A Note on Semi-Supervised Learning using Markov Random Fields

Wei Li and Andrew McCallum
{weili, mccallum}@cs.umass.edu
Computer Science Department
University of Massachusetts Amherst

February 3, 2004

Abstract

This paper describes conditional-probability training of Markov random fields using combinations of labeled and unlabeled data. We capture the similarities between instances learning the appropriate distance metric from the data. The likelihood model and several training procedures are presented.

1 Introduction

While obtaining labeled data is often expensive, unlabeled data is usually plentiful and cheap. Several models have been proposed for *semi-supervised learning* in which a combination of labeled and unlabeled data is used to train a learner for some supervised task. However, some results with such models—including mixtures of multinomials trained by EM [Nigam *et al.*2000], and SVM transduction [Joachims1999]—show that incorporating unlabeled data only sometimes improves accruacy, and other times hurts accuracy [Nigam2001].

This paper characterizes situations in which unlabeled data hurts accuracy as occurring when the "natural clustering" of the data is not in "good correspondence" with the class labels. In other words, poor performance results when instances of differing class labels are often found "near" each other, and "large distance spans containing only a few instances" do not correspond to class boundaries.

Thus, this paper contends that having a good distance metric is a key requirement for success in semi-supervised learning. In many models the distance metric is implicit and treated as a given, but the right distance metric is task-specific, and should be learned. Consider that, given a set of inputs, it is reasonable to expect that differing classification tasks could be performed on this set. For example, given a set of news articles, one might classify them according to topic or according to style. Different distance metrics would be appropriate to each task.

Previous work [McCallum and Minka1999] addresses this issue by using the limited labeled data and large quantities of unlabeled data to bootstrap the learning of a distance metric for a multinomial naive Bayes classifier. In this classifier the metric is represented with Polya trees (Dirichlet trees) as the prior over multinomial mixture components. Polya trees are a distribution over multinomials, but unlike the traditional Dirichlet, they can represent differing variance in different dimensions (and thus can emphasize some dimensions while down-playing others). This model has demonstrated some promise, however, discriminative or conditionally-trained models tend to perform better classification than generative models. Thus we seek a discriminative model that also learns a distance metric as an integral part of its training.

This paper outlines a model for conditionally-trained, semi-supervised learning in which distance metric learning is an integral part of training. The approach is based on Markov random fields with tied parameters, in which there are some parameters impacting classification of individual instances dependent on their features, and other parameters encouraging instances with related, relevant features to share the same class label.

2 A Purely Supervised Model for Classification

Consider a traditional log-linear, conditionally-trained classfier, (also known as a maximum entropy classifier, or, in the case of a binary class label, logistic regression). Let y be a class label from a finite set of classes, $\mathcal{Y}$, and let x be some arbitrary input on which various feature functions, $f(x, y)$ return real values. The model defines the conditional likelihood

$$P_\Lambda(y|x) = \frac{1}{Z_x} \exp\left(\sum_k \lambda_k f_k(x, y)\right),$$

where $Z_x = \sum_{y' \in \mathcal{Y}} \exp\left(\sum_k \lambda_k f_k(x, y')\right)$ is the partition function.

The parameters $\Lambda = \{\lambda ...\}$ are trained to maximize penalized log-likelihood of a set of training labeled data $\mathcal{D}_L = \{(x_1, y_1), ...(x_l, y_l)\}$. We will use the notation $\mathbf{x}_L$ for the x's in this labeled training set and $\mathbf{y}_L$ for the y's in this labeled training set. The penalized log-likelihood is

$$\mathcal{L} = \log(P_\Lambda(\mathbf{y}_L|\mathbf{x}_L) = \sum_{i=1}^{l} \log\left(P_\Lambda(y_i|x_i)\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2},$$

where the second sum is a Gaussian prior on the parameters to handle sparsity in the training data. For each $y \in \mathcal{Y}$ there is a separate "default feature" that is always on independent of x; this allows the model to represent a class prior probability.

