Exploiting Relations Among Concepts to Acquire Weakly Labeled Training Data

Joseph Bockhorst JOEBOCK@CS.WISC.EDU
Department of Computer Sciences, University of Wisconsin, Madison, WI 53706 USA

Mark Craven CRAVEN@BIOSTAT.WISC.EDU
Department of Biostatistics and Medical Informatics, University of Wisconsin, Madison, WI 53706 USA

Abstract

We consider a learning setting in which there are well defined relations that exist among instances of certain classes. In particular, we consider the domain of predicting various types of gene-regulation elements in bacterial genomes. Given instances of one class, we can often acquire "weakly labeled" training data for another class by taking advantage of known relationships that exist between the two classes. The examples are weakly labeled in that either the class label is incompletely specified, the exact extent of an instance is only partially specified, or both. We use an EM-based approach to handle this hidden state during learning. Our experimental results show that, when only small training sets are available, there can be significant value in augmenting the training sets with weakly labeled examples acquired from relationships among concepts.

1. Introduction

As machine learning methods are embedded into richer domains, it will be increasingly common to encounter situations in which there are known relations that link the instances being classified, as well as their classes. We argue that, in this type of domain, more accurate predictions can be gained by taking into account these relations. We present a new approach that is intended for the setting in which there are known, well-defined relationships between classes of interest. Our approach uses instances of one class, either known or predicted, to acquire "weakly labeled" training data for a related class. We demonstrate the value of our approach in the domain of predicting regulatory elements in a genome.

Each gene in a genome represents a blueprint for its product, another molecule (either protein or RNA) that is involved in the function of the cell. The products encoded by individual genes are produced at varying rates depending upon the cell type, the environmental conditions of the cell, the point in its life cycle and other factors. Many of the mechanisms of gene regulation are encoded as sequences in the genome themselves. In our research, we are focusing on bacterial genomes and trying to identify the basic units being regulated (sequences of genes called *operons*) (Craven et al., 2000), as well as certain classes of subsequences that play a role in gene regulation (Bockhorst & Craven, 2001).

An interesting aspect of this domain is that there are well defined relationships that exist among the instances of certain classes of regulatory elements. For example, an *operon* is always preceded by a sequence called a *promoter* and succeeded by a sequence called a *terminator*. We are interested in learning models to predict instances of multiple classes of regulatory elements. For example, we want to learn models to identify new instances of all three of classes above. However, we do not treat this problem as a single multiclass learning task since suitable representations for the individual classes are quite different. The question we consider in this paper is: Can we learn more accurate models for these classes by taking into account relationships that hold among their instances?

In particular, the approach we present is based on learning more accurate models by using the relationships among target concepts to get additional training data for some of the concepts. We refer to these additional instances as "weakly labeled" training data since (i) the class labels may be incompletely specified, (ii) they may contain more hidden state than our hand-labeled training instances, and (iii) they may contain class noise. Nevertheless, our experiments show that we can often improve the predictive accuracy of our models by training on this weakly labeled data.

There are a number of methods aimed at gaining predictive leverage from related learning tasks. Methods for *transfer* (Thrun & Pratt, 1997) try to improve the learned model for a given task by modifying the representation given to the learner based on learned models from other tasks. Similarly, *multitask learning* (Caruana, 1997) involves trying to improve a model by simultaneously training on related tasks. These approaches are designed for a setting in which there is some imprecisely characterized relationship among the learning tasks. Our method, in contrast, assumes that there are well defined relationships among the classes of interest, but it does not assume that the learners will share any part of their representations, nor does it even assume that we are learning models for all of the classes. Moreover, the learning process in our approach is aided with an enlarged training set rather than a changed representation.

Slattery and Mitchell (2000) also present a method that uses relationships among classes to improve accuracy in the classification process. Our method, on the other hand, operates during the learning process.

There are also several recent methods that aim to take advantage of unlabeled data by automatically imputing labels for some instances. Nigam et al. (2000) present an approach in which the EM algorithm is used to train on unlabeled instances after a model has been seeded with a small labeled training set. Instead of augmenting training sets with unlabeled examples, our approach uses domain relationships to obtain weakly labeled examples. However, in cases where our weak labels are only partially constrained, we use EM in a similar manner. In the *co-training* method (Blum & Mitchell, 1998), two models with different "views" of the same task try to bootstrap each other by assigning labels to unlabeled instances. The method of Goldman and Zhou (2000) is similar except that instead of employing different views, the two models are trained from different supervised leaning algorithms. In our second set of experiments, we also have one learned model bootstrap another one. However, our two models are trained for different tasks, and the ability of one to help the other arises from relationships between their target classes.

