Learning from Labeled and Unlabeled Data with Label Propagation

Xiaojin Zhu*
$*$ School of Computer Science
Carnegie-Mellon University
zhuxj@cs.cmu.edu

Zoubin Ghahramani*†
†Gatsby Computational Neuroscience Unit
University College London
zoubin@gatsby.ucl.ac.uk

Abstract

We investigate the use of unlabeled data to help labeled data in classification. We propose a simple iterative algorithm, label propagation, to propagate labels through the dataset along high density areas defined by unlabeled data. We analyze the algorithm, show its solution, and its connection to several other algorithms. We also show how to learn parameters by minimum spanning tree heuristic and entropy minimization, and the algorithm's ability to perform feature selection. Experiment results are promising.

1 Introduction

Labeled data are essential for supervised learning. However they are often available only in small quantities, while unlabeled data may be abundant. Using unlabeled data together with labeled data is of both theoretical and practical interest. Recently many approaches have been proposed for combining unlabeled and labeled data [1] [2]. Among them there is a promising family of methods which assume that closer data points tend to have similar class labels in a manner analogous to k-Nearest-Neighbor (kNN) in traditional supervised learning. As a result, these methods propagate labels through dense unlabeled data regions.

We propose a new algorithm to propagate labels. We formulate the problem as a particular form of label propagation, where a node's labels propagate to all nodes according to their proximity. Meanwhile we fix the labels on the labeled data. Thus labeled data act like sources that push out labels through unlabeled data. We prove the convergence of the algorithm, find a closed form solution for the fixed point, and analyze its behavior on several datasets. We also propose a minimum spanning tree heuristic and an entropy minimization criterion to learn the parameters, and show our algorithm learns to detect irrelevant features.

2 Label propagation

2.1 Problem setup

Let $(x_1, y_1) \dots (x_l, y_l)$ be labeled data, where $Y_L = \{y_1 \dots y_l\} \in \{1 \dots C\}$ are the class labels. We assume the number of classes C is known, and all classes are present in the labeled data. Let $(x_{l+1}, y_{l+1}) \dots (x_{l+u}, y_{l+u})$ be unlabeled data where $Y_U =$

$\{y_{l+1} \dots y_{l+u}\}$ are unobserved; usually $l \ll u$. Let $X = \{x_1 \dots x_{l+u}\} \in R^D$. The problem is to estimate Y_U from X and Y_L.

Intuitively, we want data points that are close to have similar labels. We create a fully connected graph where the nodes are all data points, both labeled and unlabeled. The edge between any nodes i, j is weighted so that the closer the nodes are in local Euclidean distance, d_{ij}, the larger the weight w_{ij}. The weights are controlled by a parameter σ:

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) = \exp\left(-\frac{\sum_{d=1}^{D}(x_i^d - x_j^d)^2}{\sigma^2}\right) \tag{1}$$

Other choices of distance metric are possible, and may be more appropriate if the x are, for example, positive or discrete. We have chosen to focus on Euclidean distance in this paper, later allowing different σ's for each dimension, corresponding to length scales for propagation.

All nodes have soft labels which can be interpreted as distributions over labels. We let the labels of a node propagate to all nodes through the edges. Larger edge weights allow labels to travel through more easily. Define a $(l+u) \times (l+u)$ probabilistic transition matrix T

$$T_{ij} = P(j \rightarrow i) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{kj}} \tag{2}$$

where T_{ij} is the probability of jumping from node j to i. Also define a $(l+u) \times C$ label matrix Y, whose ith row representing the label probabilities of node y_i. The initialization of the rows of Y corresponding to unlabeled data points is not important. We are now ready to present the algorithm.

2.2 The algorithm

The label propagation algorithm is as follows:

1. All nodes propagate labels for one step: $Y \leftarrow TY$
2. Row-normalize Y to maintain the class probability interpretation.
3. Clamp the labeled data. Repeat from step 2 until Y converges.

Step 3 is critical: Instead of letting the labeled data points 'fade away', we clamp their class distributions to $Y_{ic} = \delta(y_i, c)$, so the probability mass is concentrated on the given class. The intuition is that with this constant 'push' from labeled nodes, the class boundaries will be pushed through high density data regions and settle in low density gaps. If this structure of data fits the classification goal, our algorithm can use unlabeled data to help learning.