Standard optimization techniques such as conjugate gradient or limited-memory BFGS [Byrd *et al.*1994, Malouf] can find the parameters that maximize this function. The gradient is

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \left(\sum_{i=1}^{l} f_k(x_i, y_i) - \sum_{y' \in \mathcal{Y}} P_\Lambda(y'|x_i) f(x_i, y') \right) - \frac{\lambda_k}{\sigma^2}.$$

Note that this model corresponds to an undirected graphical model (also known as a Markov random field) in which the observed random variables x_i are connected by an undirected edge to their corresponding random varibles y_i, and the cliques of the graph are exactly these pairs, with clique potential $\Phi_i = \exp\left(\sum_k \lambda_k f_k(x_i, y_i)\right)$.

3 A Semi-Supervised Model for Classification

Now consider the case where, in addition to labeled data $\mathcal{D}_\mathcal{L} = \{(x_1, y_1), ...(x_l, y_l)\}$, we also have available unlabeled data $\mathcal{D}_U = \{(x_{l+1}), ...(x_{l+u})\}$.

We will use the notation $\mathbf{x}_U$ for the x's in the unlabeled set, and the notation $\mathbf{y}_U$ for the (unknown) labels of $\mathbf{x}_U$. Furthermore, let $\mathbf{x} = \mathbf{x}_L \cup \mathbf{x}_U$ and $\mathbf{y} = \mathbf{y}_L \cup \mathbf{y}_U$.

We wish to use the unlabeled data to improve parameter estimation. If we use the classification model from the previous section, the unlabeled data have no impact. One can see this by considering maximizing the likelihood

$$\begin{aligned}
\mathcal{L} &= \log P_\Lambda(\mathbf{y}_L|\mathbf{x}_L, \mathbf{x}_U) = \log \sum_{\mathbf{y}_U} P_\Lambda(\mathbf{y}_L, \mathbf{y}_U|\mathbf{x}_L, \mathbf{x}_U) \\
&= \log \sum_{\mathbf{y}_U} P_\Lambda(\mathbf{y}_L, |\mathbf{x}_L) P_\Lambda(\mathbf{y}_U|\mathbf{x}_U) = \log P_\Lambda(\mathbf{y}_L, |\mathbf{x}_L) \sum_{\mathbf{y}_U} P_\Lambda(\mathbf{y}_U|\mathbf{x}_U) \\
&= \log P_\Lambda(\mathbf{y}_L, |\mathbf{x}_L)
\end{aligned}$$

and noting that $\mathbf{x}_U$ does not appear in the gradient $\partial\mathcal{L}/\partial\lambda_k$.

For the unlabeled data to make a difference in conditional-likelihood training, we must allow the unlabeled training data $\mathbf{x}_U$ to have some impact on the labels $\mathbf{y}_L$. This is accomplished by introducing dependencies between $\mathbf{x}_U$ and $\mathbf{y}_L$.

One way (but not the only way) to do this is to add new features and weights (and thus corresponding cliques of the undirected graphical model) that assess aspects of pairs of examples—(x_i, y_i) and (x_j, y_j)—across all pairs, (including both labeled data, and unlabeled data with hypothesized labels). The corresponding new complete-data likelihood model is

$$P_\Lambda(\mathbf{y}_L, \mathbf{y}_U|\mathbf{x}_L, \mathbf{x}_U) = \frac{1}{Z_\mathbf{x}} \exp\left(\sum_i \sum_k \lambda_k f_k(x_i, y_i) + \sum_{i<j} \sum_{k'} \lambda_{k'} f_{k'}(x_i, x_j, y_i, y_j) \right),$$

where $Z_\mathbf{x} = \sum_{\mathbf{y}'} \exp\left(\sum_i \sum_k \lambda_k f_k(x_i, y'_i) + \sum_{i<j} \sum_{k'} \lambda_{k'} f_{k'}(x_i, x_j, y'_i, y'_j) \right)$ is the new partition function.