As in our task, *multiple instance* problems (Dietterich et al., 1997) also involve uncertainty in the training examples. In a multiple instance problem, only some of the instances in a positive bag are positive. In our task, only a subsequence of a positive example is of interest for the concept. Additionally, in our task there may be incompleteley specified and noisy class labels.

In summary, the key novel aspect of the method we present here is that we exploit relationships among classes in the domain to get additional, weakly labeled, training instances for some of the classes. Although our discussion throughout the paper focuses on the gene-regulation domain, we believe that our approach has applications to other domains, such as Web page classification (Slattery & Mitchell, 2000) and natural language processing, that involve regular relationships among the objects being classified.

2. Problem Setting

In this section we describe our problem domain in more detail and discuss how relationships among classes can be used to acquire weakly labeled training data.

Figure 1 illustrates the regulatory elements that we are concerned with in this paper. These regulatory elements are all involved in *transcription*, which is the first of two processes in which a gene's product is produced. The horizontal line in Figure 1 represents a segment of DNA containing seven genes. Genomic DNA is double-stranded, and genes are located on both strands. Genes are illustrated as boxes in the figure, and the strand of each gene is indicated by whether the corresponding box is above or below the line. In the figure, five of the genes, $\{g_2, ..., g_6\}$, are on one strand, and two, $\{g_1, g_7\}$, are on the opposite strand. In some organisms, such as bacteria, there are certain sets of contiguous genes, called *operons*, that are "turned on" or "turned off" as a unit. Figure 1 shows three operons, $\{o_1, o_2, o_3\}$. The first operon, o_1, consists of two genes, g_2 and g_3. The genes in an operon must all be on the same strand. Some genes, such as g_6 in the figure, are controlled individually. We will consider these cases to be operons of length one. Each operon has an associated *promoter* and *terminator*. In the figure, promoters and terminators are denoted by $\{p_1, p_2, p_3\}$ and $\{t_1, t_2, t_3\}$, respectively. These two types of elements can be thought of as signals that indicate "start transcribing an operon here" and "stop transcribing an operon here." There are actually two subclasses of terminators, which we refer to as *RI-terminator* and *RD-terminator*. Although instances of these two subclasses perform the same function, the sequences which encode them are quite different.

All of the regulatory elements described above can be thought of as subsequences of the DNA strand in which they are encoded. We use the convention that the subscripts for a given class of element indicate their ordering on the genome[1]. So, for example, gene g_i

[1] The two DNA strands are actually "read" in opposite

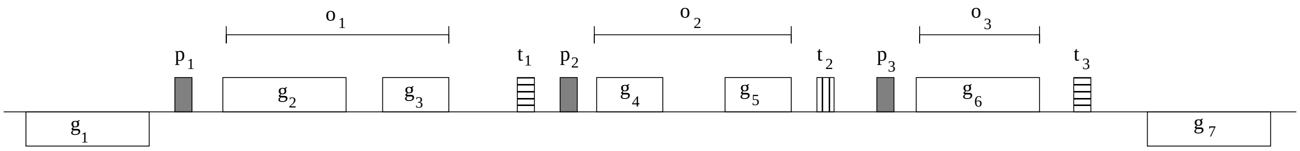

Figure 1. Some regulatory elements in bacterial genomes. The horizontal line represents a segment of DNA containing seven genes, $\{g_1, ..., g_7\}$. Five of the genes, $\{g_2, ..., g_6\}$, are on one strand, and two, $\{g_1, g_7\}$, are on the opposite strand. Also shown are the three operons, $\{o_1, o_2, o_3\}$, and the promoters, $\{p_1, p_2, p_3\}$, and terminators, $\{t_1, t_2, t_3\}$, for these operons. An operon is a sequence of genes that is regulated as a unit. Promoters and terminators are subsequences that flank operons. There are two subclasses of terminators, as indicated here by the orientation of the hatching.

