Learning from Labeled and Unlabeled Data with Label Propagation

Xiaojin Zhu Zoubin Ghahramani

June 2002
CMU-CALD-02-107

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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 give the analysis of 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 do feature selection. Experiment results are promising.

Keywords: I.2.6 [Artificial Intelligence]:Learning; I.5.1 [Pattern Recognition]:Models–Statistical, I.5.2 [Pattern Recognition]:Design Methodology–Classifier design and evaluation; General Terms: Algorithms; Additional Key Words: semi-supervised learning, label propagation.

1 Introduction

Labeled data are essential for supervised learning. However they are often available in small quantities, while unlabeled data may be abundant. Using unlabeled data together with labeled data is of both theoretical and practical interest. There has been many proposed approaches recently [See01]. Among them there is a promising family of methods analogous to k-Nearest-Neighbor (kNN) in traditional supervised learning. These methods assume that closer data points tend to have similar class labels. As a result, they 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 neighboring nodes according to their proximity. Meanwhile we clamp 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, 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 can learn 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 $Y_L = \{y_1 \dots y_l\}$ 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 $X = \{x_1 \dots x_{l+u}\}$ where $x_i \in R^D$. The problem is to estimate Y_U from X and Y_L, which is a transductive learning setting.

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, 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 report, later allowing different σ's for each dimension, corresponding to length scales in Guassian processes.

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

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

where T_{ij} is the probability to jump from node j to i. Also define a $(l+u) \times C$ label matrix Y, whose ith row representing the label probability distribution of node x_i. The initialization of 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. Propagate $Y \leftarrow TY$
2. Row-normalize Y.
3. Clamp the labeled data. Repeat from step 1 until Y converges.

In step 1, all nodes propagate their labels (including self loop) for one step. In step 2 we row-normalize Y to maintain the label probability interpretation. Step 3 is critical: we want persistent label sources from labeled data. So instead of letting the initially labeled nodes fade away, we replenish them by clamping their label distribution to $Y_{ic} = \delta(y_i, c)$, i.e. the probability mass is concentrated on the given class. With this constant 'push' from labeled nodes, the class boundaries will be pushed through high density data filaments and settle in low density gaps. If this structure of data fits the classification goal, then our algorithm can use unlabeled data to help learning.

2.3 Convergence

We now show the algorithm converges to a simple solution. First, step 1 and 2 can be combined into

$$Y \leftarrow \bar{T}Y \tag{3}$$

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 $l \times C$ matrix formed by the top l rows of Y (the labeled data) and Y_U be the $u \times C$ matrix of the remaining u rows. Notice Y_L never change because of the clamping so 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

$$\bar{T} = \begin{bmatrix} \bar{T}_{ll} & \bar{T}_{lu} \\ \bar{T}_{ul} & \bar{T}_{uu} \end{bmatrix} \tag{4}$$

It can be shown that our algorithm is

$$Y_U \leftarrow \bar{T}_{uu} Y_U + \bar{T}_{ul} Y_L \tag{5}$$

which leads to

$$Y_U = \lim_{n\to\infty} \bar{T}_{uu}^n Y^0 + [\sum_{i=1}^{n} \bar{T}_{uu}^{(i-1)}]\bar{T}_{ul}Y_L \tag{6}$$

where Y^0 is the initial Y. We need to show $\bar{T}_{uu}^n Y^0 \to 0$. By construction, all elements in $\bar{T}$ is greater than zero. Since $\bar{T}$ is row normalized, and $\bar{T}_{uu}$ is a sub-matrix of $\bar{T}$, it follows

$$\exists \gamma < 1, \sum_{j=1}^{u} \bar{T}_{uu_{ij}} \leq \gamma, \forall i = 1 \ldots u \tag{7}$$

Therefore

$$\begin{aligned} \sum_j \bar{T}_{uu_{ij}}^n &= \sum_j \sum_k \bar{T}_{uu_{ik}}^{(n-1)} \bar{T}_{uu_{kj}} && (8)\\ &= \sum_k \bar{T}_{uu_{ik}}^{(n-1)} \sum_j \bar{T}_{uu_{kj}} && (9)\\ &\leq \sum_k \bar{T}_{uu_{ik}}^{(n-1)} \gamma && (10)\\ &\leq \gamma^n && (11)\end{aligned}$$

So the row sums of $\bar{T}_{uu}^n$ converges to zero, which means $\bar{T}_{uu}^n Y^0 \to 0$. Thus the initial point Y^0 is inconsequential. Obviously

$$Y_U = (I - \bar{T}_{uu})^{-1}\bar{T}_{ul}Y_L \tag{12}$$

is a fixed point. Therefore it is the unique fixed point and the solution to our iterative algorithm. This gives us a way to solve the label propagation problem directly without iterative propagation. The solution can often be obtained by solving a sparse system of linear equations (i.e. if we truncate small elements on $\bar{T}_{uu}$), which can be efficient.