In some cases[1] it is reasonable to simplify this expression by ignoring the individual values of y_i and y_j, and only determining whether or not the two classes are equal. Let $y_{ij} = 1$ iff $y_i = y_j$, and then the complete-data likelihood in this simpler model is

$$P_\Lambda(\mathbf{y}_L, \mathbf{y}_U | \mathbf{x}_L, \mathbf{x}_U) = \frac{1}{Z_\mathbf{x}} \exp\left(\sum_i \sum_k \lambda_k f_k(x_i, y_i) + \sum_{i<j} \sum_{k'} \lambda_{k'} f_{k'}(x_i, x_j, y_{ij})\right).$$

Parameter estimation in this new semi-supervised model maximizes the incomplete penalized log-likelihood:

$$\begin{aligned}
\mathcal{L} &= \log P_\Lambda(\mathbf{y}_L | \mathbf{x}_L, \mathbf{x}_U) \\
&= \log \sum_{\mathbf{y}_U} P_\Lambda(\mathbf{y}_L, \mathbf{y}_U | \mathbf{x}_L, \mathbf{x}_U) \\
&= \left(\log \frac{1}{Z_\mathbf{x}} \sum_{\mathbf{y}_U} \exp\left(\sum_i \sum_k \lambda_k f_k(x_i, y_i) + \sum_{i<j} \sum_{k'} \lambda_{k'} f_{k'}(x_i, x_j, y_{ij})\right)\right) \\
&\quad - \sum_k \frac{\lambda_k^2}{2\sigma^2} - \sum_{k'} \frac{\lambda_{k'}^2}{2\sigma'^2}.
\end{aligned}$$

The features $f_{k'}$ and parameters $\lambda_{k'}$ parameterize the distance measure used to compare two examples. For example, for a document classification task, one such feature might be 0 in most cases, and 1 iff document x_i and document x_j both contain the word "airplane" and y_{ij} is 1; another corresponding feature would be 1 iff the same word appeared in both, but y_{ij} is 0. In such a scheme, parameters $\lambda_{k'}$ correspond to weights on each "dimension of the input space." Of course, the features $f_{k'}$ could capture arbitrarily complex features of the input, including conjunctions, and conjunctions could also be induced [McCallum2003].

Maximum likelihood parameter estimation in this model thus corresponds to simultaneously learning a classifier and a distance metric that maximize the (penalized) conditional likelihood of the labels given the labeled and unlabeled inputs.

When the number of labeled points is very small, the model might severely overfit and the priors σ and σ' will become more important. The relative importance of (1) using the features of an instance to classify it independently of other instances, and (2) using the distance measure to encourage nearby points to agree on their class label can be controlled by using different values for σ and σ'. One might also consider enforcing constraints based on the class label proportions found in the labeled data, as in co-training [Blum and Mitchell1998].

[1]Situations in which this simplication may not be desirable include: (1) when there is structure in the class label "output" space, $\mathcal{Y}$, (such as a class hierarchy, DAG, distance measures, or other relations); (2) when different pairs of classes should have their compatibility measured differently *i.e.* class-specific distance metrics. We discuss this latter scenario further in Section 4.

Note that we could also use distinct means and variances on the Gaussian priors for different features f_k and $f_{k'}$, and that this would be a simple and convenient way to inject domain knowledge.

3.1 Approximate Classification

Exact inference in this model is intractable since the sum necessary for calculating $Z_{\mathbf{x}}$ includes an exponential number of addends, $|\mathbf{x}|^{|\mathcal{Y}|}$. We can apply some approximate inference algorithms to this problem, such as loopy belief propogation or TRP, a particular schedule for loopy belief propogation that has better convergence properties. But since the graphical model is very densely connected with many loops, these algorithms may not work very well. Another approach tries to approximate the joint classification with a graph mincut algorithm [Blum and Chawla2001]. This method constructs a graph of data instances and the edges between them are assigned weights based on their similarities. Then the nodes are partitioned into disjoint subsets by removing the edges with the least weights.