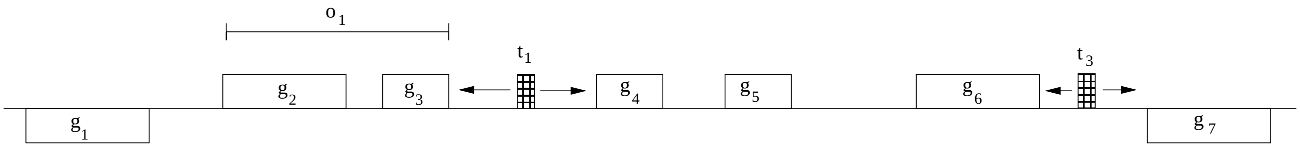

Figure 2. Inferring weakly labeled terminator examples from relations in the domain. In this scenario, we do not know how genes $\{g_4, ..., g_6\}$ are grouped into operons, nor do we know the coordinates of any promoters or terminators. From known relationships among genes, operons and terminators, however, we can infer that there must be terminators between g_3 and g_4, and between g_6 and g_7. These terminator examples are weakly labeled in that we do not know the precise coordinates of the terminators, nor do we know which terminator subclass each one is in.

comes before gene g_{i+1}.

In the setting considered here, we assume that gene boundaries (i.e. the coordinates of the corresponding subsequences) are given, since genes can be predicted with high accuracy in bacterial genomes. The overall task we are addressing can be stated as follows. Given the sequence of a genome, the coordinates of putative genes in the genome, and known instances of operons, promoters and terminators, we want to learn models for predicting operons, promoters and terminators; then scan the genome predicting new instances of these classes. Promoter and terminator predictions consist of DNA subsequences that appear to be members of these classes. Operon predictions, on the other hand, consist of sequences of genes.

We hypothesize that, for certain training set sizes, we can get more accurate models by exploiting relationships among instances of the different classes. Consider learning a model of terminators. We know:

- operons rarely overlap,
- terminators are likely to occur only in the subsequences between operons,
- operons do not "bridge" genes on the opposite strand.

Given these facts, we note that the following relationship holds between operon and terminator instances.

- If we know that a given gene, g_i, is the last (or only) gene in an operon, then we know that a terminator is likely to be found between g_i and the next gene in the DNA sequence, g_{i+1}.

For example, Figure 2 illustrates a case in which we know the coordinates of the genes shown in Figure 1, but we do not know the coordinates of any of the promoters or terminators, nor do we know how genes $g_4, ..., g_6$ are grouped into operons. Given the relationship above, we would still be able to conclude that there are terminators somewhere between g_3 and g_4, and between g_6 and g_7. In the first case, we are able to conclude that there must be a terminator between g_3 and g_4 because we know the extent of the operon o_1. In the second case, even though we do not know the extent of the operon including g_6, we conclude that there must be a terminator between g_6 and g_7 because *some* operon must end at g_6.

The example above suggests that we may be able to increase the size of our terminator training set by taking advantage of these relationships. However, it is important to note that the training examples we get from these relationships are not as informative as those we have in our conventional training set – hence we refer to them as weakly labeled examples.

A "strongly" labeled example, like those we have in our data set of known terminator sequences, consists of a sequence that is fairly precisely delineated along with a label indicating to which of two subclasses the terminator belongs. With our weakly labeled termina-

directions. However, we pick the direction of one strand to give us our canonical ordering.

tor examples, on the other hand, there are two sources of hidden information. First, the example consists of a longer sequence that includes the terminator as a subsequence; the start and end coordinates of the terminator sequence are not specified. Second, the labeling for each example does not indicate the terminator subclass to which the example belongs. This is significant because the properties of the two subclasses of terminators are quite different.

After discussing the learning methods we use, we empirically consider the value of augmenting our training sets with these weakly labeled examples. In Section 4, we explore the case where we get weakly labeled examples strictly from relationships in the domain, like the case above of ascertaining that there likely is a terminator between g_6 and g_7. In Section 5, we consider augmenting our training data based on the combination of domain relationships *and* predictions from related models. This case would arise, for example, if we had posited the existence of a terminator between g_3 and g_4 based on a prediction that o_1 is an operon. In Section 5 we use a learned operon model to generate such predictions.