2.4 Parameter Setting

We set the parameter σ with the following heuristic. We find a minimum spanning tree over all data points with Euclidean distances d_{ij}, with Kruskal's Algorithm [Kru56]. 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 to the minimum distance between class regions. We arbitrarily set $\sigma = d^0/3$ following the 3σ rule of Normal distribution, so that the weight of this edge is close to 0, with the hope that local propagation is then mostly within classes.

2.5 Rebalancing Class Proportions

For classification purpose, once Y_U is computed, we can take the most likely (ML) class of each unlabeled point as its label. This way we have no control over the final proportion of classes, because it is implicitly determined by the distribution of data. It is appropriate when classes are well separated or labeled data abound. If it is not the case, however, 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

  After computing Y_U, we normalize the class mass to fit the class proportion constraint. The class mass is the column sums of Y_U, denoted by $Y_{U_{.1}} \ldots Y_{U_{.c}}$. We scale each column such that $Y_{U_{.1}} : \ldots : Y_{U_{.c}} = P_1 : \ldots : P_C$. The label of a point is the class with the maximum element in that row after scaling. The approach does not guarantee strict label proportion.

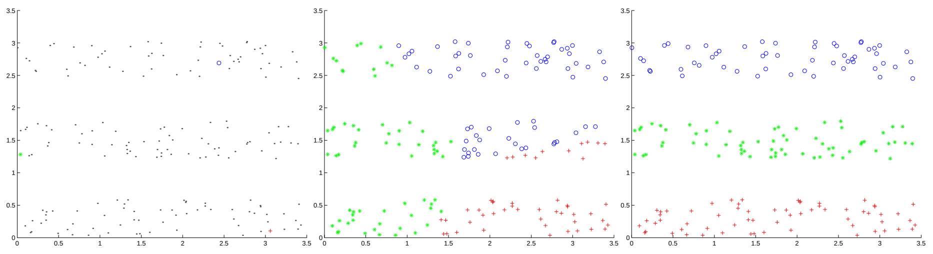
- Label Bidding

  We have uP_c class c labels for sale, for $c = 1 \ldots C$. After computing Y_U, we view each point i as having bids $\$Y_{U_{ic}}$ for class c. Bids are processed from high to low. Assuming $Y_{U_{ic}}$ is currently the highest bid. If class c labels remain, a c label is sold to point i, and point i quits the bidding. Otherwise the bid $Y_{U_{ic}}$ is ignored and the second highest bid is processed, and so on. Label bidding guarantees that strict label proportions will be met.

3 Experimental Results

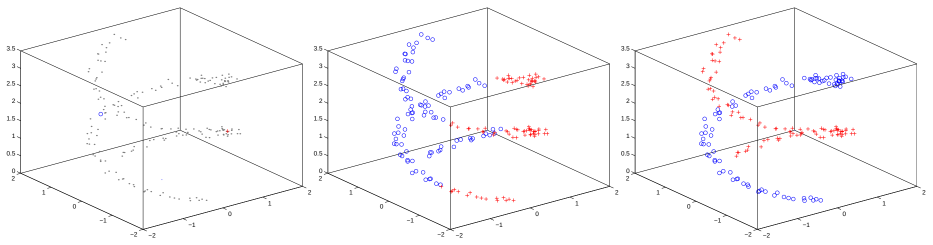
To demonstrate properties of this algorithm we investigate both synthetic datasets and a real-world classification problem. Figure 1 shows a synthetic dataset with 3 classes, each being a narrow horizontal band. Data points are randomly drawn from the classes. There are 3 labeled points and 178 unlabeled points. As expected, kNN algorithm ($k = 1$) ignores the band structure within the dataset, while our algorithm takes advantage of it and propagates labels along the bands. In this example, we used $\sigma = 0.22$ from the MST heuristic, and ML classification.

Similarly, Figure 2 shows a synthetic dataset with 2 classes as intertwined three dimensional springs. There are 2 labeled points and 184 unlabeled points. Again, kNN fails to notice the structure of unlabeled data, while our algorithm finds the springs. We used $\sigma = 0.43$ from MST and ML classification.



(a) The data (b) kNN (c) Label propagation

Figure 1: The 3 Bands dataset. Labeled data are color symbols and unlabeled data are dots in (a). kNN ignores unlabeled data structure, while label propagation uses it.



(a) The data (b) kNN (c) Label propagation

Figure 2: The Springs dataset.

For a real world example, we test label propagation on a handwritten digits dataset. The original dataset was from the Cedar Buffalo binary digits database [Hul94]. The digits were then preprocessed to reduce the size of each digit image down to 16x16 by down-sampling and Gaussian smoothing. This interpolation thus created gray-scale with pixel values range from 0 to 255 [CBD+90]. We use digits 1, 2 and 3 in our experiment as three classes. Each class has 1100 images, with a total of 3300. Each image is represented by a 256 dimensional vector. Figure 3(a) shows a random sample of 100 images from the dataset.

