Semi-Supervised Learning Using Randomized Mincuts

Avrim Blum AVRIM@CS.CMU.EDU
John Lafferty LAFFERTY@CS.CMU.EDU
Mugizi Robert Rwebangira RWEBA@CS.CMU.EDU
Rajashekar Reddy RREDDY@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213-3891

Abstract

In many application domains there is a large amount of unlabeled data but only a very limited amount of labeled training data. One general approach that has been explored for utilizing this unlabeled data is to construct a graph on all the data points based on distance relationships among examples, and then to use the known labels to perform some type of graph partitioning. One natural partitioning to use is the minimum cut that agrees with the labeled data (Blum & Chawla, 2001), which can be thought of as giving the most probable label assignment if one views labels as generated according to a Markov Random Field on the graph. Zhu et al. (2003) propose a cut based on a relaxation of this field, and Joachims (2003) gives an algorithm based on finding an approximate min-ratio cut.

In this paper, we extend the mincut approach by adding randomness to the graph structure. The resulting algorithm addresses several shortcomings of the basic mincut approach, and can be given theoretical justification from both a Markov random field perspective and from sample complexity considerations. In cases where the graph does not have small cuts for a given classification problem, randomization may not help. However, our experiments on several datasets show that when the structure of the graph supports small cuts, this can result in highly accurate classifiers with good accuracy/coverage tradeoffs. In addition, we are able to achieve good performance with a very simple graph-construction procedure.

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004.

1. Introduction

Learning algorithms often face a shortage of labeled training data. In many situations we would like a learning algorithm to do well with only a few labeled training examples. Fortunately, in many cases large numbers of *unlabeled* examples are often available. As a result, there has been a good deal of work in recent years on how unlabeled data can be usefully employed in order to produce better predictions.

If one believes that "similar examples ought to have similar labels," then a natural approach to using unlabeled data is to combine nearest-neighbor prediction—predict a given test example based on its nearest labeled example—with some sort of self-consistency criteria, e.g., that similar *unlabeled* examples should, in general, be given the same classification. The graph mincut approach of Blum and Chawla (2001) is a natural way of realizing this intuition in a transductive learning algorithm. Specifically, the idea of this algorithm is to build a graph on all the data (labeled and unlabeled) with edges between examples that are sufficiently similar, and then to partition the graph into a positive set and a negative set in a way that (a) agrees with the labeled data, and (b) cuts as few edges as possible. (An edge is "cut" if its endpoints are on different sides of the partition.) Zhu et al. (2003) propose an alternative "soft-partitioning" method that assigns fractional classifications to the unlabeled data so as to minimize a quadratic cost function, and Joachims (2003) gives a method based on spectral partitioning, that produces an approximate minimum ratio cut in the graph.

The graph mincut approach has a number of attractive properties. It can be found in polynomial time using network flow; it can be viewed as giving the most probable configuration of labels in the associated Markov Random Field (see Section 2); and, it can also be motivated from sample-complexity considerations, as we discuss further in Section 4. However, it also suffers from several drawbacks. First, from a practical perspective, a graph may have many minimum cuts and the mincut algorithm produces just one, typically the "leftmost" one using standard network flow algorithms. For instance, a line of n vertices between two labeled points s and t has $n-1$ cuts of size 1, and the leftmost cut will be especially unbalanced. Second, from an MRF perspective, the mincut approach produces the most probable joint labeling (the MAP hypothesis), but we really would rather label nodes based on their *per-node* probabilities (the Bayes-optimal prediction). Finally, from a sample-complexity perspective, if we could average over many small cuts, we could improve our confidence via PAC-Bayes style arguments.

In this paper, we propose a simple method for addressing a number of these drawbacks using randomization. Specifically, we repeatedly add artificial random noise to the edge weights,[1] solve for the minimum cut in the resulting graphs, and finally output a fractional label for each example corresponding to the fraction of the time it was on one side or the other in this experiment. This is not the same as sampling directly from the MRF distribution, and is also not the same as picking truly random minimum cuts in the original graph, but those problems appear to be much more difficult computationally on general graphs (see Section 2).

A nice property of the randomized mincut approach is that it easily leads to a measure of confidence on the predictions; this is lacking in the deterministic mincut algorithm, which produces a single partition of the data. The confidences allow us to compute accuracy-coverage curves, and we see that on many datasets the randomized mincut algorithm exhibits good accuracy-coverage performance.