Although we focus on the case of learning terminators in this section and the remainder of the paper, we emphasize that the applicability of this approach is broader. For example, just with the regulatory elements we have discussed here, there are other relationships we could exploit:

- If we know that a given gene, g_i, is the first (or only) gene in an operon, then we know that a promoter is likely to be found between g_{i-1}, the previous gene in the DNA sequence, and g_i.
- If we know that there is a terminator, t_i, and a promoter, p_{i+1}, between g_i and g_{i+1}, then we know that t_i must be located before p_{i+1}. In other words, terminators occur before promoters in inter-operon sequences.

Moreover, we have limited our discussion here to only a few types of regulatory elements, but we are interested in predicting other types, including *activators*, *attenuators*, *regulons*, *repressors*, *transcription factor binding sites* and *UP elements*. There are therefore multiple relationships we can potentially exploit to get weakly labeled training examples for multiple classes.

3. Learning Methods

In this section we describe the models that we use in our experiments. The first prediction task that we consider is to recognize terminators in DNA sequences. More specifically, given a subsequence occurring between genes, we would like to determine if the subsequence contains an *RI-terminator*, an *RD-terminator*, or neither (*negative* class). Our approach is based on Bayes' rule. Given a sequence s, we calculate the probability that s belongs to class c_i as follows:

$$\Pr(c_i|s) = \frac{\Pr(s|c_i)\Pr(c_i)}{\sum_j \Pr(s|c_j)\Pr(c_j)}$$

where c_j ranges over all three of the possible classes. The key aspect of our terminator model is how we compute the probabilities $Pr(s|c_i)$ for the three classes.

For the *negative* class, we use a second-order Markov chain model. Using this model, the probability of a given sequence s is calculated as follows:

$$\begin{aligned}\Pr(s|neg) = \; & \Pr(l|neg)\Pr(s_1|neg)\Pr(s_2|s_1,neg) \\ & \textstyle\prod_{i=3}^{l}\Pr(s_i|s_{i-1},s_{i-2},neg).\end{aligned}$$

Here, l represents the length of sequence s, and s_i represents the ith character in s. Given a training set of negative sequences, we use a maximum likelihood procedure with Laplace smoothing to train this model

The models we use for the *RI-terminator* and *RD-terminator* classes are somewhat more complex. Each of these models is a stochastic context free grammar (SCFG). Elsewhere, we describe our SCFG approach to terminator recognition in more detail (Bockhorst & Craven, 2001). Here, we give just an overview of the approach. The reason for choosing the SCFG representation is that the defining characteristics of terminators involve strong dependencies between sequence elements that may be separated by up to 30 positions.

The structure of our SCFG for *RI-terminators* consists of a central component, called the *stem-loop*, flanked by a first-order Markov chain model, on either side. These Markov models are similar to the one described above for the *negative* class. The *stem-loop* portion of the grammar was designed to capture known features of *RI-terminators* that have been identified in the literature (Brendel et al., 1986; Carafa et al., 1990). Table 1 shows the productions of our grammar with the exception of those embedded in the Markov chains.

RD-terminators have not been nearly as well characterized. However, they have some of the same properties as *RI-terminators*, so we use a model with the same architecture for them. Productions in the *RI-terminator* model are only used by *RI-terminators* and likewise for the *RD-terminator* model. Similarly, the two models have separate parameters.

Given a training set of strongly labeled *RI-terminator* and *RD-terminator* examples, we train our terminator models using an EM algorithm for SCFGs called the

Table 1. The structure of our *RI-terminator* and *RD-terminator* grammars. Nonterminals are capitalized and the terminals are a, c, g and t. The notation X $\rightarrow$ Y | Z is shorthand for the two productions X $\rightarrow$ Y and X $\rightarrow$ Z. Productions containing t_l and t_r are shorthand for the family of 16 productions where t_l and t_r can be any of the four terminals. Productions containing t_l^* and t_r^* have a similar interpretation except that t_l^* and t_r^* can also be null allowing for unpaired bases in the interior of the stem.

