Using Classification for Gene Function Determination

Jeffrey Bigham
CSE 527
jbigham@cs.washington.edu

December 18, 2003

1 Introduction

In the recent past the genome sciences have focused largely on the development of methods for sequencing genomes and concentrated much effort on performing the sequencing itself. For many species, however, the sequencing has now been completed and the field has shifted to include the classification of the sequenced genes in regard to their functional classes. Because of the vast scale of genome information it would be preferrable to hand-classify only a small percentage of the total genes and use those classifications and some computational methods to arrive at the classifications of the other genes.

The genome data itself has many features rather unique to this domain which make this classification task particular difficult. Even gather the data inot one resource is though for it is usually spread out among numerous data resources and must be joined together. The number of examples available in public resources is also somewhat scarce and the number of attributes in each of those examples is quite high. Finally, the data values themselves are quite heterogeneous and likely needs to be normalized before many machine learning methods can work effectively on it.

Many methods for machine classification are readily available but which of these has the most potential in this domain is not immediately apparent. The peculiar aspects of the data as described in the previous paragraph make this data an exciting collection on which to test classification algorithms. Also, given its unique nature it seems reasonable to try many different methods because each will have its own associated tradeoffs. This project attempts to explore those tradeoffs by first giving a brief overview of various approaches to classification and then testing implementations of those algorithms on real data.

2 Approaches

This project attempts to compare the relative performance of several popular classification algorithms on gene function prediction tasks. The algorithms that were chosen for implementation are described below. While obviously not a completely exhaustive list of all useful classification algorithms, the implemented classifiers include a number of the most popular and best-performing classifiers available both experimentally and theoretically.

2.1 *k*-Nearest Neighbor

The Nearest Neighbor algorithm [5] is perhaps the most natural and intuitive classifier. Given a set of training examples whose classifications are known, new examples are simply assigned to be in the class to which they are closest. “Closest” in this context is defined based on a distance metric defined by the client. In the case of gene expression data formulating on appropriate distance measure can be difficult although extremely importnant because the heterogeneity of the data. For example because the numbers on the chromosonal distance are much greater in absolute terms than those of the microarray expression data, using a simple Euclidean distance measure will result in examples being much further apart than they otherwise would be. In the experiments presented later this possible source of error is counteracted by rather naively scaling the different data sources so they are approximately of the same scale.

The *k*-Nearest Neighbor algorithm is a simple extension of the NN algorithm that simply allows the *k* neighbors closest to a new example to vote on what its classification should be. This helps counteract noisy data although introduces obvious problems in the case that the class sizes are not homogeneous.

2.2 Decision Trees

In its basic form, a decision tree classifier essentially performs the same logical procedure as a disjunction of conjunctions over a set of tests using using one attribute per level. While one could imagine many ways these tests could be performed, if the attribute is continuous the test usually takes the form of a threshold to distinguish two branches and if it is *n*-way discrete then each discrete value or set of values is assigned to a branch. Because decision trees are able to break the space down into arbitrarily sized hyper-rectangles they have a tendency to overfit, but fortunately techniques that work rather well have been developed to prune decision trees. A popular way of doing this is to hold out a subset of the training data and build the tree so it completely classifies the rest and then prune away any paths on the tree that lead to incorrect answers on the hold-out set.

The most popular implementation of the decision tree algorithm is called C4.5 and it has been shown to be quite good at both constructing and pruning decision trees [5]. “Decision tree” is used throughout this paper to mean C4.5.

2.3 Boosting

Boosting[2] is one version of what is called ensemble learning and works by basically constructing a number of different experts and then asking them to vote to decide which class a particular example should belong to. It is fairly easy to prove that if the experts have independent errors and if each is correct even slightly greater than half the time that given enough experts you should expect to get the answer correct with an arbitrarily high probability approaching 1. Unfortunately the independent error hypothesis is difficult to reach in real life where some examples are simply "harder" to classify (and therefore are more likely to be incorrectly labelled by all experts) than others and the underlying learners we have to serve as experts come from a very limited pool. Boosting attempts to construct such independent experts by essentially iteratively creating new experts trained on harder and harder problems constructed by sampling from the training set giving higher preference to examples missclassified by the current collective hypothesis. It has been shown that our guarantee of asymptotically decreasing error holds in this framework given basic assumptions and that it somewhat surprisingly tends to avoid overfitting.

