Bagging in Computer Vision

Bruce A. Draper

Department of Computer Science
Colorado State University
Fort Collins, CO. 80523
draper@cs.colostate.edu

Kyungim Baek

Department of Computer Science
Colorado State University
Fort Collins, CO. 80523
baek@cs.colostate.edu

Abstract

Previous research has shown that aggregated predictors improve the performance of non-parametric function approximation techniques. This paper presents the results of applying aggregated predictors to a computer vision problem, and shows that the method of bagging significantly improves performance. In fact, the results are better than those previously reported on other domains. This paper explains this performance in terms of the variance and bias.

1 Introduction

Function approximation and classification are ubiquitous problems in computer vision. The pattern recognition task of mapping from a set of (not necessarily independent) features to a predicted function value or label rises in object recognition, target recognition, visual navigation, and almost every other application of computer vision. For many years, the traditional approach to this problem was to estimate the parameters of the feature distributions, and apply maximum likelihood classification or related approximation techniques to predict labels/values. Unfortunately, this approach often performs poorly in the presence of unmodeled dependencies between features.

To counteract this problem, many researchers have turned to non-parametric techniques such as neural networks, decision trees, and nearest neighbor classifiers. While these techniques often provide improved results on computer vision data sets, they are all too often "black boxes" to the researchers who use them. As a result, many researchers do not know how to optimize the performance of these techniques short of extensive trial-and-error.

This paper presents a technique called *bagging*[1] for improving the performance of non-parametric function approximation techniques. This approach was developed by Breiman [1], and is similar to work by Dietterich on improving non-parametric classification [2, 4, 5]. (See also [3, 6].) The basic idea is to randomly sample the training set and produce multiple function approximations. The aggregated or "bagged" predictor is then the average of the component predictions on a given sample. The non-obvious result is that the bagged predictor is significantly better than any of the component predictors.

Since results from other domains do not always apply in computer vision, this paper applies bagging to the task of evaluating image regions via neural networks. Our results prove to be consistent with - if not better than - those presented by Breiman [1]. In addition, we predict which tasks will benefit from bagging by measuring the variance and bias of the component non-parametric function approximations (predictors).

2 Bagging

The basic idea behind bagging is simple: train multiple function approximations and average their results. To clearly explain when and why this works, however, we have to get a little more formal. A function approximation task can be defined in terms of a set W:

$$W = \{(F_n, y_n), n = 1, ..., \infty\},$$

where F_n is a feature vector describing a data instance, and y_n is the function value associated with instance n. The goal is to learn a function approximation φ such that $y_n \approx y'_n = \varphi(F_n)$ from a training set $T \subset W$.

Training algorithms (such as backpropagation) for non-parametric function approximation can be thought of as processes that map training sets onto functions (i.e. $T \rightarrow \varphi$). They are typically used to train a single approximation $\varphi = Train(T)$. In bagging, however, T is randomly subsampled (with replacement) to produce M subsets T_M, and each of

[1] Bagging is an acronym for *b*ootstrap *agg*regat*ing*.

these subsets is used to train a function approximation $\varphi_M = Train(T_M)$. The bagged approximation can then be written as:

$$\varphi_{bag}(F_i) = \frac{(\sum_{k=1}^{M} W_k \varphi_k(F_i))}{(\sum_{k=1}^{M} W_k)}$$

where

$$W_k = \frac{1}{MSE(\varphi_k)}$$

(In other words, $\varphi_{bag}(F_i)$ is the weighted average of $\varphi_M(F_i)$.) The weighting term, W_M, gives more weight to predictors that have lower overall mean squared error (MSE) than others. (Breiman does not use a weighting term, but we found that it significantly improved performance.)