As for our model, we also use a graph partitioning algorithm to approximate classification. But besides the similarities between instances, we want to consider their individual classification scores as well. So we construct the graph in a slightly different way. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $\mathcal{V}$ correspond to both the data points $\mathbf{x} = \mathbf{x}_L \cup \mathbf{x}_U$ and class labels $\mathcal{Y}$. There are two types of weighted edges in $\mathcal{E}$: For each pair of data node x and class node y, they are connected with a weight of $\exp\left(\sum_k \lambda_k f_k(x, y)\right)$, the score of assigning class label y to x. And the edge weight between each pair of data nodes (x_i, x_j) measures the probability that they share the same class label. In the simplified case where $y_{ij} \in \{0, 1\}$ is used, we set the weight to be $\exp\left(\sum_{k'} \lambda_{k'}(f_{k'}(x_i, x_j, 1) - f_{k'}(x_i, x_j, 0))/2\right)$.

The graph partitioning algorithm we use is similar to the standard agglomerative clustering algorithm, which iteratively merges the closest clusters. In our case, we always find the most similar node and cluster based on some criteria and then add the node to the cluster. We could also use correlation clustering [Bansal *et al.*2002], but we have found this method to usually work better in practice [McCallum and Wellner2003]. The algorithm is described below:

1: Initialize clusters C_1, C_2, ..., $C_{|\mathcal{Y}|}$ such that C_i contains one class node y_i.
2: Add the data nodes in $\mathbf{x}_L$ to the clusters that contain their corresponding class nodes.
3: $D = \mathbf{x}_U$
4: **while** D is not empty **do**
5: Find the heaviest edge between a data node $x \in D$ and a class node.
6: Let C_i be the cluster that maximizes the average edge weight between its members and x.
7: $C_i = C_i + x$, $D = D - x$.
8: **end while**

This algorithm puts all the data nodes into $|\mathcal{Y}|$ clusters, and for each cluster C_i, its members will be assigned the same class label y_i. As we can see, this is a greedy algorithm in the sense that the membership of a data node will not change once it is determined. According to step 6, we choose the cluster for a node based on its independent classification scores as well as its similarities to existing members. So the ordering to consider the nodes will affect the clustering result. As shown in step 5, we always pick the node with the highest classification score first. We have also tried merging the node and cluster with the highest average edge weight, it works worse (but not significantly).

3.2 Parameter Estimation

To learn the parameters in our model, we need to maximize the incomplete penalized log-likelihood:

$$\mathcal{L} = \log P_\Lambda(\mathbf{y}_L|\mathbf{x}_L, \mathbf{x}_U) = \log \sum_{\mathbf{y}_U} P_\Lambda(\mathbf{y}_L, \mathbf{y}_U|\mathbf{x}_L, \mathbf{x}_U).$$

Note that the $\sum_{\mathbf{y}_U}$ also includes an exponential number of addends, $|\mathbf{x}_U|^{|\mathcal{Y}|}$, and we experiment with various approximations to avoid calculating this sum.

To begin, we use only labeled data to learn the parameters by maximizing the penalized log-likelihood:

$$\begin{aligned}
\mathcal{L} &= \log P_\Lambda(\mathbf{y}_L|\mathbf{x}_L) \\
&= \log \frac{1}{Z_{\mathbf{x}_L}} \exp\left(\sum_i \sum_k \lambda_k f_k(x_i, y_i) + \sum_{i<j} \sum_{k'} \lambda_{k'} f_{k'}(x_i, x_j, y_{ij})\right) \\
&\quad - \sum_k \frac{\lambda_k^2}{2\sigma^2} - \sum_{k'} \frac{\lambda_{k'}^2}{2\sigma'^2}.
\end{aligned}$$