The algorithm converges to a simple solution. First, step 1 and 2 can be combined into $Y \leftarrow \bar{T}Y$ with $\bar{T}$ being the row-normalized matrix of T, i.e. $\bar{T}_{ij} = T_{ij} / \sum_k T_{ik}$. Let Y_L be the top l rows of Y (the labeled data) and Y_U the remaining u rows. Notice that Y_L never really changes since it is clamped in step 3, and we are solely interested in Y_U. We split $\bar{T}$ after the l-th row and the l-th column into 4 sub-matrices

$$\bar{T} = \begin{bmatrix} \bar{T}_{ll} & \bar{T}_{lu} \\ \bar{T}_{ul} & \bar{T}_{uu} \end{bmatrix} \tag{3}$$

It can be shown that our algorithm is $Y_U \leftarrow \bar{T}_{uu}Y_U + \bar{T}_{ul}Y_L$, which leads to $Y_U = \lim_{n\rightarrow\infty} \bar{T}_{uu}^n Y^0 + [\sum_{i=1}^{n} \bar{T}_{uu}^{(i-1)}]\bar{T}_{ul}Y_L$, where Y^0 is the initial Y. We need to show $\bar{T}_{uu}^n Y^0 \rightarrow 0$. By construction, all elements in $\bar{T}$ are greater than zero. Since $\bar{T}$ is row normalized, and $\bar{T}_{uu}$ is a sub-matrix of $\bar{T}$, it follows that $\exists \gamma < 1$, such that $\sum_{j=1}^{u} \bar{T}_{uu_{ij}} \leq \gamma, \forall i = 1 \dots u$. Therefore $\sum_j \bar{T}_{uu_{ij}}^n = \sum_j \sum_k \bar{T}_{uu_{ik}}^{(n-1)} \bar{T}_{uu_{kj}} = \sum_k \bar{T}_{uu_{ik}}^{(n-1)} \sum_j \bar{T}_{uu_{kj}} \leq$

$\sum_k \bar{T}^{(n-1)}_{uu_{ik}} \gamma \le \gamma^n$. So the row sums of $\bar{T}^n_{uu}$ converge to zero, which means $\bar{T}^n_{uu} Y^0 \to 0$. Thus the initial point Y^0 is inconsequential. Obviously

$$Y_U = (I - \bar{T}_{uu})^{-1} \bar{T}_{ul} Y_L \tag{4}$$

is a fixed point. Therefore it is the unique fixed point and the solution to our iterative algorithm.

2.3 Parameter setting

We set the parameter σ with a heuristic. We find a minimum spanning tree (MST) over all data points under Euclidean distances d_{ij}, with Kruskal's Algorithm [3]. In the beginning no node is connected. During tree growth, the edges are examined one by one from short to long. An edge is added to the tree if it connects two separate components. The process repeats until the whole graph is connected. We find the first tree edge that connects two components with different labeled points in them. We regard the length of this edge d^0 as a heuristic of the minimum distance between classes. We set $\sigma = d^0/3$ so that the weight of this (and longer) edge is close to 0, with the hope that local propagation is then mostly within classes. Later we will propose an entropy-based criterion to learn the σ parameters.

2.4 Rebalancing class proportions

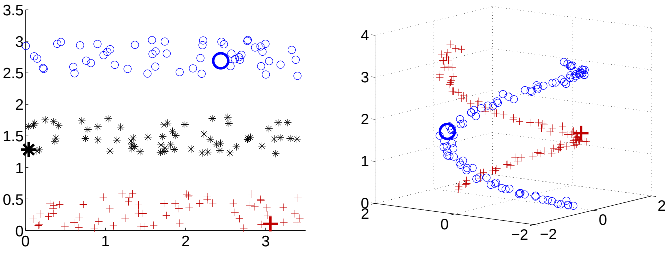
For classification purposes, once Y_U is computed, we can take the most likely (ML) class of each unlabeled point as its label. However, this procedure does not provide any control over the final proportions of the classes, which are implicitly determined by the distribution of data. If classes are not well separated and labeled data is scarce, incorporating constraints on class proportions can improve final classification. We assume the class proportions $P_1 \ldots P_C$ ($\sum_c P_c = 1$) are either estimated from labeled data or known a priori (i.e. from an oracle). We propose two post-processing alternatives to ML class assignment:

- **Class Mass Normalization:** Find coefficients λ_c to scale columns of Y_U s.t. $\lambda_1 \sum Y_{U.1} : \ldots : \lambda_C \sum Y_{U.C} = P_1 : \ldots : P_C$. Once decisions are made for each point, this procedure does not guarantee that class proportion will be exactly equal to $P_1 \ldots P_C$.
- **Label Bidding:** We have uP_c class c labels for sale. Each point i bids $\$Y_{U_{ic}}$ for class c. Bids are processed from high to low. Assume $Y_{U_{ic}}$ is currently the highest bid. If class c labels remain, a c label is sold to point i, who then quits the bidding. Otherwise the bid is ignored and the second highest bid is processed, and so on. Label bidding guarantees that strict class proportions will be met.

