

An Analysis of Neural Network Versus Decision Tree Performance on a Bio-Informatics Problem¹

Lawrence O. Hall, Xiaomei Liu², Kevin W. Bowyer², Robert Banfield
Department of Computer Science & Engineering

University of South Florida

Tampa, Florida 33620-5399

² Computer Science & Engineering

384 Fitzpatrick Hall

Notre Dame, IN 46556

{hall,banfield}@csee.usf.edu, {xliu5,kwb}@cse.nd.edu

Abstract

Bio informatics data sets may be large in the number of examples and/or the number of features. Predicting the secondary structure of proteins from amino acid sequences is one example of high dimensional data for which large training sets exist. The data from the KDD Cup 2001 on the binding of compounds to thrombin is another example of a very high dimensional data set. This type of data set can require significant computing resources to train a neural network. In general, decision trees will require much less training time than neural networks. There have been a number of studies on the advantages of decision trees relative to neural networks for specific data sets. There are often statistically significant, though typically not very large, differences. Here, we examine one case in which a neural network greatly outperforms a decision tree; predicting the secondary structure of proteins. The hypothesis that the neural network learns important features of the data through its hidden units is explored by using a neural network to transform data for decision tree training. Experiments show that this explains some of the performance difference, but not all. Ensembles of decision trees are compared with a single neural network. It is our conclusion that the problem of protein secondary structure prediction exhibits some characteristics that are fundamentally better exploited by a neural network model.

1 Introduction

Neural networks have been compared with other machine learning algorithms in various ways previously [1, 2, 3, 4, 5, 6]. In this paper, we compare feedforward back propagation neural networks with decision tree learning on a well-known, public bio-informatics data set. Bio-informatics data sets present several challenges for neural network learning. They are often high dimensional and can potentially contain a large number of labeled examples. For example, in the KDD Cup 2001 problem on binding of compounds to thrombin,

¹This work was supported in part by the United States Department of Energy through the Sandia National Laboratories LDRD program and ASCI VIEWS Data Discovery Program, contract number DE-AC04-76DO00789, and the National Science Foundation under grant EIA-0130768. Thanks to Steven Eschrich who developed usfC4.5.

each individual data element had 139,351 binary features [7]. It can be a lengthy and computationally expensive process to train a neural network using a large number of high dimensional training examples. Compared with neural networks, decision tree learning is more efficient with large numbers of labeled examples. Decision tree learning is less efficient with large numbers of features or high dimensionality, but still much quicker to build or learn a model than the typical neural network.

A feedforward back propagation neural network repeatedly examines all the training data in the process of updating its weights [8]. A decision tree learning algorithm recursively partitions the training data into ever smaller subsets on which a test is made [9]. Typically, these tests induce axis-parallel decision boundaries while neural networks with one or more hidden layers can arbitrarily closely approximate non-linear functions [20, 10, 11]. Hence, it would seem that for highly non-linear boundaries between classes, neural networks are more likely to find appropriate boundaries because decision trees will have to approximate a non-linear boundary with a series of axis parallel splits.

We began this project with the hypothesis that an ensemble of decision trees might be a close competitor to a neural network in accuracy, but could be built much quicker. This hypothesis was tested on the domain of prediction of secondary structure of proteins from their amino acid sequence. This data is high dimensional (315) and has training sets varying in size from 200,000 examples to 3.7 million examples [12, 13]. We found that the best single neural network trained on a data set of just over 200,000 examples is more accurate in its predictions by over 14% relative to a single decision tree created using default parameter settings. Further, a single neural network outperforms any ensemble of decision trees obtainable.

We have investigated potential reasons for this. One possibility is that the decision tree badly overfits the data because it has noise in it. To explore this hypothesis, we have pruned the decision tree as strongly as we could. Another possibility is that the neural network creates important new features through its hidden units. To explore this possibility, we have transformed the training data into the outputs of a trained neural network's hidden units for each example and given this to the decision tree as training data. Experiments will show that these two hypotheses cover most of the reason the neural network is better in this domain. However, the neural network is always better in our experiments at predicting the secondary structure of proteins.

