Appears in *Proceedings of the 17th International Conference on Machine Learning (ICML 2000).*

Using Multiple Levels of Learning and Diverse Evidence Sources to Uncover Coordinately Controlled Genes

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

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

Jeremy Glasner JEREMY@GENOME.WISC.EDU
Department of Genetics, University of Wisconsin, Madison, WI 53706 USA

Abstract

Now that the complete genomes of numerous organisms have been determined, a key problem in computational molecular biology is uncovering the relationships that exist among the genes in each organism and the regulatory mechanisms that control their operation. We are developing computational methods for discovering such regulatory mechanisms and relationships. Toward this end, we have developed a machine learning approach to identifying sets of genes that are coordinately controlled in the *E. coli* genome. A number of factors make this an interesting application for machine learning: (i) there is a rich variety of data types that provide useful evidence for this task, (ii) the overall problem of uncovering regulatory mechanisms can be decomposed in multiple machine learning subtasks operating at different levels of detail, (iii) there are not any known negative training examples, and (iv) some of the features are misleading in their predictiveness.

1. Introduction

The complete genomic sequences of more than 30 organisms have been determined in the last few years, more than 130 other genomes are currently being sequenced, and a "working draft" of the human genome is expected to be completed this year. A significant challenge now confronting biologists is to determine the functions of the genes contained in these sequences. One aspect of understanding the function of a given gene is to determine the conditions under which it is active and the interactions it has with other genes. Toward this end, we have begun a research project that is investigating computational approaches to uncovering the regulatory mechanisms and interactions of genes in the heavily studied organism *E. coli*. A key aspect of our approach is to learn predictive models from existing data. As a first step, we have used machine learning methods to induce models for predicting *operons*. Informally, operons are consecutive sequences of genes that are "turned on" or "shut off" as a unit.

We argue that this is an interesting machine learning application for several reasons:

- The problem of uncovering regulatory mechanisms can be decomposed into multiple, related machine learning tasks.
- There is a rich variety of data types that provide useful evidence for the task including DNA sequence data, numeric gene-expression data, and hierarchically organized gene annotations.
- We do not have any negative examples per se for the target concept.
- A group of features that is apparently the most predictive group, is probably misleading in its value. In other words, the values we have for these features on unlabeled data are less reliable than the values we have for labeled data.

We consider these issues in more detail throughout the paper and revisit them again in the concluding section. The first issue – multiple related learning tasks – we discuss more here. The main task that we consider in this paper is predicting which sequences of genes con-

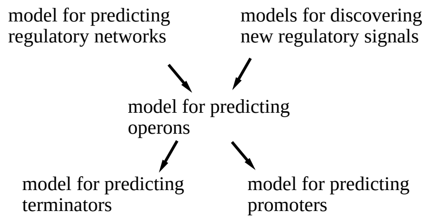

Figure 1. Our multi-level learning approach to discovering gene regulatory mechanisms. The tasks at the highest level represent those that are motivating our research. The task at the middle level represents the main focus of this paper. Those at the lowest level represent learning tasks that we address in order to make better predictions for the middle level task. The arrows represent subtask relationships.

stitute operons. This task is not the overall goal of our work, however, but only an intermediate step in addressing such problems as inferring networks of regulatory interactions, and discovering new subclasses of sequences involved in controlling gene transcription. Moreover, we hypothesize that operon predictions can be improved by first identifying certain control sequences (*promoters* and *terminators*) in the genome. The recognition of these sequences is itself not a well understood task, and thus we also address the learning subtasks of constructing predictive models of these sequences. Figure 1 illustrates the relationships among the learning tasks involved in our work. The predictions made by a model learned at one level are passed to learning components at the next level to be used as input features. This idea of decomposing a learning task into subtasks has been around for some time (Fu & Buchanan, 1985; Shapiro, 1987), and has been applied recently in robotic systems (O'Sullivan, 1998; Stone, 1998).

