Gene Expression Classification: Decision Trees vs. SVMs

Xiaojing Yuan and **Xiaohui Yuan** and **Fan Yang** and **Jing Peng** and **Bill P. Buckles**
Electrical Engineering and Computer Science Department
Tulane University
New Orleans, Louisiana 70118, U.S.A
{yuan, yuanx, yangf, jp, buckles}@eecs.tulane.edu

Abstract

In this article, we compare decision trees (DT) and support vector machines (SVM) in classifying gene expressions. With the explosion of genome research, tremendous amount of data have been made available and a deep insight study becomes demanding. Among various kinds of gene analysis approaches being developed, sequence based gene expression classification shows the importance due to its ability to identify existence of some specific gene pieces. In this article, we focus on two major categories of classification methods, namely decision trees and support vector machines. By comparing various versions of decision tree algorithms, SVMs, and a particular SVM that integrates structural information of the gene sequence, it is shown that the structural information does help in achieving better performance with respect to the classification accuracy.

Introduction

A gene is a functional unit along the DNA of a chromosome. Nucleotide sequencing and other physical studies have shown that between genes there are sequences of DNA, where functions are mostly unknown. Therefore, classifying gene sequences is fundamental in gene expression analysis. (Mount 2001) The strong sequence similarity often implies functional and structural relationships. In particular, sequence classification algorithms can be applied to problems such as determining whether a DNA sequence is part of a gene-coding or a non-coding region, identifying the introns or exons of a eukaryotic gene sequence, predicting the secondary structure of a protein, and assigning a protein sequence to a specific protein family. Various machine learning techniques have been applied to gene expression classification,(Deshpande & Karypis 2002; Kuramochi & Karypis 2001; Lesh, Zaki, & Ogihara 1999; Ohler *et al.* 1999; Yuan *et al.* 2002) among which decision tree algorithms and support vector machines are widely recognized as the most effective and efficient approaches dealing with that huge amount of data. (Kuramochi & Karypis 2001; Wang *et al.* 1999) However, few if any studies have been conducted that compare their performance over such particular kind of data. In addition, whether the characteristics of the gene data can be employed to improve the classification performance remains an open question. In this paper, we compare the performance of various kinds of decision tree algorithms, namely traditional decision trees (such as C4.5), bagged (bagging) decision trees, and boosted (boosting) decision trees, and SVM classifier with respect to conditions such as varying training set size and different number of possible classes. The experiments show that these methods generate good classifiers and handle large amount of data very well. Although decision tree algorithms may suffer from over fitting, both bagged (bagging) and boosted (boosting) decision trees perform as well as SVMs and even better in some cases. However, due to the characteristics of gene data, a *SVM Bank* is created to integrate the structural knowledge of a gene into the classifier. The classification is determined by the weighted majority vote of SVMs from the bank and the improvement on performance is clearly shown by the experiments. The rest of this article is organized as follows. A brief review on decision trees and SVMs, as well as ensemble techniques such as bagging and boosting are given in the next section. A further extension of SVMs integrating gene structural characteristics is also described. The experiments are carried out with widely used gene data sets and followed by discussion and conclusion.

Background

Decision Tree

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. (Mitchell 1997) Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies the test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the sub-tree rooted at the new node. Some widely used algorithms include ID3, ASSISTANT, and C4.5. (Quinlan 1986; 1993) Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down and greedy search through the space of possible decision trees. The central issue is selecting which attribute to test at each node in the tree. This is determined based on measures, such as information gain, that evaluate how well a single node classifies the training examples. The *Bagging* algorithm votes classifiers built from different bootstrap samples, which are generated by uniformly sampling N replacement instances from the original training set (of size N). (Breiman 1994; 1996) The new training set has the same size as that of the original training set, but is more randomized in that some instances may not appeared while others appear more than once. For trials t, $t = \{1, 2, \ldots, T\}$, bootstrap sample sets $D_1, D_2, \ldots, D_T$ are generated and a classifier C_i is built by a learning system for each bootstrap sample set D_i. The final classifier C^* is built from $C_1, C_2, \ldots, C_T$ by majority vote, so that the result classification is the class that output most often by the sub-classifiers. *Boosting* is another ensemble technique to improve the performance of a weak learning system. A *weak learning system* is a system whose prediction accuracy is at least 50It is proven that the error rate of the boosted classifier C^* on the given examples approaches zero exponentially as number of trials T increases, and the error rate of trials is less than 0.5. (Freund & Schapire 1999) Therefore, a succession of *weak* classifiers are boosted to a *strong* classifier, which is usually more accurate than the best *weak* classifier is on the given training data.