This paper proceeds as follows. Section 2 covers the details of the protein secondary structure prediction problem. Section 3 contains the experimental results. Section 4 contains a discussion of the results. Finally, Section 5 contains a summary and conclusions.

2 Predicting the secondary structure of proteins

The goal of protein secondary structure prediction is to predict the secondary structure for each amino acid in a protein chain. There are 20 commonly occurring amino acids. We have used the psi-blast package [14] to obtain log likelihood values between -17 and 17 that indicate whether any of the 20 amino acids could be in any particular position in a chain.

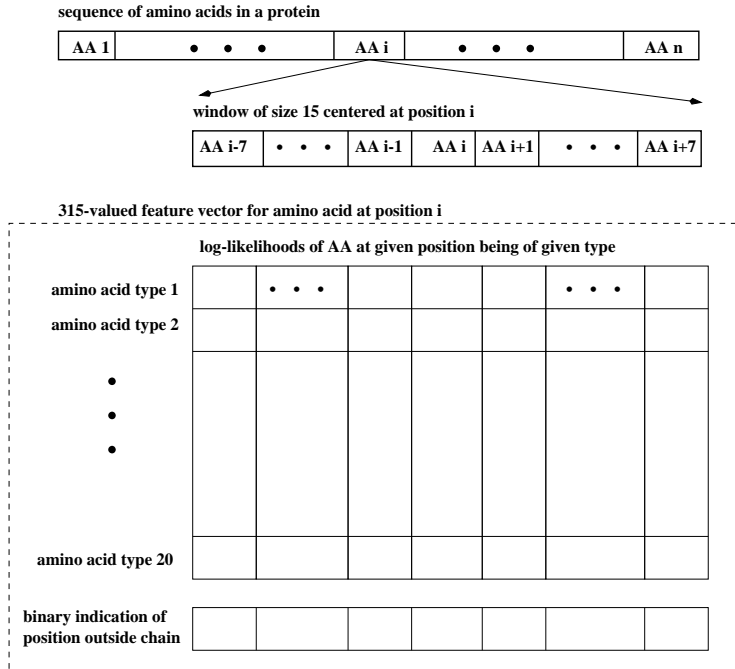


Figure 1: The construction of the feature vector for training.

In order to do prediction, a window of size n (typically $n = 15$) is used. With a window size of 15 there will be 7 amino acids to the left of the amino acid whose secondary structure is being predicted and 7 amino acids to its right. We denote the case that a window contains an N or C terminus (left and right end of the chain) with a single binary feature. Hence, a window will be of size $n \star (20 + 1)$. For window sizes of 15, the feature vector will be of size 315 as shown in Figure 1. We have used the convention of the community assessment of structural prediction (CASP) contests [15, 16] in which three classes are predicted: Helix (H), Coil (C), and Sheet (E).

For analysis purposes we have focused on a publicly available training set and test set used in [15]. It consists of a training set of chains which Jones calls non-homologous because they share little structural similarity. This set consists of 1156 chains which contain a total of 209,529 amino acids. The test set consists of 63 chains (which encompass 17,731 amino acids) which are non-homologous to the training set. Non-homologous means that the test chains share minimal structural similarity with the train set chains.

We are interested in the accuracy per amino acid because amino acids are what learning algorithms train on. However, biologists are interested in the average accuracy in secondary structure prediction per chain. Hence, we report both accuracies. Our conclusions about the relative performance of neural networks and decision trees are true for either way of reporting accuracy.

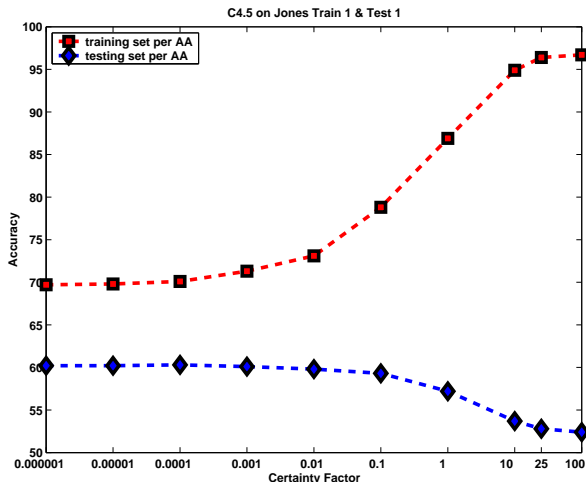


Figure 2: Decision tree accuracy per amino acid as pruning is increased by lowering the certainty factor in usfC4.5.

