PASTING BITES TOGETHER FOR PREDICTION IN LARGE DATA SETS AND ON-LINE

Leo Breiman*
Statistics Department
University of California
Berkeley, CA. 94708
leo@stat.berkeley.edu

Abstract

The size of many data bases have grown to the point where they cannot fit into the fast memory of even large memory machines, to say nothing of current workstations. If what we want to do is to use these data bases to construct predictions of various characteristics, then since the usual methods require that all data be held in fast memory, various work-arounds have to be used. This paper studies one such class of methods which give accuracy comparable to that which could have been obtained if all data could have been held in core and which are computationally fast. The procedure takes small bites of the data, grows a predictor on each small bite and then pastes these predictors together. The methods are also applicable to on-line learning.

*Partially supported by NSF Grant 1-444063-21445

1. Introduction.

Suppose that the data base D is a large collection of examples $(y_n, \mathbf{x}_n)$ where the $\mathbf{x}_n$ are input vectors and the y_n are class labels. What we want to do is use this data to construct a classifier which can accurately assign a class label to future inputs $\mathbf{x}$. But D is too large to hold in the memory of any currently available computer.

What we show in this paper is that the aggregation of many classifiers, each grown on a training set of modest size N selected from the data base D, can achieve almost optimum classification accuracy. The same idea can also be used in the prediction of numerical outputs (regression). The procedure, which we refer to as pasting bites together, has two key ideas in its implementation:

i) Suppose that up to the present, K predictors have been constructed. A new training set of size N is selected from D by unweighted (bagging) or weighted (arcing) sampling. The (K+1)st predictor is grown on this new training set and aggregated with the previous K. The aggregation is by voting in classification and averaging in regression.

ii) An estimate e(K) of the generalization error for the Kth aggregation e(K) is updated. The pasting stops when e(K) stops decreasing. The estimate of e(K) can be gotten using a test set, but our implementation uses out-of-bag estimation (Breiman[1996a]).

CART is used as a test bed predictor, but it is clear that pasting works with many prediction methods. In Section 2 we explore two versions in classification-pasting arcbites and pasting bagbites. Using bagbites is less complicated, but pasting arcbites gives considerably more accuracy. We experiment on five moderate sized data sets. The accuracy of pasting arcbites is compared to trees grown using the entire data set and to arcing such trees using the whole data set (Breiman[1996b]). In Section 3 pasting arcbites together is applied to on-line learning using a synthetic data set as a test bed. Section 4 looks at pasting bagbites in regression. Comments, conclusions, and plans for future research are given in Section 5.

2. <u>**Pasting Bites In Classification**</u>

2.1 <u>Method Description</u>

The simplest version of pasting is to select each training set of size N by random sampling from the data base D, grow the classifier, repeat a preassigned number K of times, stop and aggregate the classifiers by voting. This is certainly workable and cheap. If, after aggregation, accuracy is checked on a test set, then further runs can be used to optimize the values of K and N. A more sophisticated version estimates e(k) after the kth aggregation and stops when e(k) stops decreasing.

There are three methods that can be used to estimate e(k):

<u>First</u>: Set aside a fixed test set of examples in D. Run the kth aggregated classifier on the test set. Estimate e(k) by the error on this test set.

<u>Second</u>: If T is the (k+1)st training set, let r(k) be the error rate of the kth aggregated classifier on T. Since N is small, r(k) with be a noisy estimate of e(k). Smooth it by defining e(k)=p*e(k-1)+(1-p)*r(k). The value p=.75 was used in all of our experiments, but results are not sensitive to the value of p.

If the total number of examples used in the repeated sampling of training sets gets above an appreciable fraction of the number in D, the second estimate will be biased downward because some of the examples in the (k+1)st training set will have been used to construct the previous classifiers.

<u>Third</u>: To eliminate the bias in the second method, if T_h is the hth training set, and C($\mathbf{x}$,h) the classifier for input vector $\mathbf{x}$ constructed using T_h, then classify an example (y,$\mathbf{x}$) that is a candidate for the (k+1)st training set by aggregating all of the classifiers C($\mathbf{x}$,h), h<k+1, such that (y,$\mathbf{x}$) is not in T_h. This is the out-of-bag classifier $C^{OB}(\mathbf{x},k)$. Estimate the error r(k) as the proportion of missclassifications made by C^{OB} . Smooth the r(k) as in the second method to get the estimate $e^{OB}(k)$.

Out-of-bag (OB) estimation is shown in Breiman[1996a] to give effective estimates of the generalization error. (Regarding OB estimation, see also Wolpert[1996] and Tibshirani[1996]) In the examples we run, both the test set $e^{TS}(k)$ and the estimate $e^{OB}(k)$ are computed and compared. Also compared are two methods for selecting the examples for the (k+1)st training sets.

<u>Bagging:</u> This is simple random selection from D with all examples having the same probability of being selected. Continue until N examples are selected. The classifier grown on this training set is called a **bagbite**.

<u>Arcing:</u> In this procedure, an example (y,$\mathbf{x}$) is selected at random from D with all examples having the same probability of being selected. Let $C^{OB}(\mathbf{x},k)$ be the out-of-bag classifier at stage k. If $y \neq C^{OB}(\mathbf{x},k)$ then put this example in the training set. Otherwise, put it in the training set with probability e(k)/(1-e(k). Repeat until N examples have been collected. Refer to the classifier grown on this training sets as an **arcbite**.

The rational for the arcing selection is that the new training set will contain about equal numbers of missclassified and correctly classified examples (see Appendix). Arcing was introduced in Breiman[1996b] as an acronym standing for **a**daptive **r**esampling and **c**ombin**ing.** Although the algorithm for arcing used here differs essentially from those used in the 1996b paper, it retains the basic idea--put increased weight on those examples more likely to be missclssified. The first effective arcing algorithm is due to Freund and Schapire [1995,1996] and is called boosting by them (see also Drucker and Cortes[1996], and Quinlan[1996]) It is referred to in Breiman[1996b] as arc-fs. Bagging is introduced in Breiman[1996c].

In our experiments, N is taken to be a few hundred examples out of data sets having up to 43,500 examples. In the four of the data sets we use in our experiments arc-fs CART was shown to have a lower overall error rate than any of the other 22 well-known classification methods reported on in the Statlog Project (Michie et.al.[1994]). The 5th data base is the famous Post Office handwritten digit data on which arc-fs CART is competitive with hand tailored neural nets.

Two surprising and gratifying things emerge in the experiments

1) Pasting together arcbites, each one grown using only a few hundred examples, gives accuracy comparable to running arc-fs using the whole data set at each iteration.

2) The computing times to construct the pasted classifiers are very nominal, making the procedure computationally feasible even for large data bases.

This procedure works for large data bases. Pasting never requires storage of the entire data base D in core. Examples are selected, tested, and pulled out to form the small training sets. The K trees constructed to date need to be stored. For J class data, this takes about (7+J)*K*N bytes--very workstation feasible. The trees produced are not pruned--the aggregation seems to eliminate the overfitting. Each tree construction takes order NlogN flops but the selection of the arced training sets adds additional computational burden.

2.2. Pasting Arcbites.

The data sets used these experiments are summarized in Table 1.

Table 1 Data Set Summary

Data Set	Classes	Inputs	Training	Test
letters	26	16	15,000	5,000
satellite	6	36	4,435	2,000
shuttle	7	9	43,500	14,500
dna	3	60	2,000	1,186
digit	10	256	7,291	2,007

The first four of these data sets were used in the Statlog project and are described in Michie et.al.[1994]. The last data set is the well-known handwritten digit recognition data set. The data exists as separated into test and training set. The results of 10cv CART and arc-fs CART are given in Table 2. On the digit data 100 iterations were used in arc-fs. The other arc-fs results are based on 50 iterations.

Table 2 Test Set Error (% Missclassification)

Data Set	10cv-CART	Arc-fs CART
letters	12.4	3.4
satellite	14.8	8.8
shuttle	.062	.007
dna	6.2	4.2
digit	27.1	6.2

To track the effect of N, the size of the training sets, we ran pasting on all data sets with N=100,200,400,800. The number of total iterations was chosen, in each data set, to be past the point of decreasing test set error. In letters, this was 1000, 500 for the digits data, 250 for the satellite data, 100 for the dna data, and 50 for the shuttle data. In these runs, we kept track of the test set error, the OB estimates and compute times.