We also discuss design criteria for constructing graphs likely to be amenable to our algorithm. Note that some graphs simply do not have small cuts that match any low-error solution; in such graphs, the mincut approach will likely fail even with randomization. However, constructing the graph in a way that is very conservative in producing edges can alleviate many of these problems. For instance, we find that a very simple minimum spanning tree graph does quite well across a range of datasets.

PAC-Bayes sample complexity analysis (McAllester, 2003) suggests that when the graph has many small cuts consistent with the labeling, randomization should improve generalization performance. This analysis is supported in experiments with data sets such as handwritten digit recognition, where the algorithm results in a highly accurate classifier. In cases where the graph does not have small cuts for a given classification problem, the theory also suggests, and our experiments confirm, that randomization may not help. We present experiments on several different data sets that indicate both the strengths and weaknesses of randomized mincuts, and also how this approach compares with the semi-supervised learning schemes of Zhu et al. (2003) and Joachims (2003). For the case of MST-graphs, in which the Markov random field probabilities *can* be efficiently calculated exactly, we compare to that method as well.

In the following sections we give some background on Markov random fields, describe our algorithm more precisely as well as our design criteria for graph construction, provide sample-complexity analysis motivating some of our design decisions, and finally give our experimental results.

[1] We add noise only to existing edges and do not introduce new edges in this procedure.

2. Background and Motivation

Markov random field models originated in statistical physics, and have been extensively used in image processing. In the context of machine learning, what we can do is create a graph with a node for each example, and with edges between examples that are similar to each other. A natural energy function to consider is

$$E(f) = \frac{1}{2}\sum_{i,j} w_{ij}|f(i)-f(j)| = \frac{1}{4}\sum_{i,j} w_{ij}(f(i)-f(j))^2$$

where $f(i) \in \{-1,+1\}$ are binary labels and w_{ij} is the weight on edge (i,j), which is a measure of the similarity between the examples. To assign a probability distribution to labelings of the graph, we form a random field

$$p_\beta(f) = \frac{1}{Z}\exp\left(-\beta E(f)\right)$$

where the partition function Z normalizes over all labelings. Solving for the lowest energy configuration in this Markov random field will produce a partition of the entire (labeled and unlabeled) dataset that maximally optimizes self-consistency, subject to the constraint that the configuration must agree with the labeled data.

As noticed over a decade ago in the vision literature (Greig et al., 1989), this is equivalent to solving for a minimum cut in the graph, which can be done via a number of standard algorithms. Blum and Chawla (2001) introduced this approach to machine learning, carried out experiments on several data sets, and explored generative models that support this notion of self-consistency.

The minimum cut corresponds, in essence, to the MAP hypothesis in this MRF model. To produce Bayes-optimal predictions, however, we would like instead to sample directly from the MRF distribution. Unfortunately, that problem appears to be much more difficult computationally on general graphs. Specifically, while random labelings can be efficiently sampled *before* any labels are observed, using the well-known Jerrum-Sinclair procedure for the Ising model (Jerrum & Sinclair, 1993), after we observe the labels on some examples, there is no known efficient algorithm for sampling from the conditional probability distribution; see (Dyer et al., 2000) for a discussion of related combinatorial problems. This leads to two approaches: (1) try to approximate this procedure by adding random noise into the graph, or (2) make sure the graph is a tree, for which the MRF probabilities can be calculated exactly using dynamic programming. In this paper, we consider both.

3. Randomized Mincuts

The randomized mincut procedure we consider is the following. Given a graph G constructed from the dataset, we produce a collection of cuts by repeatedly adding random noise to the edge weights and then solving for the minimum cut in the perturbed graph. In addition, now that we have a collection of cuts, we remove those that are highly unbalanced. This step is justified using a simple ϵ-cover argument (see Section 4), and in our experiments, any cut with less than 5% of the vertices on one side is considered unbalanced.[2] Finally, we predict based on a majority vote over the remaining cuts in our sample, outputting a confidence based on the margin of the vote. We call this algorithm "Randomized mincut with sanity check" since we use randomization to produce a distribution over cuts, and then throw out the ones that are obviously far from the true target function.

In many cases this randomization can overcome some of the limitations of the plain mincut algorithm. Consider a graph which simply consists of a line, with a positively labeled node at one end and a negatively labeled node at the other end with the rest being unlabeled. Plain mincut may choose from any of a number of cuts, and in fact the cut produced by running network flow will be either the leftmost or rightmost one depending on how it is implemented. Our algorithm will take a vote among all the mincuts and thus we will end up using the middle of the line as a decision boundary, with confidence that increases linearly out to the endpoints.