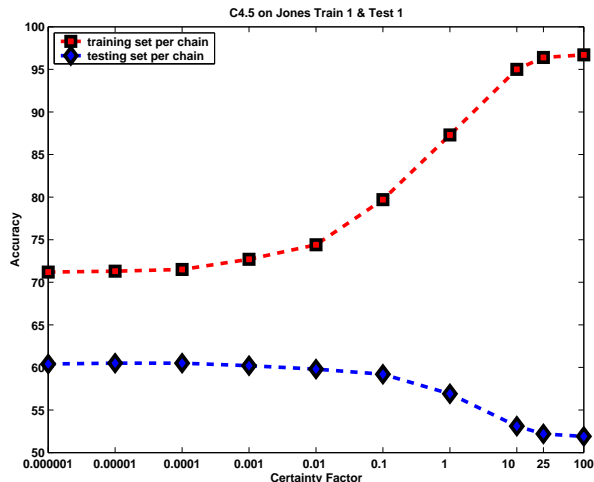


Figure 3: Decision tree accuracy per chain as pruning is increased by lowering the certainty factor in usfC4.5.

2.1 Learning methods

The decision tree software used for the experiments reported here is usfC4.5 which is an enhanced version of C4.5 release 8 [9, 17]. The C4.5 decision tree learning algorithm uses an error based pruning method [18] to simplify the tree built from all the data. The amount of the tree that will be pruned is controlled by a certainty factor parameter. The default value for the certainty factor is 25. Smaller certainty factor values cause stronger pruning, leading to trees that are smaller and more general.

We used a feedforward back propagation neural network [8] with a single hidden layer of 75 hidden units. This architecture was described by Jones [15] as the one that enabled the training of the classifier which won the CASP 3 contest. The input vector consists of the 315 values described earlier. The neural network learning parameters for all experiments are the following unless otherwise stated. The weight update was done after each pattern was presented. The learning rate is 0.00005. The weight decay is -0.0001. The momentum is 0.09. The initial weight range was ± 0.02 . The train set was randomized once.

3 Experimental Results

The decision tree learning algorithm with the default certainty factor of 25 trained on Jones train set one and tested on Jones test set one produces an accuracy of 52.8% per amino acid and 52.22% per chain. The best trained neural network has an accuracy of 74.2% per amino acid and 73.01% per chain after training for 150 epochs. This difference is over 20% and while we were not surprised that the neural network was better on this data set we were surprised at the amount by which it was better.

Figure 2 shows the decision tree accuracy on the training and test set per amino acid as

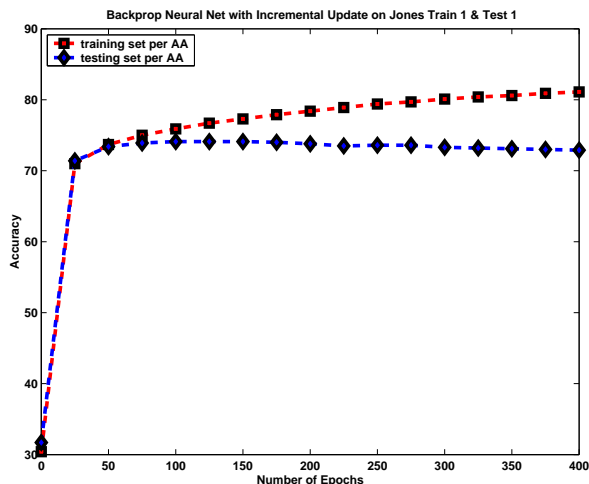


Figure 4: Neural network accuracy for selected epochs per amino acid.