Support Vector Machine

Support vector machines are introduced to solve two-class pattern recognition problems using the Structural Risk Minimization principle. (Vapnik 1995; 1998; Burges 1998; Cristianini & Shawe-Taylor 2000; Scholkopf, Burges, & Smola 1999) Given a training set in a vector space, SVMs find the best decision hyperplane that separates two classes. The quality of a decision hyperplane is determined by the distance (i.e. margin) between two hyperplanes defined by support vectors. The best decision hyperplane is the one that maximizes this margin. Figure 1 shows these concepts in two-dimensional feature space. By defining the hyperplane in this fashion, SVM is able to generalize to unseen instances quite effectively. SVM extends its applicability on the linearly non-separable data sets by either using soft margin hyperplanes, or by mapping the original data vectors into a higher dimensional space in which the data points are linearly separable. The mapping to higher dimensional spaces is done using appropriate kernels such as Gaussian kernel and polynomial kernel.

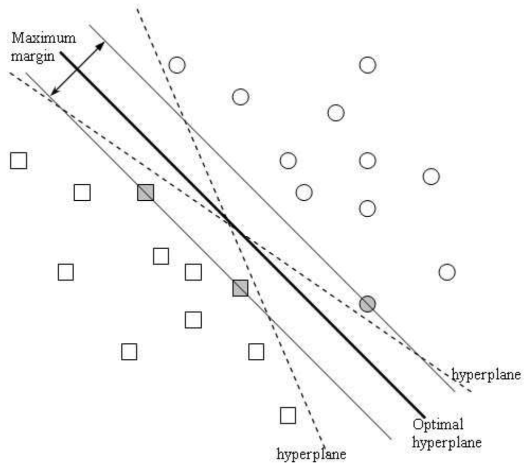

Figure 1: Classification with SVMs.

Integrating Gene Structural Information

Gene Representation

A gene can be defined as a region of the chromosomal DNA that can be transcribed into a functional RNA during development. DNA is constructed with four nitrogenous bases: adenine, guanine, cytosine, and thymine, which are referred by A, G, C, and T respectively. In turn, the gene data are most commonly arranged in sequence as illustrated below.

. . . GAAGTCCAGTTGCTACAGGCCCTGGAG
ACTGACATGCCAAAGTACCATCCGAAGGG
AGCCCCGTGCTCGGCAGCCCCGATGCTCT
AAACTAGGAAGTGGCAGCATGTCAGAC. . .

From computer science point of view, to analyze gene data, it is necessary to convert from symbolic representation to numerical representation, especially when applying methods such as neural networks and SVMs. Obviously, there are infinite ways of mappings. The question is will simple ways, such as assigning each nucleotide a two-digit number, make classification difference from using a complicated coding strategy? To study the potential differences, we compare the following two coding schemes with Hamming distance.

Scheme 1:	A = 00, C = 01, G = 10, T = 11
Scheme 2:	A = 0001, C = 0010, G = 0100, T = 1000
Sequences:	{A, A, G, T, C, C, A, G, T, T, G, C}
	{A, G, T, A, C, C, A, T, T, C, G, G}

The Hamming distance using the first coding scheme is 8, while it is 12 using the second coding scheme. It is clear that the coding scheme affects the value of similarity measurement.

Integrating Gene Structural Information into SVM

Given a DNA sequence, since there is no a priori information on how each nucleotide relates to each other, it is important to preserve the sequential structure in it. From above discussion, it is obvious that different coding schemes affect the similarity measure, thus may affect the learned classifier. This requires that when translating symbols to numerical representation, no bias should be introduced by these coding schemes. The best choice is to treat these nucleotides is to consider these bases as equally correlated individual. It is suggested that the ideal mapping should be as shown in Figure 2. (Xu & Buckles 2002) However, with any set of values, it is infeasible to guarantee an unbiased assignment. To approach this ideal model, we take into account all the possible encoding schemes and build a SVM bank. The four nucleotides, *A, C, G, T*, formed a base so that a gene sequence can be fully represented by their combinations. Thus, for gene sequence analysis, a maximum of 24 coding combinations are necessary, as shown in Table 1. These 24 coding schemes can be used for any application that can be fully described by four characteristics and their combinations, which could be a very large number. It is noted that even though coding combination may increase exponentially when the underline base increases, there are several ways to simplify the coding schemes.