Dettling and Buhlmann presented several ehancements to the regular boosting algorithm [2] that made it more robust for use in multi-class problems as well as adding a feature preselection model that helped the boosting algorithm deal with the large numbers of features combined with relatively few examples which are common in biological data. This seems similar to what Zenzen Yao did in her quals project [7].

In general extending the boosting algorithm to multi-class problems presents difficulty. One of the original boosting algorithms, AdaBoost.MH, is capable of multi-class problems and usually shows good performance. Unfortunately, the time required for training increases by a factor equal to the number of classes. Another version of multi-class boosting constructs binary learners based on random partitions of the data and interprets this sequence of codes as an error-correcting code. The implementation portion of this project uses AdaBoost.MH.

2.4 Support Vector Machines

SVMs (Support vector machines) [1,6] are a relatively new method for machine learning developed by Vapnik. They provide an efficient method to find the "best" hyperplane separating examples with m features in an m-dimensional space - that is how to maximize the minimum distance from any point in the training set to the separating hyperplane. Most real-world problems that one would like to classify have examples that cannot be readily separated straightforwardly in this manner, but if the input space is mapped to a higher-dimensional feature space this is often made possible. Perhaps the most powerful feature of SVMs is that the input space can be mapped to a higher-dimensional feature space in which the examples are linearly separable. Through the clever use of kernel functions the "best" hyperplane in the feature space can be computed using the input space which helps preserve the process' efficiency.

The basic SVM algorithm only does binary classification, but in real world cases such as gene functional classification as with boosting we are interested in how it can be extended to deal with multiple classes. The most straightforward way of doing this involves simply creating *m* binary learning problems for each of *m* classes in a one-vs-all fashion although this has associated problems such as in the case when an example is labeled to more than one class (unless examples can have multiple labels). The other method involves expanding the optimization problem to consider all of the classes at once. The latter is not that practical for large example spaces but the former has been shown to perform quite well and is the basis for what is explored here.

2.5 Adaptive Decision Tree

The Adaptive Decision Tree algorithm [3] is a classification method that combines the power of decision trees with some of the theoretical performance guarantees of boosting. These trees are constructed during the interative rounds of the boosting procedure and at each round three nodes are added to a growing tree. One node is a splitter node that attempts to split the training examples up to that point into two pure subsets and the others are prediction nodes corresponding to each of the predictor's subsets. The exact point that signals a split in the splitter node is determined by setting it equal to the value that most improves the prediction. The tree structure representation gives ADTrees the nice property of being able to be merged together which is a useful addition when one wants to construct a multi-class classifier from the binary classifiers that the basic algorithm produces.

3 Similar Work

Determining gene function based on data such as microarray expression levels is not a new idea and has been previously studied quite extensively. Kuramochi and Karypis experimented with functional prediction of yeast genes based on microarray expression level data and found that kNN-based classifiers performed better than a SVM-based classifier while Brown et al. found that SVM's performed better than decision trees and two statistical techniques called Parzen Windows and Fisher's Linear Discriminant again using yeast gene data. Dettling and Buhlmann meanwhile studied various improvements to the boosting algorithm for this sort of data and showed that boosting generally performed reasonably well and with their additions performed even better. Their main alteration to the boosting algorithm was a mechanism to preselect important features. There seems to be no study that pits the major classification algorithms against one another head to head on this type of data which provides further motivation for such a study because it seems rather unclear which will perform the best. Yao explored the k-NN algorithm and variuos improvements and approaches to combining heterogeneous data. Here normalized data forms the basis for the experiments presented next.