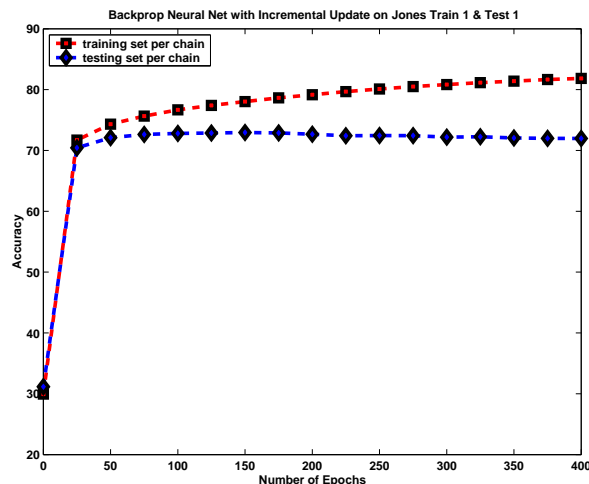


Figure 5: Neural network accuracy for selected epochs per chain.

the certainty factor is changed. The maximum accuracy obtained on the test set is 60.3% per amino acid with a very low certainty factor resulting in a highly pruned tree. This narrows the gap between the two learning algorithms to approximately 14%. The fact that more aggressive pruning of the tree produces a significant performance increase over the default pruned tree suggests that the default tree is fitting noise.

Figure 3 demonstrates that as the certainty factor is lessened, the per chain training set accuracy is non-increasing and the test set accuracy is non-decreasing. Assuming that these trends continue with a more powerful pruning method, and that training set accuracy will not be worse than test set accuracy, there exists a *generous* upper bound of 70% accuracy on the test set, a percentage which falls far short of the neural network.

Figures 4 and 5 show the neural network results on the training and test sets per amino acid and per chain respectively as the number of training epochs increases. Accuracy above 70% is reached after only a few epochs. As expected, after some number of epochs the accuracy on the train set is increasing but accuracy on the test set begins to decrease because of over fitting.

On a machine running the Linux operating system with a Pentium 4 processor running at 2.53 GHz with 2 gigabytes of memory, the neural network requires 2 minutes per epoch. The decision tree can be grown and pruned in 63 minutes. In the same time it takes to grow and prune a decision tree, 31 epochs of neural network training can be done. The neural network reaches its maximum accuracy per amino acid on the test set at around 100 epochs. Hence, for this application it is only three times slower than a decision tree in training. Even so, the neural network accuracy is greater given the same amount of training time.

One potential reason the neural network approach is greatly outperforming the decision tree approach is that a neural network could be creating new features, which enable more accurate boundaries, via its hidden units. To test this hypothesis we took the outputs of

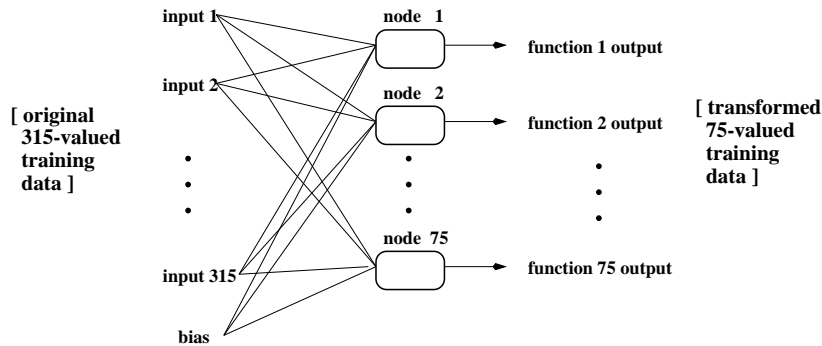


Figure 6: Transforming the training data through the neural network for the decision tree.

