Optimal Artificial Neural Network Architecture Selection for Bagging

Tim Andersen
Mike Rimer
Tony Martinez

Iarchives
1401 S. State, Provo, UT 84097 USA
timothyandersen@yahoo.com

Abstract

This paper studies the performance of standard architecture selection strategies, such as cost/performance and CV based strategies, for voting methods such as bagging. It is shown that standard architecture selection strategies are not optimal for voting methods and tend to underestimate the complexity of the optimal network architecture, since they only examine the performance of the network on an individual basis and do not consider the correlation between responses from multiple networks.

1 Introduction

There are several well-known methods for combining the predictions of multiple classifiers in order to obtain a single prediction. These include Bayesian methods [16], bagging [6], boosting[13], and other voting methods [19]. However, little work has been done on the problem of model selection when using these methods. This paper examines the problem of selecting an appropriate neural network architecture when using bagging and other voting methods to combine the predictions of multiple neural networks. We show that standard architecture selection strategies do not always select optimal neural network architectures for such methods.

Section 2 discusses voting methods and the problem of selecting an optimal network architecture for such methods. Section 3 discusses related work in the field of architecture selection. Section 4 gives experimental results, and section 5 gives the conclusion.

2 Architecture Selection for Voting Methods

Neural network architecture selection strategies studied in the literature have focused on choosing the single best performing architecture from a group of architectures, generally using some kind of cost/performance tradeoff or the performance of the network on a holdout set as the selection criteria. Under certain assumptions, these architecture selection criteria can be shown to be optimal. However, such performance measures are only optimal in the case where a single network is to be used as the final predictor, and are not optimal for the architecture selection problem when using bagging or other voting methods to combine the predictions of several neural networks. From a Bayesian standpoint, the optimal prediction is obtained by calculating a weighted average of all possible network architectures and all possible weight settings for those architectures, where each network is weighted by its posterior probability. From a purely Bayesian standpoint, any architecture selection strategy which chooses a single network architecture using a cost/performance tradeoff is sub-optimal, since it entirely ignores a large number of possible architectures that could significantly impact the solution.

Obviously, the calculation of this weighted average is computationally infeasible, however, the optimal prediction can be approximated in a number of different ways. Bagging, which can be viewed as an approximation to the Bayes optimal solution, generates a prediction by calculating a weighted average of several predictors. With bagging, the weight is usually set to 1 for each predictor, which amounts to the assumption that all of the predictors are equal-probable from a Bayesian standpoint. This assumption is not unreasonable since the predictors are often not likely to greatly differ in their posterior probabilities, and it may be difficult to accurately estimate the true, relative a-priori probabilities.

Bagging and other voting methods work best when the errors between the various predictors are uncorrelated, and the correct responses between the predictors are correlated. Generally speaking, very simple predictors tend to have both correlated errors and correlated correct responses. For example, one of the simplest ways to formulate a predictor is to always predict the majority class of the training set. Obviously, using multiple such predictors cannot increase classification accuracy, since the errors (and correct responses) of such predictors are 100 percent correlated. As the complexity of the predictors is increased, the correlation between the responses of the predictors tends to decrease. This is because with increasing complexity there is a corresponding increase in the number of different solutions (minimum error for the training set) that the predictor can produce.

Since bagging and other voting methods work best when the correct responses between predictors are correlated and the incorrect responses are uncorrelated, when bagging or other voting methods are used to combine the results of multiple networks the goal for neural network architecture selection is to choose the network architecture which maximizes the correlation between multiple trained copies of the network when the networks are producing the correct response, and minimizes the correlation between the networks on incorrect responses. So, the network architecture which maximizes a cost/performance tradeoff, or even that performs the best on a holdout set, is not guaranteed to be the best architecture for bagging, since it does not examine this correlation.

There are a number of factors that can influence the choice of the appropriate network architecture for voting methods such as bagging. These include but are not limited to:

- Number of bagged predictors
- Number of training examples
- Underlying problem domain
- Idiosyncrasies of the training algorithm

For example, lowering the number of training examples is likely to require lowering the complexity of the network architecture in order to achieve optimal performance. It is also possible that increasing the number of predictors "in the bag" may allow for a corresponding increase in the complexity of the network architectures being bagged.

3 Related Work in Architecture Selection

There have been a number of different architecture selection strategies studied in the literature. These strategies are all ultimately based on either the use of a holdout set or a cost/performance tradeoff to determine the 'optimal' network architecture. These strategies include the following:

Network Construction Algorithms

The majority of network construction methods start from a very simple basis, usually one node, and add nodes and connections as needed in order to learn the training set. These strategies include Cascade Correlation [8], DNAL [4], Tiling [14], Extentron[3], Perceptron Cascade [7], the Tower and Inverted Pyramid algorithms [10], and DCN [17]. Other construction algorithms include Meiosis [11] and node splitting (Wynne-Jones 1992).