We perform label propagation on random labeled / unlabeled splits of the dataset, and measure classification error rates on the unlabeled data. We vary labeled data size l from 3 up to 100. For a given labeled data size, we perform 20 trials. In each trial we randomly sample labeled data from the whole dataset, and use the rest 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 then run the minimum spanning tree algorithm on the split to find σ. All trials have σ close to 340. To speed up computation, only the top 150 neighbors are considered for each image when constructing the transition matrix T. We measure the error rates of:

1. ML: The most likely labels (see section 2.5).
2. CNe: Class mass normalization post processing, with maximum likelihood estimate of class proportions from labeled data.
3. LBe: Label bidding post processing, with maximum likelihood estimate of class proportions from labeled data.
4. CNo: Class mass normalization post processing, 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 alternative algorithms as baselines. The first one is standard kNN. We report 1NN error rate since it is the best among $k = 1 \dots 11$. 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 Figures 1(c) and 2(c).

Figure 3(b)–(f) shows the results. In (b), when l is small, ML labeling is worse than 1NN, but when $l \geq 40$, ML is better. However if we rebalance class proportions, we can do much better. If we estimate class proportions from labeled data by class frequency and perform class mass normalization, we improve the performance when l is small (c). (Note, however, that we required all classes to be present in training set. This biases class frequency towards uniform, which happens to be the true proportion in this example. This is especially true when $l = 3$, and explains the initial ramp in (c)(d).) If we have a priori knowledge of the true class proportion on the unlabeled data, the performance is even better (e,f), with label bidding being slightly superior to class mass normalization. But it should be noted that we are using extra information not employed by the baseline algorithms. 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. Meanwhile 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 bold face.

4 Parameter Learning in Label Propagation

4.1 The Effect of σ on Label Propagation

We used minimum spanning trees as a heuristic to set the parameter σ. Let's see how different σ values affect label propagation. Figure 4(a) shows a synthetic

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

Table 1: Average error rate over 20 trials with different post processing methods and sizes of labeled data.

dataset 'Bridge' which consists of a upper block and a lower block, connected by a thin bridge. The upper block is on a grid with side 0.95, while the lower one is on a looser grid with side 1. Points on the bridge have distance 0.67 between them. There are two labeled points. The dataset simulates two otherwise well separated classes, corrupted by a few points (the bridge). The p1NN algorithm will propagate the + label in the upper block first, since the block is denser. And once reaching the bridge, the + label travels down and competes with the o label in the lower block. As a result, the + label takes the majority of the territory (Figure 4(b)). This also shows that p1NN is susceptible to small noise in dataset. The standard 1NN algorithm finds a decision boundary bisecting the labeled points, regardless of the structure of data (Figure 4(c)). Figure 4(d)–(h) shows the results of label propagation with different σ and ML classification.

When $\sigma \to 0$, label propagation result approaches p1NN, because under the exponential weights (1) the influence of the nearest point dominates. when $\sigma \to \infty$, it is easy to see that all unlabeled points will have similar class probabilities, which is the class frequency on labeled data. This is because the whole dataset essentially shrinks to a single point with large σ, and all unlabeled points receive the same influence from all labeled points. The 'appropriate' σ is in between (Figure 4(f)). How to learn it? Data likelihood does not make sense as a criterion in our setting because they are clamped (We can of course 'unclamp' labeled points at the end and let their labels be determined by nearby unlabeled points. But intuitively we should pay more attention to how unlabeled data are assigned labels). Instead we minimize the *entropy* H of the result

$$H = -\sum_{ij} Y_{ij} \log Y_{ij} \tag{13}$$

which is the sum of entropy on individual data points. H is a function of σ. The intuition is that a good σ should label all points confidently, and thus minimize H. 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 $\sigma \to 0$ approaches p1NN, and p1NN gives each point a hard label), as Figure 4(i) shows. This (p1NN) is not always desirable (Figure 4(b,d)), and can be fixed by smoothing the transition matrix, as we will see next.

4.2 Smoothing the Transition Matrix T

Inspired by the analysis on the PageRank algorithm [NZJ01], we smooth T by interpolating it with a uniform transition matrix $\mathcal{U}$, where $\mathcal{U}_{ij} = 1/(l+u), \forall i, j$:

$$\tilde{T} = \epsilon \mathcal{U} + (1-\epsilon)T \tag{14}$$

$\tilde{T}$ is then used in place of T in the algorithm. ϵ is the interpolation parameter. With the smoothed transition matrix $\tilde{T}$, when $\sigma \to 0$ the uniform component dominates (outside the self loop). And the class probabilities are close to uniform on unlabeled data, resulting in high entropy H. On the other hand when σ is not very small, the original T dominates and the results are the same as the unsmoothed version. Figure 4(i) shows the curve of entropy H vs. σ before and after smoothing, with different ϵ values. In the following we use the value $\epsilon = 0.0005$.

