldsymbol{\theta})$ for convergence check.

Second, in computing $Q(\widehat{\boldsymbol{\theta}}|\delta, q)$ by (10), one needs to compute $\int q(\boldsymbol{\Lambda}) \ln p(\widehat{\boldsymbol{\theta}}|\boldsymbol{\Lambda}) d\boldsymbol{\Lambda}$, which is found equal to

$$\widehat{\boldsymbol{\theta}}^T \mathrm{diag}\left[\mathbb{E}_q(\lambda_1), \mathbb{E}_q(\lambda_2), \cdots, \mathbb{E}_q(\lambda_d)\right] \widehat{\boldsymbol{\theta}}$$

with $\mathbb{E}_q(\lambda_i) = (\alpha_i + \frac{1}{2})(\beta_i + \frac{1}{2}\theta_i^2)^{-1}$.

5. EXPERIMENTAL RESULTS

The proposed semi-supervised algorithm is evaluated on three benchmark data sets — Pima Indians Diabetes Database (PIMA), Wisconsin Diagnostic Breast Cancer (WDBC) data, and Johns Hopkins University Ionosphere database (Ionosphere). These data sets and their descriptions are publicly available at the UCI machine learning repository [8]. The evaluation is performed in comparison to four other semi-supervised algorithms, namely, the transductive SVM [3], the algorithm of Szummer & Jaakkola [4], GRF [5], and Logistic GRF [6]. The performance is evaluated in terms of classification accuracy, defined as the ratio of the number of correctly classified data over the total number of data being tested.

We consider two testing modes: transductive and inductive. In the transductive mode, the test data are the unlabeled data that are used in training the semi-supervised algorithms; in the inductive mode, the test data are a set of holdout data unseen during training. We follow the same procedures as used in [6] to perform the experiments. Denote by $\mathcal{X}$ any of the three benchmark data sets and $\mathcal{Y}$ the associated label set. In the transductive mode, we randomly sample $\mathcal{X}_L \subset \mathcal{X}$ and assume the associated label set $\mathcal{Y}_L$ are available. The semi-supervised algorithms are trained by $\mathcal{X} \cup \mathcal{Y}_L$ and tested on $\mathcal{X} \setminus \mathcal{X}_L$. In the inductive mode, we randomly sample two disjoint data subsets $\mathcal{X}_L \subset \mathcal{X}$ and $\mathcal{X}_U \subset \mathcal{X}$, and assume the label set $\mathcal{Y}_L$ associated with $\mathcal{X}_L$ are available. The semi-supervised algorithms are trained by $\mathcal{X}_L \cup \mathcal{Y}_L \cup \mathcal{X}_U$ and tested on 200 data randomly sampled from $\mathcal{X} \setminus (\mathcal{X}_L \cup \mathcal{X}_U)$.

The comparison results are summarized in Figures 1 and 2, where the results of the proposed algorithm and the algorithm of Szummer & Jaakkola are calculated by us, and the results of remaining algorithms are cited from [6]. Each curve in the figures is a result averaged from T independent trials, with $T = 20$ for the transductive results and $T = 50$ for the inductive results. In the inductive case, the comparison is between the proposed algorithm and the Logistic GRF, as the others are transductive algorithms.

For the proposed algorithm, we can either use the regular classifier in (4) or the neighborhood-based classifier in (3) to predict the labels of unlabeled data seen in training (the transductive mode). In the inductive mode, however, the $\{a_{ij}^{(t)}\}$ are not available for the test data (unseen in training) since they are not in the graph representation, therefore we can only employ the regular classifier. In the legends of Figures 1 and 2, a suffix "II" to proposed algorithm indicates that the neighborhood-based classifier is invoked in testing; when no suffix is attached, the regular classifier is invoked.

In general, the proposed algorithm outperforms all the competing algorithms on the data sets considered here. The improvements are particularly significant on PIMA and Ionosphere. The margin of improvements achieved by the proposed algorithm over Logistic GRF is striking and encouraging — the proposed algorithm virtually performs better in al-