R[I,D]-TERM	$\rightarrow$ FRONT STEM_LOOP REAR
FRONT	$\rightarrow$ *First-order Markov chain model*
REAR	$\rightarrow$ *First-order Markov chain model*
STEM_LOOP	$\rightarrow$ PREFIX STEM_BOT1 SUFFIX
PREFIX	$\rightarrow P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7\ P_8\ P_9\ P_{10}$
STEM_BOT1	$\rightarrow t_l$ STEM_BOT2 t_r
STEM_BOT2	$\rightarrow t_l^*$ STEM_MID t_r^* | t_l^* STEM_TOP2 t_r^*
STEM_MID	$\rightarrow t_l^*$ STEM_MID t_r^* | t_l^* STEM_TOP2 t_r^*
STEM_TOP2	$\rightarrow t_l^*$ STEM_TOP1 t_r^*
STEM_TOP1	$\rightarrow t_l$ LOOP t_r
LOOP	$\rightarrow$ B B B LOOP_END
LOOP_END	$\rightarrow$ B LOOP_END | B
SUFFIX	$\rightarrow S_1\ S_2\ S_3\ S_4\ S_5\ S_6\ S_7\ S_8\ S_9\ S_{10}$
B	$\rightarrow$ a | c | g | t
P_i	$\rightarrow$ a | c | g | t
S_i	$\rightarrow$ a | c | g | t

inside-outside algorithm (Lari & Young, 1990). The hidden values estimated during the E-step are the expected number of times each production is used over all the sequences in the training set.

If the training set includes weakly labeled examples the training procedure must be adjusted because the label of the subclass, either *RI-terminator* or *RD-terminator*, is hidden for weakly labeled examples,. We handle this case by joining the terminator models with the two productions

$$\text{TERM} \rightarrow \text{RI-TERM} \mid \text{RD-TERM}$$

which are both assigned a probability of 0.5. To compute the expected production counts for a weakly labeled example s during the E-step, we first compute the expected counts of productions in both terminator models as if the subclass were known. Then, the counts of productions in the *RI-terminator* model are weighted by a factor of $\Pr(RI-term|s, pos)$, the probability that the sequence belongs to the *RI-terminator* class given that it contains a terminator. Because of the uniform prior, this value is calculated as

$$\frac{\Pr(s|RI-term)}{\Pr(s|RI-term) + \Pr(s|RD-term)}.$$

The counts of productions in the *RD-terminator* model are weighted analogously.

In some of our experiments, we consider learning with unlabeled examples. Unlabeled examples in the training set are treated similarly to weak examples. In this case though, all three models are joined by the productions

$$\text{ALL} \rightarrow \text{RI-TERM} \mid \text{RD-TERM} \mid \text{NEG}$$

where NEG here refers to our second-order Markov chain model for negative sequences. The expected counts of the productions of the three models from unlabeled sequences are weighted in an analogous manner to the counts from weak examples.

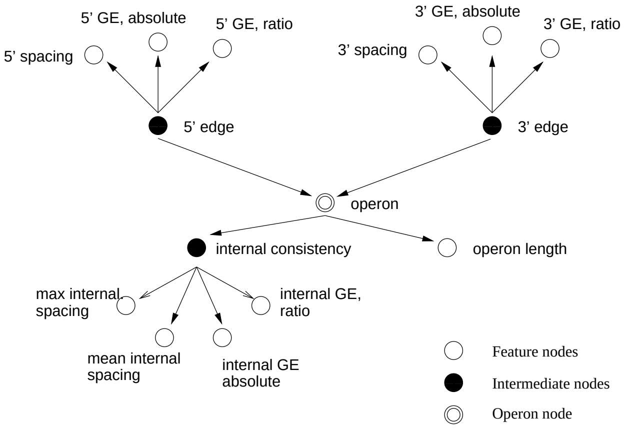

Figure 3. Our Bayes net operon model. Instances are sequence of genes and all nodes are discrete valued. On test examples feature nodes are always observable, intermediate nodes may be observable or hidden and the boolean valued operon node is always hidden. Classification is done by setting a threshold on the posterior probability of the operon node given the values of all its observable nodes.

In Section 5, in addition to learning models for terminators, we also learn models to predict operons. For this task, we employ the Bayesian network shown in Figure 3. Instances in this task are sequences of contiguous genes. An instance is considered *positive* if its sequence of genes constitutes a complete operon and *negative* otherwise. Each instance is represented by a set of features that correspond to the nodes in the Bayes net. The features represent such aspects as the number of genes in the instance, the spacing between genes within and neighboring the instance, and transcription rates for the genes in the instance measured under various conditions. For a more detailed explanation see Craven et al. (2000).