To apply standard optimization techniques such as limited-memory BFGS, we need to provide the values of the objective function and its gradient:

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \left(\sum_{i=1}^{l} f_k(x_i, y_i) - \sum_{i=1}^{l} \sum_{y' \in \mathcal{Y}} P_\Lambda(y'|\mathbf{x}_L) f_k(x_i, y')\right) - \frac{\lambda_k}{\sigma^2},$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{k'}} = \left(\sum_{i<j} f_{k'}(x_i, x_j, y_{ij}) - \sum_{i<j} \sum_{y'_i \in \mathcal{Y}} \sum_{y'_j \in \mathcal{Y}} P_\Lambda(y'_i, y'_j|\mathbf{x}_L) f_{k'}(x_i, x_j, y'_{ij})\right) - \frac{\lambda_{k'}}{\sigma'^2}.$$

This involves calculating the joint probability $P_\Lambda(\mathbf{y}_L|\mathbf{x}_L)$ and the marginals $P_\Lambda(y_i|\mathbf{x}_L)$ and $P_\Lambda(y_i, y_j|\mathbf{x}_L)$. Since there is no efficient algorithm for exact inference, we have tried various approximations to obtain these values.

First, we use *voted perceptron* for optimization, which only requires the gradient. In order to avoid calculating the marginals in the gradient, we approximate the feature expectation $\sum_{y' \in \mathcal{Y}} P_\Lambda(y'|\mathbf{x}_L) f_k(x_i, y')$ with a single value $f(x_i, y^*)$, where $y^* = arg\,max_{y' \in \mathcal{Y}} P_\Lambda(y'|\mathbf{x}_L)$. And we find an approximate y^* with the graph partitioning algorithm described above, treating all the training instances as unlabeled data. Similarly, $\sum_{y'_i \in \mathcal{Y}} \sum_{y'_j \in \mathcal{Y}} P_\Lambda(y'_i, y'_j|\mathbf{x}_L) f_{k'}(x_i, x_j, y'_{ij})$ is replaced with $f(x_i, x_j, y^*_{ij})$.

The second solution is to calculate the joint and marginal probabilities using approximate inference algorithms, and then apply limited-memory BFGS to train the parameters. We choose TRP to perform the inference.

Another approach is to approximate $P_\Lambda(\mathbf{y}_L|\mathbf{x}_L)$ with a simpler function form $P'_\Lambda(\mathbf{y}_L|\mathbf{x}_L)$, for which it is easy to perform exact inference. Therefore, we could calculate the accurate values for the new log-likelihood $\mathcal{L}'$ we want to maximize and its gradient. For instance, *pseudo-likelihood training* approximates the joint probability of a set of random variables with the product of the conditional probabilities of each variable given the others. In our case, we define $P'_\Lambda(\mathbf{y}_L|\mathbf{x}_L)$ as

$$\begin{aligned} P'_\Lambda(\mathbf{y}_L|\mathbf{x}_L) &= \prod_n P_\Lambda(y_n|\mathbf{x}_L, \mathbf{y}_{-n}) \\ &= \prod_n \frac{1}{Z_{\mathbf{x}_L, \mathbf{y}_{-n}}} \exp\left(\sum_i \sum_k \lambda_k f_k(x_i, y_i) + \sum_{i<j} \sum_{k'} \lambda_{k'} f_{k'}(x_i, x_j, y_{ij})\right), \end{aligned}$$

where $\mathbf{y}_{-n}$ indicates all the y_i's in $\mathbf{y}_L$ except y_n and $Z_{\mathbf{x}_L, \mathbf{y}_{-n}}$ is the normalization factor.