3 Experimental results

To demonstrate properties of this algorithm we investigated both synthetic datasets and a real-world classification problem. Figure 1 shows label propagation on two synthetic datasets. Large symbols are labeled data, other points are originally unlabeled. For '3-Bands' $C = 3, l = 3, u = 178, \sigma = 0.22$; for 'Springs' $C = 2, l = 2, u = 184, \sigma = 0.43$. Both σ's are from the MST heuristic. Simple ML classification is used. Here, obviously kNN would fail to follow the structure of data.

For a real world example we test label propagation on a handwritten digits dataset, originally from the Cedar Buffalo binary digits database [4]. The digits were preprocessed to reduce the size of each image down to 16×16 by down-sampling and Gaussian smoothing, with pixel values ranging from 0 to 255 [5]. We use digits '1', '2' and '3' in our experiment as three classes, with 1100 images in each class. Each image is represented by a 256



(a) 3-Bands

(b) Springs

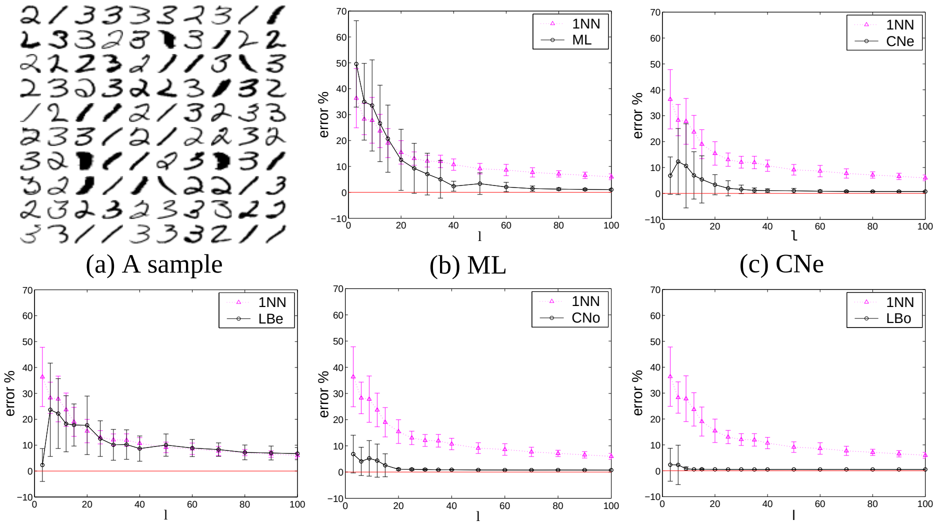
Figure 1: Label propagation on two synthetic datasets.

dimensional vector. Figure 2(a) shows a random sample of 100 images from the dataset. We vary labeled data size l from 3 up to 100. For each size, we perform 20 trials. In each trial we randomly sample labeled data from the whole dataset, and use the rest of images as unlabeled data. If any class is absent from the sampled labeled set, we redo the sampling. Thus labeled and unlabeled data are approximately iid. We find σ by the MST heuristic (all trials have σ close to 340). To speed up computation, only the top 150 neighbors of each image are considered to make T sparse. We measure 5 error rates on unlabeled data (see section 2.4): (1) **ML**: The most likely labels; (2) **CNe**: Class mass normalization, with maximum likelihood estimate of class proportions from labeled data; (3) **LBe**: Label bidding, with maximum likelihood estimate of class proportions from labeled data; (4) **CNo**: Class mass normalization, with knowledge of the oracle (true) class proportions (i.e. $1/3$); (5) **LBo**: Label bidding post processing, with oracle class proportions. We use two algorithms as baselines. The first one is standard kNN. We report 1NN results since it is the best among $k = 1 \ldots 11$. The second baseline algorithm is 'propagating 1NN' (p1NN): Among all unlabeled data, find the point x_u closest to a labeled point (call it x_l). Label x_u with x_l's label, add x_u to the labeled set, and repeat. p1NN is a crude version of label propagation. It performs well on the two synthetic datasets, with the same results as in Figure 1.