4.3 The Derivative of H

Although we introduced a nuisance parameter ϵ in order to learn σ, the advantage will be apparent when we introduce multiple parameters $\sigma_1 \ldots \sigma_D$, one for each dimension. That is, let the weight now be

$$w_{ij} = \exp\left(-\sum_{d=1}^{D} \frac{(x_i^d - x_j^d)^2}{\sigma_d^2}\right) \tag{15}$$

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. We find the derivatives of H w.r.t. σ_d's by the chain rule,

$$\frac{\partial H}{\partial \sigma_d} = \sum_{i=l+1}^{l+u} \sum_{c=1}^{C} \frac{\partial H}{\partial Y_{ic}} \frac{\partial Y_{ic}}{\partial \sigma_d} \tag{16}$$

$$\frac{\partial H}{\partial Y_{ic}} = -\log Y_{ic} - 1 \tag{17}$$

The value $\partial Y_{ic}/\partial\sigma_d$ can be read off the matrix $\partial Y_U/\partial\sigma_d$, which is

$$\begin{aligned}
\frac{\partial Y_U}{\partial\sigma_d} &= \frac{\partial}{\partial\sigma_d}[(I-\bar{T}_{uu})^{-1}\bar{T}_{ul}Y_L] && (18)\\
&= [\frac{\partial}{\partial\sigma_d}(I-\bar{T}_{uu})^{-1}]\bar{T}_{ul}Y_L + (I-\bar{T}_{uu})^{-1}[\frac{\partial}{\partial\sigma_d}\bar{T}_{ul}]Y_L && (19)\\
&= (I-\bar{T}_{uu})^{-1}[\frac{\partial\bar{T}_{uu}}{\partial\sigma_d}](I-\bar{T}_{uu})^{-1}\bar{T}_{ul}Y_L + (I-\bar{T}_{uu})^{-1}[\frac{\partial\bar{T}_{ul}}{\partial\sigma_d}]Y_L && (20)\\
&= (I-\bar{T}_{uu})^{-1}[\frac{\partial\bar{T}_{uu}}{\partial\sigma_d}]Y_U + (I-\bar{T}_{uu})^{-1}[\frac{\partial\bar{T}_{ul}}{\partial\sigma_d}]Y_L && (21)
\end{aligned}$$

where we used the fact $dX^{-1} = -X^{-1}(dX)X^{-1}$. Both $\partial\bar{T}_{uu}/\partial\sigma_d$ and $\partial\bar{T}_{ul}/\partial\sigma_d$ are sub-matrices of $\partial\bar{T}/\partial\sigma_d$. Remember $\bar{T}$ is $\tilde{T}$ row-normalized,

$$\begin{aligned}
\frac{\partial\bar{T}_{ij}}{\partial\sigma_d} &= \sum_{m,n=1}^{l+u}\frac{\partial\bar{T}_{ij}}{\partial\tilde{T}_{mn}}\frac{\partial\tilde{T}_{mn}}{\partial\sigma_d} && (22)\\
&= \sum_{m,n=1}^{l+u}[\frac{\delta(m,i)}{\sum_k\tilde{T}_{ik}}(\delta(n,j)-\bar{T}_{ij})]\frac{\partial\tilde{T}_{mn}}{\partial\sigma_d} && (23)\\
&= [\frac{\partial\tilde{T}_{ij}}{\partial\sigma_d} - \bar{T}_{ij}\sum_n\frac{\partial\tilde{T}_{in}}{\partial\sigma_d}]/\sum_k\tilde{T}_{ik} && (24)
\end{aligned}$$

$\tilde{T}$ is the smoothed transition matrix (eq. (14)), so

$$\frac{\partial\tilde{T}_{mn}}{\partial\sigma_d} = (1-\epsilon)\frac{\partial T_{mn}}{\partial\sigma_d} \tag{25}$$

T is the original transition matrix (eq. (2)),

$$\begin{aligned}
\frac{\partial T_{mn}}{\partial\sigma_d} &= \sum_{s,t=1}^{l+u}\frac{\partial T_{mn}}{\partial w_{st}}\frac{\partial w_{st}}{\partial\sigma_d} && (26)\\
&= \sum_{s,t=1}^{l+u}[\frac{\delta(t,n)}{\sum_r w_{rn}}(\delta(s,m)-T_{mn})]\frac{\partial w_{st}}{\partial\sigma_d} && (27)\\
&= [\frac{\partial w_{mn}}{\partial\sigma_d} - T_{mn}\sum_s\frac{\partial w_{sn}}{\partial\sigma_d}]/\sum_r w_{rn} && (28)
\end{aligned}$$

finally,

$$\frac{\partial w_{st}}{\partial\sigma_d} = 2w_{st}(x_s^d - x_t^d)^2/\sigma_d^3 \tag{29}$$