We also explore a strategy we call *local-joint training* to construct the approximating function. Although it is difficult to calculate the joint probability when there is a large number of instances, it is not a problem for a small set of variables. So for each pair of nodes (x_i, x_j), we calculate the probability $P_\Lambda(y_i, y_j|x_i, x_j)$ by normalizing the potentials locally and then approximate the global probability $P_\Lambda(\mathbf{y}_L|\mathbf{x}_L)$ as their product:

$$\begin{aligned} P'_\Lambda(\mathbf{y}_L|\mathbf{x}_L) &= \prod_{i<j} P_\Lambda(y_i, y_j|x_i, x_j) \\ &= \prod_{i<j} \frac{1}{Z_{x_i, x_j}} \exp\left(\sum_k \lambda_k (f_k(x_i, y_i) + f_k(x_j, y_j)) + \sum_{k'} \lambda_{k'} f_{k'}(x_i, x_j, y_{ij})\right). \end{aligned}$$

To better understand the effect of approximating the joint probability with the product of all the pair-wise probabilities, consider the simple case where there are only four variables $\{a, b, c, d\}$. Then we have

$$\begin{aligned} P(a, b, c, d) &= P(a, b) P(c, d|a, b) \\ &= P(a, b) P(c, d), \end{aligned}$$

if we assume the independence between a, b and c, d. Similarly,

$$P(a, b, c, d) = P(a, c)P(b, d),$$

$$P(a, b, c, d) = P(a, d)P(b, c).$$

So given these independence assumptions, we approximate the joint as follows:

$$\begin{aligned} P'(a, b, c, d) &= (P(a, b)P(c, d))\,(P(a, c)P(b, d))\,(P(a, d)P(b, c)) \\ &= P(a, b, c, d)^3. \end{aligned}$$

In the general case, this approximation roughly raises the joint probability to some power and thus multiplies the log-likelihood by a constant. Although the values for P and P' might be very different, the parameters that maximize them should be similar, assuming independences. So we expect it to be a reasonable approximation. When the independence assumption is violated, we might double-count the evidence just as Naive Bayes classification.

For both pseudo-likelihood training and local-joint training, we can easily evaluate the approximating log-likelihood functions and gradients, which allow us to learn the parameters efficiently using limited-memory BFGS.

It is easy to extend the local-joint training method to incorporate unlabeled instances. We still approximate $P_\Lambda(\mathbf{y}_L|\mathbf{x}_L, \mathbf{x}_U)$ with the product of pair-wise probabilities. However, the half labeled pairs, i.e., pairs with only one instance labeled, also contribute to this product. And we calculate their local probabilities as $P_\Lambda(y_l|x_l, x_u) = \sum_{y\in\mathcal{Y}} P_\Lambda(y_l, y|x_l, x_u)$. We could also consider triples with one unlabeled example: $P_\Lambda(y_l1, y_l2|x_l1, x_l2, x_u) = \sum_{y\in\mathcal{Y}} P_\Lambda(y_l1, y_l2, y|x_l1, x_l2, x_u)$

4 Experiment Results

We present experimental results for the document classification task using the 20 newsgroup dataset. The preprocessing on the documents includes downcasing and removing headers and stopwords. The features $f_k(x)$ are term frequencies and $f_{k'}(x_i, x_j) = \min\{f_k(x_i), f_k(x_j)\}$ are the shared term frequencies.

The Gaussian means for the priors on λ_k and $\lambda_{k'}$ are set to 0 and *idf* respectively. Idf is the inverse of the number of documents that a term occurs in. We use it as the means for $\lambda_{k'}$ because in the information retrieval area, it is often combined with *tf*, term frequency, to measure document similarities. Since there is only a small set of labeled instances, we impose relatively tight variances on the parameters to reduce overfitting. Currently we have $\sigma^2 = \sigma'^2 = 0.001$, and the performance with larger variances is worse.