It is interesting to consider for which graphs our algorithm produces a true uniform distribution over minimum cuts and for which it does not. To think about this, it is helpful to imagine we collapse all labeled positive examples into a single node s and we collapse all labeled negative examples into a single node t. We can now make a few simple observations. First, a class of graphs for which our algorithm *does* produce a true uniform distribution are those for which all the s-t minimum cuts are disjoint, such as the case of the line above. Furthermore, if the graph can be decomposed into several such graphs running in parallel between s and t ("generalized theta graphs" (Brown et al., 2001)), then we get a true uniform distribution as well. That is because any minimum s-t cut must look like a tuple of minimum cuts, one from each graph, and the randomized mincut algorithm will end up choosing at random from each one.

On the other hand, if the graph has the property that some minimum cuts overlap with many others and some do not, then the distribution may not be uniform. For example, Figure 1 shows a case in which the randomized procedure gives a much higher weight to one of the cuts than it should ($\geq 1/6$ rather than $1/n$).

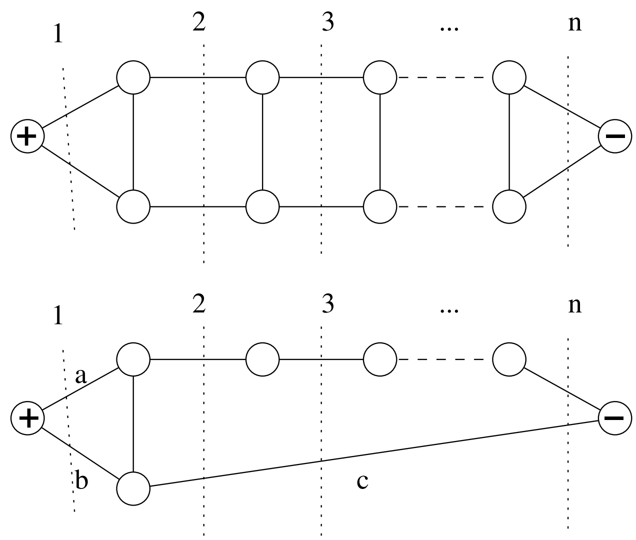

Figure 1. In the top graph, each of the n cuts of size 2 has probability $1/n$ of being minimum when random noise is added to the edge lengths. However, in the bottom graph this is not the case. In particular, there is a constant probability that the noise added to edge c exceeds that added to a and b combined (if a, b, c are picked at random from $[0, 1]$ then $\Pr(c > a + b) = 1/6$). This results in the algorithm producing cut $\{a, b\}$ no matter what is added to the other edges. Thus, $\{a, b\}$ has a much higher than $1/n$ probability of being produced.

4. Sample complexity analysis

4.1. The basic mincut approach

From a sample-complexity perspective, we have a transductive learning problem, or (roughly) equivalently, a problem of learning from a known distribution. Let us model the learning scenario as one in which first the graph G is constructed from data without any labels (as is done in our experiments) and then a few examples at random are labeled. Our goal is to perform well on the rest of the points. This

[2] With only a small set of labeled data, one cannot in general be confident that the true class probabilities are close to the observed fractions in the training data, but one *can* be confident that they are not extremely biased one way or the other.

means we can view our setting as a standard PAC-learning problem over the uniform distribution on the vertices of the graph. We can now think of the mincut algorithm as motivated by standard Occam bounds: if we describe a hypothesis by listing the edges cut using $O(\log n)$ bits each, then a cut of size k can be described in $O(k \log n)$ bits.[3] This means we need only $O(k \log n)$ labeled examples to be confident in a consistent cut of k edges (ignoring dependence on ϵ and δ).

In fact, we can push this bound further: Kleinberg (2000), studying the problem of detecting failures in networks, shows that the VC-dimension of the class of cuts of size k is $O(k)$. Thus, only $O(k)$ labeled examples are needed to be confident in a consistent cut of k edges. Kleinberg et al. (2004) reduce this further to $O(k/\lambda)$ where λ is the size of the *global* minimum cut in the graph (the minimum number of edges that must be removed in order to separate the graph into two nonempty pieces, without the requirement that the labeled data be partitioned correctly).