Each node in this network is discrete valued. Training is done by computing counts over the training examples and setting probabilities to the observed frequencies. Again, Laplace smoothing is used for all parameters. Classification is done by computing the posterior distribution for the operon node after setting all observable nodes.

4. Using Relations to Acquire Weakly Labeled Examples

In this section, we present a set of experiments designed to test our hypothesis that weakly labeled training examples can lead to more accurate learned models. The weak examples that we use in these experiments are derived from the relationship between terminators and the ends of operons that was discussed in Section 2. In particular, we use *known* operon boundaries to generate weakly labeled examples. We evaluate our hypothesis by analyzing our terminator model's accuracy when it is trained on varying numbers of strong and weak examples.

The data set we use for our experiments consists of 365 known operons, 105 known *RI-terminators* and 84 *RD-terminators* from the most heavily studied bacterium, *E. coli*. Although *E. coli* provides a good testbed for our experiments, the real targets of our approach are the many bacterial genomes about which little is known except for gene coordinates. We use the knowledge that terminators rarely exist between genes in the same operon to construct negative examples for the terminator model. We assemble a negative example from each consecutive pair of genes that are in the same operon and are at least 50 bases apart. This gives us 158 negative examples.

We run a series of five-fold cross-validation experiments for the binary classification problem of distinguishing terminators (both classes) from non-terminators. A single experiment involves training our terminator models with a training set consisting of S strong examples and W weakly labeled examples. To construct the training and test sets, we first partition our complete pool of known operons and known terminators into five folds. The negative examples for a fold consist of all those that can be derived from the operons in the fold. This partitioning is the same for all experiments. For an experiment with S strong terminators and W weak terminators, S terminators and W known operons are randomly sampled from the appropriate training folds. We derive W weakly labeled terminator examples from the sequences following the last genes of these operons.

We have explored two different training methods for utilizing weak examples. The first method pools together the strong and weak examples and runs the inside-outside algorithm to convergence. The second method attempts to start the EM procedure near a good local maximum by first running inside-outside to convergence using only the strong examples. Then the strong and weak examples are pooled and inside-outside is run to convergence again. The experiments

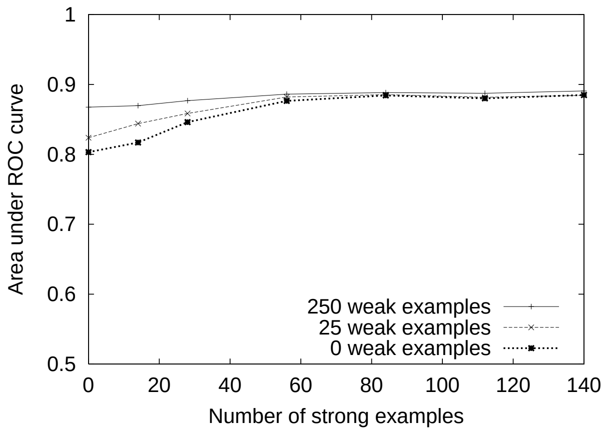

Figure 4. Learning curves showing the area under ROC curves vs. the number of strong training examples for 0, 25 and 250 weakly labeled examples. Note the y-axis range is from the expected value of a random classifier (0.5) to the value of a perfect classifier (1.0).

we have run have shown no significant difference in the accuracy of the models learned by these two methods and so here we present results for the first method.

For each (S, W) experiment, we construct ROC curves for each test-set in the cross validation, and then construct a composite ROC curve by averaging the individual ROC curves (Provost et al., 1998). An ROC curve plots the true positive (TP) rate $(\frac{TP}{TP+FN})$ versus the false positive (FP) rate $(\frac{FP}{FP+TN})$ as we vary a threshold on the confidence of our classifications. The measure we use to evaluate accuracy in a given experiment is the area under the composite ROC curve. This value can range from 0.5 for a random classifier to 1.0 for a perfect classifier.