One of the drawbacks of most current MLP construction algorithms is that they do not have built in mechanisms to prevent the network from overlearning, rather treating this important subject as an afterthought. For example, Burgess states that "for good generalization it is necessary to restrict the size of the network to match the task," [7] but no specific algorithm is presented on how to do so. Left uncontrolled, all of these methods will suffer from over learning, and so in some respects they do not avoid the architecture selection problem but must utilize some type of architecture selection strategy (such as CV or MDL based strategies) in an attempt to avoid over learning. This is due to the fact that, left uncontrolled, the network structure can grow to fit the training set data exactly. But with many problems the training data may contain noise that will cause the algorithm to perform worse if the noisy instances are memorized. Also, the network can grow to the point that the amount of training data is insufficient to properly constrain the network weights.

Early Stopping

Early stopping strategies [1,9,18,23] utilize overly complex network architectures. One of the main advantages of using a network that is more complex than is actually needed is that larger networks tend to have fewer local minima in the error surface. However, with a larger network there is a higher likelihood that over learning will occur. In other words, larger network architectures are more likely to converge to a lower training set error, but often tend to produce higher error on non-training examples. In order to avoid this, early stopping strategies try to determine when the network has been trained sufficiently to do well on the problem but has not yet over learned (or memorized) the training data. One way to do this is to occasionally test the performance of the network on a holdout set and stop training when the performance on the holdout set begins to degrade.

Cross Validation (CV)

CV is often used to select an optimal architecture from amongst a set of available network architectures. In a comparison of CV with two other MLP architecture selection strategies in a recent paper [20] CV was found to be the best at choosing the optimal network architecture, at least on the data sets tested. However, the comparison was based on only a single type of artificial data and did not look at any real world problem domains.

In a larger study CV was found to not perform well when selecting an optimal architecture from a large set of relatively similar architectures [2]. Several strategies are suggested which can be applied when using CV based MLP architecture selection to significantly improve the performance CV based architecture selection.

Weight Decay

Weight decay adds a penalty term to the error function that favors smaller weights [5, 12]. The rate of weight decay is often chosen by training several different networks with different rates of decay and then using CV to estimate which rate is optimal.

Network Pruning

Pruning techniques start with an overly large network and iteratively prune connections that are estimated to be unnecessary. CV is often used to assist in the estimation process. The pruning can take place during the training process or training cycles can be alternated with pruning cycles. Pruning strategies include Optimal Brain Damage [21], Skeletonization [15], and Optimal Brain Surgeon [22].

4 Experiments and Results

Experiments were conducted several data sets in order to empirically determine the efficacy of cost/performance tradeoff and CV based methods in determining the optimal network architecture for bagging. The real world data sets were obtained from the UC Irvine machine learning database repository.

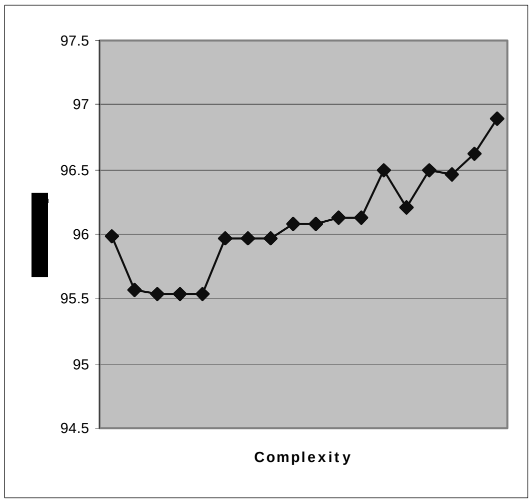


Figure 1. Breast Cancer Wisconsin and Bagging.

For the results reported in this paper the complexity level of the tested network architectures ranges from 2 to 20 hidden nodes arranged in a single hidden layer of a fully connected network. In order to determine the best architecture for bagging, 30 sets of network weights are trained for each complexity level in this range, and the performance of the 30 bagged networks is evaluated for each of the network architectures. The performance and complexity level of the best architecture for bagging is then compared against bagging's performance and complexity level using the network architecture chosen by Akaike's information based measure (AIC), and with the architecture selected by CV.

Figure 1 shows the test set results of the bagged networks for each of the network architectures tested on the Breast Cancer Wisconsin data set. This data set is interesting because it shows a significant general upward trend in test set accuracy as the complexity of the bagged networks is increased. However, there is not a significant upward trend in the test set scores of the networks taken individually (nor in the training set scores), as can be seen in figure 2. Because of this, architecture selection strategies which only examine the performance of the individual networks, such as most cost/performance measures and also CV based measures, are unlikely to find the optimal architecture for bagging for this particular problem. Indeed, for this particular problem the AIC criteria chooses the simplest network architecture, which has a bagged network performance which is significantly worse than the best performance of the tested architectures (95.9% vs 96.9% on the test set).