2. Problem Domain

Currently, our primary task is to predict operons in the *E. coli* genome. The approach that we are developing is applicable to other *prokaryotic*[1] organisms as well. The genome of *E. coli*, which was sequenced at the University of Wisconsin (Blattner et al., 1997), consists of a single circular chromosome of double-stranded DNA. The chromosome of the particular strain of *E. coli* (K-12) in our data set has 4,639,221 base pairs (this can be thought of as a string consisting of 4,639,221 characters from a four-letter alphabet).

A *gene* is a sequence of DNA bases that carries the information required for constructing a particular *protein.* Proteins perform most of the essential functions within cells, including structural support, transport of other substances, response to chemical stimuli, etc. *E. coli* has approximately 4,400 genes, located on both strands of the chromosome.

[1] *Prokaryotes* are single-celled organism that do not have separate nuclei.

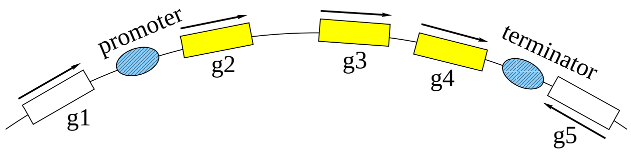

Figure 2. The concept of an operon. The curved line represents part of the *E. coli* chromosome and the rectangular boxes on it represent genes. An operon is a sequence of genes, such as $[g2, g3, g4]$ that is transcribed as a unit. Transcription is controlled via an upstream sequence, called a *promoter*, and a downstream sequence, called a *terminator*. A promoter enables the molecule performing transcription to bind to the DNA, and terminator signals the molecule to detach from the DNA. Each gene is transcribed in a particular direction, determined by which of the two strands it is located. The arrows in the figure indicate the direction of transcription for each gene.

The process by which proteins are produced from their encoding genes is referred to as *gene expression.* Individual genes are expressed at differing levels depending on (i) the extent to which certain other genes in the cell are active, and (ii) environmental factors such as temperature and the presence of particular nutrients. In multicellular organisms, for example, some genes may be completely shut off in certain cells or tissues.

The first step in the process of a gene being expressed is for it to be *transcribed* into a similar RNA sequence. Although the expression of a gene can be regulated at various points, the most significant regulatory controls are exerted on the transcription process. For example, a gene can be "shut off" by preventing it from being transcribed. In some organisms, such as *E. coli*, there are certain sets of contiguous genes, called *operons* that are transcribed coordinately. In other words, the genes in an operon are "turned on" or "shut off" as a unit. Thus, a complete understanding of the regulatory mechanisms in organisms such as *E. coli* requires that we know which sets of genes constitute operons.

Figure 2 illustrates the concept of an operon. The transcription process is initiated when a molecule called *RNA polymerase* binds to the DNA before the first gene in an operon. The RNA polymerase binds to a special sequence called a *promoter*. It then moves along the DNA using it as a template to produce an RNA molecule. When the RNA polymerase gets past the last gene in the operon, it encounters a special sequence called a *terminator* that signals it to release the DNA and ceases transcription. Some genes are transcribed individually; we refer to these special cases as *singleton* operons.

The data that we have available for learning a model of operons, some of which come from the RegulonDB (Salgado et al., 2000), include the following:

- complete DNA sequence of the genome,
- beginning and ending positions of 3,033 known genes and 1,372 putative genes,
- positions and sequences of 438 known promoters, and 289 known terminators,
- functional annotation codes characterizing 1,668 genes; these are taken from a three-level, 123-leaf hierarchy (Riley, 1996),
- gene expression data characterizing the activity levels of 4,097 genes and putative genes across 39 experiments,
- 365 known operons.

It is estimated (F. Blattner, personal communication) that there are several hundred undiscovered operons in *E. coli*. Our immediate goal is to predict these operons using a model learned from the data described above.

3. Problem Representation

In this section we describe the features that our learning methods use to assess the probability that a given candidate actually is an operon.

