Working Paper Series
ISSN 1170-487X

Learning from Batched Data: Model Combination vs Data Combination

by Kai Ming Ting, Boon Toh Low & Ian H Witten

Working Paper 97/14
May 1997

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Learning from batched data: Model combination *vs* data combination

Kai Ming Ting
University of Waikato, Hamilton, New Zealand.

Boon Toh Low
Chinese University of Hong Kong, Shatin, Hong Kong.

Ian H. Witten
University of Waikato, Hamilton, New Zealand.

Abstract

When presented with multiple batches of data, one can either combine them into a single batch before applying a machine learning procedure or learn from each batch independently and combine the resulting models. The former procedure, data combination, is straightforward; this paper investigates the latter, model combination. Given an appropriate combination method, one might expect model combination to prove superior when the data in each batch was obtained under somewhat different conditions or when different learning algorithms were used on the batches. Empirical results show that model combination often outperforms data combination even when the batches are drawn randomly from a single source of data and the same learning method is used on each. Moreover, this is not just an artifact of one particular method of combining models: it occurs with several different combination methods.

We relate this phenomenon to the learning curve of the classifiers being used. Early in the learning process when the learning curve is steep there is much to gain from data combination, but later when it becomes shallow there is less to gain and model combination achieves a greater reduction in variance and hence a lower error rate.

The practical implication of these results is that one should consider using model combination rather than data combination, especially when multiple batches of data for the same task are readily available. It is often superior even when the batches are drawn randomly from a single sample, and we expect its advantage to increase if genuine statistical differences between the batches exist.

Keywords: model combination, data combination, empirical evaluation, learning curve, near-asymptotic performance.

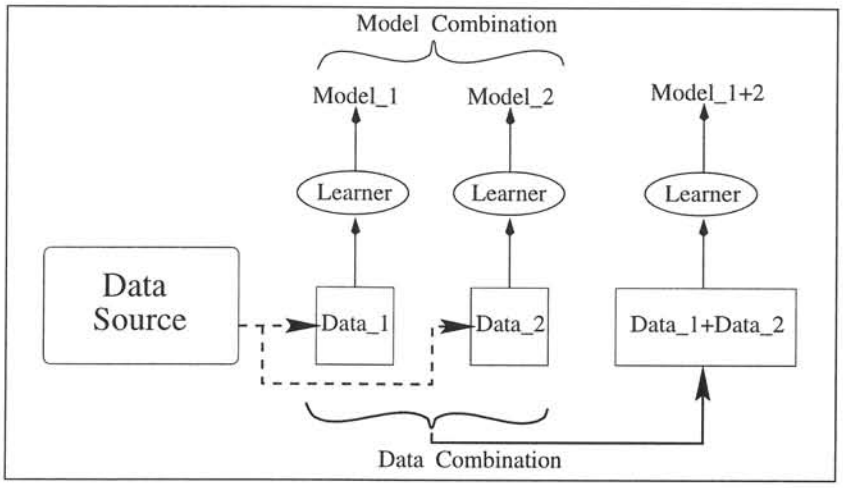


Figure 1: Model combination vs. data combination

1 Introduction

When different batches of data for the same machine learning task are available, the natural approach is to combine them into a single data set and use it to produce a single classifier. This seems intuitively to conform to the conventional wisdom that "the more data the better." However, this paper challenges that notion. We investigate an alternative strategy: to use each batch of data individually to learn a classifier—using the same learning algorithm—and then combine the predictions from these separate classifiers. We call this method "model combination" to distinguish it from the traditional "data combination" approach. Figure 1 illustrates the difference: data combination combines the individual data sets before learning, whereas model combination combines the individual models that have been learned.