One implication of this analysis is that if we imagine data is being labeled for us one at a time, we can plot the size of the minimum cut found (which can only increase as we see more labeled data) and compare it to the global minimum cut in the graph. If this ratio grows slowly with the number of labeled examples, then we can be confident in the mincut predictions.

4.2. Randomized mincut with "sanity check"

As pointed out by Joachims (2003), minimum cuts can at times be very unbalanced. From a sample-complexity perspective we can interpret this as a situation in which the cut produced is simply not small enough for the above bounds to apply given the number of labeled examples available. From this point of view, we can think of our mincut extension as being motivated by two lines of research on ways of achieving rules of higher confidence. The first of these are PAC-Bayes bounds (McAllester, 2003; Langford & Shawe-Taylor, 2002). The idea here is that even if no single consistent hypothesis is small enough to inspire confidence, if many of them are "pretty small" (that is, they together have a large prior if we convert our description language into a probability distribution) then we can get a better confidence bound by randomizing over them. Even though our algorithm does not necessarily produce a true uniform distribution over all consistent minimum cuts, our goal is simply to produce as wide a distribution as we can to take as much advantage of this as possible. Results of Freund et al. (2003) show furthermore that if we weight the rules appropriately, then we can expect a lower error rate on examples for which their vote is highly biased. Again, while our procedure is at best only an approximation to their weighting scheme, this motivates our use of the bias of the vote in producing accuracy/coverage curves.

The second line of research motivating aspects of our algorithm is work on bounds based on ϵ-cover size, e.g., (Benedek & Itai, 1991). The idea here is that suppose we have a known distribution D and we identify some hypothesis h that has many similar hypotheses in our class with respect to D. Then if h has a high error rate over a labeled sample, it is likely that *all* of these similar hypotheses have a high true error rate, *even if some happen to be consistent with the labeled sample*. In our case, two specific hypotheses we can easily identify of this form are the "all positive" and "all negative" rules. If our labeled sample is even reasonably close to balanced — e.g., 3 positive examples out of 10 — then we can confidently conclude that these two hypotheses have a high error rate, and throw out *all highly unbalanced cuts*, even if they happen to be consistent with the labeled data. For instance, the cut that simply separates the three positive examples from the rest of the graph is consistent with the data, but can be ruled out by this method.

This analysis then motivates the second part of our algorithm in which we discard all highly unbalanced cuts found before taking majority vote. The important issue here is that we can confidently do this even if we have only a very small labeled sample. Of course, it is possible that by doing so, our algorithm is never able to find a cut it is willing to use. In that case our algorithm halts with failure, concluding that the dataset is not one that is a good fit to the biases of our algorithm. In that case, perhaps a different approach such as the methods of Joachims (2003) or Zhu et al. (2003) or a different graph construction procedure is needed.

5. Graph design criteria

For a given distance metric, there are a number of ways of constructing a graph. In this section, we briefly discuss design principles for producing graphs amenable to the graph mincut algorithm. These then motivate the graph construction methods we use in our experiments.

First of all, the graph produced should either be connected or at least have the property that a small number of connected components cover nearly all the examples. If t components are needed to cover a $1 - \epsilon$ fraction of the points, then clearly any graph-based method will need t labeled examples to do well.[4]

[3]This also assumes the graph is connected — otherwise, a hypothesis is not uniquely described by the edges cut.

[4]This is perhaps an obvious criterion but it is important to keep in mind. For instance, if examples are uniform random points in the 1-dimensional interval $[0, 1]$, and we connect each point to its nearest k neighbors, then it is not hard to see that if k is fixed and

Secondly, for a mincut-based approach we would like a graph that at least has some small balanced cuts. While these may or may not correspond to cuts consistent with the labeled data, we at least do not want to be dead in the water at the start. This suggests conservative methods that only produce edges between very similar examples.

Based on these criteria, we chose the following two graph construction methods for our experiments.

MST: Here we simply construct a minimum spanning tree on the entire dataset. This graph is connected, sparse, and furthermore has the appealing property that it has no free parameters to adjust. In addition, because the exact MRF per-node probabilities *can* be exactly calculated on a tree, it allows us to compare our randomized mincut method with the exact MRF calculation.

δ-MST: For this method, we connect two points with an edge if they are within a radius δ of each other. We then view the components produced as supernodes and connect them via an MST. Blum and Chawla (2001) used δ such that the largest component had half the vertices (but did not do the second MST stage). To produce a more sparse graph, we choose δ so that the largest component has $1/4$ of the vertices.