4.4 Examples: Learning Irrelevant Dimensions

We find the optimal σ for the datasets in Figures 1 and 2, assuming a single σ. For the '3 Bands' dataset, we start from the minimum spanning tree heuristic $\sigma = 0.22$ ($H = 110.1$), and find the optimal $\sigma = 0.26$ ($H = 94.9$). Similarly for the 'Springs' dataset, we start from the heuristic $\sigma = 0.43$ ($H = 61.3$) and find it is already the optimal. Both classification remains the same (for all experiments in this section, we used ML classification). The MST heuristic for these well separated datasets is close to optimal.

With multiple σ's, for the 'Bridge' dataset in Figure 4, we start from $\sigma_1 = \sigma_2 = 0.67$ ($H = 34.2$) which is a third the distance between the upper and lower blocks. After substantial amount of gradient descent iterations, we reach $\sigma_1 = 121.3$, $\sigma_2 = 0.55$ and $H = 21.1$. And σ_1 is still increasing while σ_2 and H asymptote. The classification result is the same as Figure 4(f). Local minimum is a problem though, for example, if we start from $\sigma_1 = \sigma_2 = 0.33$ which is the minimum spanning tree heuristic, we would end up with $\sigma_1 = 0.44$, $\sigma_2 = 0.69$ and $H = 35.8$.

The ever increasing σ_1 is expected. In the 'Bridge' dataset, the horizontal dimension (corresponding to σ_1) is irrelevant to classification. A large σ means labels can be propagated freely along this dimension. Thus our algorithm detects that dimension 1 is irrelevant to classification. To further illustrate this, we create a synthetic dataset 'Ball', with 400 data points uniformly sampled inside a 4-dimensional unit hypersphere, and with a $45°$ gap in dimensions 1-2 (Figure 5). There is no gap in all other two dimensional projection of the dataset. The gap splits the dataset into two classes, while dimensions 3 and 4 are irrelevant to classification. There are two labeled points. We start from $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 0.2$ ($H = 240.9$), and reach $\sigma_1 = 0.17$, $\sigma_2 = 0.19$, $\sigma_3 = 2.29$, $\sigma_4 = 7.41$ with $H = 130.1$. While not as dramatic as in the 'Bridge' dataset, σ_3 and σ_4 are very large compared to the radius of the data, and signify the irrelevance of these two dimensions. The classification at the optimal σ's obeys the gap.

To show that our method is not merely looking for structures in unlabeled data, consider a similar dataset 'Ball2' which is the same as 'Ball' except that there is now also a gap in dimensions 3-4 (Figure 6. There is no gap in dimensions 1-3, 2-3, 2-4). So from unlabeled data point of view, dimensions 3, 4 is as interesting as dimensions 1, 2. 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 4 labeled points (Figure 6(a,b)). As before we start from $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 0.2$ ($H = 140.6$), and reach $\sigma_1 = 0.18$, $\sigma_2 = 0.19$, $\sigma_3 = 14.8$, $\sigma_4 = 13.3$ with $H = 78.7$. Again our method thinks dimensions 3, 4 are irrelevant, 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 [SJ01]. Both utilize the manifold structure defined by large amount of unlabeled data, and assume the structure is correlated to classification goal. 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 labels, one 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 of labeled data 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 [PA87] [JGJS99]. In label propagation, upon convergence we have the equations (for unlabeled data)

$$Y_{ic} = \frac{\sum_j T_{ij} Y_{jc}}{\sum_{c'} \sum_j T_{ij} Y_{jc'}} \tag{30}$$

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 $Y_i = (\delta(y_i, 1), \ldots, \delta(y_i, C))$. The probability of a particular configuration Y in $\mathcal{F}$ is

$$P_{\mathcal{F}}(Y) = \frac{1}{Z} \exp[\sum_{ij} w_{ij} Y_i Y_j^T] \tag{31}$$

As we will see later, $\mathcal{F}$ is related to the Mincut algorithm. We now show label propagation (30) is approximately a mean field solution to a Markov random field $\mathcal{F}'$ that approximates $\mathcal{F}$. The Markov random field $\mathcal{F}'$ has the same structure as $\mathcal{F}$, but besides pairwise interactions, $\mathcal{F}'$ has all higher order interactions. Specifically $\mathcal{F}'$ is defined as

$$P_{\mathcal{F}'}(Y) = \frac{1}{Z} \exp[\log(\sum_{ij} w_{ij} Y_i Y_j^T)] \tag{32}$$