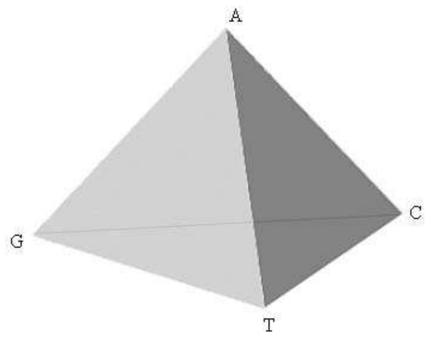

Figure 2: The ideal mapping structure. Keep symbols equally apart.

Table 1: Possible coding combinations, assuming number basis is (00 01 10 11).

Nuc.	A	G	C	T	Nuc.	A	G	C	T
1	00	01	10	11	13	00	01	11	10
2	00	10	01	11	14	00	10	11	01
3	00	11	01	10	15	00	11	10	01
4	01	00	10	11	16	01	00	11	10
5	01	10	00	01	17	01	10	01	00
6	01	11	00	10	18	01	11	10	00
7	10	00	01	11	19	10	00	11	01
8	10	01	00	11	20	10	01	11	00
9	10	11	00	01	21	10	11	01	00
10	11	00	01	10	22	11	00	10	01
11	11	01	00	10	23	11	01	10	00
12	11	10	00	01	24	11	10	01	00

Using different coding schemes, 24 support vector machines are built from training data, $S = \{s_1, s_2, \dots, s_{24}\}$, which can be considered as a *SVM Bank* designed for gene expression analysis. To predict an unlabelled sequence, each SVM, s_i ($i = \{1, 2, \dots, 24\}$) is applied and gives its classification result p_i. The final judgment is determined by the majority vote on all predictions. In addition, weights, $W = \{w_1, w_2, \dots, w_{24}\}$, are assigned to SVMs according to their accuracy during prediction. Initially, an equal weight is assigned to each SVM (Here, $w_i = 1$ is used in our experiments.) Each weight is dynamically updated with adjustment δ based on the correctness of each prediction. Let the final classification on a unlabeled instance be p^*.

$$w_i = w_i + \phi(s_i) \times \delta$$

and
$$\phi(s_i) = \begin{cases} 1 & p_i = p^* \\ -1 & \text{otherwise} \end{cases}$$

For reference purpose, we call such SVM as Gene Structural SVM (GS-SVM).

Experiments

In this section, the experiments on the classification of two gene sequence data sets, namely ALU gene and splice junctions, are performed and followed by the comparison and discussion.

Data Set Description

Two widely used gene sequence analysis sets, ALU gene sequences and splice junction sequence, are used as test beds, which are available for download at (ALU 2002; Blake & Merz 1998). (Jurka *et al.* 1993; Wang *et al.* 1999) ALU gene is the most abundant single sequence in the entire human genome. The data set we use has 327 positive instances which are downloaded from (ALU 2002; Blake & Merz 1998) and 654 negative instances. These negative instances were generated randomly while keeping the nucleotide frequencies unchanged. (Wang *et al.* 1999) The sequence length varies from 69bp to 379bp (where bp stands for base pair, a biological metric for gene length.) Splice junctions (SJ) are sections on a DNA sequence at which *superfluous* DNA is removed during the process of protein creation in higher level organisms. There exist two forms of boundaries: (1) the transition from intron to exon, called *acceptor*, and (2) the transition from exon to intron, called *donor*. All the sequences are in length of 60bp, among which 767 sequences contain acceptor, 768 sequences contain donor, and 1655 sequences contain neither of them. In all experiments, training sets are randomly selected from the original data set without replacement according to specified percentage. The remaining data will be used as testing set. Table 2 summarizes the properties of the experimental data sets.

Table 2: Properties of the experimental data sets.

Set	Cases	Classes	Positive	Negative
ALU	981	2	327	654

Set	Cases	Classes	Positive		Neg.
			Acceptor	Donor	
SJ	3190	3	767	768	1655

Experimental Setup

Both bagging and boosting decision trees are developed by extending C4.5 algorithm. (Quinlan 1993) The implementation of SVM for our experiments is developed by S. R. Gunn. (Gunn 1998) Considering possible errors made during data collection stage, a soft margin SVM is used.