Another natural method to consider would be a k-NN graph, say connected up via a minimum spanning tree as in δ-MST. However, experimentally, we find that on many of our datasets this produces graphs where the mincut algorithm is simply not able to find even moderately balanced cuts (so it ends up rejecting them all in its internal "sanity-check" procedure). Thus, even with a small labeled dataset, the mincut-based procedure would tell us to choose an alternative graph-creation method.

6. Experimental Analysis

We compare the randomized mincut algorithm on a number of datasets with the following approaches:

PLAIN MINCUT: Mincut without randomization.

GAUSSIAN FIELDS: The algorithm of Zhu et al. (2003).

SGT: The spectral algorithm of Joachims (2003).

EXACT: The exact Bayes-optimal prediction in the MRF model, which can be computed efficiently in trees (so we only run it on the MST graphs).

Below we present results on handwritten digits, portions of the 20 newsgroups text collection, and various UCI datasets.

6.1. Handwritten Digits

We evaluated randomized mincut on a dataset of handwritten digits originally from the Cedar Buffalo binary digits database (Hull, 1994). Each digit is represented by a 16 X 16 grid with pixel values ranging from 0 to 255. Hence, each image is represented by a 256-dimensional vector.

For each size of the labeled set, we perform 10 trials, randomly sampling the labeled points from the entire dataset. If any class is not represented in the labeled set, we redo the sample.

One vs. Two: We consider the problem of classifying digits, "1" vs. "2" with 1128 images. Results are reported in Figure 2. We find that randomization substantially helps the mincut procedure when the number of labeled examples is small, and that randomized mincut and the Gaussian field method perform very similarly. The SGT method does not perform very well on this dataset for these graph-construction procedures. (This is perhaps an unfair comparison, because our graph-construction procedures are based on the needs of the mincut algorithm, which may be different than the design criteria one would use for graphs for SGT.)

Odd vs. Even: Here we classify 4000 digits into Odd vs. Even. Results are given in Figure 3. On the MST graph, we find that Randomized mincut, Gaussian fields, and the exact MRF calculation all perform well (and nearly identically). Again, randomization substantially helps the mincut procedure when the number of labeled examples is small. On the δ-MST graph, however, the mincut-based procedures perform substantially worse, and here Gaussian fields and SGT are the top performers.

In both datasets, the randomized mincut algorithm tracks the exact MRF Bayes-optimal predictions extremely closely. Perhaps what is most surprising, however, is how good performance is on the simple MST graph. On the Odd vs. Even problem, for instance, Zhu et al. (2003) report for their graphs an accuracy of 73% at 22 labeled examples, 77% at 32 labeled examples, and do not exceed 80% until 62 labeled examples.

6.2. 20 newsgroups

We performed experiments on classifying text data from the 20-newsgroup datasets, specifically PC versus MAC (see Figure 4). Here we find that on the MST graph, all the methods perform similarly, with SGT edging out the others

the number of points goes to infinity, the number of components will go to infinity as well. That is because a local configuration, such as two adjacent tight clumps of k points each, can cause such a graph to disconnect.