Figure 4 presents a set of learning curves showing, for various numbers of strong training examples, how additional weakly labeled examples affect predictive accuracy. The x-axis in the figure indicates the number of strong examples the learner is given. The y-axis represents predictive accuracy in terms of the area under the composite ROC curves. The three curves in the figure represent differing numbers of weak examples that the learner is given. The curve for zero weak training examples represents a baseline. It shows that the accuracy of our models without any training examples is about 0.80 (due to strong priors in the SCFG) and has a maximum near 0.89 with little improvement seen after more than 60 strong examples. The other two curves show that augmenting our training set with weakly labeled training examples gives a boost in predictive accuracy, most noticeably when the number of strong examples is small. In fact, with 250

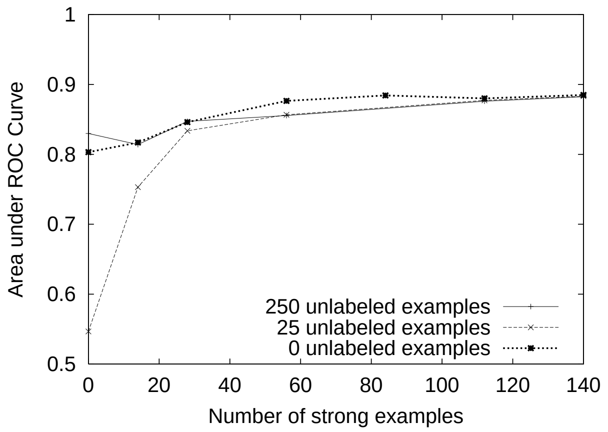

Figure 5. Learning curves showing the area under ROC curves vs. number of the strong training examples for 0, 25 and 250 *unlabeled* examples. Note the y-axis range is from the expected value of a random classifier (0.5) to the value of a perfect classifier (1.0)

weakly labeled training examples and no strong examples, the accuracy is comparable to the best that can be achieved with strong examples only.

In order to judge the significance of the accuracy differences, we conduct pairwise t-tests for points at fixed numbers of strong examples. In these tests, the null hypothesis is that there is no difference in the areas under the ROC curves for varying numbers of weak examples. These tests show that at zero and 14 strong examples, the differences are significant at the 95% level, with the exception of the difference between zero and 25 weak examples at the point of zero strong examples. For the larger strong-example sets, t-tests do not show that the differences are significant.

As a control, we also consider constructing the same type of learning curves by augmenting training sets with *unlabeled* examples instead of weakly labeled ones. The difference between these two types of examples is that the weakly labeled examples have been identified as belonging to one or other of the terminator classes. We train on these examples by using our EM method to iteratively assign "soft" labels to these examples. This is analogous to the approach used by Nigam et al. (2000) for training document classifiers.

Figure 5 shows the results for this set of experiments. The curve showing zero unlabeled examples is the same, by definition, as the curve in Figure 4 that shows zero weakly labeled examples. The other two curves illustrate that augmenting our training set with unlabeled data yields no appreciable accuracy boost, and in some cases accuracy suffers.

We use pairwise t-tests to compare corresponding points on the weak-example curves and the unlabeled-example curves (i.e. we compare the 250-weak curve to the 250-unlabeled curve, and the 25-weak curve to the 25-unlabeled curve). The differences between the two pairs of curves at all points representing 56 and fewer strong examples are significant at the 95% level.

From the results presented in this section, we conclude that weakly labeled examples acquired from domain relationships can have significant value with small training sets. Moreover, the value of the weakly labeled examples exceeds that of unlabeled examples.

5. Using Predictions and Relations to Acquire Weakly Labeled Examples

In our experiments in the previous section, we acquired weakly labeled examples by exploiting (i) a known relationship between the boundaries of operons and terminators, and (ii) knowledge of some actual operon boundaries. Now we turn our attention to the case where our weakly labeled examples are based not on *known* instances of another class, but on *predicted* instances. Specifically, we ask if we can improve the accuracy of our terminator models by using weak terminator examples that are based on operons predicted by the Bayes net model described in Section 3.

As before, we run a set of five-fold cross-validation experiments. The partitioning of our complete data set is the same as before but the construction of the training sets for the individual experiments differs. For the experiment where the training set has S strong examples and W weak examples, we sample S known terminators from the appropriate training folds of the complete data set, as before. In addition, we also sample S known operons from the training folds. From these known operons we can infer an associated set of negative operons and then train the Bayes net operon model described in Section 3. Using this operon model we scan the genome keeping track of our most confident operon predictions. From these operon predictions, we infer W weakly labeled terminator and negative examples. These weak examples are then used to augment the training set for the terminator model.