3.1 Length and Spacing Features

We use several features that relate to the size and intergenic spacing of operons and non-operons:

- Operon size: The number of genes in the candidate operon.
- Within-operon spacing: The *mean* and the *maximum* spacing (# DNA base pairs) between the genes in a candidate operon (e.g., the distances between *g2* and *g3* and between *g3* and *g4* in Figure 2). Since the genes in an operon are transcribed together, there might be constraints on inter-gene spacing. This feature is not defined for singleton operons (consisting of one gene).
- Distance to neighboring genes: The distances to the *preceding* (*g1* in Figure 2) and *following* (*g5* in Figure 2) genes. Notice that these genes are not part of the candidate operon.
- Directionality of the neighboring genes: These are two Boolean-valued features, indicating whether or not the directionality of the preceding and following genes individually match the directionality of the genes within the candidate operon. Recall that all the genes within an operon are transcribed in the same direction.

We refer to these last two features collectively as our neighboring genes features.

3.2 Functional Annotation Features

Since the genes in an operon typically act together to perform some common function, we expect the functions of the individual genes to be related to each other. The functions of many genes in *E. coli* have been described using a three-level hierarchy (Riley, 1996). The levels of this hierarchy represent broad, intermediate, and detailed functions. For example one path from the root to a leaf in this hierarchy is: root→metabolism of small molecules→carbon energy metabolism→anaerobic respiration.

We measure the "functional distance" between two genes in terms of how deeply into this hierarchy they match: genes with completely identical functions have a distance of 0, genes with identical broad and intermediate functions only have a distance of 2, genes with identical broad functions only have a distance of 4, and genes without common function have a distance of 6. In cases where the function is unknown for one or both genes at a given level, we split the difference: e.g., if two genes share a broad function but the intermediate function of one is unknown, the distance is 5.

Given the distance measure between the annotations for pairs of genes, we use three features to measure the functional homogeneity of a candidate operon. Collectively, we refer to these as the functional annotation features. The first feature represents the mean pairwise functional distance for genes within the candidate operon. We also consider the functional distance between the gene preceding the candidate and genes within the candidate. Specifically, we compute the mean of the pairwise distances between the preceding gene and each of the genes within the operon. The third feature is analogous except that it uses the first gene *after* the candidate operon.

3.3 Transcription Signal Features

As discussed in Section 2, each operon has special subsequences, called *promoters* and *terminators*, upstream and downstream respectively. These subsequences act as *signals* to the molecule that performs transcription, effectively telling it to *begin transcription here* and *end transcription here*. Thus, to decide if a given sequence of genes represents an operon or not, we would like to look upstream from the first gene in the sequence to see if we find a promoter, and to look downstream to see if we find a terminator.

The task of recognizing promoters and terminators, however, is not easily accomplished. Although there are known examples of both types of sequences, the sufficient and necessary conditions for them are not known. Thus, to use promoters and terminators as evidence for operons, we first need some method that can be used to predictively identify them. Our approach is to learn *interpolated Markov models* (IMMs) (Jelinek & Mercer, 1980) that characterize the known promoters and terminators. IMMs have been used previously for modeling biological sequences (Salzberg et al., 1998), although the particular task here is somewhat different. An interpolated Markov model consists of a set of Markov models of different orders. An IMM makes a prediction in a given case by interpolating among the statistics represented in the models of different order.

Each IMM is trained to recognize the presence of a particular signal (promoter or terminator). Our training set for promoters consists of 438 sequences, each of which is 81 bases long and contains a promoter in it. Since these promoter sequences are aligned to a common reference point, we can obtain statistics about the likelihood of a particular base at a particular position in a promoter. The terminator data set is similar, except that it consists of 289 sequences of length 58.

Our IMMs represent the probability of seeing each of the possible DNA bases at each position in the given signal. To assess the probability that a given sequence S is a promoter (or terminator), we calculate the product of the IMM's estimated probability of seeing each base in the sequence.

$$\Pr(S|\text{model}) = \prod_{i=1}^{n} \text{IMM}(S_i) \tag{1}$$

Here S_i is the base at the ith position in sequence S and n is the length of the sequence we are evaluating.