It seems clear that this new approach is likely to outperform the old one in certain situations. First, if there is some fundamental difference between the individual data batches, combining them would blur the statistical properties of each batch and perhaps confound the ability to learn regularities that they exhibit. In this case one would expect the advantage of model combination to be more marked as the difference between batches increased—although it may be hard to quantify such differences. Second, if different learning algorithms were used for the batches, there might be an advantage in model combination provided it was done in a way that allowed the superior model to dominate the combination. Third, if a particularly clever method of model combination were used, along with a particularly naive method of learning from the individual batches, the model combination operation might counteract deficiencies in the learners and thereby produce good results.

Surprisingly, we find that model combination can outperform data combination even

in situations where the batches are drawn at random (without replacement) from a single source of data, and even when the same learning algorithm is used for each batch, and even when different methods of model combination are used. This counter-intuitive result is strongly supported by the experimental results reported in this paper.

While research on methods of combining multiple models has been widely reported (e.g., Brodley, 1993; Breiman, 1996a, 1996b; Freund & Schapire, 1996; Perrone & Cooper, 1993; Krogh & Vedelsby, 1995), the combination of models of the same type, induced by a single learning algorithm from completely disjoint sets of data, has received much less attention. Most work shows that combining multiple models induced using either a single learning algorithm or different learning algorithms from a single dataset outperforms a single model induced from that dataset.

This paper examines the situation where multiple batches of data are available for a classification task. The data might have been collected from a family of separate sources, or from the same source in consecutive years, or it might result from different, but related, events, as long as they are for the same task. Alternatively, it might be generated by randomly sampling a single data set. We expect this last condition to be the most challenging for the model combination methodology, for in the other circumstances results will depend on the degree of variation between the batches. Consequently our goal is to determine the conditions under which model combination performs better than data combination in the random-batch scenario.

Our focus here is not on the relative merits of different methods for model combination. Specifically, we address the question of whether model combination, in the multiple-data-batches scenario, is a viable option as compared to data combination in classification tasks. If the answer is yes, when should one use it? In order that the results do not depend on a particular model combination method, we describe three different ones and perform experiments with all of them.

The intuition behind this work is that different batches of data—even if they are produced by random sampling—exhibit some variation of data representation in the description space. Models, or theories, induced separately from these independent batches become "specialists" in different regions of the space. Model combination allows cooperation between these specialists. Combining the specialists, if done in a plausible manner, can smooth over the variation that the individual models represent and form a representation that is robust and valid globally.

Combining models in this way applies very naturally to an incremental batch learning scenario where a model is derived from each batch of data as it arrives. Alternatively, if a large database of examples is already available, the multiple-data-batches scenario may be simulated using sampling, random or otherwise.

Some researchers have investigated the combination of independently-learned multiple

models which are formed by varying the *induction bias* of the learning algorithm to generate models with uncorrelated errors. For example, one might either vary the parameters of a single learning algorithm, or use different types of learning algorithm. Other researchers use sampling methods to create multiple overlapping data subsets from a given dataset. Section 2 reviews this work on the use of multiple models to enhance performance.

In Section 3 we describe three different ways of combining models. The first and simplest, majority vote, is used in most of the previous work. The second involves the use of an *a priori* "measure of characterization" that is used to estimate a model's predictive accuracy, the quantitative relationship between the two being calibrated using a cross-validation technique on the training data. The third, stacked generalization, involves a meta-level learner that learns the circumstances under which each model's output should be used. The results in this paper show that all three methods of model combination can, under certain circumstances, outperform data combination. This follows from our central hypothesis, presented in Section 4, that the relative performance of model combination and data combination can be explained with reference to the learning curve of the algorithm used. This hypothesis gives, in qualitative terms, the conditions under which model combination outperforms data combination for randomly-chosen batches. Section 4 also describes the parameters of an experiment designed to test this hypothesis. The experimental results are reported in Section 5, and discussed in the following section. In Section 7 we briefly consider some further issues; Section 8 summarizes our conclusions.