2.2.1 Test Set Error

The resulting test set errors are shown in Figures 1a-e which give graphs of the test set error $e^{TS}(k)$ at the kth iteration as a function of k. Each graph contains plots of $e^{TS}(k)$ for N=100,200,400,800. These can usually be distinguished by the fact that for a given value of k, test set error is highest for N=100, and decreases to N=800. The higher horizontal line is the error rate for 10cv-CART; the lower for arc-fs CART.

These conclusions can be read from the graphs:

i) The test set error falls rapidly as the number k of iterations increases, and then reaches an asymptotic value for large k.

ii) The accuracy in pasting arcbites together is similar to that of arc-fs, especially for the larger values of N. For instance, the test set error percentages at the end of the N=800 runs are: letters, 3.8 %, satellite 8.7%, shuttle .007%, dna 3.8%, digit 6.5%.

iii) In all of the data sets, using even the smallest training sets (N=100,200) gave test set error comparable to or lower than 10cv-CART after a small number of iterations.

iv) The effect of increasing N is data dependent. In the dna and shuttle data sets, N=100 gives the same accuracy as N=800. Generally, the asymptotic value is approached faster in the large N runs

v) In all data sets, the major decrease in test set error has occurred by 100 iterations. In the letters data set, it appears as though the error is decreasing out to K=1000, although the decrease is small after K=200. In the other data sets, the decrease has stopped by K=200.

2.2.2 Monitoring and Selecting Using the OB Estimates.

Suppose that the out-of-bag test set estimates $e^{OB}(k)$ are used to monitor the pasting procedures in the sense that we stop when the values of $e^{OB}(k)$ become flat and select the pasted classifier corresponding to the lowest value of $e^{OB}(k)$ seen to date. Then if we decide to stop after k iterations, the true test set error will be $e^{TS}(h(k))$ where $h(k)=\text{argmin}\{e^{OB}(h), h=1, ... ,k\}$. The loss in using this method depends on how close or far apart the values of $e^{TS}(k)$ and $e^{TS}(h(k))$ are.

Graphs 2a-e give plots of $e^{OB}(k)$, $e^{TS}(k)$, and $e^{TS}(h(k))$ vs. k for each of the five data sets for N=200. The plot of $e^{OB}(k)$ can be recognized as the noisiest. The plots of $e^{TS}(k)$ and $e^{TS}(h(k))$ lie of top of each other. In all 5 data sets $e^{TS}(k)$ and $e^{TS}(h(k))$ differ very little. Using the out-of-bag estimates to select a pasted classifier works well. However, one thing that emerges from these graphs is that while e^{OB} tracks e^{TS} quite well, it may be systematically low or high. There are two reasons for this bias:

i) For the same classifier, the "true" test set classification error may differ from the "true" training set classification error.
ii) The out-of-bag estimates, early in the iterations, tend to overestimate the test set error rate.

One way we have of checking on i) is to apply out-of-bag estimation to the entire training data instead of to the little bite data, and compare this estimate to the test set error rates. Based on this check, and looking at the pasted classifiers for large K and N=200, it appears that the test data has an error rate about 3.0% higher than the training set for the digit data; 1.5% lower than the training data for the letters data; .07% less than the training data for the shuttle data and 1.5% less than the training data for the dna data. Test and training error rates appear about equal in the satellite data.

In the initial stages of pasting, the out-of-bag estimates are usually systematically high. One obvious cause is that since $e^{OB}(k)$ is a smoothed version of past out-of-bag estimates, it reflects the earlier and higher values of the error. The less obvious is this: in data sets of moderate size with low missclassification error, examples that are prone to be missclassified are used over and over again in the arcbite training sets. These examples will tend to be out-of-bag in a relatively small number of the arcbites. Therefore, their missclassification rate will be elevated.

Both of the two sources of overestimation by e^{OB} taper down as the number of iterations increase. In fact, in all of the checks that we have run of e^{OB} versus the unsmoothed out-of-bag estimates based on the whole training set show that they converge together as the number of iterations increase.

Another question in using arcbites is how big to take N. In general, it seems that the bigger, the better. But taking N larger also slows down the compute time. The out-of-bag estimates can be used to resolve this issue, since modulo a possible offset due to systematic bias , they will track the true test set error consistently. That is, looking at the out-of-bag estimates for different N will give a fairly accurate idea of how much one buys by using larger N. To illustrate, Figures 3a-e are graphs of $e^{OB}(k)$ vs.k for all 4 values of N for the five data sets. In the data sets where there are significant differences in test set error with N, the order of the OB test set error estimates is the same as the order of the true test set error. In the shuttle and dna data, where there is very little effect as N changes, the OB estimates for N=800 are curiously higher than for the other N.

2.2.3 Compute Times

Table 3 gives the compute times per 100 iterations for the five data sets by training set size. The times are scaled to a SUN Ultrasparc 2 and do not include the time to load the data.

Table 3 Compute Time Per 100 Iterations (Minutes)

Data Set	N=100	200	400	800
letters .	15	.24	.59	.84
satellite	.06	.12	.26	.55
shuttle	.40	.48	.62	.70
dna	.10	.18	.43	.47
digit	.34 `	.92	1.94	3.61

These times were computed from the total elapsed time of the run. Thus, if the run was 500 iterations, the time was divided by 5. The selection of the arced training sets can be an appreciable proportion of the computing time.

2.3 Pasting Bagbites

Pasting bagbites does not work as well as pasting arcbites. To illustrate, bagging was used on the five data sets to form training sets of size N=200. The number of iterations was set at the upper limits specified in Section 2.2. The final test set error was divided by the corresponding test set error for pasting arcbites. These ratios are given in Table 4.

Table 4 . Ratio of Test Set Errors (Bagbites/Arcbites)

letters	satellite	shuttle	dna	digit
2.41	1.32	73.95	1.26	1.68

These ratios are disappointingly large and show that pasting bagbites is not competitive with pasting arcbites in terms of accuracy..

3. Semi-Infinite Data Sets and On-Line Learning.

In situations where there is a steady flow of new examples being formed, incremental or on-line learning research has focused on the continual updating of a prespecified architecture. For instance, given a flow of examples how is a neural net with specified architecture updated as each new example occurs? Or given a binary tree structure, find efficient ways to update the tree as new examples are added.

What we propose instead is to do on-line learning by the steady pasting on of new bites. Thus the architecture grows as the information flows in. The drawback is that the storage requirements cannot be set in advance. Bites are added until an asymptote is reached. But this is offset by two important advantages--accuracy and speed. As seen in the previous section, pasting arcbites gives low test set error. Further, the compute times required per example are small.

Semi-infinite data sets and on-line learning have one thing in common--the possibility of a sampled example being sampled again is zero or negligible. This simplifies the algorithms. In forming test set error estimates, one can assume that none of the incoming examples have been previously used in a training set. Similarly, in deciding whether an incoming example is correctly classified or not, we can assume that it has not previously been used as a training example.

The on-line procedure generates arced training sets of size N in the way similar to the Section 2 method. But instead of dealing with a fixed data base D, there is a flow of incoming examples.

With semi-infinite data bases, the incoming example is randomly sampled from the data base. As the new examples come in, they are checked to see if how they are classified by the current pasted classifier. If missclassified, they are accepted into the training set. If not, they are accepted with probability $e^{TR}/(1-eTR)$. Continue this procedure until there are N examples in the training set. The error estimates $e^{TR}(k)$ are computed as the smoothed version of r(k), where r(k) is the proportion of missclassified examples found in forming the kth training set.

To study on-line performance, 10 class, 61 input synthetic data is manufactured (see Appendix). Before going on-line, a test set of 5000 examples is generated. At the kth iteration, both $e^{TS}(k)$ and $e^{TR}(k)$ are computed. Again, we use N=100,200,400, 800. Figure 4 a shows the test set errors vs. the number of iterations out to 250 iterations . Figure 4b plots the values of e^{TR} for the 250 iterations. All the test set plots show a rapid decrease to an error of about 20% followed by a slow decrease. Although we stopped at 250 iterations, the test set error was still slowly decreasing. To see what happens if we kept going, we did the N=400 pasting out to 750 iterations. The test set error dropped from 17.7 at 250 to 16.6 at 750.