In large data sets, calculating an affinity measure between all $|\mathbf{x}|^2$ pairs of instances is prohibitively expensive. The "canopies" method [McCallum *et al.*2000] for efficiently pruning the set of pairs to be measured has been used with success

in several related situations, including [Pasula *et al.*2002]. It enables us to use a large number of documents for training and classification. We use this method in local-joint training with unlabeled instances, and the similarity measure is tf-idf. In our experiments, we keep approximately 5000 pairs for about 2000 examples. We do not use this method in testing because it significantly hurts the performance.

For all the training models discussed in the previous section, we conduct experiments using the simplified features $f_{k'}(x_i, x_j, y_{ij})$, where we only care if the two classes are equal or not. But it is easy to extend this model to consider the individual values of y_i and y_j with additional parameters associated with them. In this case, different classes could have different distance metrics. Currently we are experimenting with this more complicated model.

The results reported here are all based on the graph partitioning algorithm for *test*-time classification. Although we also experimented with TRP to perform inference used in classification, it does not work as well as this algorithm.

A complete comparison of various *training* methods on the binary problem PC vs. MAC is presented in Table 1. Furthermore, we compare their performances with three traditional models that classify each instance independently: Naive Bayes (NB), maximum entropy (ME) and support vector machines (SVM). We use 5 different amounts of labeled instances (shown in the first column of the table), and perform 5 trials for each of them. After the labeled instances are randomly sampled, we use the remaining instances as unlabeld data. The average classification accuracies (and standard deviations in parentheses) are listed in the table.

As we can see, the results are very different for the four training methods with labeled data only. Voted perceptron (VP), TRP and pseudo-likelihood (PL) didn't improve over the independent classifiers, while local-joint training (LJ) has significantly better performance. To show the impact of learning the distance metric, we conduct another experiment (IDF), which only learns λ_k but fixes $\lambda_{k'}$ to idf. The performance of this fixed distance metric is less than the metric learned by local-joint training. We also use unlabeled data in local-joint training (LJU), and while the performance is generally better, it is not statistically significantly so.

Test-time classification in our model is approximated by graph partitioning. To evaluate the loss in performance due to this approximation, we compare the clustering result produced by the graph partitioning algorithm and the correct clustering where each data node is placed in the same cluster as its class node. The criterion is the difference between average inter- and intra-cluster similarities. And it shows that the correct clustering is actually worse. This means that the graph partitioning algorithm misses the perfect classification because it is not the most likely configuration, not because the algorithm is a bad approximation.

	NB	ME	SVM	VP	TRP	PL	IDF	LJ	LJU
20	64.42(6)	64.31(7)	65.40(6)	58.46(1)	68.45(4)	66.24(5)	70.88(7)	76.92(6)	75.20(11)
40	64.90(7)	69.78(6)	69.16(4)	57.65(4)	66.78(6)	69.54(9)	72.15(4)	79.38(5)	76.81(8)
60	69.87(4)	73.40(2)	72.57(2)	68.20(14)	70.25(12)	73.50(13)	80.79(3)	86.63(2)	85.57(3)
80	73.44(4)	76.84(2)	74.83(2)	71.92(14)	71.16(13)	78.30(14)	82.41(2)	87.13(2)	87.56(2)
100	73.77(5)	77.09(2)	75.36(2)	70.23(14)	82.18(4)	77.67(8)	82.20(3)	86.92(1)	87.84(1)

Table 1: Classification Accuracies (%)

5 Related Work

Semi-supervised learning has become an important research topic in the machine learning community. A review of various techniques in this field is given in [Seeger2000]. Some recent work has focused on exploring the structure described by the unlabeled data to change the classification boundaries. A common assumption of these methods is to give the same classification to close data points. Szummer and Jaakkola [Szummer and Jaakkola2001] propose a Markov random walk representation over unlabeled examples, resisting rapid change of class labels in high-density regions. They use fixed affinity function for density calculation. Chapelle et al. [Chapelle *et al.*2002] introduce a framework to design "cluster kernels" so that the distance between two points is smaller when they are in the same cluster. Blum and Chawla [Blum and Chawla2001] use a graph partitioning algorithm for classification, minimizing the number of close pairs that are assigned different class labels. They discuss several ways to define similar examples, but they do not learn the distance metric.