To assess the probability of seeing a given base in a particular position, our IMM interpolates between a 0th-order Markov model and a 1st-order Markov model.

$$\text{IMM}(S_i) = \frac{\lambda_{i-1,1}(S_{i-1}) \Pr_{i,1}(S_i) + \lambda_{i,0} \Pr_{i,0}(S_i)}{\lambda_{i-1,1}(S_{i-1}) + \lambda_{i,0}} \tag{2}$$

The notation $\Pr_{i,1}(S_i)$ represents the probability of seeing base S_i at the ith position under a 1st-order model, and $\Pr_{i,0}(S_i)$ represents the same under a 0th-order model. Whereas the 0th-order model simply represents the marginal probabilities of seeing each base at the given position, the 1st-order model represents the conditional probabilities of seeing each base given the previous base in the sequence S_{i-1}.

$$\Pr_{i,0}(S_i) = \Pr_{i}(S_i) \qquad \Pr_{i,1}(S_i) = \Pr_{i}(S_i|S_{i-1}) \tag{3}$$

We use a simple scheme to set the values of the λ parameters, which represent the amount of weight we give each model being interpolated. For all positions i, $\lambda_{i,0}$ is set to 1. The parameter $\lambda_{i-1,1}(S_{i-1})$ is set to 1 if the training data included at least m occurrences of the base S_{i-1} at position $i-1$, and is set to 0 otherwise. The intuition here is that we trust a 1st-order probability only if we had sufficient data from which to estimate it. In all of our experiments, we set m to 40.

Once we have induced our promoter and terminator IMMs, we can use them to look for instances of these two signals in the neighborhood of a candidate operon. We do this by "scanning" the promoter model along the 300 bases immediately preceding the first gene in a candidate operon, and similarly by scanning the terminator model along the 300 bases immediately following the last gene in the candidate. We then characterize a candidate operon by two features: the promoter feature is the strongest predicted promoter we find upstream from the candidate, and the terminator feature is the strongest predicted terminator we find downstream.

3.4 Gene Expression Features

Recent *microarray* technology enables scientists to measure the activity level of thousands of genes under various experimental conditions. The Wisconsin *E. coli* Genome Project has begun generating such data, and for the work presented in this paper we use data from the first 39 experiments. We have two measurements for each gene in each experiment: intensity values representing the relative amount of RNA produced by the gene under some experimental condition versus the relative amount under some baseline condition. We refer to these two sets of measurements from a given array as the two *channels* of the array. From this data, we compute two sets of features for our learning algorithm. The first set of features is based on the ratios of the two measurements, and the second set is based on the raw measurements themselves.

For the features based on ratios, we associate with each gene g_k a vector $\vec{r}_k$ of length m, where m is the number of experiments and the elements of the vector are the measured ratios for the gene. Since the genes within an operon are coordinately controlled, we expect the expression vectors of two such genes to be more correlated than the expression vectors of two randomly chosen genes. Hence, the first set of features we use for classifying candidate operons are based on pairwise correlations between the expression vectors of genes of interest. Specifically, we use three features to characterize a candidate operon c: (i) the mean correlation of every pair of genes in c, (ii) the correlation between the first gene in c and the previous gene in the sequence, (iii) the correlation between the last gene in c and the next gene in the sequence.

The ratio correlation measure is appropriate to identify genes that have similar expression profiles. However, we expect an even stronger association among genes that are in the same operon. Since these genes are transcribed together, the *absolute* amount of RNA produced should be similar. Consequently, we also derive features from the raw measurements.

For these features, we treat each channel as a separate experiment. Our approach is based on the model that each intensity value is the result of some true underlying signal plus normally distributed random noise. Let $\vec{a}_{cj}$ be a vector of expression values from the jth experiment for the genes in candidate operon c. If c actually is an operon, the likelihood of the observed measurements $\vec{a}_{cj}$ is given by:

$$\Pr(\vec{a}_{cj}|O = \text{ true}) \propto \prod_i N(\vec{a}_{cj}(i), \mu_{cj}, \sigma_{cj}). \quad (4)$$

Here O is the random variable indicating whether or not the candidate c is an operon, $\vec{a}_{cj}(i)$ represents the intensity value of the ith gene in the candidate for the jth experiment, and $N(\vec{a}_{cj}(i), \mu_{cj}, \sigma_{cj})$ represents the density value of $\vec{a}_{cj}(i)$ under a normal distribution with parameters μ_{cj} and σ_{cj}. In our model, μ_{cj} represents the true signal for the operon in this experiment, and σ_{cj} represents the standard deviation of the noise for the given operon and experiment. Our approach estimates μ_{cj} by the mean of the values in the jth experiment for the genes in the candidate. The parameter σ_{cj} is estimated from the training set operons by fitting a linear function in (μ_j, σ_j) space, where each data point in this space is determined from a known operon in the training set (singletons are ignored).