Figure 2(b)–(f) shows the results. ML labeling is better than 1NN when $l \geq 40$ (b). But if we rebalance class proportions, we can do much better. If we use class frequency of labeled data as class proportions and perform class mass normalization, we improve the performance when l is small (c). If we know the true class proportion, the performance is even better (e,f), with label bidding being slightly superior to class mass normalization. On the other hand since label bidding requires exact proportions, its performance is bad when the class proportions are estimated (d). To summarize, label bidding is the best when exact proportions are known, otherwise class mass normalization is the best. p1NN consistently performs no better than 1NN. Table 1 lists the error rates for p1NN, 1NN, ML, CNe and LBo. Each entry is averaged over 20 trials. All differences are statistically significant at α level 0.05 (t test), except for the pairs in thin face.

4 Parameter learning by entropy minimization

When $\sigma \to 0$, the label propagation result approaches p1NN, because under the exponential weights (1) the influence of the nearest point dominates. When $\sigma \to \infty$, the whole dataset effectively shrinks to a single point. All unlabeled points receive the same influence from all labeled points, resulting in equal class probabilities. The 'appropriate' σ is in between. We used MST as a heuristic to set the parameter σ and have shown in the previous section that it can work very well; in this section we propose a criterion for learning the model parameters that can be applied in more general settings. Data label likelihood does not



(a) A sample (b) ML (c) CNe

(d) LBe (e) CNo (f) LBo

Figure 2: The digits dataset. Each point is an average of 20 random trials. The error bars are ±1 standard deviation.

make sense as a criterion in our setting, especially with very few labeled points, since intuitively the 'quality' of a solution depends on how unlabeled data are assigned labels.

We propose to minimize the *entropy* $H = -\sum_{ij} Y_{ij} \log Y_{ij}$, which is the sum of the entropy on individual data points. This captures the intuition that good σ should label all points confidently. There are many arbitrary labelings of the unlabeled data that have low entropy, which might suggest that this criterion would not work. However, it is important to point out that most of these arbitrary low entropy labelings cannot be achieved by propagating labels using our algorithm. In fact, we find that the space of low entropy labelings achievable by label propagation is small and lends itself well to tuning the σ parameters. One complication remains, which is that H has a minimum 0 at $\sigma \to 0$ (notice p1NN gives each point a hard label), but this (p1NN) is not always desirable. Figure 3(a,b) shows the problem on the 'Bridge' dataset, where the upper grid is slightly tighter than the lower grid.

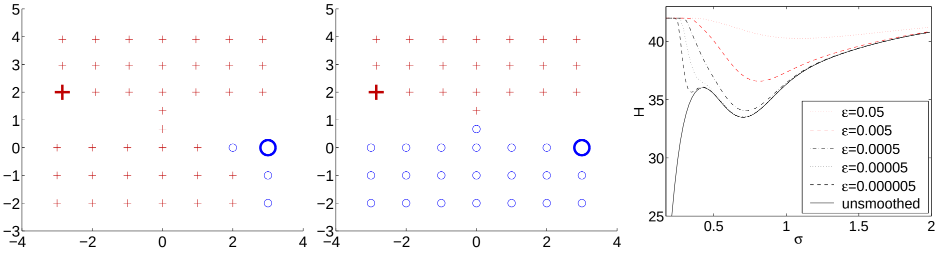
Table 1: Digits: error rate of different post processing methods.

l	3	6	9	12	15	20	25	30	35	40	50	60	70	80	90	100
p1NN	46.1	34.2	35.0	29.2	23.2	17.7	15.0	14.3	12.9	11.5	11.7	10.1	9.6	8.8	8.2	7.4
1NN	**36.4**	28.3	27.8	23.7	19.0	15.4	13.1	12.1	11.9	10.7	9.1	8.6	7.7	7.1	6.6	6.0
ML	49.6	35.0	33.5	26.6	20.7	12.6	9.3	**7.0**	**5.0**	**2.4**	**3.4**	**2.0**	**1.5**	**1.2**	**1.1**	**1.0**
CNe	**6.9**	**12.3**	**10.6**	**7.0**	**5.4**	**3.4**	**2.0**	**1.6**	**1.1**	**1.1**	**1.0**	**0.8**	**0.8**	**0.7**	**0.7**	**0.7**
LBo	**2.3**	**2.3**	**0.8**	**0.6**	**0.6**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**