3.1 A Modification to Bound Compute Times and Memory Requirements

At the (k+1)st iteration each candidate example for the arced training set has to be passed down k trees. The compute time to do this for the kth iteration increases linearly with k, and the total compute time increases quadratically with k. Furthermore, the memory used grows unboundedly.

This is not as problematic with finite data sets, since the out-of-bag condition bounds the number of trees used to classify each candidate example and the size of the run is limited. To explore a possible remedy for the problem in on-line learning , we revised the procedure so that the arced training set is selected based only on the missclassifications in the pasting together of the last NBACK trees where NBACK is set by the user. The compute times now increase linearly. At 500 iterations the total compute time using NBACK =100, and N=400 is 8.3 minutes compared to 22.0 minutes for the unrestricted arcing.

The final test set error is similar--17.2% for the unrestricted arcing vs. 17.4% for the restricted version. One drawback is that the e^{TR} estimates in the restricted arcing now reflect the accuracy of the last NBACK bites, instead of all bites. Therefore they have a pessimistic bias. In the original procedure, the final value of e^{TR} is 18.0%. In the restricted procedure it is 19.8%. Further experiments are needed to see if the NBACK restriction gives generally satisfactory results. If it does, then the memory requirements for the on-line learning can be bounded.

3.1 Out-Of-Bag Can't Be Ignored For Finite Data Sets

Keeping track of which examples are in and out-of-bag is an extra complication. A natural question is what happens if this complication is ignored and all incoming examples treated as if they had never been seen--that is, everything is out-of-bag. If the data base is semi-infinite, so that duplicate examples in the training sets used in the pasting is infrequent, then this works. But if the data base is finite in the sense that duplication is not infrequent, then performance degenerates.

One consequence is that the $e^{TR}(k)$, being resubstitution estimates, are increasingly biased low and generally go to zero as the iterations continue. Then the arced training sets have to be filled with fewer and fewer missclassified examples and the procedure bogs down. For instance, using the letters data and N=400, it bogged down at around 90 iterations with a test set error of 10. 3, almost twice as large as the error gotten using out-of-bag .

One way to try and prevent bogging down is to set the acceptance level for correctly classified examples at a preset level instead of at $e^{TR}/(1\text{-}eTR)$. This was tried using the acceptance level .062 based on the asymptotic error (.0584) for letters at N=400. This did slightly better, not bogging down until about 200 iterations. The test set error at that point was 9.22%, about 60% higher than the out-of-bag procedure asymptotic test set error rate.

4. **Pasting Bites in Regression**

In regression, we did not find an effective analogue of arced training sets. Therefore, our experiments were done using bagbites. At the kth iteration, a training set of size N is randomly selected from the data set. A regression tree is grown on this training set. For any new set of inputs **x** the predicted output is the average of the k tree predictions. A running estimate $e^{OB}(k)$ of the mean-squared generalization error is computed by by smoothing the raw-out-of-bag estimates r(k).

To get r(k), for each example **x** in the kth training set, compute its predicted value as the average of the predictions by all previous trees such that **x** was not in the training set used to construct the tree. Then r(k) is the average sum-of-squares of the differences between the actual response corresponding to **x** and its predicted value. The out-of-bag estimate $e^{OB}(k)$ is computed as $e^{OB}(k) = .95e^{OB}(k)+.05r(k)$. The heavier smoothing reflects the larger variance of r(k) in regression as compared to classification.

The data used in our regression experiment was made available to us by Michael Jordan. The inputs consist of 12 measurements on the configuration of a robot arm. There are 4 output accelerations. The training set has 15,000 examples with a separate 5000 example test set. We used the first output acceleration in our testing. We ran both a single 10cv tree on this data and 50 iterations of bagging. The test set error (multiplied by 100) is 6.93 for the single tree and 4.75 for the bagged trees.

We used training sets of size 100,200,400,800 and ran 200 iterations. The results are graphed in Figure 5. The higher horizontal line is the the single tree error rate, the lower is the bagged tree error rate. Note that for training set sizes greater than 100, the test set error drops below the single tree error after a small number of iterations. But the pasted predictors do not get down to the error level of the 50 trees bagged on the whole data sets.

Figure 6 graphs $e^{OB}(k)$ vs. k for N=100,200,400 and 800. These estimates are somewhat noisy due to the large standard deviation to mean ratio of the individual training set OB estimates. However, their relative sizes for different N reflect fairly well the relative sizes of the test set error for N, and they flatten out where the test set error flattens. Choosing the best model on the basis of the minimum value to date of the out-of-bag estimates tracks the minimum test set error closely.

The computing of the regression pasted bites goes rapidly. In particular, there is no time spent testing candidate examples for inclusion in the training set. The compute ime per 100 iterations is .76 minutes for the N=800 case and is roughly proportional to N.

4. Comments, Conclusions, and Further Research

Our prime conclusion is that pasting arcbites together is a promising approach to constructing classifiers for large data sets and for on-line learning. On the data sets we experimented with, it is fast, accurate, and has modest memory requirements. Its performance raises interesting issues and possibilities.

The increase in accuracy using arcbites as compared to bagbites is newsworthy. It emphasizes that in classification, concentrating on the examples near the classification boundaries pays off in terms of reduced generalization error. The arcing approach used here is differs considerably from the Freud-Schapire algorithm. In arc-fs sampling weights for the entire training set are updated in terms of the classifications done by the last classifier constructed. In pasting arcbites, the sampling for the new training set is done on the basis of the classifications of the entire past sequence of classifiers and the current estimate of test set error.

Important questions regarding the present procedure are: i) After K iterations, K trees are being stored. Are these all necessary? How can we decide, on the fly, which are redundant and deletable? ii) More generally, given K trees acting as a single classifier by voting, can the pasted classifier be stored in a more compact form? iii) Is there a good way of deciding, after the Kth tree has been constructed, whether to paste that tree onto the previous trees or to discard it and try again?

The process of creating arcbites does not depend on the classifier used--it is simply a method of forming the successive training sets. This leads to a question that will be explored in the near future: suppose that there are a number of classifications methods available in our repertoire, i.e. trees, nearest neighbor, neural nets, etc. Suppose that each of these is run on the Kth arced training set and one is selected to paste onto the previous classifiers. Can this significantly upgrade performance over the use of a single type of cl;assifier?

While the results on regression show that pasting bagbites gives predictors more accurate than growing a single tree on the whole data set, there is a loss in accuracy as compared to bagging in the whole data set. This contrasts with pasting together arcbites in classification. It may be that we don't know yet how to select appropriate training set in our bites.

References

Breiman, L. [1996a] Out-of-bag Estimation, submitted to Machine Learning (available ftp.stat.berkeley.edu/users/pub/breiman/OOBestimation.ps)

Breiman, L. [1996b] Bias, Varaince, and Arcing Classifiers, submitted Annals of Statistics (available in the above ftp directory as arcall.ps)

Breiman, L. [1996c] Bagging Predictors , Machine Learning 26, No. 2, 123-140

Michie, D., Spiegelhalter, D. and Taylor, C. [1994] Machine Learning, Neural and Statistical Classification, Ellis Horwood, London

Drucker, H. and Cortes, C. [1996] Boosting decision trees, Neural Information Processing 8, Morgan-Kaufmann, 1996

Freund, Y. and Schapire, R. [1995] A decision-theoretic generalization of on-line learning and an application to boosting. http://www.research.att.com/orgs/ssr/people/yoav or http://www.research.att.com/orgs/ssr/people/schapire

Freund, Y. and Schapire, R. [1996] Experiments with a new boosting algorithm, "Machine Learning: Proceedings of the Thirteenth International Conference," July, 1996.