To assess a candidate operon we compute the following likelihood ratio in which we consider all experiments and treat them as independent of one another:

$$L(c) = \frac{\prod_j \Pr(\vec{a}_{cj}|O = \text{ true})}{\prod_j \Pr(\vec{a}_{cj})}. \quad (5)$$

Here $\Pr(\vec{a}_{cj})$ is the marginal probability of seeing the observed intensity values in the jth experiment. It is modeled by assuming that the distribution of intensity values in a given experiment is exponential. We determine the parameter of this exponential distribution using a maximum likelihood estimate.

Using this approach, we calculate three features for a candidate operon c: (i) the likelihood ratio $L(c)$ for all of the genes c, (ii) the likelihood ratio for the first gene in c and the previous gene in the sequence, and (iii) the likelihood ratio for the last gene in c and the next gene in the sequence. Collectively, we refer to all six of the features described in this section (the three ratio-based features and the three absolute features) as the expression data features.

4. Negative Examples

An interesting aspect of our learning task is that we do not have a set of known non-operons. The nature of scientific inquiry is such that biologists have identified several hundred operons in *E. coli*, but they have not focused attention on identifying sequences of genes that *do not* constitute operons.

We are able, however, to assemble a set of 6633 putative non-operons by exploiting the fact that operons rarely overlap with each other. Given this rule, we generate a set of negative examples by enumerating every sequence of consecutive genes, from the same strand, that overlaps, but does not coincide with a known operon. We know that some of these generated non-operons are actually true operons, because operons do overlap in some cases. However, the probability of this is small and our learning algorithms are robust in the presence of noisy data.

5. Machine Learning Algorithms

Given the representation of candidate operons presented in the preceding section, we could employ a number of supervised learning algorithms to induce a model for predicting whether a candidate operon really is an operon or not. In our work to date we have primarily used naive Bayes for this task. In this section, we describe the specifics how we apply naive Bayes. In the empirical evaluation presented in the following section, we also evaluate the predictive accuracy of C5.0 (Quinlan, 1993; Quinlan, 1999).

Given a candidate operon – any consecutive sequence of genes on the same strand – we would like to estimate the probability that the candidate is an operon given the available data. Using naive Bayes, we can determine this as follows:

$$\Pr(O|D) = \frac{\Pr(O)\Pr(D|O)}{\Pr(D)} \approx \frac{\Pr(O)\prod_i \Pr(D_i|O)}{\Pr(D)} \quad (6)$$

where O is a random variable indicating whether or not a candidate is an actual operon, D represents the data available to make our determination, and D_i is the ith feature. The heart of the task is to estimate the likelihood of the data of interest given the two possible outcomes for O.

All of our features provide numeric characterizations of candidate operons. To represent the conditional distribution of each feature given the class, we use a histogram approach. The first step of this procedure is to choose the "cutpoints" that define the bins. This procedure selects the bin boundaries such that each bin contains about 150 training examples (pooling positive and negative examples). Should some bin have no examples of one class, we assume 0.5 examples of that class fell into that bin. This smoothing method avoids zero-valued probabilities which cause Bayes' rule to produce zero as its estimated probability.

6. Empirical Evaluation

We run a 10-fold cross-validation experiment with a data set consisting of 365 known operons and 6633 sequences of genes thought not to be operons.[2] There are several questions that we want to answer: What level of accuracy can we achieve with our approach? How does the accuracy of naive Bayes compare with C5.0? What is the predictive value of individual features?

To address the first two questions, we compare naive Bayes to C5.0 using the same partitioning of the data for cross validation. We give C5.0 essentially the same feature representation as naive Bayes, and set all parameters of the algorithm to their default settings. We treat our naive Bayes model as a classifier; if the posterior probability of a candidate operon is greater than 0.5, we classify it as an operon. Table 1 shows the overall accuracy rates for the two learners as well as the false positive and true positive rates. The false positive rate is defined as $\frac{FP}{FP+TN}$, and the true positive rate is defined as $\frac{TP}{TP+FN}$. As the table indicates, the two classifiers, in their present configurations, tend to make different types of mistakes.