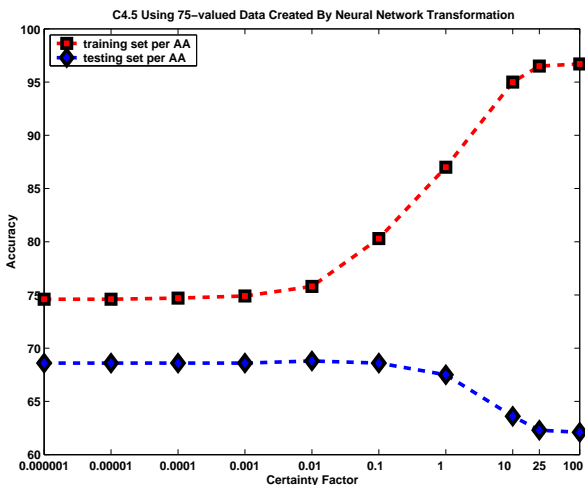


Figure 7: Decision tree per amino acid accuracy at different levels of pruning. Built on neural network transformed data.

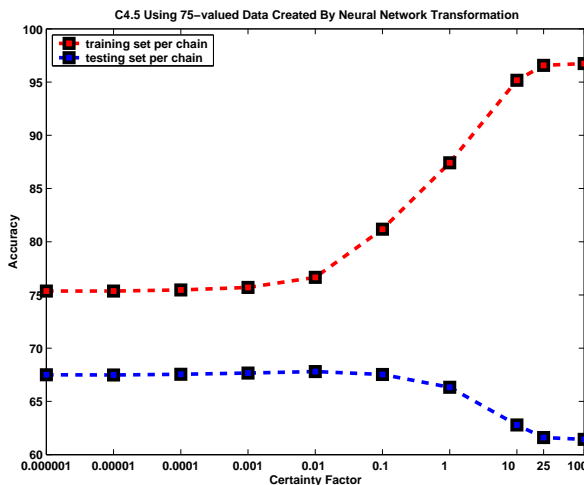


Figure 8: Decision tree per chain accuracy at different levels of pruning. Built on neural network transformed data.

the 75 hidden units for the best trained neural network for each of the 209,529 training patterns and used these values to create a new training set with the same number of examples, each of which had the same classification. The difference is that now there are 75 features or attributes each of which is the output of a particular hidden unit from the best trained neural network, as shown in Figure 6.

The accuracy results, computed per amino acid, from training a decision tree on the 75-feature transformed data from the neural network with varying levels of pruning (a progressively smaller certainty factor) are shown in Figure 7. The result with default pruning was 62.3%. The best result with the tree created using a certainty factor of 0.01 was an accuracy of 68.8% per amino acid and 67.8% per chain. So, comparing to the results from the decision tree grown using the original 315-valued data, using the transformed features results in an increase in accuracy of about 8%, which closes approximately 62% of the performance gap between the neural network and decision tree classifiers. The comparative results per chain are shown in Figure 8.

We also built a decision tree from features transformed by the neural network’s hidden units obtained after 30 epochs. At that point the neural network’s accuracy is 72.0% per amino acid and 70.75% per chain, which is clearly less than maximal accuracy. However, the decision tree built with the transformed features has an accuracy of 70.9% per amino acid and 69.68% per chain after pruning with a certainty factor of 0.001. This is 2% more accurate. The increase in accuracy accounts for 77% of the gap between classifiers in labeling the secondary structure of proteins.

At first glance, it seems strange that a transformation from a less accurate neural network could result in a more accurate decision tree. However, the optimization of weights for the hidden units to better approximate the training data might not improve a decision tree built from transformed data because the weight updates are not looking at the decision tree performance. We did look at a couple of transformations at different epochs for the neural network and the best one is reported above.

So, utilizing the hidden units as features gets us part of the way. Once the transformation is done, it is easy to build multiple decision trees. Next, we tried building an ensemble of decision trees to see if a voted ensemble of classifiers could approach or exceed a single neural network. The ensemble was built on the transformed data obtained from a neural network trained for 30 epochs. UsfC4.5 was used to create an ensemble of 100 trees in which the test at every node was chosen at random among the top 20 tests [19]. The accuracy of the voted ensemble per amino acid was 71.8% for unpruned trees and 71.9% for pruned trees with a certainty factor of 5. The average accuracy of the trees in the ensemble was 62.4% and 67.8%, respectively. The increase in accuracy of the voted classifiers over the average classifier is strong, but the increase over the best decision tree is only 1%.