Quinlan, J.R.[1996] Bagging, Boosting, and C4.5, Proceedings of AAAI'96 National Conference, on Artificial Intelligence

Tibshirani, R.[1996] Bias, Variance, and Prediction Error for Classification Rules, Technical Report, Statistics Department, University of Toronto

Wolpert, D.H. and Macready, W.G.[1996] An Efficient Method to Estimate Bagging's Generalization Error

Appendix

a) Acceptance Level.

Suppose that a classifier has probability p of incorrectly classifying an example and let r be the rejection probability. Let q=1-p. Consider the following scheme: pick an item at random and classify it. If incorrect accept it into the sample. If not, then reject it with probability r. The event that the first item accepted will be a missclassifed item is the union over k of the disjoint events that k-1 rejections of a correctly classified item occur followed by the acceptance of a missclassifed item. The probability of this event is $p+p(qr)+p(qr)^2 +p(qr)^3 + \dots$. The sum equals p/(1-qr). Putting this quantity equal to 1/2 and solving for 1-r gives p/(1-p). Thus, setting the acceptance level at p/(1-p) leads to a sample that is approximately half missclassified and half correctly classified items.

b) Synthetic Data.

Let w(k,m) be a triangular function in m, with a maximum of 10 at m=10k, becoming zero at 10k-10,10k+10, and zero for m outside this range. There are 10 distinct subsets of size 3 of the integers {1,2,3,4,5}. Assigning each of these subsets to a class and denote by T(j) the subset assigned to class j. Then the 61 dimensional inputs corresponding to class j are given by:

$$x(m) = \sum_{k \in T(j)} z_k w(k,m) + u_m$$

where the u_m are independent mean zero normals with sd=10, and the z_k are independent mean zero normals with sd=1.

The data for each of the classes has a multivariate 61-dimensional mean-zero normal distribution. The optimal classification scheme for this data is quadratic discrimination, and the curved classification boundaries are hard for trees to approximate.

LETTER DATA

FIGURE 1a. TEST SET ERROR (%)

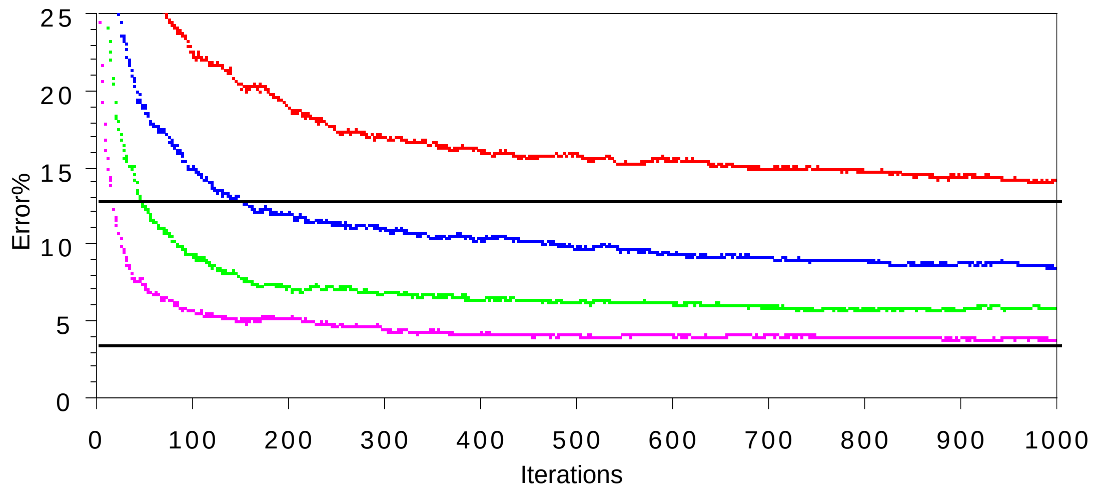


FIGURE 2a. OB, TEST SET, AND OB MINIMUM ERROR ESTIMATES

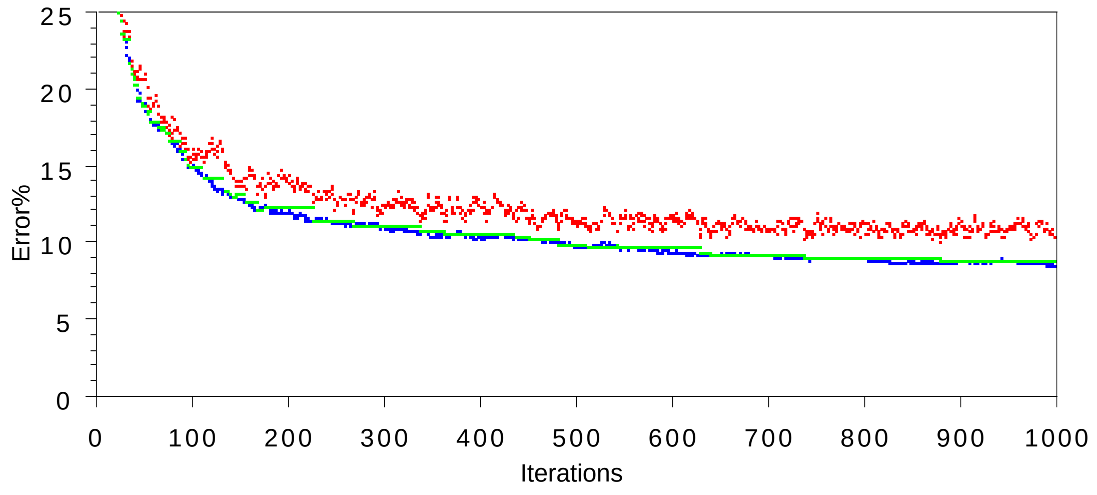
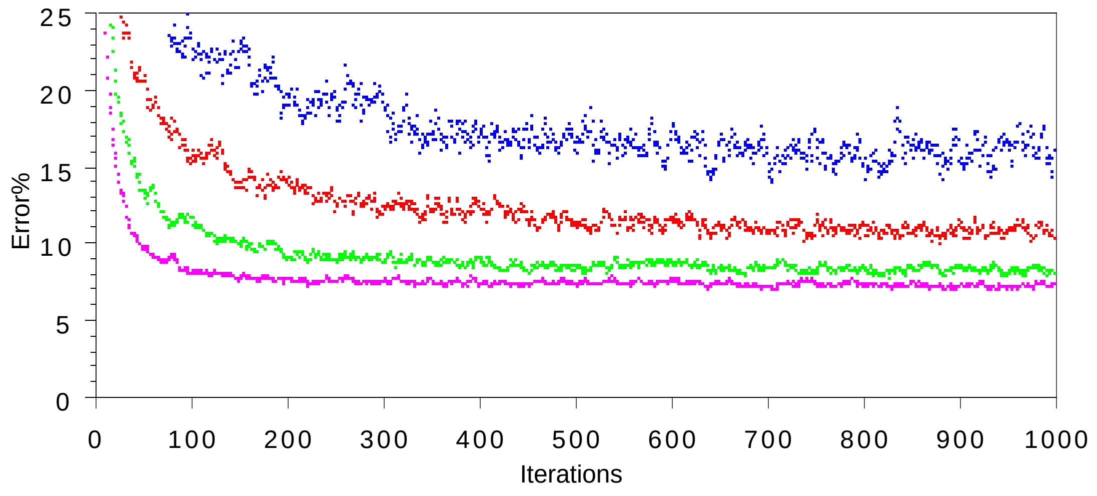