$\mathcal{F}'$ is the same as $\mathcal{F}$ up to the first order:

$$\begin{aligned} P_{\mathcal{F}'}(Y) &\approx \frac{1}{Z}\exp[\sum_{ij} w_{ij} Y_i Y_j^T - 1] && (33)\\ &= \frac{1}{Z}\exp[\sum_{ij} w_{ij} Y_i Y_j^T] && (34)\\ &= P_{\mathcal{F}}(Y) && (35)\end{aligned}$$

To find the mean field approximation to $\mathcal{F}'$, we replace each node i's state Y_i with its mean value $< Y_i >$ and obtain the self consistent equations

$$\begin{aligned} < Y_i > &= \sum_{Y_i} \frac{1}{Z'}\exp[\log(\sum_j w_{ij} Y_i < Y_j >^T)] Y_i && (36)\\ &= \sum_{Y_i} \frac{1}{Z'}(\sum_j w_{ij} Y_i < Y_j >^T) Y_i && (37)\end{aligned}$$

which leads to the mean field solution to $\mathcal{F}'$:

$$< Y_{ic} >= \frac{\sum_j w_{ij} < Y_{jc} >}{\sum_{c'} \sum_j w_{ij} < Y_{jc} >} \qquad (38)$$

(30) is an approximation to (38) 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 (30). Therefore we find that label propagation is approximately the mean field approximation to $\mathcal{F}$.

With this view, it is easy to compare label propagation to the graph mincut algorithm [BC01]. Mincut algorithm finds the minimum cut through a graph to separate labeled data of different classes. This corresponds to a state configuration with minimum energy, or equivalently the most likely state configuration, of the same Markov random field $\mathcal{F}$. Label propagation, as we have seen, finds the most likely state configuration of the *approximate mean field solution* of $\mathcal{F}$. There is a subtle difference between the two algorithms, as in Figure 7. The dataset is perfectly symmetric with two labeled points. Mincut will label the middle band with either all + or all o, since these two are the two equally most likely state configurations [Blu]. But label propagation, being more in the spirit of a mean field approximation, will split the middle band, classifying points in the upper half as o and lower half as + (with low confidence though). In addition, mincut is limited to binary labels while label propagation is not.

Unlike many semi-supervised learning algorithms, in label propagation labeled data are fixed and cannot change during iterations. One might argue that it would reinforce mistakes when labeled data are noisy. This is true, and label propagation assumes noise free labels. Under this assumption it is important to be able to insist on labels, without being carried away by unlabeled data distribution. In practice since labeled dataset is small, it might be relatively easy to make sure it is correctly labeled. An interesting open question is when we are not confident about a label, whether we can clamp it to some soft distribution in Y_L to express the uncertainty.

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 [ZG02].

6 Summary and Discussion

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 algorithm 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 [KL02].

Acknowledgement

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

[BC01] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincut. In *Proc. 18th International Conf. on Machine Learning*, 2001.

[Blu] A. Blum. Personal Communication.

[CBD$^+$90] 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.

[Hul94] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5), 1994.

[JGJS99] 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.

[KL02] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proc. 19th International Conf. on Machine Learning*, 2002.

[Kru56] 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.

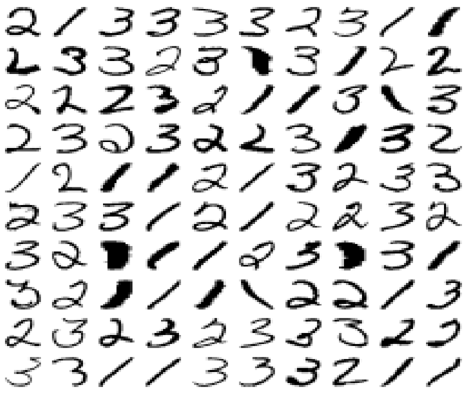
[NZJ01] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

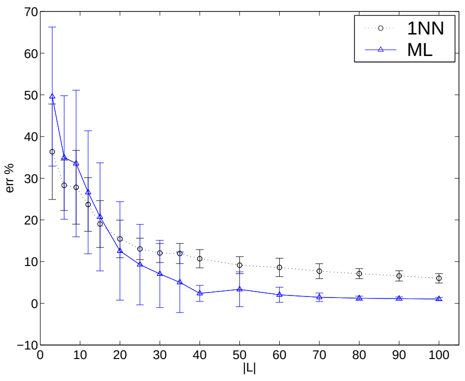
[PA87] Carsten Peterson and James R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[See01] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

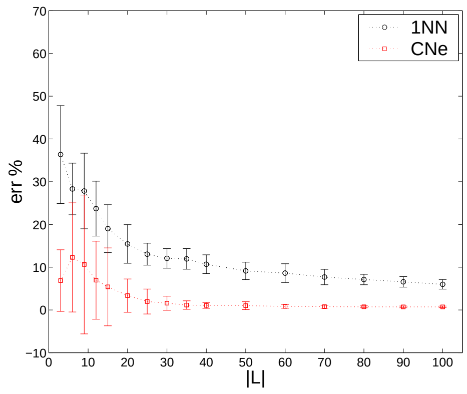
[SJ01] Martin Szummer and Tommi Jaakkola. Partially labeled classification with Markov random walks. In *NIPS*, 2001.