2 Related work

The use of multiple models generated from training sets that were derived from a single dataset by different sampling methods is a popular topic of current research. For example, in bagging, training sets are generated by sampling the dataset with replacement (Breiman, 1996a). In boosting, a sequence of training sets is generated and successive classifiers are built for them (Freund & Schapire, 1996; Quinlan, 1996; Breiman, 1996b). The first training set is formed by equal-weighted sampling but subsequent ones are weighted in favour of instances that are misclassified by the classifiers built so far; the process is repeated iteratively in an attempt to improve performance. The general approach has been called "adaptive resampling and combining" by Breiman (1996b). Ali and Pazzani (1996) use k-fold partitioning to generate k models by training on all but the ith partition k times. In all these approaches the models that are generated are combined using the method of voting or weighted voting. Ali and Pazzani (1996) also investigate other combination methods such as Bayesian combination, distribution summation, and likelihood combination. They all differ from the approach taken in the present paper in that each sample dataset contains most, or all, of the total instances.

Breiman (1996c) investigates boosting models derived using small bites of the entire dataset. In this formalism, like ours, the multiple models are produced from a single learning algorithm. However, we do not use the adaptive resampling that lies at the heart of boosting.

An alternative way of producing multiple models is to vary the parameters of a single learning algorithm. For example, multiple neural networks can be generated by using different initial random weight configurations and/or orders of training data (Hansen & Salamon, 1990; Perrone & Cooper, 1993). Multiple decision trees can be generated by selecting different tests at each node, generating option trees, or pruning a tree in different ways (Kwok & Carter, 1990; Buntine, 1991; Oliver & Hand, 1995). Multiple rules can be generated by stochastic search guided by heuristics (Kononenko & Kovačič, 1992). These models are combined by averaging—possibly using different weights—the outputs of neural networks or the class probabilities of trees, or by using a Bayesian combination of different rules.

Chan and Stolfo (1995, 1997) investigate various methods for combining models, e.g. (weighted) voting, Bayesian combination and stacked generalization (Wolpert, 1992). They show that models learned from disjoint partitions of a dataset can sometimes be combined to outperform a single model learned from the entire dataset. While this work overlaps to some extent with ours, it is only tested on two datasets. Moreover, Chan and Stolfo do not address the conditions under which a combination model is better than a single model learned from the entire dataset.

Many researchers have investigated how to partition the description space. Some use the information gain criterion (Utgoff, 1989), user-provided information (Tcheng *et al.*, 1989), or hand-crafted rules (Brodley, 1993) to guide the recursive partitioning process in a tree structure. Others employ a confidence measure provided by one particular learned model to decide at classification time which of two different models shall be used for the final prediction (Ting, 1994; Wettschereck, 1994). Most of the former methods apply different types of learning algorithm for each of the mutually exclusive partitions, and the latter methods derive different types of model independently using the entire dataset.

Baxt (1992) describes a situation where the data is pre-sorted manually into two groups according to some criterion (e.g. high- and low-risk groups in a medical diagnostic task). He trains separate neural networks on each group of data. During classification, the network trained using the low-risk group is used if its output falls below a certain threshold, otherwise the high-risk network is used. This method is only applicable when appropriate information about the sorting criterion is available.

Provost and Hennessy (1996) describe a distributed approach to learning a single ruleset from rulesets induced from disjoint partitions of a dataset. They ensure that the ruleset is a superset of the rules induced from the entire dataset, and achieve this by

maintaining the invariant-partitioning property during the rule-learning process. This property guarantees that any rule that is satisfactory over the entire dataset will be satisfactory over at least one subset. The goal is to accelerate the process of learning a set of rules that cover the entire dataset, rather than to enhance the performance of the rule set. Brazdil and Torgo (1990) also consider how to convert different models into one.

Most of this work assumes that a single dataset is used to generate models which are then combined. Exceptions are Chan and Stolfo (1997), Provost and Hennessy (1996), Breiman (1996c) and Baxt (1992). Only the first two share our assumption that multiple batches of data for the same task are available without any prior information about them.