FIGURE 3a OB ERROR ESTIMATES



SATELLITE DATA

FIGURE 1b. TEST SET ERROR%

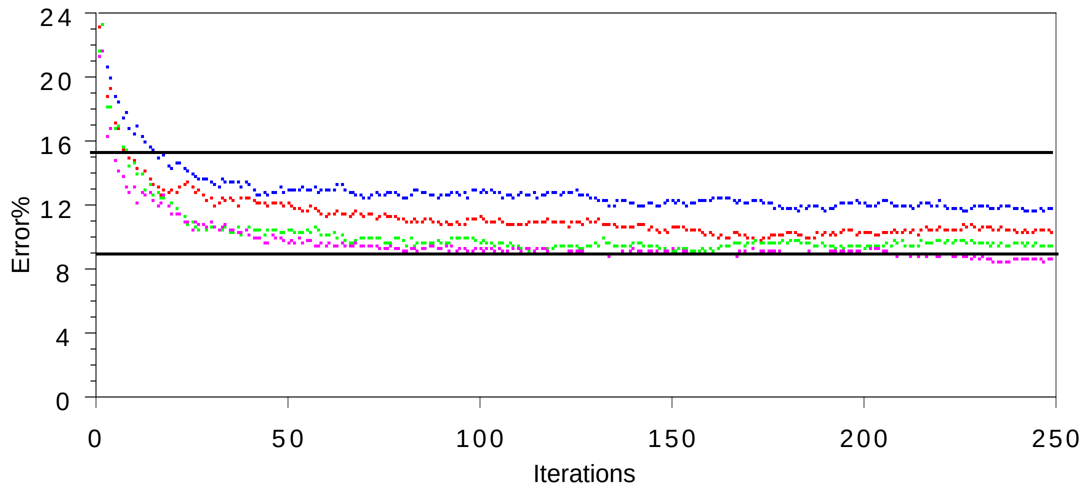


FIGURE 2b. OB, TEST SET, AND OB MINIMUM ERROR ESTIMATES

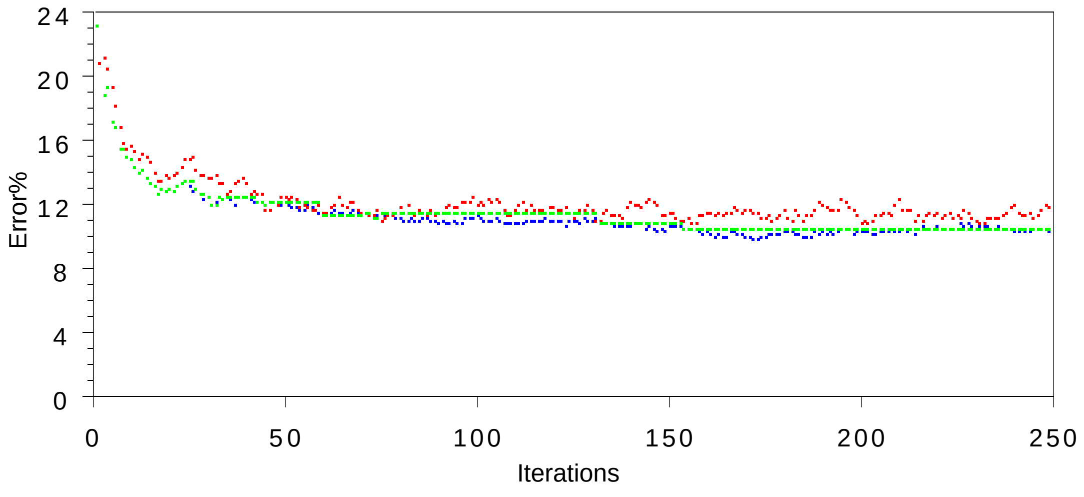
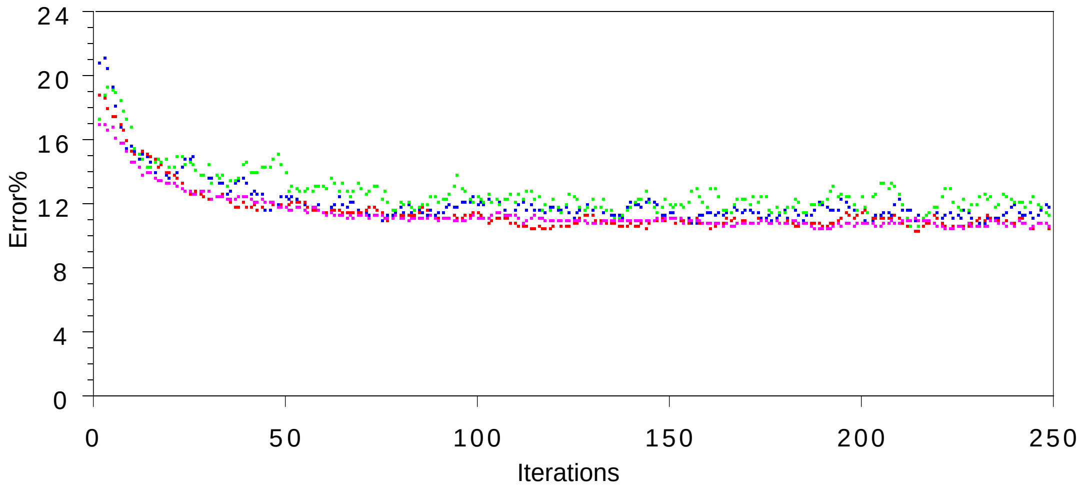