Figure 6 shows how the area under the ROC curves of our trained terminator models changes as a function of the number of strong examples used to train both the operon and terminator models. We show learning curves for training sets augmented with 0, 25, 100 and 200 weak examples. As before, when we have few strong examples we get improvement in accuracy by using weak examples. A paired t-test shows that the

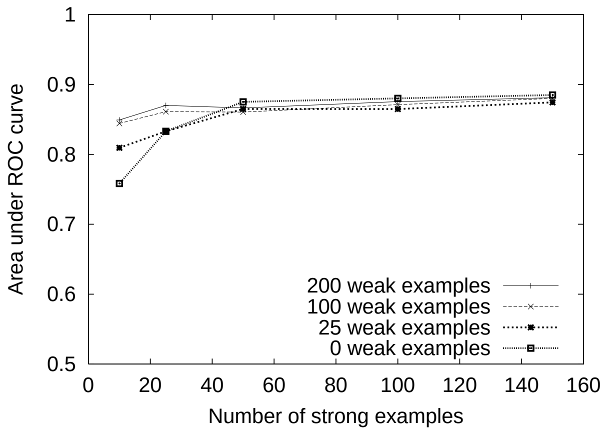

Figure 6. Learning curves showing the area under ROC curves vs. the number of strong training examples for various numbers of weak examples. The weak examples are inferred from *predicted* operons.

areas under the ROC curves for the models trained with 10 strong examples and 100 or 200 predicted weak examples are significantly different, at greater than a 95% confidence level, than the models trained without weak examples. For larger sets of strong examples, we cannot show that the differences are significant.

6. Conclusions

We have addressed the task of learning in domains where well defined relations exist among instances of certain classes. We have argued that, in such domains, there is the potential to infer weakly labeled training examples for some learning tasks that can lead to more accurate models.

To test this hypothesis, we conducted experiments in which we generated weakly labeled examples for learning to recognize terminators in bacterial genomes. In our first set of experiments, we used a known relationship between the boundaries of operons and terminators, and knowledge of some actual operon boundaries to generate weakly labeled examples. In our second set of experiments, we used predicted operon boundaries instead of known ones. Both sets of experiments demonstrated that, for small training set sizes, the weakly labeled examples resulted in significantly more accurate models. For larger training set sizes, the weak examples did not offer much benefit, but they did not degrade accuracy either.

The experiments presented herein represent a preliminary investigation of our hypothesis. In future work, we plan to apply our approach using other target concepts and other relations in our domain. Additionally, we plan to investigate using *both* known and predicted instances of a given class in order to infer weakly labeled examples for another class.

Acknowledgments

This research was supported in part by NSF grant IIS-0093016. Thanks to Soumya Ray for helpful comments.

References

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proc. of the 11th Conf. on Computational Learning Theory* (pp. 92–100).

Bockhorst, J., & Craven, M. (2001). Refining the structure of a stochastic context-free grammar. *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (pp. 1315–1320). Seattle, WA: Morgan Kaufmann.

Brendel, V., Hamm, G., & Trifinov, E. (1986). Terminators of transcription with RNA polymerase from Escherichia coli: What they look like and how to find them. *Journal of Biomolecular Structure and Dynamics*, *3*, 705–723.

Carafa, Y. A., Brody, E., & Thermes, C. (1990). Prediction of rho-independent Escherichia coli transcription terminators: A statistical analysis of their RNA stem-loop structures. *Journal of Molecular Biology*, *216*, 835–858.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Craven, M., Page, D., Shavlik, J., Bockhorst, J., & Glasner, J. (2000). A probabilistic learning approach to whole-genome operon prediction. *Proceedings of the 8th International Conf. on Intelligent Systems for Molecular Biology* (pp. 116–127). La Jolla, CA: AAAI Press.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Goldman, S., & Zhou, Y. (2000). Enhancing supervised learning with unlabled data. *Proc. of the 17th Int. Conf. on Machine Learning* (pp. 327–334). Stanford, CA

Lari, K., & Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, *4*, 35–56.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. *Proc. of the 15th Int. Conf. on Machine Learning* (pp. 445–453). Madison, WI: Morgan Kaufmann.

Slattery, S., & Mitchell, T. (2000). Discovering test set regularities in relational domains. *Proc. of the 17th Int. Conf. on Machine Learning* (pp. 895–902). Stanford, CA: Morgan Kaufmann.

Thrun, S., & Pratt, L. (1997). *Learning to learn*. Boston, MA: Kluwer.