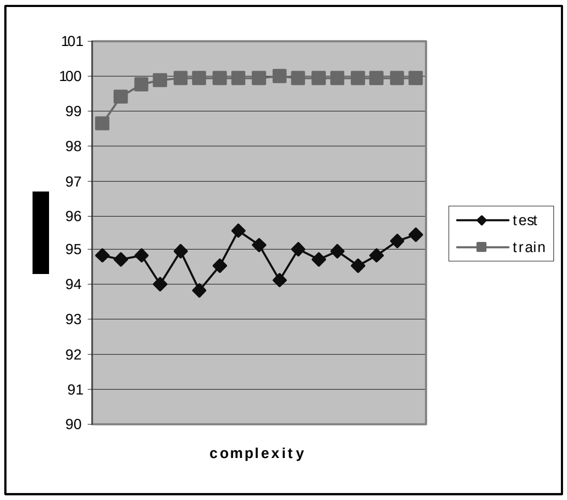
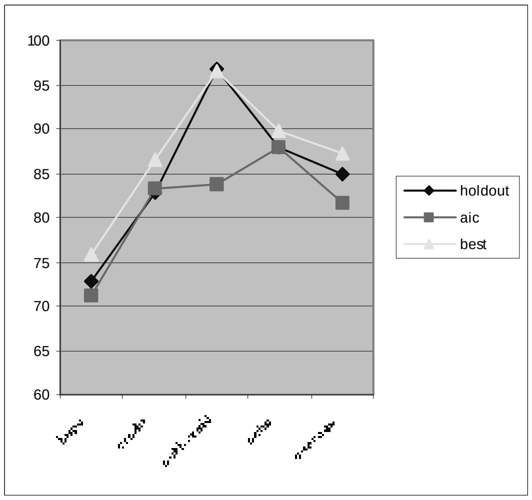


Figure 2. Breast Cancer Wisconsin - no Bagging.



Fi

Figure 3. AIC vs holdout vs optimal test set accuracy.

Figure 3 shows the test set performance of AIC vs CV for selecting the network architecture for bagging, and compares this against the ‘optimal’ network architecture for bagging. On average, the AIC criteria is significantly worse than using CV to choose an architecture for bagging, and both generally fail to pick the optimal network architecture for these problems.

Both AIC and CV significantly underestimate the complexity of the best architecture for bagging for these problems, as can be seen in figure 4,

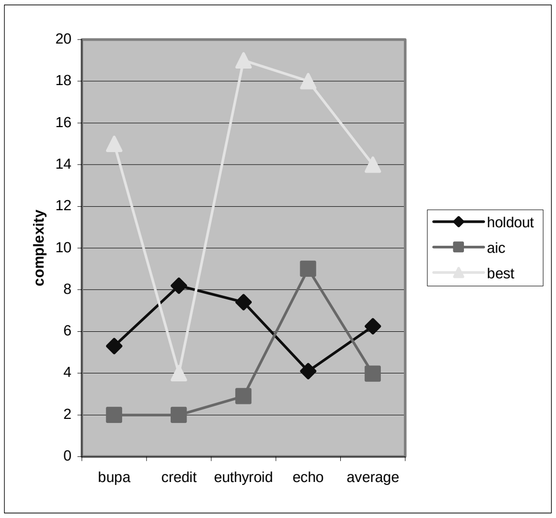


Figure 4. Average complexity of chosen architecture for each problem.

with AIC on average choosing a network with 4 hidden nodes and CV choosing an architecture with 6 hidden nodes, with the optimal network architecture for bagging containing (on average) 14 hidden nodes.

5 Conclusion

The experimental results show that, for the problems tested in this paper, the optimal network architecture for bagging (and by extension other voting methods) is more complex than the network architecture chosen by cost/performance tradeoff methods such as MML and MDL, and also more complex than the network architecture chosen by CV based methods which only examine the performance of individual networks. We have argued that this empirical result will hold for most learning problems, since these strategies are only designed to identify the optimal network architecture if a single network will be used as the final predictor. When multiple networks are combined using a voting method, then these strategies tend to underestimate the complexity of the optimal network architecture since they cannot estimate the degree to which the responses of the different network architectures will be correlated, and this estimate is critical in the determination of the optimal network architecture for voting methods.

The factors which may affect the optimal complexity for bagging and other voting based methods include the number predictors that will b e voted, the number of training examples, the underlying problem domain, and idiosyncrasies of the training algorithm. Future work will focus on studying the effects of each of these factors, as well as developing a systematic methodology for selecting the optimal network architecture for voting methods.