This can be fixed by smoothing T. Inspired by the analysis on the PageRank algorithm [6], we smooth T with a uniform transition matrix $\mathcal{U}$, where $\mathcal{U}_{ij} = 1/(l+u), \forall i, j$:

$$\tilde{T} = \epsilon \mathcal{U} + (1-\epsilon) T \tag{5}$$

$\tilde{T}$ is then used in place of T in the algorithm. Figure 3(c) shows H vs. σ before and after smoothing on the 'Bridge' dataset, with different ϵ values. Smoothing helps to get rid of the nuisance minimum at $\sigma \to 0$. In the following we use the value $\epsilon = 0.0005$. Although we have to add one more parameter ϵ to learn σ, the advantage is apparent when we introduce multiple parameters $\sigma_1 \ldots \sigma_D$, one for each dimension. Now the weights are $w_{ij} = \exp(-\sum_{d=1}^{D}(x_i^d - x_j^d)^2/\sigma_d^2)$. The σ_d's are analogous to the relevance or length scales in Gaussian process. We use gradient descent to find the parameters $\sigma_1 \ldots \sigma_D$ that minimizes H. Readers are referred to [7] for a derivation of $\partial H/\partial \sigma_d$.



(a) p1NN or $\sigma \to 0$ (b) optimal $\sigma = 0.72$ (c) smoothing

Figure 3: The Bridge dataset. p1NN (or $\sigma \to 0$) result is undesirable. Smoothing T helps to remove the minimum of H at $\sigma \to 0$.

The learned single σ is 0.26 and 0.43 for the '3-Bands' and 'springs' datasets respectively, very close to the MST heuristic. Classification remains the same. With multiple σ_d's, our algorithm can detect irrelevant dimensions. For the 'Bridge' dataset σ_1 keeps increasing while σ_2 and H asymptote during learning, meaning the algorithm thinks the horizontal dimension (corresponding to σ_1) is irrelevant to classification (large σ_d allows labels to freely propagate along that dimension). Classification is the same as Figure 3(b). Let us look at another synthetic dataset, 'Ball', with 400 data points uniformly sampled within a 4-dimensional unit hypersphere with gaps (Figure 4). There are two $45°$ gaps when the dataset is projected onto dimensions 1-2 and 3-4 respectively, but no gap in dimensions 1-3, 1-4 or 2-3. The gap in dimensions 1-2 is related to classification while the one in 3-4 is not. But this information is only hinted by the 4 labeled points. The learned parameters are $\sigma_1 = 0.18$, $\sigma_2 = 0.19$, $\sigma_3 = 14.8$, and $\sigma_4 = 13.3$, i.e. the algorithm learns that dimensions 3, 4 are irrelevant to classification, even though the data are clustered along those dimensions. Classification follows the gap in dimensions 1, 2.

5 Related work

The proposed label propagation algorithm is closely related to the Markov random walks algorithm [8]. Both utilize the manifold structure defined by large amount of unlabeled data, and assume the structure is correlated to the goal of classification. Both define a probabilistic process for labels to transit between nodes. But the Markov random walks algorithm approaches the problem from a different perspective. It uses the transition process to compute the t-step ancestorhood of any node i, that is, given that the random walk is at node i, what is the probability that it was at some node j at t steps before. To understand the algorithm, it is helpful to imagine that each node has two separate sets of labels, one

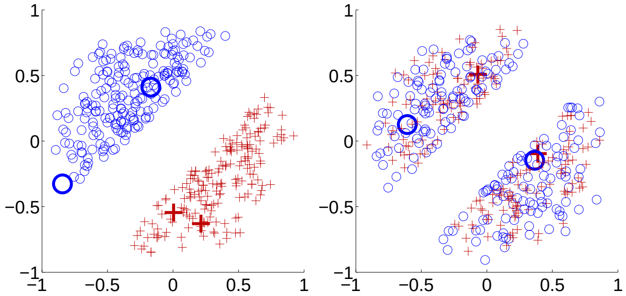


Figure 4: The Ball dataset. The algorithm learns that dimensions 3, 4 are irrelevant.

hidden and one observable. A node i's observable label is the average of all nodes' hidden labels, weighted by their ancestorhood. This is in fact kernel regression, with the kernel being the t-step ancestorhood. The hidden labels are learned such that the likelihood or margin of the observed labels are optimized. The algorithm is sensitive to the time scale t, since when $t \to \infty$ every node looks equally like an ancestor, and all observable labels will be the same. In our algorithm, labeled data are constant sources that push out labels, and the system achieves equilibrium when $t \to \infty$.