An interesting question is how does the accuracy of our predictions vary as we raise or lower the threshold at which we classify a candidate as an operon? Figure 3 shows the resulting ROC curve as we vary this threshold for our naive Bayes model. This curve is informative since, unlike the overall accuracy numbers in Table 1, it does not depend on the class distribution (this is a property of ROC curves). In practice, we do not know what the class distribution of operons vs. non-operons is. Also, depending on how we use our operon predictions we may want to associate different costs with the different types of mistakes the classifier can make. Figure 3 also shows the ROC curve for C5.0; we generate this curve by varying misclassification costs when running C5.0. This figure also suggests, that overall, the predictive accuracy of naive Bayes is about the same as C5.0.

Table 1. Predictive accuracy for the 10-fold cross-validation experiment.

method	accuracy (%)	FP rate (%)	TP rate (%)
Bayes	90.0	8.9	74.5
C5.0	94.5	2.3	48.9

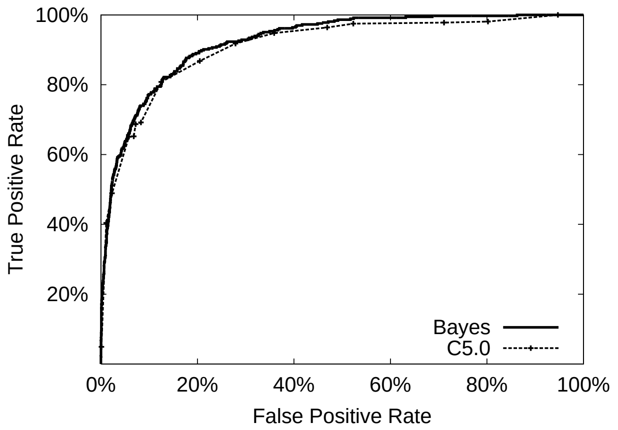

Figure 3. ROC curves for naive Bayes and C5.0.

We perform two experiments to evaluate the contribution of our individual features to the predictive accuracy of our probabilistic model. In the first experiment, we consider making our predictions *using only* a single feature, or a small group of closely related features. We collectively refer to both of these cases as "feature groups." In the second experiment, we learn models that *leave out* a single group. Figure 4 shows the resulting ROC curves for the models that consist of single feature groups, and Figure 5 shows the ROC curves for the models that leave one feature group out.

Figures 4 and 5 illustrate three interesting points. First, the features vary quite a bit in their predictive accuracy. The terminator feature, for example, seems to have negligible predictive value, whereas the functional annotation, neighboring genes, expression data and promoter features carry quite a bit of predictive in-

[2]Of the 6633 non-operons, only 5145 actually appear in some test set. The reason for this is subtle. We randomly distribute the known operons into the 10 test sets. For each known operon, we assign all of the overlapping negative examples to the same test set *except* for those examples that *also* overlap a known operon in the corresponding *training* set. This process ensures that no test-set example overlaps an example in the corresponding training set. Since non-operons can overlap multiple known operons, this process leads to 1488 non-operons not appearing in any test set.

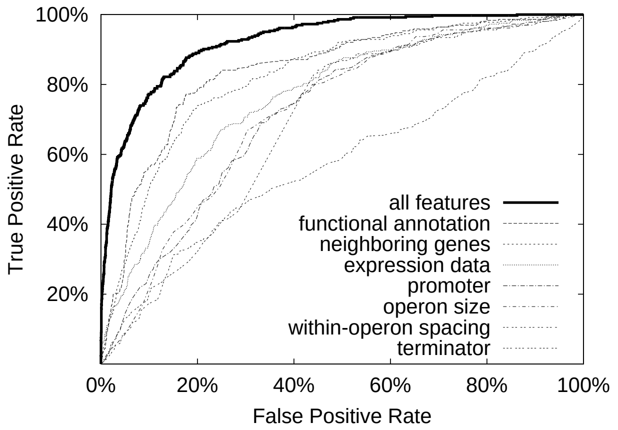

Figure 4. ROC curves for predictions made using individual feature groups. The reader should notice that the order of listings in the key reflects the order of the curves.