FIGURE 3b. OB ERROR ESTIMATES



. SHUTTLE DATA

FIGURE 1c TEST SET ERROR%

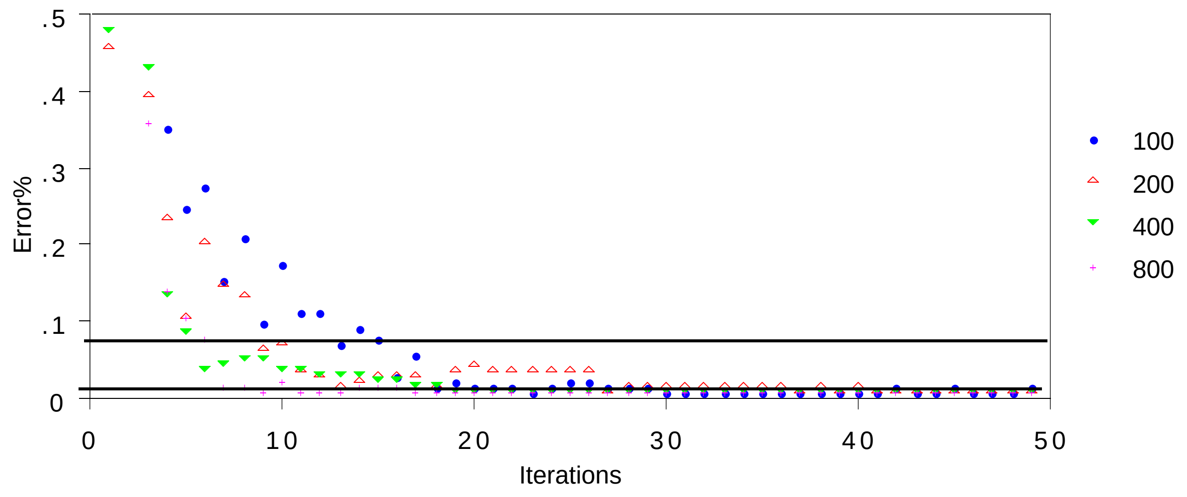


FIGURE 2c OB,TEST SET AND OB MINIMUM ERROR ESTIMATES

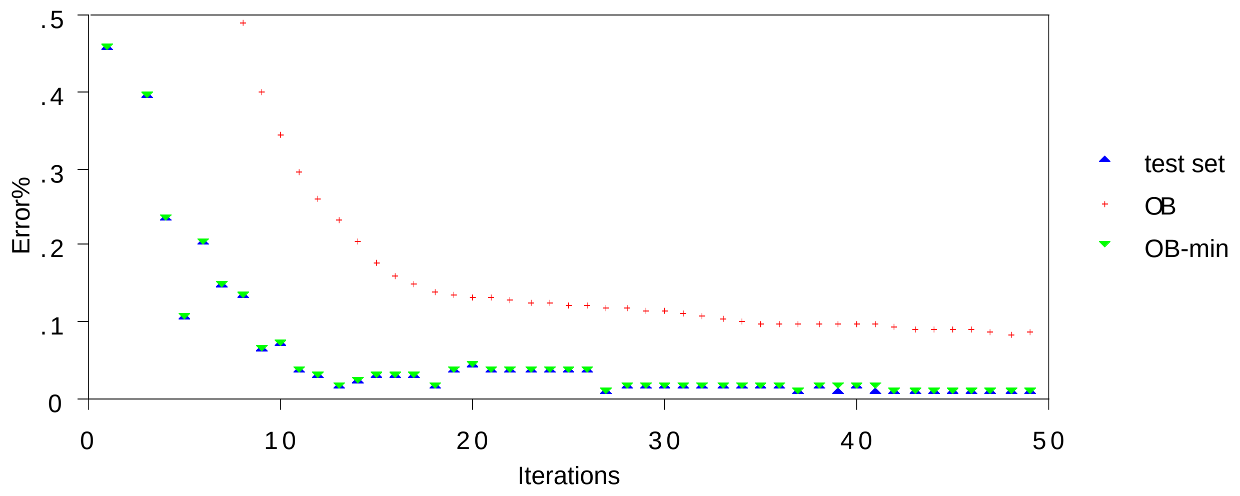
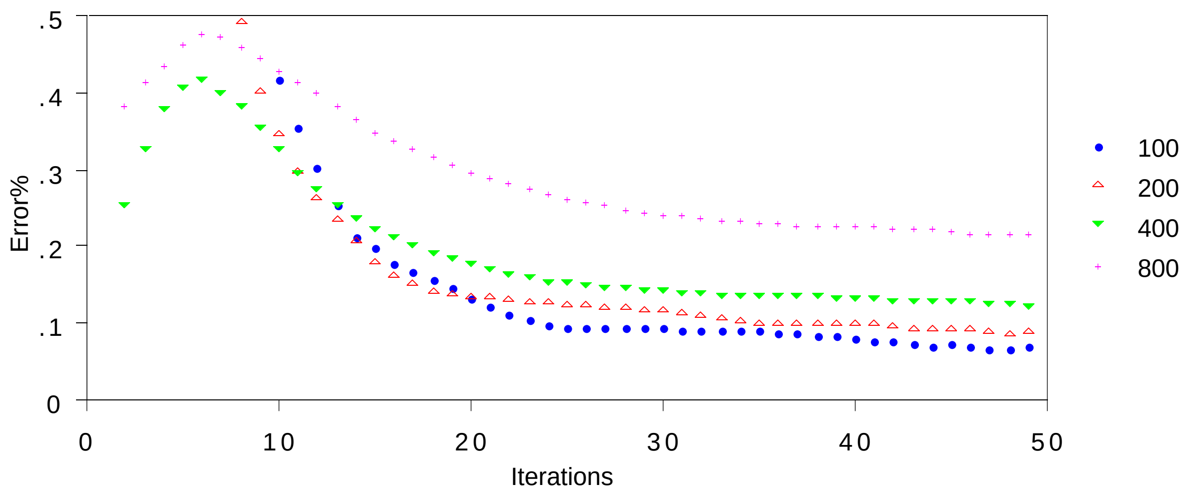


FIGURE 3c OB ERROR ESTIMATES



DNA DATA

FIGURE 1d TEST SET ERROR%

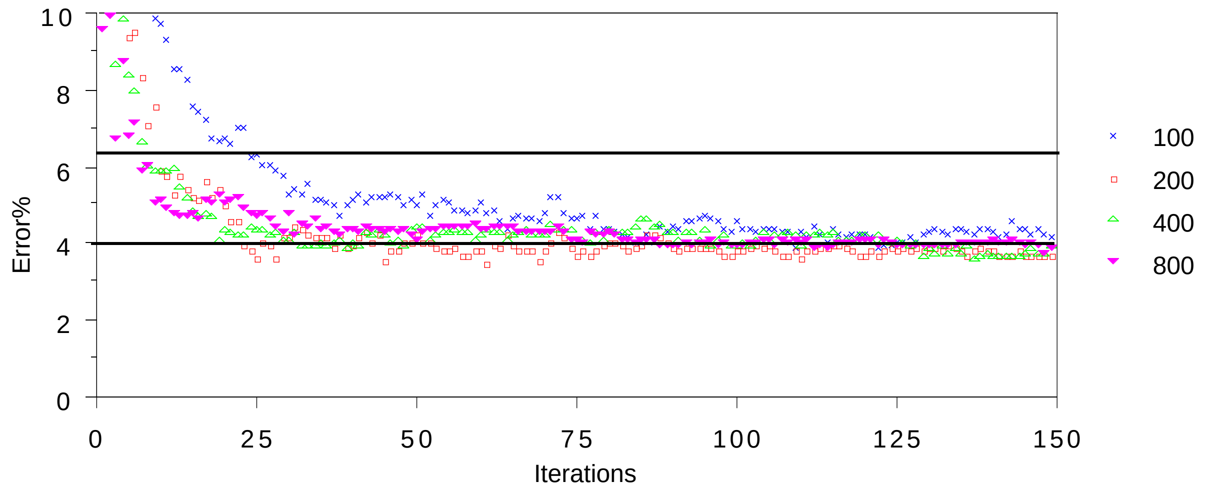


FIGURE 2d OB, TEST SET, AND OB MINIMUM ESTIMATES

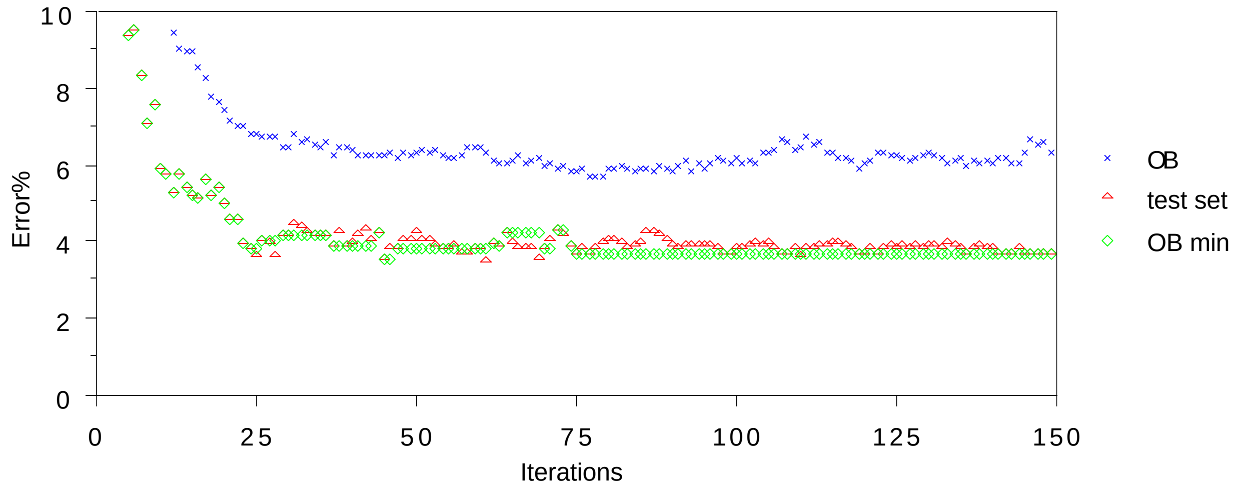
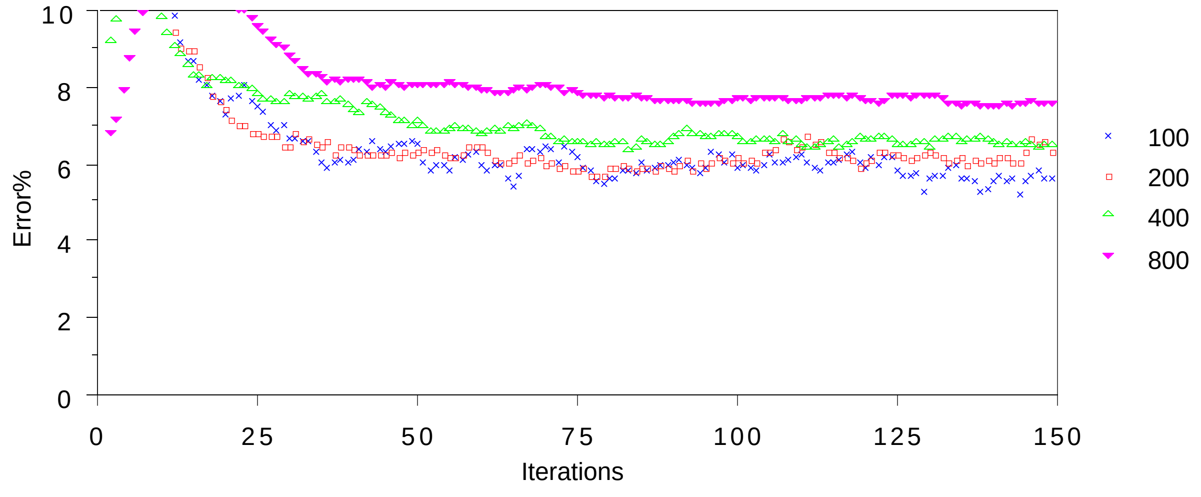