There are two bases for arguing the φ_{bag} should be a better approximation than $\varphi = Train(T)$. The first assumes that the training algorithm is unstable, in the sense that a small change to the training set T_M results in a large change to approximation φ_M. (This assumption is true for backpropagation, and Breiman argues that it is true for decision tree inference algorithms as well [1].) In this case, the function approximations $\varphi_1, ..., \varphi_M$ are largely independent of each other. If they are also unbiased, then $\varphi_1(F_i), ..., \varphi_M(F_i)$ can be thought of as M independent samples drawn from a distribution whose mean is y_i. As a result, the expected value of $\varphi_{bag}(F_i)$ is y_i, even if the errors in the individual approximations φ are large.

The second argument says that most training algorithms get caught in local optima, and as result produce approximations φ that are accurate over part but not all of the feature space. Once again, the difference among the subsampled training sets T_M suggest that the approximations φ_M will correspond to different local optima. If each φ_M is accurate for most of the feature space, then the majority of estimates $y'_{iM} = \varphi_M(F_i)$ will accurately reflect y_i. Moreover, we can once again assume that the outliers will be unbiased, so their expected value is zero. As a result, the bagged predictor should be accurate across the entire feature space, or at least a larger portion of the space than any component φ_M.

Both of these arguments can be cast in terms of variance and bias. In general, bias is the tendency of an approximation to overestimate or underestimate, while variance is (informally) the noise in the estimation process. In regression, it is a well-known theory that the expected squared error of an approximation algorithm on test data can be decomposed into bias and variance terms [7]:

$$Error(F_i) = Bias^2(F_i) + Var(F_i)$$

where

$$Bias(F_i) = \overline{y_i}' - y_i$$

$$Var(F_i) = \mathrm{E}_T(\overline{y_i}' - \varphi_T(F_i))^2$$

and

$$\overline{y_i}' = \mathrm{E}_T(y_i)$$

The general result of bagging is to reduce the variance, so that the error of an aggregate approximation approaches the bias error as the number of component approximations approaches infinity. If the component predictors are unbiased (which they should be overall, but may not be for specific samples F_i), then the aggregate error approaches zero.

3 The Test Problem

Our test case evaluates bagging on a region evaluation problem. As part of another project, we have an object recognition system that generates hypotheses in the form of labeled image regions. Working in an aerial image domain, we collected a data set of 3,374 house hypotheses, each represented by a mask and a corresponding image chip. The task is to estimate the quality of hypotheses from features describing the regions.

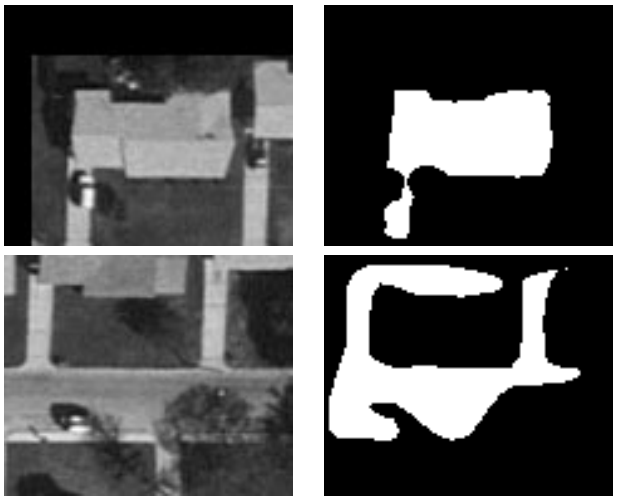

Figure 1: Image chips(left) and masks(right) for two house hypotheses. The top hypothesis gets a score of .87(the inclusion of the driveway is most of the penalty), while the lower hypothesis scores only .15 since it barely overlaps a house.