The kernel we employed is a 6th order polynomial function. For both Bagging and Boosting, the parameter T governing the number of classifiers generated was set at ten. This is based on the results from (Breiman 1994; 1996) that most of the performance improvement from bagging is evident within the first ten trials. Ten complete tests were carried out for each training set. In each classification experiment, Training sets are randomly selected from original data set and the training set size range from 10% to 90% of original data set. The corresponding remaining data sets are used as testing set. All parameters of C4.5 maintained their default values, and the pruned trees were used for decision trees. However, there are several issues regarding the choice of the training set. First, extreme situations, such as all training instances are negative or positive examples, will compromise learning the classifier. Second, margin samples may missed from training set. To minimize the possible misleading caused by the training data, ten experiments were conducted for each experiment using same data sets and parameters. The average of the prediction accuracy is used in the performance comparison. Three sets of experiments were done to systematically compare the classification performance of different ensemble methods (bagging or boosting), different encoding methods (symbolic or gene-structural based), and in general, different learning methods (decision trees or SVMs).

Results and Discussion

Figure 3 illustrates the classification performance with splice junctions sequence. Without considering gene structural information, the accuracy of decision tree algorithms and SVM algorithm are very close, especially when using less than 50% data as training set. In some cases, bagging and boosting are better than SVMs. When increasing training set size, the accuracy of SVMs continues to improve while decision tree based classifiers are less satisfactory for little improvement and they show degradation due to over fitting. Also, it is shown that by incorporating gene structural information, GS-SVM classifiers gain improvement for all population sizes, where it gives the best performance than any of others most of time.

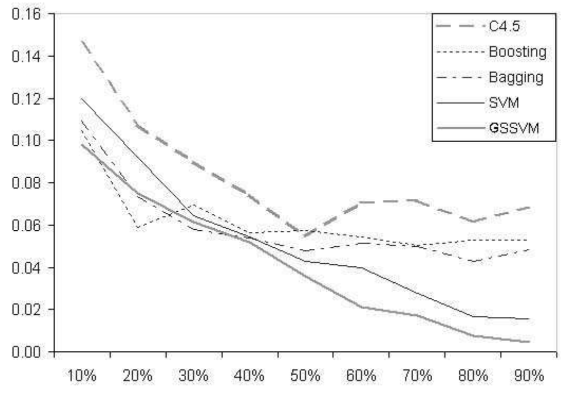

Figure 3: Classification performance with splice junctions sequence.

Figure 4 illustrates the classification performance with ALU gene sequence data. It is clear that the gene structural information plays an important role in learning the classifier and an average accuracy gain is over 70%.

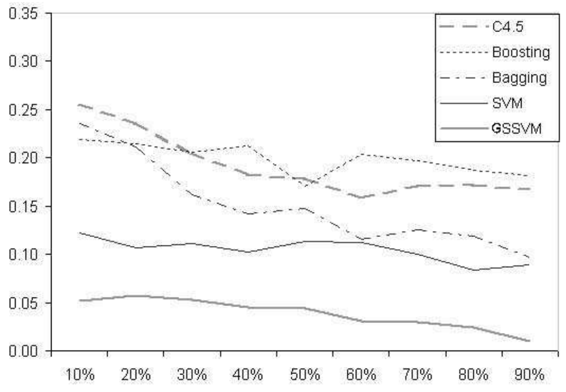

Figure 4: Classification performance for ALU gene sequence.

Table 3 lists the error rate improvement of GS-SVM in terms of Average Error Rate (AER) of decision trees and SVMs and in terms of average error rate of SVMs. The second column lists the classification error rate of GS-SVM. The next two columns are average error rate and the improvement of GS-SVM compared to AER. The last two columns show the error rate of SVM classifier and GS-SVM's performance gain.

Table 3: Error rate reduction of GS-SVM.
Splice junctions sequence

TDR	GS-SVM	AER	Impro.	SVM	Impro.
10%	0.0980	0.1200	18%	0.1200	18%
20%	0.0750	0.0829	10%	0.0921	19%
30%	0.0616	0.0703	12%	0.0643	4%
40%	0.0519	0.0598	13%	0.0546	5%
50%	0.0355	0.0507	30%	0.0428	17%
60%	0.0211	0.0539	61%	0.0396	47%
70%	0.0169	0.0498	66%	0.0276	39%
80%	0.0074	0.0433	83%	0.0164	55%
90%	0.0046	0.0463	90%	0.0158	71%

ALU gene sequence

TDR	GS-SVM	AER	Impro.	SVM	Impro.
10%	0.0522	0.2081	75%	0.1226	57%
20%	0.0572	0.1921	70%	0.1066	46%
30%	0.0531	0.1705	69%	0.1107	52%
40%	0.0448	0.1600	72%	0.1025	56%
50%	0.0440	0.1523	71%	0.1128	61%
60%	0.0305	0.1473	79%	0.1120	73%
70%	0.0300	0.1482	80%	0.1000	70%
80%	0.0247	0.1404	82%	0.0839	71%
90%	0.0101	0.1337	92%	0.0893	89%