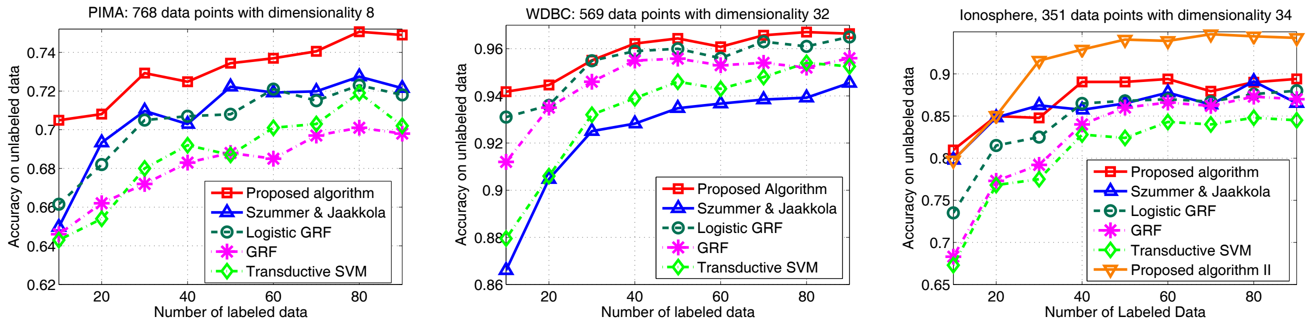

Fig. 1. Transductive results. Each curve is an average from 20 independent trials. The horizontal axis is the size of $\mathcal{X}_L$. The algorithms are tested on $\mathcal{X}_U$. The algorithm of Szummer & Jaakkola [4] and ours use $\sigma_i = \min_j \|\mathbf{x}_i - \mathbf{x}_j\|/3$ and $t = 100$.

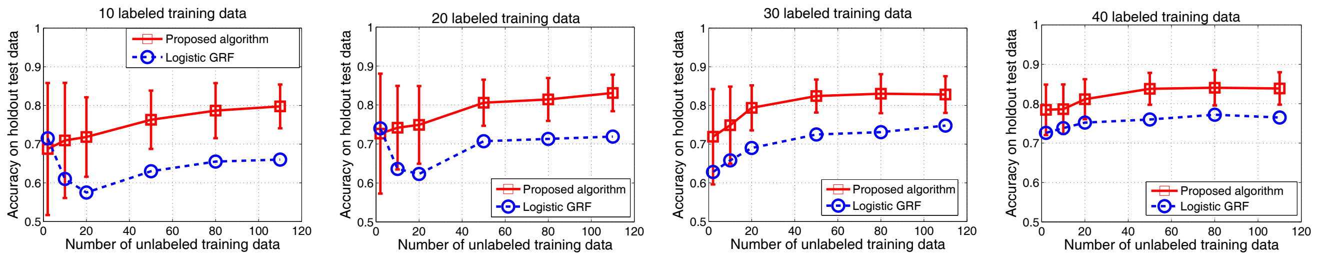

Fig. 2. Inductive results. Each curve is an average from 50 independent trials. The horizontal axis is the size of $\mathcal{X}_U$. From left to right in the sub-figures, the size of $\mathcal{X}_L$ is $10, 20, 30, 40$. The algorithms are tested on 200 data randomly sampled from $\mathcal{X} \setminus (\mathcal{X}_L \cup \mathcal{X}_U)$. Error bars are shown for the proposed algorithm, which uses $\sigma_i = \min_j \|\mathbf{x}_i - \mathbf{x}_j\|/3$ and $t = 100$.

most all individual trials, as indicated by the error bars shown in Figure 2. Seen from the last sub-figure of Figure 1, the neighborhood-based classifier boosts the improvement a lot on Ionosphere. Also noted in Figure 2, the advantage of our algorithm becomes more conspicuous as the number of labeled training data gets smaller.

6. CONCLUSIONS

We have presented a new semi-supervised learning algorithm for classification. By basing the learning on neighborhoods built from Markov random walks on a partially labeled data manifold, we efficiently employ the unlabeled data to overcome the over-fitting problem that plagues supervised learning due to insufficient labeled data. The proposed algorithm produces a parametric classifier, which makes it easy to predict the labels of new data that are unseen during training. The experimental results on three benchmark data sets demonstrate that our algorithm yields marked improvements over competing state-of-the-art algorithms.

7. REFERENCES

[1] S. Ganesalingam, "Classification and mixture approaches to clustering via maximum likelihood," *Applied Statistics*, vol. 38, no. 3, pp. 455–466, 1989.

[2] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *The Annual Conference on Learning Theory (COLT)*.

[3] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th International Conf. on Machine Learning (ICML)*. 1999, pp. 200–209, Morgan Kaufmann, San Francisco, CA.

[4] M. Szummer and T. Jaakkola, "Partially labeled classification with markov random walks," in *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[5] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *The Twentieth International Conference on Machine Learning (ICML)*, 2003, pp. 912–919.

[6] B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo, "On semi-supervised classification," in *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[7] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006.

[8] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, "UCI repository of machine learning databases," *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, 1998.