A total of twenty features are used to describe hypothesized regions. The features themselves are not unusual. They include shape features, size features, texture features, and brightness features. One feature

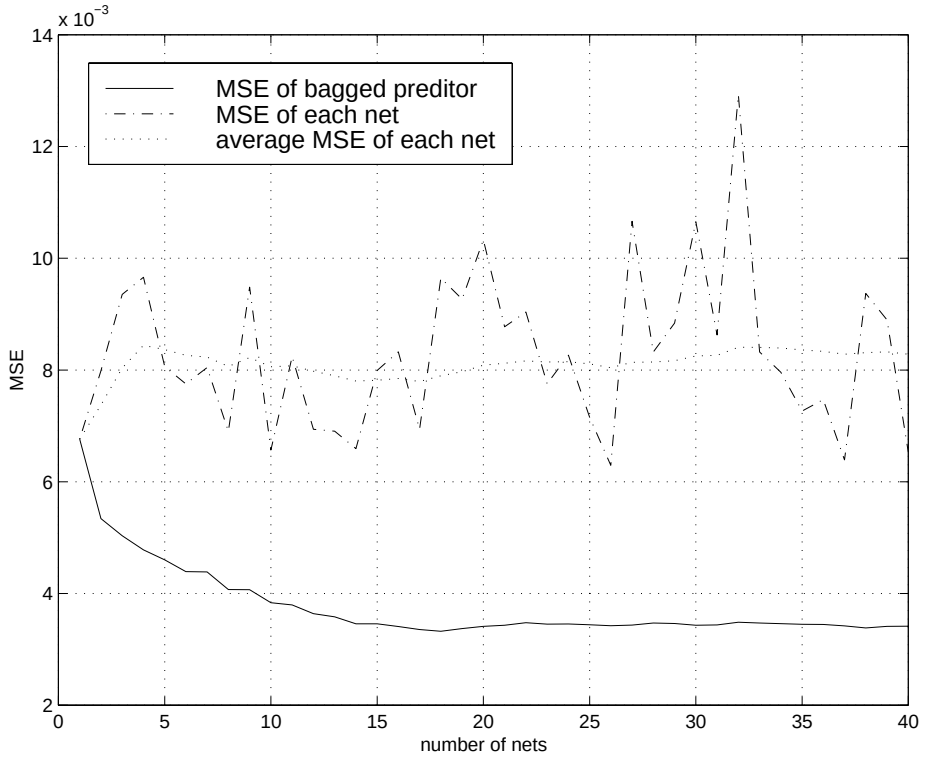

Figure 2: Mean Squared Error of Forty Nets vs. Bagged(Aggregated) Prediction

	Individual Nets	Bagged Net : 15 nets (decrease)	Bagged Net : 40 nets (decrease)
MSE average	0.0083	0.0036 (56.6%)	0.0034 (58.8%)
MSE min(best)	0.0063	0.0031 (50.8%)	0.0034 (45.8%)
MSE max(worst)	0.0129	0.0040 (69.0%)	0.0034 (73.6%)
Standard Deviation	0.0014	0.0002	–

Figure 3: MSE Comparison of individual nets and bagged net. The 3rd column is the result of bagging 100 times with 15 nets randomly selected from 40 nets in each time.

is a shadow measure that relies on knowing the sun angle. In many respects, this task is typical of function approximation tasks that arise in computer vision.

Part of what makes this task difficult is that the answer is not binary. Most of the hypotheses are good to the extent that they overlap the true position of a house, but they do not match it exactly. The quality of a hypothesis depends on how closely it matches the ground truth data, and is measured according to:

$$\frac{|H \cap T|}{|H \cup T|}$$

where H is the set of hypothesized pixels, and T is the true set of house pixels. (Note that this function - the size of the intersection of the pixel sets divided by the size of their union - is bounded between zero and one.) Figure 3 shows two hypotheses, one with a high score (.87) because it matches the true house region closely, the other with a lower score (.15).

4 Results

The goal of this experiment is to test whether bagging improves accuracy in computer vision problems, and if the improvement is in line with the results reported in [1]. Secondarily, we are also interested in whether the reason for the improvement (if any) matches the explanation given in Section 2.