The most related work to this paper is [Zhu *et al.*2002] and [Zhu *et al.*2003a]. There are several differences in our approach, but it is not yet clear if our model provides any strong advantages. Their latter paper uses Gaussian random fields instead of discrete random fields, the continuous state space providing easier inference since there is an efficient closed form solution to the most likely configuration. Similar to the parameters $\lambda_{k'}$ in our model, they also have different weights associated with different features to measure the similarities between instances. However, these weights do not depend on class labels. We think it is reasonable to have class-dependent distance metrics. One can imagine that some features are meaningful to a particular class but not to others. So we need to consider the (hypothesized) labels to judge if two instances are close or not. Another difference between our approach and Zhu's is parameter estimation. Instead of maximizing the conditional probability of labels given the instances, they try to minimize the average label entropy for the unlabeled data. Again, this can be done in an efficient way. In the following work [Zhu *et al.*2003b], they also try to maximize the data likelihood. In their basic model, the classification is only based on the data manifold (the potentials between data points—there are no potentials to the class prototypes). It can be further extended to incorporate an external classifier that also produces labels for unlabeled

data. However, the relative importances of these two kinds of information are determined arbitrarily. On the other hand, we capture both the data structure and their individual features in one model and learn the two sets of parameters simultaneously. So they can automatically adjust their values to appropriately balance with each other.

6 Conclusion

This paper presents conditionally-trained Markov random fields with tied parameters, which affect classification of individual instances based on their own features and also capture the similarities between pairs of instances. Unlabeled data can be used in both training and classification. We use a graph partitioning algorithm to approximate inference and describe several training procedures.

Currently, we are exploring several directions to improve the performance. For example, the similarity features $f_{k'}(x_i, x_j, y_{ij})$ are now the shared term frequencies between two instances. We are experimenting with more options. Furthermore, we are using the simplified features that only care if the two classes are equal or not. And we will extend them to include the individual class values. As the experiments show, the local-joint training method has the best performance. The idea of approximating the joint probability with the product of locally normalized probabilities can be applied to larger pieces too. So we will continue working on this approach and try to further improve parameter estimation.

References

[Bansal *et al.*2002] N. Bansal, A. Blum, and S. Chawla. Correlation clustering, 2002.

[Blum and Chawla2001] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conf. on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA, 2001.

[Blum and Mitchell1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

[Byrd *et al.*1994] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.

[Chapelle *et al.*2002] O. Chapelle, J. Weston, and B. Schoelkopf. Cluster kernels for semi-supervised learning. In *Neural Information Processing Systems*, 2002.

[Joachims1999] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[Malouf] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation.

[McCallum and Minka1999] Andrew McCallum and Tom Minka. Semi-supervised learning using distance metrics learned via dirichlet trees (unpublished work), 1999.

[McCallum and Wellner2003] Andrew McCallum and Ben Wellner. Object consolidation by graph partitioning with a conditionally-trained distance metric, 2003.

[McCallum *et al.*2000] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining*, pages 169–178, 2000.

[McCallum2003] Andrew McCallum. Efficiently inducing features of conditional random fields. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*, 2003.

[Nigam *et al.*2000] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[Nigam2001] Kamal Nigam. *Using Unlabeled Data to Improve Text Classification.* PhD thesis, Pittsburgh, US, 2001.

[Pasula *et al.*2002] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching, 2002.

[Seeger2000] M. Seeger. Learning with labeled and unlabeled data, 2000.

[Szummer and Jaakkola2001] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[Zhu *et al.*2002] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation, 2002.

[Zhu *et al.*2003a] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML-03*, 2003.

[Zhu *et al.*2003b] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes., 2003.