[ZG02] Xiaojin Zhu and Zoubin Ghahramani. Towards semi-supervised classification with Markov random fields. Technical Report CMU-CALD-02-106, Carnegie Mellon University, 2002.

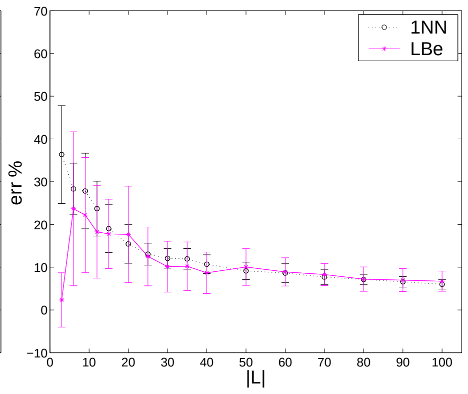

(a) A random sample



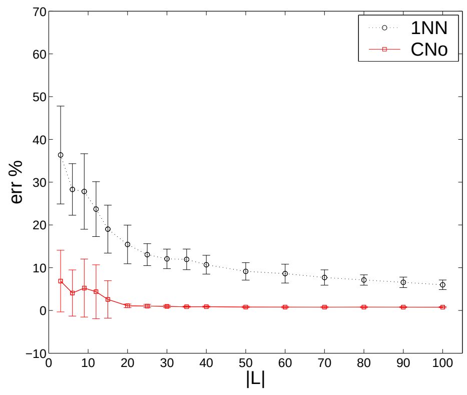
(b) ML labels



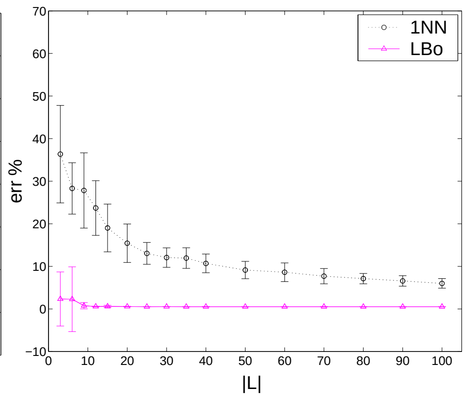
(c) CNe labels



(d) LBe labels

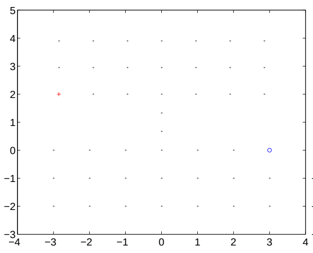


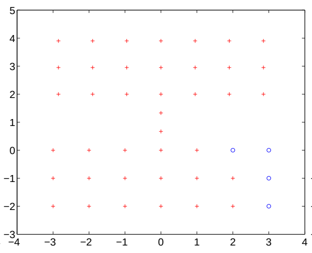
(e) CNo labels

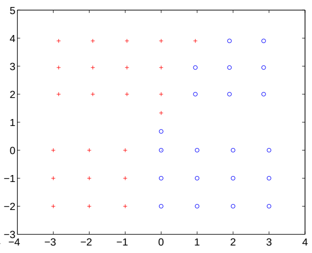


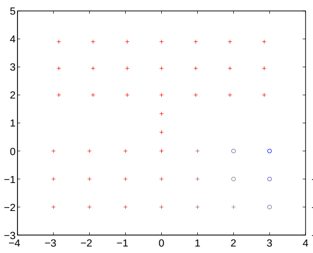
(f) LBo labels

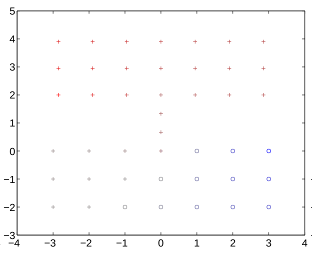
Figure 3: The digits dataset. (a) shows 100 randomly sampled images. (b)–(f) show the error rates of different post processing methods. Each point is an average of 20 random trials. The error bars are ±1 standard deviation. LBo is the best if we have oracle class proportion knowledge, otherwise CNe is the best.