6. References

[1] Amari, S, Noboru Murata, Klaus-Robert Muller, Michael Finke, and Howard Hua Yang. 1997. Asymptotic Statistical Theory of Overtraining and Cross Validation. IEEE Transactions on Neural Networks, vol 8, no 5, pp 985-996, September 1997.

[2] Andersen, Tim and Tony Martinez. 1999. Cross Validation and MLP Architecture Selection. To appear in Proceedings of the International Joint Conference on Neural Networks 1999.

[3] Baffes, Paul, and John Zelle. 1992. Growing Layers of Perceptrons: Introducing the Extentron Algorithm. Proceedings of the 1992 International Joint Conference on Neural Networks, pp 392-397, Baltimore, Maryland, June 1992.

[4] Bartlett, Eric. 1994. Dynamic Node Architecture Learning: An Information Theoretic Approach. Neural Networks, vol 7, no 1, pp 129-140.

[5] Bartlett, P.L. 1997. For valid generalization, the size of the weights is more important than the size of the network. Advances in Neural Information Processing Systems 9, pp. 134-140 Cambrideg, MA: The MIT Press.

[6] Breiman, L. 1996. Bagging Predictors. *Machine Learning* **24**, 123-140.

[7] Burgess, Neil. 1994. A Constructive Algorithm that Converges for Real-Valued Input Patterns. International Journal of Neural Systems, vol 5l, no 1, pp 59-66, March, 1994.

[8] Fahlman, S. E. and C. Lebiere. 1990. The Cascade Correlation Learning Architecture. Neural Information Processing Systems 2, D. S. Touretzsky, ed. Morgan Kaufman, pp. 524-532.

[9] Finnof, William, Ferdinand Hergert, and Hans Zimmermann. 1993. Improving model selection by nonconvergent methods. Neural Networks, 6, 771-783.

[10] Gallant., S.I. 1986. Three constructive algorithms for network learning. Proc. 8th Ann Conf of Cognitive Science Soc, pp. 652-660, August 1986.

[11] Hanson, S. 1990. Meiosis networks. In Advances in Neural Information Processing Systems, D. S. Touretzky, editort, pp 533-541. Morgan Kaufman, San Mateo.

[12] Krogh, Anderse, and John Hertz. 1992. A Simple Weight Decay Can Improve Generalization. In Advances in Neural Information Processing Systems, Moody, J; Hanson S; and Lippmann, R eds, vol 4, pp 950-957. San Mateo, CA: Morgan Kauffmann publishers.

[13] Maclin, R and D Opitz. 1997. An empirical evaluation of bagging and boosting. *The Fourteenth National Conference on Artificial Intelligence.*

[14] Mezard, M. and J. P. Nadal. 1989. Learning in feedforward layered networks: The tiling algorithm. Jounral of Physics A, 22:2191-2203.

[15] Mozer, M. C. and P. Smolensky. 1988. Skeletonization: A technique for trimming the fat from a network via relevance assessment. Advances in Neural Information Processing. 1. D. S. Touretzky, Ed.. Denver 1988. pp 107-115.

[16] Neal, R. M. (1996), *Bayesian Learning for Neural Networks* 118, New York: Springer-Verlag.

[17] Romaniuk, Steve and Lawrence Hall. 1993. Divide and Conquer Neural Networks. Neural Networks, vol 6, pp 1105-1116.

[18] Sarle, W.S.. 1995. Stopped Training and Other Remedies for Overfitting. Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics, pp 352-360.

[19] Schapire, R, Y Freund, P Bartlett, and W S Lee. 1997. Boosting the margin: A new explanation for the effectiveness of voting methods. *Proceedings of the Fourteenth International Conference of Machine Learning*.

[20] Schenker, B, and M Agarwal. 1996. Cross-validated structure selection for neural networks. Computers chem. Engng, vol 20, no 2, 175-186.

[21] Solla S., Y. Le Cun, and J. Denker. 1990. Optimal Brain Damage. In Advances in Neural Information Processing Systems. NIPS) 2, pages 598--605, D. S. Touretzky, editor, San Mateo, Morgan Kaufmann Publishers Inc.

[22] Stork D. and B. Hassibi. 1993. Second order derivatives for network pruning: Optimal Brain Surgeon. in Advances in Neural Information Processing Systems. (NIPS) 5, pages 164--171, T. J. Sejnowski G. E. Hinton and D. S. Touretzky, editors, San Mateo, Morgan Kaufmann Publishers Inc.

[23] Wang, C., Venkatesh, S.S., and Judd, J.S. 1994. Optimal Stopping and Effective Machine Complexity in Learning. NIPS6, 303-310.