Algorithm	Description
kNN	Implemented by Jeffrey Bigham. Written in Perl with k=5 and the distance measure a simple Euclidean distance measure that is weighted to prevent the chromosomal distance from having too much influence
SVM	BSVM, a general multi-class boosting implementation developed by Chih-Wei Hsu and Chih-Jen Lin, National Taiwan University
Boosting	XBoost, developed by Manoj Prabhakaran and Jeffrey Bigham at Princeton University
D-Tree	Standard C4.5 Implementation
ADTree	An implementation of Freund & Mason Alternating Decision Tree Algorithm, extended to multilabel classification.

Figure 1: Descriptions of each tested algorithm.

	KEGG	COG	Multi-Fun
kNN	55.4%	42.2%	63.2%
SVM	32.0%	33.5%	52.0%
Boosting	32.9%	18.3%	40.4%
D-Tree	38.9%	17.4%	31.1%
ADTree	31.7%	18.5%	35.9%

Figure 2: Accuracy achieved per data set per algorithm.

4 Experiment

For the experimental portion of this paper each of the algorithms discussed above was applied to the three data sets from [7]. The first data set is derived from KEGG (Kyoto Encylcopadia of Genes and Genomes) and contains 18 functional classes. The second data set, COG (Clusters of Orthologous Genes), also has 18 functional classes. The final data set, MultiFun (Multidimensional Functional Classification Scheme), has 8 functional classes. For these experiments, both the paralog indicator and the block indicator were eliminated leaving 106 attributes obtained by normalizing the expression data associated with each gene and 1 attribute representing the chromosomal position of the gene.

The algorithms were either freely available implementations found online or home-grown implementations. Information describing the particular implementation of each algorithm can be found in Figure 1.

The results of the experiments I ran are presented in Figure 2. In general the k-NN algorithm performed the best, followed by the SVM algorithm, with all of the others trailing by approximately the same amount.

5 Analysis and Conclusion

Both in the experiments presented here and also in the available literature kNN classifiers outperform all others. The unique characteristics of the gene data seem to suggest an intuitive reason why this might be the case. It seems reasonable to think that the performance of many of the classification algorithms was hampered by the fact that they were given relatively few examples and that each example had a large number of attributes. The algorithms used above usually make a decision based on a single attribute at a time. The exceptions to this are of course kNN and SVM which intuitively seems to go toward explaining why this might be the case.

This area certainly seems to lend itself to more experimentation with classification algorithms taylored to the peculiarities of this particular classification task because decent performance was achieved with straightforward implementation of the algorithms. Something better suited for the type of data this algorithm may show some promise.

6 References

(1) Brown, Michael P.S., William Noble Grundy, David Lin, Nello Christianini, Charles Sugnet, Manuel Ares, Jr., David Haussler. "Support Vector Machine Classification of Microarray Gene Expression Data." UCSC-CRL-99-09, 1999.

(2) Dettling, Marcel and Peter Buhlmann. "Boosting for tumor classification with gene expression data." Bioinformatics, Vol 19, 9 November 2003.

(3) Holmes, Geoffrey, Bernhard Pfahringer, Richard Kirkby, Eibe Frank, and Mark Hall. "Multiclass Alternating Decision Trees." University of Waikato, New Zealand 2002.

(4) Kuramochi, Michihiro and George Karypis. "Gene Classification using Expression Profiles: A Feasibility Study." Techical Report 01-029. University of Minnesota, Minneapolis, MN 2001.

(5) Russell, Stuart and Peter Norvig. Artificial Intelligence: A Modern Approach. Pearson Education, Inc. Upper Saddle River, NJ 2003.

(6) Santosa, Budi, Tyrrell Conway, and T.B. Trafalis. "Knowledge Base-Clustering and Application of Multi-Class SVM for Genes Expression Analysis." University of Oklahoma.

(7) Yao, Zizhen. "Gene Functional Prediction of Escheriachia coli based on heterogeneous data sources." UW CSE Quals Talk Presented in CSE 527. November 2003.