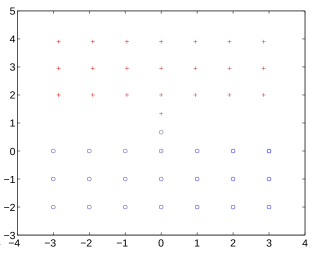

(a) The data

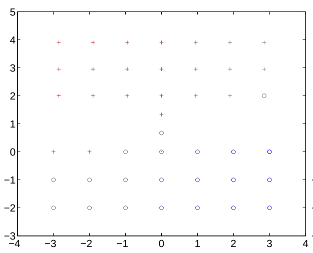

(b) p1NN

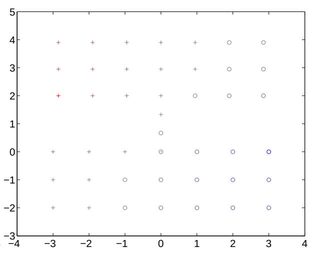

(c) 1NN

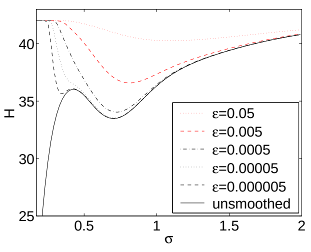

(d) $\sigma = 0.17$


(e) $\sigma = 0.40$


(f) $\sigma = 0.72$

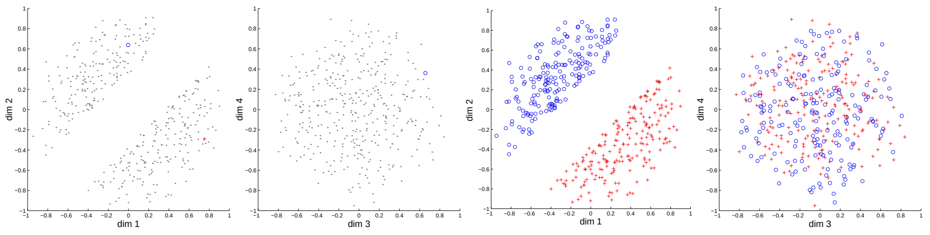

(g) $\sigma = 1.30$


(h) $\sigma = 2.00$



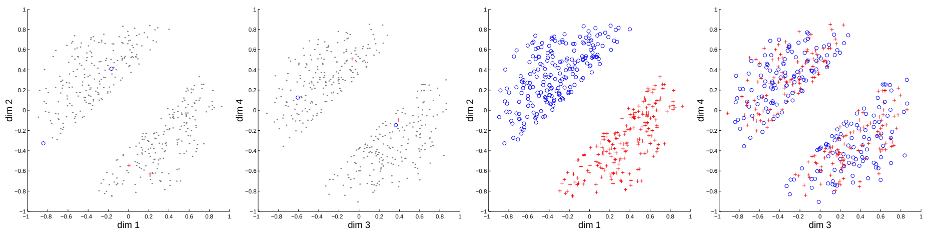
(i) Smoothing

Figure 4: The Bridge dataset. (d)–(h) are the results of label propagation with different σ values before smoothing T. (i) shows the entropy vs. σ curve for label propagation before and after smoothing T with different ϵ values.



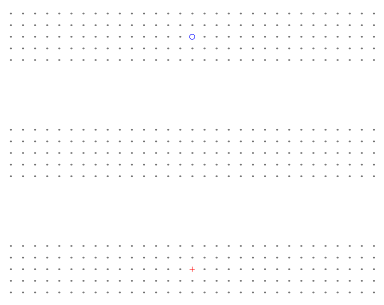
(a) data, dim 1,2 (b) data, dim 3,4 (c) result, dim 1,2 (d) result, dim 3,4

Figure 5: The Ball dataset. Dimensions 3,4 are deemed irrelevant because of the lack of structure in unlabeled data. The result is at optimal σ's learned from gradient descent.

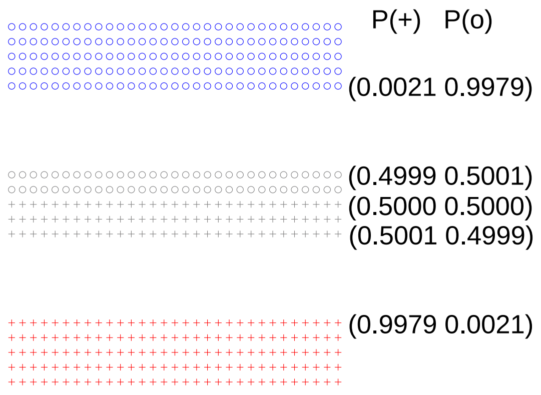


(a) data, dim 1,2 (b) data, dim 3,4 (c) result, dim 1,2 (d) result, dim 3,4

Figure 6: The Ball2 dataset. There is structure in both dimensions 1,2 and 3,4. But the four labeled points indicate the irrelevance of dimensions 3,4.



(a) The data



(b) Label propagation result

Figure 7: Label propagation on a symmetric dataset. The middle band splits in label, but with close to uniform probabilities. A graph mincut algorithm will generate either all + or all o labels for the middle band.