There seems to be a resemblance between label propagation and mean field approximation [9] [10]. In label propagation, upon convergence we have the equations (for unlabeled data)

$$Y_{ic} = \frac{\sum_j T_{ij} Y_{jc}}{\sum_{c'} \sum_j T_{ij} Y_{jc'}} \tag{6}$$

Consider the labeled / unlabeled data graph as a conditional Markov random field $\mathcal{F}$ with pairwise interaction w_{ij} between nodes i, j, and with labeled nodes clamped. Each unclamped (unlabeled) node i in $\mathcal{F}$ can be in one of C states, denoted by a vector also called $Y_i = (\delta(y_i, 1), \dots, \delta(y_i, C))$. The probability of a particular configuration Y in $\mathcal{F}$ is $P_{\mathcal{F}}(Y) = \frac{1}{Z} \exp[\sum_{ij} w_{ij} Y_i Y_j{}^\top]$. We now show label propagation (6) is approximately a mean field solution to a Markov random field $\mathcal{F}'$ that approximates $\mathcal{F}$. $\mathcal{F}'$ is defined as $P_{\mathcal{F}'}(Y) = \frac{1}{Z} \exp[\log(\sum_{ij} w_{ij} Y_i Y_j{}^\top)]$, which is the same as $\mathcal{F}$ up to the first order: $P_{\mathcal{F}'}(Y) \approx \frac{1}{Z} \exp[\sum_{ij} w_{ij} Y_i Y_j{}^\top - 1] = P_{\mathcal{F}}(Y)$. The mean field solution to $\mathcal{F}'$ is :

$$\langle Y_{ic} \rangle = \frac{\sum_j w_{ij} \langle Y_{jc} \rangle}{\sum_{c'} \sum_j w_{ij} \langle Y_{jc} \rangle} \tag{7}$$

where $\langle\rangle$ denotes the mean. Equation (6) is an approximation to (7) in the sense that if we assume $\sum_k w_{ik}$ are the same for all i, we can replace T_{ij} with w_{ij} in (6). Therefore we find that label propagation is approximately the mean field approximation to $\mathcal{F}$.

The graph mincut algorithm [11] finds the most likely state configuration of the same Markov random field $\mathcal{F}$, since the minimum cut corresponds to minimum energy. There is a subtle difference: assume the middle band in Figure 1(a) has no labeled point. Mincut will classify the middle band as either all o or all +, since these are the two most likely state configurations [12]. But label propagation, being more in the spirit of a mean field approximation, splits the middle band, classifying points in the upper half as o and lower half as + (with low confidence though). In addition, label propagation is not limited to binary classification.

In related work, we have also attempted to use Boltzmann machine learning on the Markov random field $\mathcal{F}$ to learn from labeled and unlabeled data, optimizing the length scale parameters using the likelihood criterion on the labeled points [13].

6 Summary

We proposed a label propagation algorithm to learn from both labeled and unlabeled data. Labels were propagated with a combination of random walk and clamping. We showed the solution to the process, and its connection to other methods. We also showed how to learn the parameters. As with various semi-supervised learning algorithms of its kind, label propagation works only if the structure of the data distribution, revealed by abundant unlabeled data, fits the classification goal. In the future we will investigate better ways to rebalance class proportions, applications of the entropy minimization criterion to learn propagation parameters from real datasets, and possible connections to the diffusion kernel [14].

Acknowledgments

We thank Sam Roweis, Roni Rosenfeld, Teddy Seidenfeld, Guy Lebanon, Jin Rong and Jing Liu for helpful discussions. Sam Roweis provided the handwritten digits dataset. The first author is supported in part by a Microsoft Research Graduate Fellowship.

References

[1] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

[2] Xiaojin Zhu. A very short survey on semi-supervised learning. Technical report, Carnegie Mellon University, 2002. in preparation.

[3] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50, 1956.

[4] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5), 1994.

[5] Y. Le Cun, B. Boser, J. S Denker, D. Henderson, R. E.. Howard, W. Howard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems II (Denver 1989)*. 1990.

[6] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

[7] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002. in preparation.

[8] Martin Szummer and Tommi Jaakkola. Partially labeled classification with Markov random walks. In *NIPS*, 2001.

[9] Carsten Peterson and James R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[10] Michael I. Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[11] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincut. In *Proc. 18th International Conf. on Machine Learning*, 2001.

[12] A. Blum. Personal Communication.

[13] Xiaojin Zhu and Zoubin Ghahramani. Towards semi-supervised learning with boltzmann machines. Technical report, Carnegie Mellon University, 2002. in preparation.

[14] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proc. 19th International Conf. on Machine Learning*, 2002.