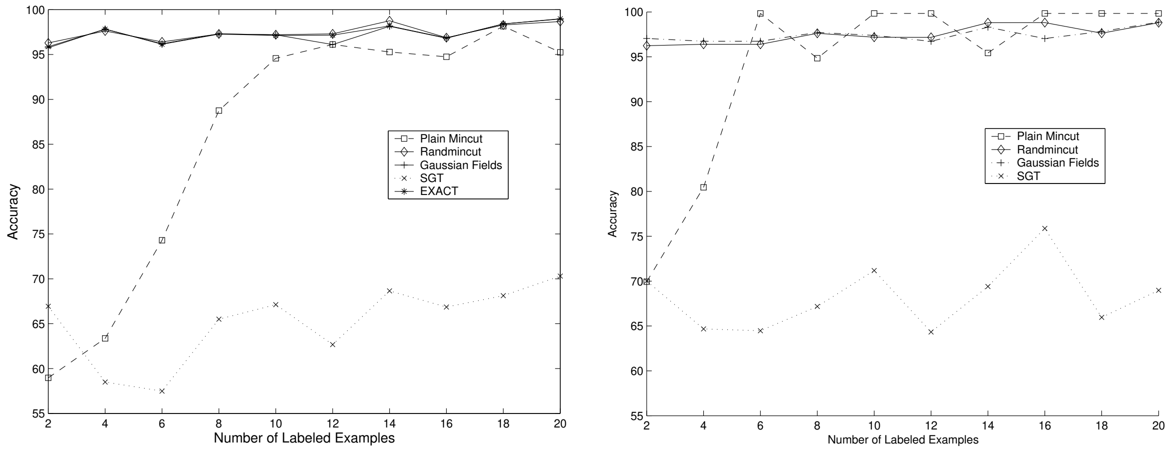

Figure 2. "1" vs "2" on the digits dataset with the MST graph (left) and $\delta_{\frac{1}{4}}$ graph (right).

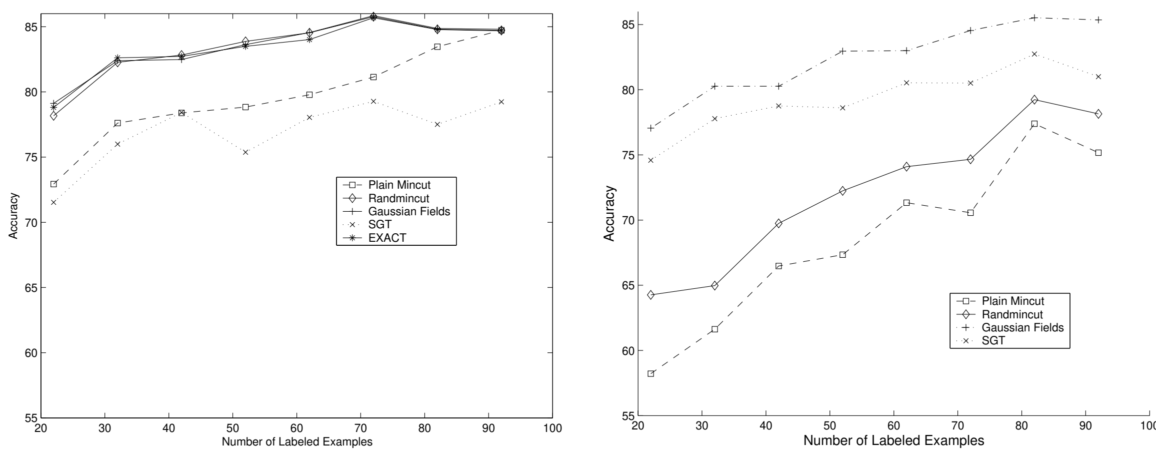

Figure 3. Odd vs. Even on the digits dataset with the MST graph (left) and $\delta_{\frac{1}{4}}$ graph (right).

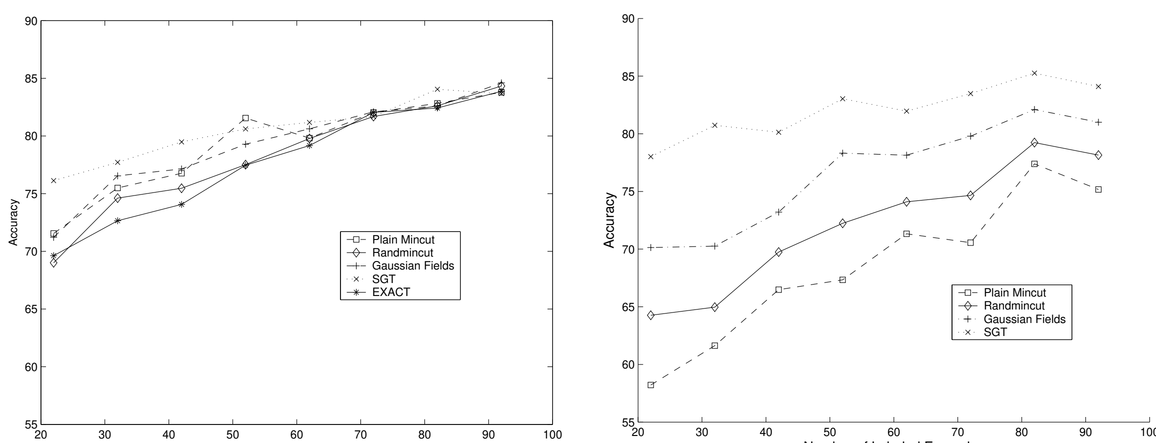

Figure 4. PC vs MAC on the 20 newsgroup dataset with MST graph (left) and $\delta_{\frac{1}{4}}$ graph (right).

on the smaller labeled set sizes. On the δ-MST graph, SGT performs best across the range of labeled set sizes. On this dataset, randomization has much less of an effect on the mincut algorithm.