FIGURE 3d OB ERROR ESTIMATES



DIGIT DATA

FIGURE 1e TEST SET ERROR%

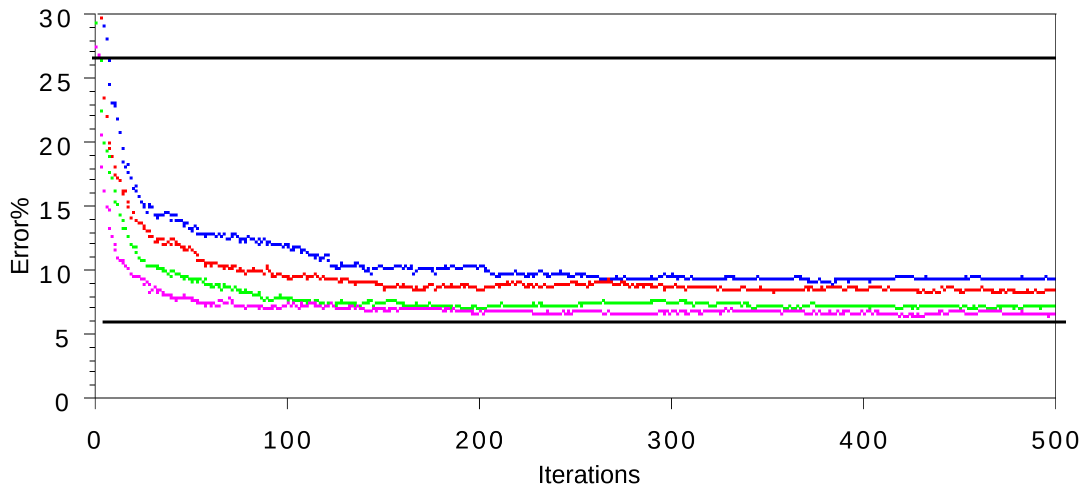


FIGURE 2e OB, TEST SET, AND OB MINIMUM TEST SET ESTIMATES

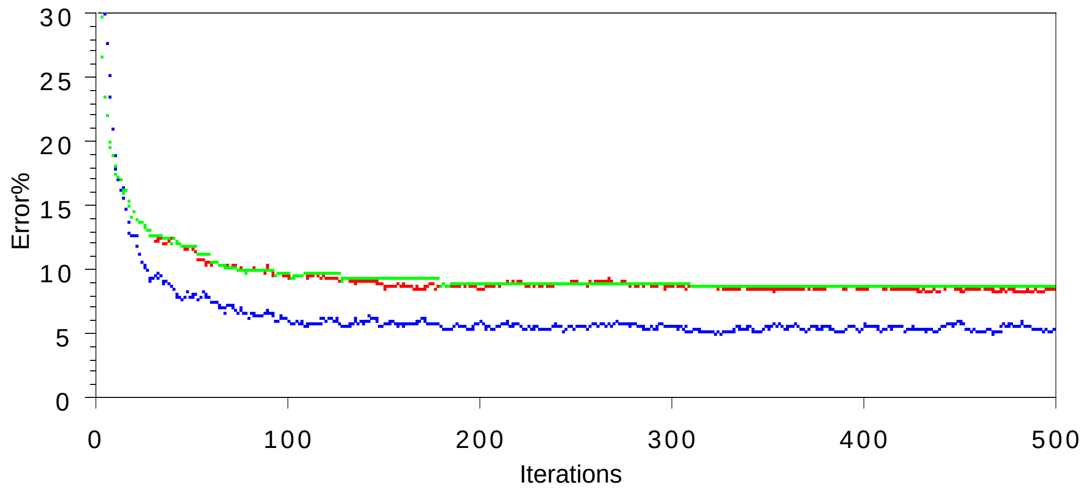
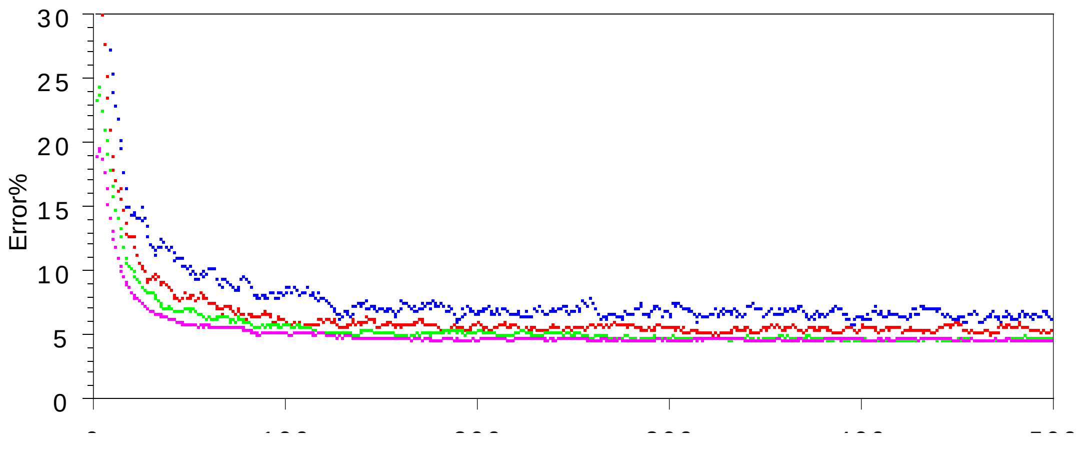