We tested the bagging technique using backpropagation to train fully-connected neural networks with 30 hidden units, a sigmoidal squashing function and a momentum term of 0. In order to test the bagging technique, we split our total database of 3,374 samples into a training set T of 2,874 samples and a test set U of the remaining 500 samples. We then trained a total

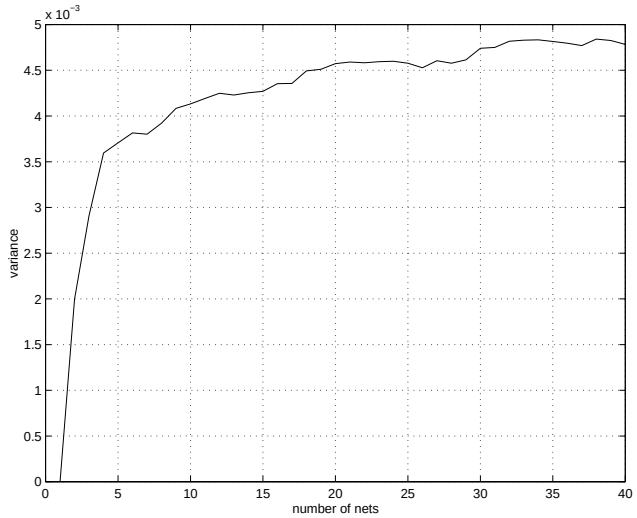

Figure 4: Cumulative Variance

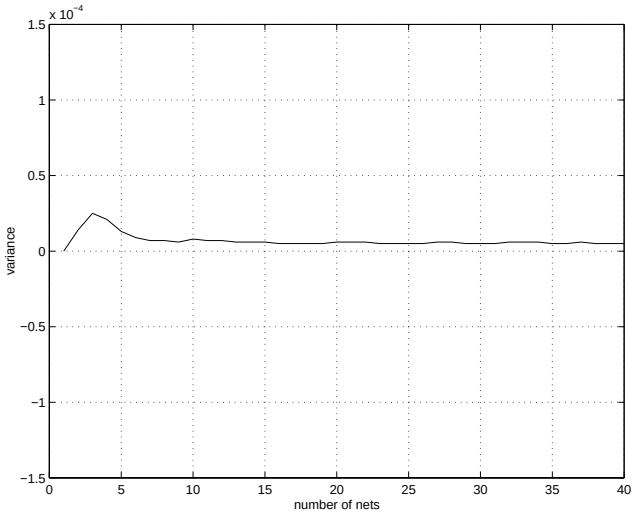

Figure 6: Variance of Bagged net

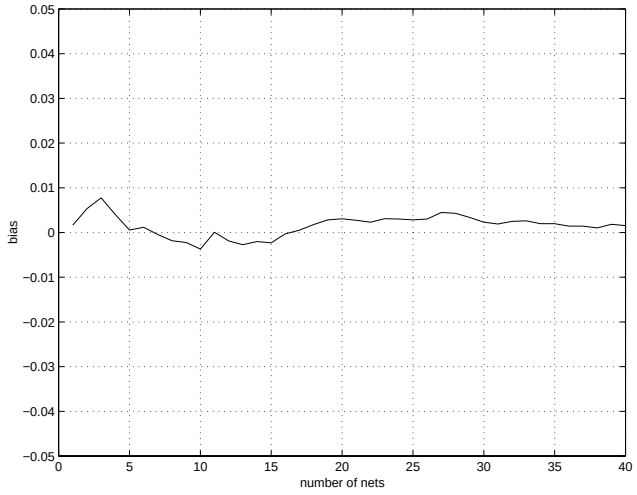

Figure 5: Cumulative Bias

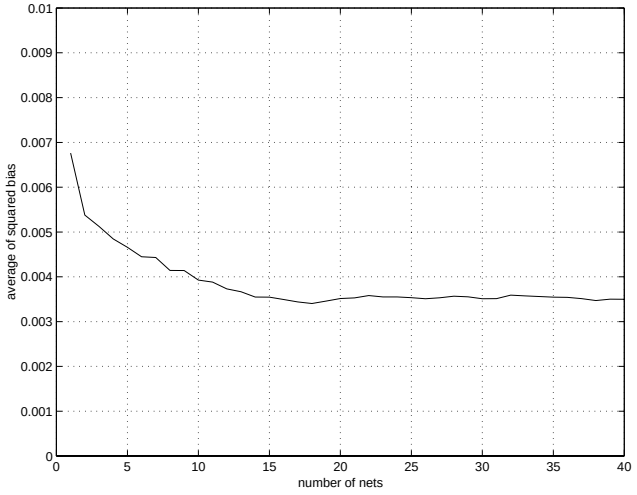

Figure 7: Bias Error

of forty networks, where network φ_i was training by randomly selecting 2,000 samples from T to form T_i, and then applying backpropagation to train network φ_i over data set T_i.

Figure 2 shows the result of applying these networks to the test set U, with and without bagging. The dash-dot(-.-) line shows the mean squared error(MSE) of the test set on each of the forty nets. The solid line is the MSEs that result from bagging, so that for example the MSE of bagging the first five nets is 0.0048, while the MSE from bagging ten nets is 0.0038. (Remember that the function being approximated has a range from 0 to 1.) For comparison's sake, the dotted line(..) is the running average of the MSEs of the nets.