4 Discussion

The problem of predicting the secondary structure of proteins is a reasonably hard one because a new protein chain is potentially different from any chain which was trained upon. This is because the process of determining the secondary structure of proteins is ongoing and new types of proteins are constantly being encountered. The training and testing sets used in this study were particularly challenging because they are designed so that there is minimal homology between them. Hence, the generalization capability of the classifier is at a premium for this data. The neural network classifier is clearly generalizing much better than the decision tree classifier.

Part of the better generalization is coming from the transformed features created by the hidden units as shown in our experiment where the decision tree learns from the transformed features. However, non-axis-parallel combination of features in the second hidden layer for the neural network is clearly important here.

One possibility to improve decision trees is to use a decision tree learning algorithm which does feature construction. QUEST [20] has been shown to be able to create linear combinations of features that allow decision trees to produce non axis parallel splits. One

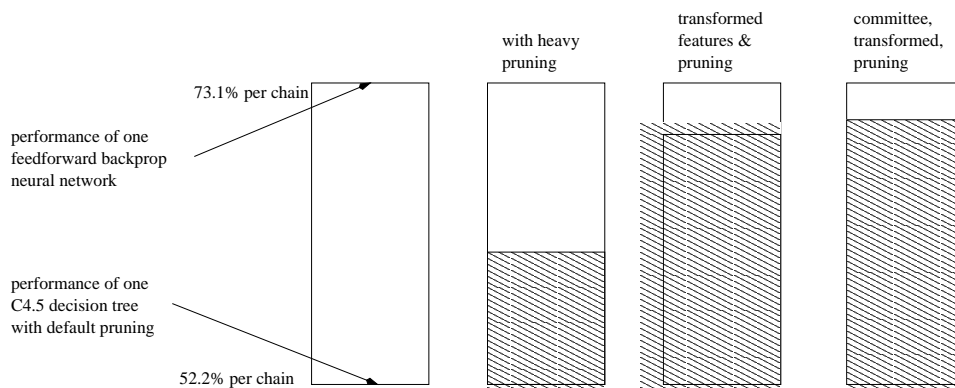


Figure 9: Closing the gap between the decision tree learning approach and the neural network approach.

concern is the time it will take, given the number of features and the amount of data. However, we have run an experiment on a randomly chosen 10% of the Jones training set one data as our training set. QUEST took 197 minutes to build a tree, but did obtain 69.62% accuracy per amino acid on the test data. This is slightly more accurate than one decision tree created from the data transformed by one neural network and slightly less than the other. In comparison, a usfC4.5 decision tree built on this data was 50.5% accurate per amino acid with default pruning and 59% accurate with a certainty factor of 0.001 when applied to Jones test set one. A usfC4.5 decision tree took about 109 seconds (under 2 minutes) to grow and prune. Hence, usfC4.5 is two orders of magnitude faster.

As another comparison, a neural network was built on the same 10% of the training data using a learning rate of 0.005, and momentum of 0.02, a weight range of ± 0.1 , no weight decay, and five epochs of training. We report results from the best neural network from three created in this way. The per amino acid accuracy of the neural network was 72.6% and it took 125 seconds to train. So, in this case it was about two orders of magnitude faster and 3% more accurate than a QUEST decision tree.

Figure 9 gives a graphical description of how our various approaches have closed the gap between neural networks and decision trees in this domain. The ensemble of decision trees resulted in a slightly better classifier than the best individual decision tree, but it was still over 2% less accurate than the best neural network in per amino acid accuracy. The more highly pruned tree ensemble resulted in a better average tree, but less accuracy increase from voting. We believe this is because the trees are more alike in predictions after pruning. A set of trees with accuracy near the best tree that individually tend to make different errors would likely provide an ensemble that further closes the gap with the neural network.

5 Conclusions

Neural networks significantly outperform decision trees in the domain of predicting the secondary structure of proteins from their amino acid sequence. One possible explanation for the difference in performance is that hidden units create new types of features that the neural network used to separate classes. To test this hypothesis, we transformed the training data into the outputs of the 75 hidden units utilized by the neural network. Each example then became a vector of 75 hidden unit outputs and a label that was the same as before. This training data was then given to our decision tree training algorithm. We were able to close 77% of the gap between classifiers by utilizing the transformation. The accuracies per amino acid were 74.1% for the best trained neural network and 70.9% for the highly pruned decision tree.