FIGURE 3e OB ERROR ESTIMATES



ONLINE SYNTHETIC DATA

FIGURE 4a TEST SET ERROR%

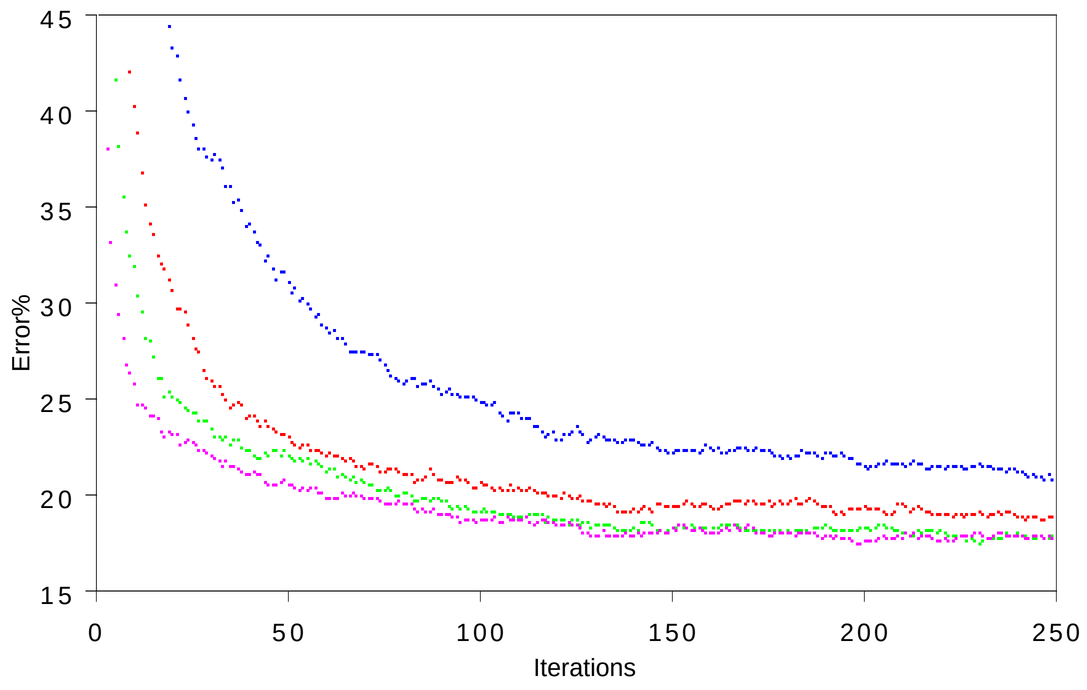


FIGURE 4b TRAINING SET ERROR ESTIMATES

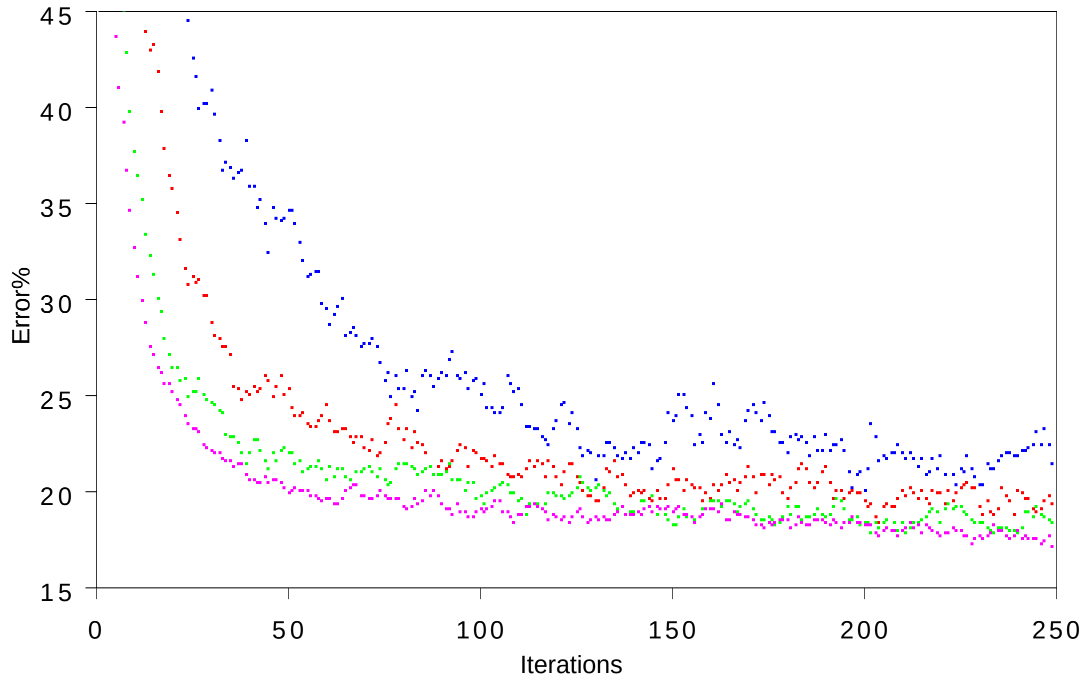


FIGURE 5. TEST SET ERROR

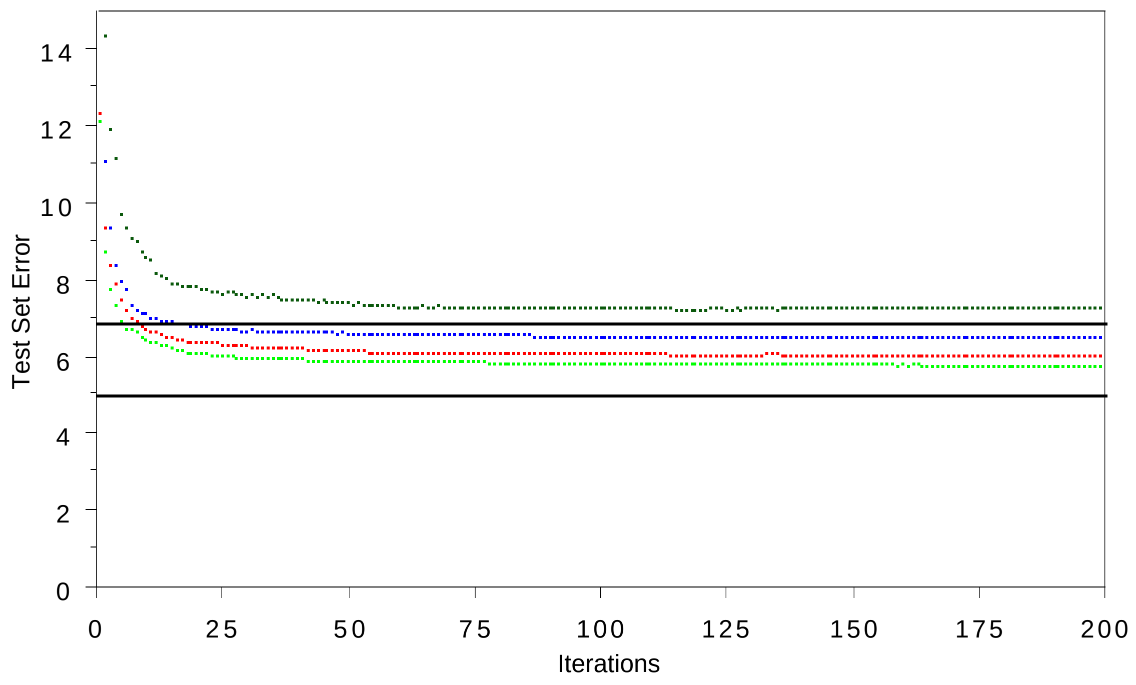
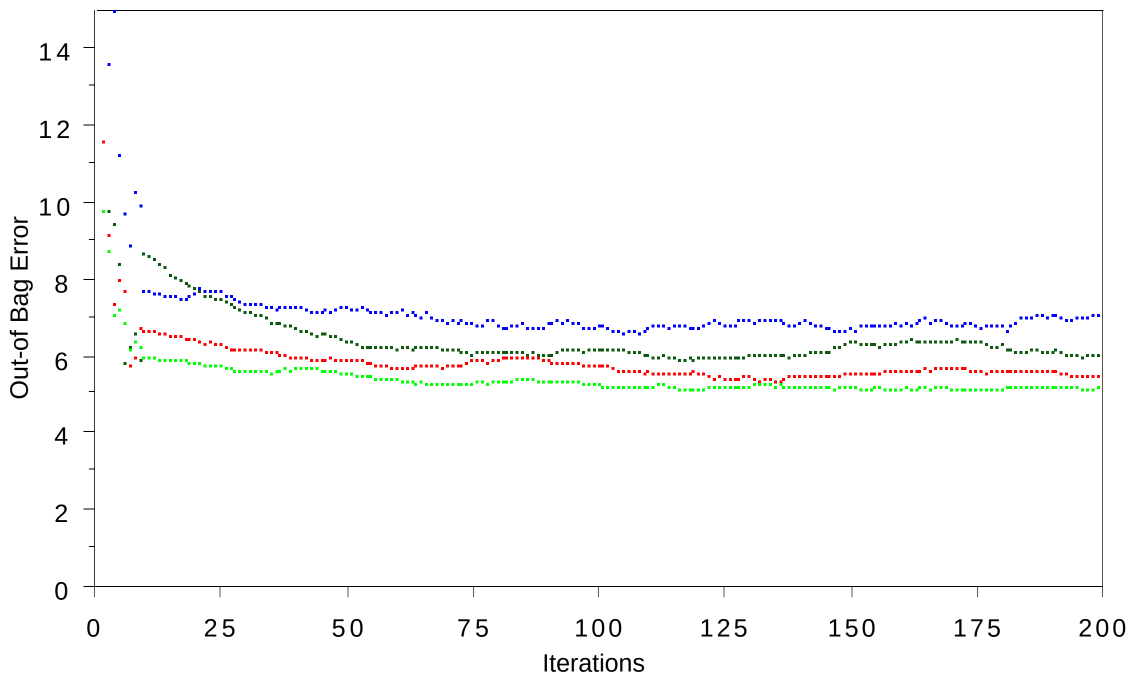


FIGURE 6. OUT-0F-BAG ESTIMATES


Out-of Bag Error
Iterations