The immediate conclusion from Figure 2 is that bagging leads to a striking improvement. The MSEs of the forty individual nets ranged from 0.0063 to 0.0129 with an average of 0.0083, while the MSE from bagging the forty networks is 0.0034 (Figure 3). Bagging therefore leads to an improvement of 58.8% over the average performance of the individual nets and an improvement of 45.8% over the best of the component nets. In addition, although bagging is clearly more expensive than other approaches (since it trains multiple nets), from Figure 2 we see significant improvement from bagging as few as five neural nets, and we see almost all of our improvement by the time fifteen nets have been bagged. The second column of Figure 3

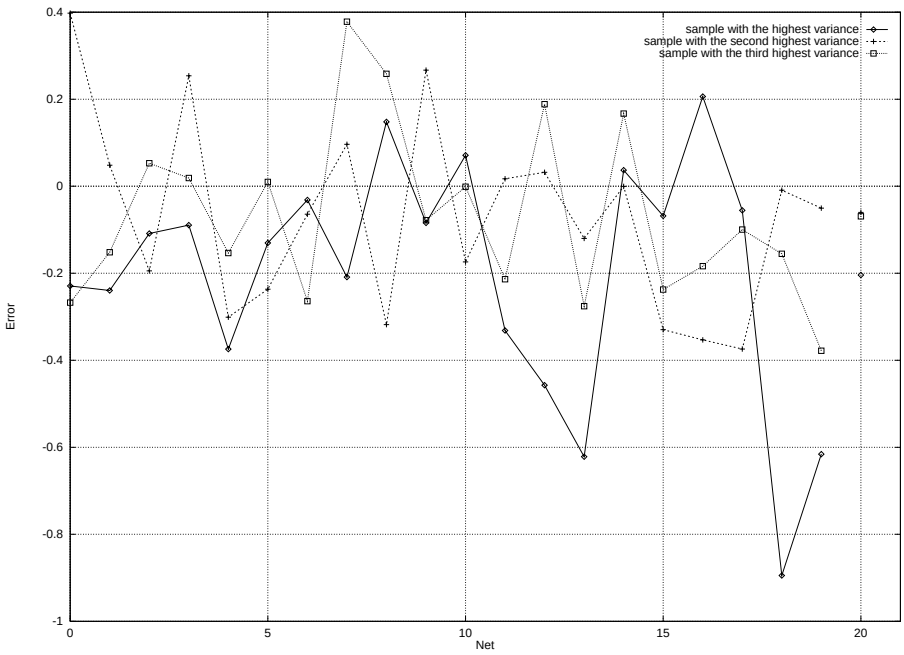

Figure 8: Prediction errors for the three samples with the highest variance across the first twenty nets(net #0 - net #19). The variance is clearly visible in the different errors produced by each net, while the average prediction(net #20) for each sample is reasonable.

shows it more clearly. The values have been drawn by bagging 15 randomly selected nets 100 times. The average error shows that the performance is very close to the results gained by bagging 40 nets.