Recognizing that features that enabled non-axis-parallel splits might be important to the decision tree, we used QUEST [20] to construct non-axis-parallel features. Due to memory and time limitations we were only able to apply it to 10% of the training data. A neural network built on the same training data is 3% more accurate than the QUEST decision tree and 13% more accurate than a usfC4.5 decision tree.

We also investigated an ensemble of decision trees. However, an ensemble of 100 decision trees created by using a method that has been favorably compared to bagging resulted in only a 1% gain in accuracy. Hence, we found no construction of decision trees which could result in a classifier better than a single neural network.

The generalization ability of neural networks for secondary structure prediction of proteins from amino acid sequences is clearly superior to that of decision trees. This appears to be partly because of complex features that are constructed to allow non axis parallel decision boundaries and partly the ability to make those boundaries nonlinear. In this domain a neural network was able to train to high accuracy in a relatively short time period. Hence, decision tree learning was not truly time effective and certainly less accurate. The results of this investigation lend clear support to the preference for neural networks in at least this type of bio-informatics problem.

References

- [1] R. Mooney, J. Shavlik, G. Towell, and A. Gove. An experimental comparison of symbolic and connectionist learning algorithms. In *IJCAI-89 Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 1, pages 775–780, 1989.
- [2] Shavlik J.W., R.J. Mooney, and Towell G.G. Symbolic and neural learning algorithms: an experimental comparison, machine learning. *Machine Learning*, 6(2):111–143, 1991.
- [3] Brown D.E., Corruble V., and Pittard C.L. A comparison of decision tree classifiers with backpropagation neural networks for multimodal classification problems. *Pattern Recognition*, 26(6):953–961, 1993.
- [4] Dietterich T.G., Hild H., and Bakiri G. A comparison of ID3 and backpropagation for english text-to-speech mapping. *Machine Learning*, 18(1):51–80, 1995.

- [5] Sethi I.K. and Otten M. Comparison between entropy net and decision tree classifiers. In *IJCNN International Joint Conference on Neural Networks*, volume 3, pages 63–68, 1990.
- [6] Petra Perner, Uwe Zscherpel, and Carsten Jacobsen. A comparison between neural networks and decision trees based on data from industrial radiographic testing. *Pattern Recognition Letters*, 22(1):47–54, 2001.
- [7] Kdd cup 2001. <http://www.cs.wisc.edu/~page/kddcup2001/>.
- [8] Martin Anthony and Peter Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University press, 1999.
- [9] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992. San Mateo, CA.
- [10] Hornik K., Stinchcombe M., and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
- [11] J. L. Castro, C. J. Mantas, and J. M. Benitez. Neural networks with a continuous squashing function in the output are universal approximators. *Neural Networks*, 13(6):561–563, 2000.
- [12] N.V. Chawla, T.E. Moore, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, and C. Springer. Distributed learning with bagging-like performance. *Pattern Recognition Letters*, 24(1-3):455–471, 2003.
- [13] N. Chawla, T.E. Moore, K.W. Bowyer, L.O. Hall, C. Springer, and W.P. Kegelmeyer. Bagging-like effects for decision trees and neural nets in protein secondary structure prediction. In *BIOKDD01: Workshop on DataMining in Bioinformatics at KDD01*, pages 50–59, 2001.
- [14] NCBI BLAST homepage. <http://www.ncbi.nlm.nih.gov/BLAST/>.
- [15] D.T. Jones. Protein secondary structure prediction based on decision-specific scoring matrices. *Journal of Molecular Biology*, 292:195–202, 1999.
- [16] Protein structure prediction center. <http://predictioncenter.llnl.gov/>.
- [17] J.R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.
- [18] L.O. Hall, R. Collins, K. W. Bowyer, and R. Banfield. Error-based pruning of decision trees grown on very large data sets can work! In *Fourteenth International conference on Tools with Artificial Intelligence*, pages 233–238. IEEE Computer Society, 2002.
- [19] Dietterich T. G. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.
- [20] T-S Lim, W-Y Loh, and Y-S Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, 2000.