6.3. UCI Datasets

We conducted experiments on various UC Irvine datasets; see Table 1. Here we find all the algorithm perform comparably.

6.4. Accuracy Coverage Tradeoff

As mentioned earlier, one motivation for adding randomness to the mincut procedure is that we can use it to set a confidence level based on the number of cuts that agree on the classification of a particular example. To see how confidence affects prediction accuracy, we sorted the examples by confidence and plotted the cumulative accuracy. Figure 5 shows accuracy-coverage tradeoffs for Odd-vs-Even and PC-vs-MAC. We see an especially smooth tradeoff for the digits data, and we observe on both datasets that the algorithm obtains a substantially lower error rate on examples on which it has high confidence.

6.5. Examining the graphs

To get a feel for why the performance of the algorithms is so good on the MST graph for the digits dataset, we examined the following question. Suppose for some $i \in \{0, \ldots, 9\}$ you remove all vertices that are not digit i. What is the size of the largest component in the graph remaining? This gives a sense of how well one could possibly hope to do on the MST graph if one had only one labeled example of each digit. The result is shown in Figure 6. Interestingly, we see that most digits have a substantial fraction of their examples in a single component. This partly explains the good performance of the various algorithms on the MST graph.

7. Conclusion

We introduce a new semi-supervised learning algorithm based on adding artificial random noise to the edge weights of a graph and averaging over the minimum cuts produced. Our algorithm addresses several shortcomings of the basic mincut approach, improving performance especially when the number of labeled examples is small, as well as providing a confidence score for accuracy-coverage curves. We provide theoretical motivation for our approach from a sample complexity and Markov Random Field perspective.

We present experimental results supporting the applicability of the randomized mincut algorithm to various settings. In the experiments done so far, our method allows mincut

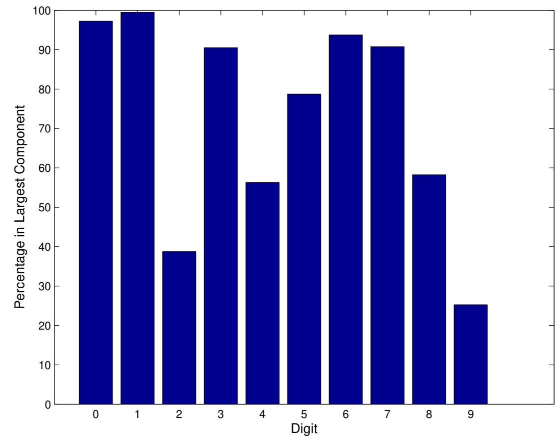

Figure 6. MST graph for Odd vs. Even: percentage of digit i that is in the largest component if all other digits were deleted from the graph.

to approach, though it tends not to beat, the Gaussian field method of Zhu et al. (2003). However, mincuts have the nice property that we can apply sample-complexity analysis, and furthermore the algorithm can often easily tell when it is or is not appropriate for a dataset based on how large and how unbalanced the cuts happen to be.

The exact MRF per-node likelihoods can be computed efficiently on trees. It would be especially interesting if this can be extended to larger classes of graphs.

Acknowledgements

This work was supported in part by NSF grants CCR-0105488, NSF-ITR CCR-0122581, and NSF-ITR IIS-0312814.

References

Benedek, G., & Itai, A. (1991). Learnability with respect to a fixed distribution. *Theoretical Computer Science*, *86*, 377–389.

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. *Proceedings of the 18th International Conference on Machine Learning* (pp. 19–26). Morgan Kaufmann.

Brown, J. I., Hickman, C. A., Sokal, A. D., & Wagner, D. G. (2001). Chromatic roots of generalized theta graphs. *J. Combinatorial Theory, Series B*, *83*, 272–297.

Dyer, M., Goldberg, L. A., Greenhill, C., & Jerrum, M. (2000). On the relative complexity of approximate counting problems. *Proceedings of APPROX'00, Lecture Notes in Computer Science 1913* (pp. 108–119).