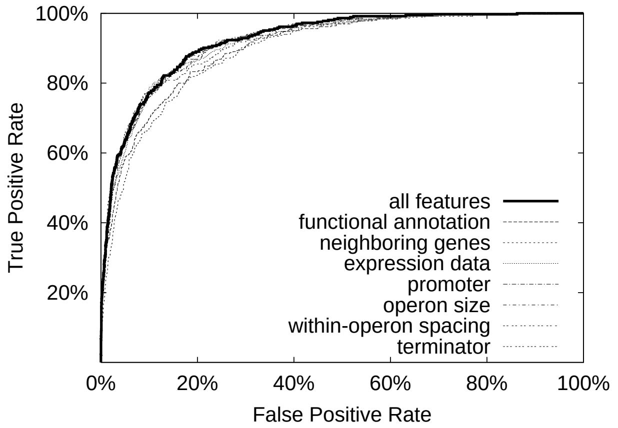

Figure 5. ROC curves for predictions made *leaving out* one of the individual feature groups. The two curves that stand out (from lowest to highest) are those leaving out the neighboring genes and functional annotation feature groups.

formation. Interestingly, these four feature groups are based on quite different types of data. Second, none of the individual feature groups is nearly as predictive as the model as a whole. This result indicates the value of combining evidence from a variety of sources. Third, as Figure 5 illustrates, the model that employs all of the features is not overly reliant on any single feature. Most of the leave-one-out curves are close to the all-features curve.

7. Discussion and Conclusions

Our research to date in predicting operons has broached several issues and lessons that should be of interest to machine learning researchers.

1. Biased features. Our experiments showed that one of the most predictive feature groups is the functional annotation features. However, we suspect that the predictive value of these features is not as high as our experiments suggest. The basic problem is that, in general, we know more about genes in known operons than we do about other genes in the genome since they have been more heavily studied. Therefore, we expect that the functional annotation values for genes in our set of known operons are more reliable and more consistent than the values for genes that are not in known operons. Unfortunately, we do not know how to estimate the extent of this bias nor correct for it.

This issue of some features being more informative for our labeled examples than they will be in general also crops up in the case of our promoter and terminator features, since the set of known promoters and terminators is concentrated around the set of known operons. However, in this case, we can ensure that our experiments are not biased. Because we are not able to represent the presence of promoters and terminators directly, we used learned models to predict them. Thus, we can ensure that our operon-prediction experiments are not biased by leaving out of the promoter (terminator) training sets those promoters (terminators) associated with test-set operons. We wonder if this general problem – some features being more informative for instances in the training set than they will be for real live test instances – is prevalent in other machine learning applications?

2. Multiple machine learning subtasks. Another lesson of our experiments is the value of addressing the learning task by decomposing it into multiple subtasks. This lesson is illustrated by the fact that the promoter features were among the most predictive. We would not have access to these features without a learned model to predict them, since the sufficient and necessary conditions of promoters are not known.

3. Naive Bayes vs. C5.0. An interesting result from our experiments is that the predictive performances of naive Bayes and C5.0 are quite comparable, as illustrated in Figure 3. Hence, this work serves as an additional data point in understanding the relative predictive accuracy of various machine learning approaches for various applications. However, we have found that naive Bayes has other advantages for this particular application. First, it outputs a probability for each candidate operon. We use these probabilities in subsequent processing in which we try to find the most optimal partitioning of the genome into operons (Craven et al., 2000). Second, naive Bayes naturally handles the situation with singleton candidate operons where we have fewer features (e.g., within-operon spacing does not apply) than we do with other candidates.

4. Absence of negative training examples. Another interesting aspect of our task is the absence of negative training examples. The nature of scientific inquiry is such that biologists have identified at least 365 operons in *E. coli*, but they have not focused attention on identifying sequences of genes that *do not* constitute operons. We were able to generate a set of putative negative examples, however, by exploiting the fact that overlapping operons are rare, and thus most gene sequences that overlap with known operons are not operons themselves. We know that there is some noise in these negative class labels, however, since there are exceptions to this general constraint.