Not only are these results consistent with those reported in [1], the improvement of 58.8% ranks it as the best. (Breiman's best was an improvement of 47% on the Heart dataset at the UCI dataset repository.) In the test reported here, the random sampling was done without replacement. However, the results of experiments with replacement are not much different from previous test. Although the MSE is slightly worse (ranging from 0.0075 to 0.0174 for individual nets, 0.0044 for bagging 40 nets), the average decrease in error is 64.2%.

Whether this is because of the domain, because we bagged neural nets as opposed to decision trees (as Breiman did), or because we added the weighting term, we cannot say. Either way, bagging is clearly beneficial in this domain. But does the reason for its success lie in the explanations posited in Section 2?

Since we observed a striking improvement by bagging, if the explanations from Section 2 hold, two other predictions should be met: 1) the bias of the neural nets (viewed as a set) should be low and the variance of the neural nets should be high. 2) bagging reduces most of the variance, so the error of bagged net should approach the bias error. Figures 4 through 7 do indeed show this to be true, which explains why there was so much improvement as a result of bagging. Over forty nets, the variance is almost 0.005, while the bias is 0.0015 (Figures 4 and 5). In Figure 6, we see that the variance of the bagged net is almost zero and the bias error (average of squared bias of each sample over multiple nets) in Figure 7 is almost same as the MSE of bagged net in Figure 2.

Finally, one intuitive way to look at these results is to plot the three test samples with the highest variance. As shown in Figure 8, all of these samples were estimated with reasonable accuracy by most of the nets. (We show just the first twenty nets and the bagged predictor in Figure 8, while the vertical dimension is the error between the true and predicted value.) Certain nets did very poorly on certain samples, however. For example, net #18 had an error of almost 0.9 (in a range from 0 to 1) on one sample. (Nets #13 and #19 also had trouble with this sample.) Another sample proved problematic for net #7, with an error of almost 0.4. Because the aggregate predictor averages across all forty nets, however, these occasional failures - which occur for every net on at least a few samples - are replaced by more reliable

estimates, resulting in an aggregate predictor with far lower error than any of its components.

5 Conclusion

Aggregate or "bagged" predictors work as well or possibly even better on a typical computer vision dataset than they do on the non-visual UCI test suites for which results have previously been reported.

References

[1] L. Breiman, *Bagging Predictors.* Technical Report No. 421, Dept. of Statistics, University of California(Berkeley). Sept. 1994.

[2] T. Dietterich and G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *Journal of Artificial Intelligence Research,* 2:263-286(1995).

[3] D. Heath, S. Kasif, and S. Salzberg, "K-dt: a Multi-tree Learning Method," 2^{nd} *Workshop on Multistrategy Learning,* Chambery, France, 1993, pp.1002-1007.

[4] E.B. Kong and T. Dietterich, *Why error-correcting output coding works with decision trees.* Technical Report, Dept. of Computer Science, Oregon State University (Corvalis), 1995.

[5] E.B. Kong and T. Dietterich, "Probability Estimation via Error-Correcting Output Coding," *IASTED Conference on Artificial Intelligence and Soft Computing,* Banff, Canada, 1997.

[6] H. Kwok and C. Carter, "Multiple Decision Trees," in *Uncertainty in Artificial Intelligence 4,* Shachter, Levitt, Kanal and Lemmer (eds.), North-Holland, 1990, pp.327-335.

[7] E.B. Kong and T. Dietterich, "Error-Correcting Output Corrects Bias and Variance," *Machine Learning: Proceedings of the* 12^{th} *International Conference,* pp.313-321.