Freund, Y., Mansour, Y., & Schapire, R. (2003). Generalization bounds for averaged classifiers (how to be a Bayesian without believing). To appear in Annals of Statistics. Preliminary

Dataset	$\|L\|\&\|U\|$	feat.	Graph	Mincut	Rand mincut	gaussian	sgt	exact
			MST	92.0	90.9	90.6	90.0	90.6
Voting	45+390	16	$\delta_{\frac{1}{4}}$	92.3	91.2	91.0	85.9	—
			MST	86.1	89.4	92.2	89.0	92.4
Mush	20+1000	22	$\delta_{\frac{1}{4}}$	94.3	94.2	94.2	91.6	—
			MST	78.3	77.8	79.2	78.1	83.9
Iono	50+300	34	$\delta_{\frac{1}{4}}$	78.8	80.0	82.8	79.7	—
			MST	63.5	64.0	63.7	61.8	63.9
Bupa	45+300	6	$\delta_{\frac{1}{4}}$	62.9	62.9	63.5	61.6	—
			MST	65.7	67.9	66.7	67.7	67.7
Pima	50+718	8	$\delta_{\frac{1}{4}}$	67.9	68.8	67.5	68.2	—

Table 1. Classification accuracies of basic mincut, randomized mincut, Gaussian fields, SGT, and the exact MRF calculation on datasets from the UCI repository using the MST and $\delta_{\frac{1}{4}}$ graph. On this data, all the algorithms perform fairly similarly.

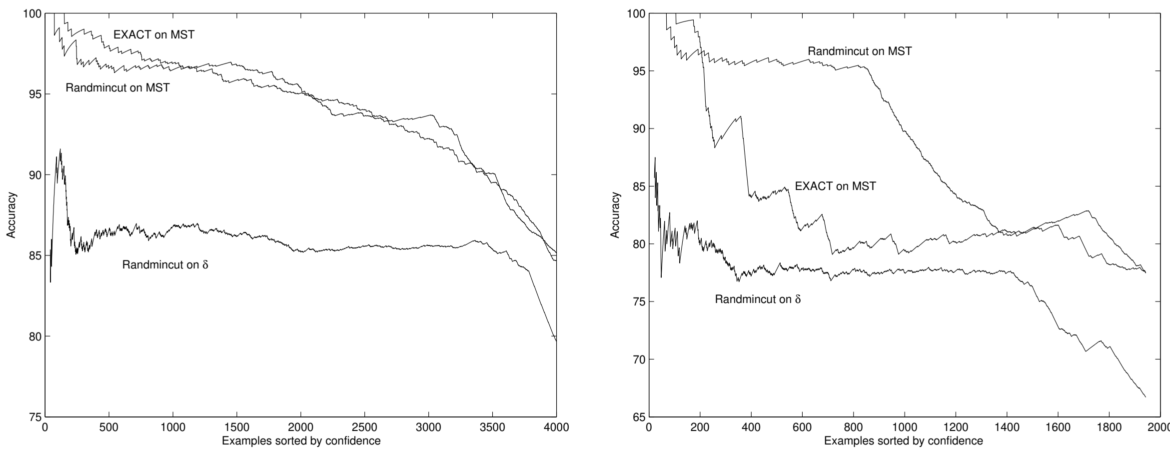

Figure 5. Accuracy coverage tradeoffs for randomized mincut and EXACT. Odd vs. Even (left) and PC vs. MAC (right). Both are at 52 labeled examples. Each curve shown here is for a single run of the algorithm, so the 100% coverage points do not exactly match the 10-run averages of Figures 3 and 4.

version appeared in Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics, 2001.

Greig, D., Porteous, B., & Seheult, A. (1989). Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, *51*, 271–279.

Hull, J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*, 550–554.

Jerrum, M., & Sinclair, A. (1993). Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, *22*, 1087–1116.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 290–297).

Kleinberg, J. (2000). Detecting a network failure. *Proc. 41st IEEE Symposium on Foundations of Computer Science* (pp. 231–239).

Kleinberg, J., Sandler, M., & Slivkins, A. (2004). Network failure detection and graph connectivity. *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms* (pp. 76–85).

Langford, J., & Shawe-Taylor, J. (2002). PAC-bayes and margins. *Neural Information Processing Systems*.

McAllester, D. (2003). PAC-bayesian stochastic model selection. *Machine Learning*, *51*, 5–21.

Zhu, X., Gharahmani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. *Proceedings of the 20th International Conference on Machine Learning* (pp. 912–919).