In addition to these conclusions that are pertinent to machine learning research, we note several lessons that should be of interest to researchers who are applying learning methods in computational biology domains.

1. Value of multiple evidence sources. Our experiments indicate that when determining regulatory relationships among genes, there is value in combining evidence from various data sources. No single feature group had nearly as much predictive power as the learned model as a whole, and the three most predictive feature groups represent diverse types of data.

2. Value of augmenting expression data. A related lesson of our experiments pertains to the relative value of gene expression data. There has been much recent interest in identifying sets of related genes and discovering regulatory relationships using microarray expression data (Eisen et al., 1998; Brown et al., 1999; Friedman et al., 2000). Our results suggest a cautionary note here: we were able to obtain much more accurate operon predictions by considering other types of data in conjunction with expression data.

A fundamental challenge facing the computational biology community is determining the functions of genes in newly determined genomes, and the relationships among these genes. We argue that this task is best addressed by employing a wide array of data sources as evidence, and we believe that the work presented here represents a promising first step in this general approach. We also argue that, increasingly, the most challenging problems for machine learning involve rich, diverse sets of data, and interrelated learning tasks. Thus, we believe that our application provides an interesting case study for machine learning researchers.

Acknowledgements

This research was supported by NSF Grant IRI-9502990, NIH Grant R01 GM35682-15A1, and the Univ. of Wisconsin Graduate School. Thanks to F. Blattner, C. Kendziorski, N. Perna and C. Richmond for helpful discussions. Thanks to the anonymous reviewers for very detailed and thoughtful reviews.

References

Blattner, F. R. et al. (1997). The complete genome sequence of Escherichia coli K-12. *Science*, *277*, 1453–1474.

Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C., Ares, M., & Haussler, D. (1999). *Support vector machine classification of microarray gene expression data* (TR UCSC-CRL-99-09). Dept. of Computer Science, University of California, Santa Cruz.

Craven, M., Page, D., Shavlik, J., Bockhorst, J., & Glasner, J. (2000). A Probabilistic Learning Approach to Whole-Genome Operon Prediction. *Proc. of the 8th International Conf. on Intelligent Systems for Molecular Biology*. San Diego, CA. AAAI Press.

Eisen, M. B., Spellman, P. T., Brown, P. O., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. of the National Academy of Sciences, USA*, *95*, 14863–14868.

Friedman, N., Linial, M., Nachman, I., & Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *Proc. of the 4th Annual International Conf. on Computational Molecular Biology (RECOMB)*. Tokyo, Japan.

Fu, L., & Buchanan, B. G. (1985). Learning intermediate concepts in constructing a hierarchical knowledge base. *Proc. of the 9th International Joint Conf. on Artificial Intelligence* (pp. 659–666). Los Angeles, CA.

Jelinek, F., & Mercer, R. L. (1980). Interpolated estimation of Markov source parameters from sparse data. In E. S. Gelsema and L. N. Kanal (Eds.), *Pattern recognition in practice*, 381–397. North Holland.

O'Sullivan, J. (1998). Transferring learned knowledge in a lifelong learning mobile robot agent. *Proc. of the 7th European Workshop on Learning Robots*.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. (1999). C5.0 decision tree software. http://www.rulequest.com/.

Riley, M. (1996). E.coli gene products: Physiological functions and common ancestries. In R. Curtiss et al. (Eds.), *Escherichia coli and Salmonella*, 2118–2202. American Society for Microbiology. 2nd edition.

Salgado, H. et al. (2000). RegulonDB (version 3.0): Transcriptional regulation and operon organization in Escherichia coli K-12. *Nucleic Acids Research*, *28*, 65–67.

Salzberg, S., Delcher, A., Kasif, S., & White, O. (1998). Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, *26*, 544–548.

Shapiro, A. (1987). *Structured induction in expert systems*. Reading, MA: Addison Wesley.

Stone, P. (1998). *Layered learning in multi-agent systems*. Doctoral dissertation, School of Computer Science, Carnegie Mellon University. TR CMU-CS-98-187.