The performance improvement of GS-SVM relies on the fact that although there are some co-occurrences of nucleotides to form a gene, little knowledge is present. By assuming an equal correlation between nucleotides, given a coding base, 24 identical combinations exist. Obviously, encoding scheme changes the information distribution, i.e. the co-occurrence of basic representative units. In addition, genes have their own features, or we can say, the regularity of both occurrence of nucleotides and spatial relationship determine if it is a gene or not and which gene it is. Encoding nucleotides is essentially mapping the sequence to a different space. This process may increase the discrimination, within one encoding scheme, or among several encoding schemes. Also, which encoding schemes play a more important role varies from gene to gene. Therefore, a weighted majority voting circumvents this by learning the weights along predictions.

Conclusions

In this article, we study the performance of the classification of gene sequence data with decision trees and SVMs. From our experiments, it is clearly shown that these methods generate good classifiers and handle large amount of data very well. Although decision tree algorithms suffer from over fitting difficulty, both bagging and boosting decision trees perform as well as or close to SVM and even better in some cases. However, due the data pre-processing requirements, i.e. converting symbolic to numeric representation, bias is inevitable. To circumvent this, a weighted voting scheme using SVM Bank is described, which is inspired by the gene structural characteristics. A 24-SVM bank is learned from data sets independently and a voting weight is assigned to each SVM and updated according to the classification performance on each prediction. The promising results with experimental data suggest that incorporating structural information greatly improves the classifier.

Acknowledgment

We would like to thank Ibrahim Gokcen and Jian Zhang for discussions and insights on preparing this article. We would also like to thank Zujia Xu and Kun Zhang for their help on developing software.

References

ALU. 2002. Alu gene data. ftp://ncbi.nlm.nih.gov/ pub/ jmc/ alu/ ALU.327.dna.ref.

Blake, C. L., and Merz, C. J. 1998. Uci: Repository of machine learning databases.

Breiman, L. 1994. Heuristic of instability in model selection. Technical report, Statistics Department, University of California.

Breiman, L. 1996. Bagging predictors. *Machine Learning* 24:123–140.

Burges, C. J. C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2):121–167.

Cristianini, N., and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machine*. Cambridge University Press.

Deshpande, M., and Karypis, G. 2002. Evaluation of techniques for classifying biological sequences. In *Proceedings of the 2002 Pacific Asia Conference on Knowledge Discovery and Data Mining*, 417–431.

Freund, Y., and Schapire, R. E. 1999. A short introduction to boosting. *Journal of Japaneses Society for Artificial Intelligence* 14(5):771–780.

Gunn, S. R. 1998. Support vector machines for classification and regression. Technical report, University of Southampton.

Jurka, J.; Kaplan, D. J.; Duncan, C. H.; Walichiewicz, J.; Milosavljevic, A.; Murali, G.; and Solus, J. F. 1993. Identification and characterization of new human medium reiteration frequency repeats. *Nucleic Acids Res.* 1273–1279.

Kuramochi, M., and Karypis, G. 2001. Gene classification using expression profiles: A feasibility study. In *IEEE International Conference on Data Mining*, 191–200.

Lesh, N.; Zaki, M. J.; and Ogihara, M. 1999. Mining features for sequence classification. In *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 342–346.

Mitchell, T. M. 1997. *Machine Learning*. McGraw-Hill Companies, Inc.

Mount, D. W. 2001. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press.

Ohler, U.; Harbeck, S.; Niemann, H.; Noth, E.; and Reese, M. G. 1999. Interpolated markov chains for eukaryotic promoter recognition. *Bioinformatics* 15(5).

Quinlan, J. R. 1986. Induction of decision trees. *Maching Learning* 1(1):81–106.

Quinlan, J. R. 1993. *C4.5.: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Scholkopf, C.; Burges, J.; and Smola, A. 1999. *Advances in Kernel methods: Support Vector Learning*. MIT Press.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. New York: Springer Verlag.

Vapnik, V. 1998. *Statistical Learning Theory*. New York: Wiley.

Wang, J. T. L.; Rozen, S.; Shapiro, B. A.; Shasha, D.; Wang, Z.; and Yin, M. 1999. New techniques for dna sequence classification. *Journal of Computational Biology* 6(2):209–218.

Xu, Z., and Buckles, B. 2002. Dna sequence classification by using support vector machine. EECS, Tulane University.

Yuan, X.; Buckles, B. P.; Yuan, Z.; and Zhang, J. 2002. Mining negative association rules. In *Proceedings of the 7th IEEE Symposium on Computers and Communications*.