3 Combining multiple models

The central phenomenon reported in this paper, that models built from individual batches of data can be combined into one that outperforms a single model built from the whole data set, has been observed using three distinct methods of model combination.

3.1 Majority vote

The simplest way of combining models is by majority vote, where the output of the combined model for a particular example is the majority class predicted by the individual models. This method is used in most of the work reported above.

3.2 Model combination using a measure of characterization

We have previously investigated the use of an *a priori* measure that characterizes predictive accuracy as a basis for model combination (Ting, 1996). The measure we choose depends on the particular learning algorithm used to produce the models. In this paper, we consider both the NB* (Naive Bayes) learning algorithm and the IB1* (instance-based) learning algorithm. For NB*, the measure of predictive accuracy is simply the posterior probability. For IB1*, the measure—which we call "typicality"—is the inter-concept distance divided by the intra-concept distance. For an instance with class p, the inter-concept distance is defined to be its average distance to instances of classes other than p; and the intra-concept distance is its average distance to other instances of the same class.

In this procedure for model combination, we first calibrate the measure of characterization of each learning algorithm in terms of its observed predictive accuracy on the training set. The aim of the calibration procedure is to find an empirical function relating classification accuracy to the characterization measure, so that the accuracy for a new test example can be predicted from its measure. Cross-validation is used to provide a

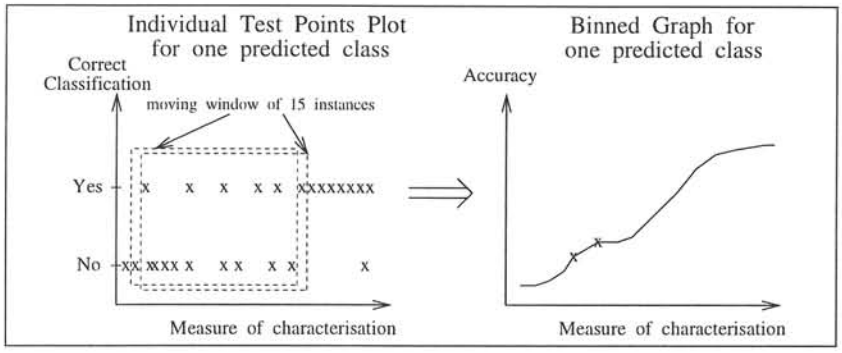


Figure 2: Transforming individual test points to a function that predicts accuracy

reliable estimate of accuracy. A separate empirical function is found for each class in each model, that is, for each batch of data.

The calibration procedure is as follows. Given a learning algorithm (in our case, one of NB* and IB1*) and a batch of data, a k-fold cross-validation is performed. In k-fold cross-validation, the data batch is partitioned into k equal-size subsets, and training is performed using all subsets except the ith, which is used as the testing set. The procedure is repeated for all values of i, that is, k times. Thus by the end of the cross-validation, there will be one (and only one) test result for each instance of the original data batch. We set k to three in our experiments.

For each predicted class, the individual instances' test results from the k-fold cross-validation are sorted according to the values of the characterization measure. The left part of Figure 2 illustrates the result. On the x-axis is the characterization value (i.e., posterior probability for NB*, or typicality for IB1*), and each data point corresponds to a single training example. In order to produce a function that allows accuracy to be estimated from characterization value, illustrated in the right part of the Figure, a sliding window containing a fixed number of instances is used, and the average predictive accuracy is calculated from the instances within the window and plotted on the vertical axis. The process is repeated by sliding the window along, dropping the leftmost instance and adding one in on the right.

Once calibration is complete, a model is induced for each batch of data from all the instances in it. This, along with the accuracy prediction function for each class value, is stored and used for future classification. The procedure is repeated for the other batches.

Now consider the classification process. Given an instance whose class is unknown, it is fed into the model induced from each data batch. This model will predict a particular class, and the likely accuracy of the prediction can be estimated from the prediction function for that class value. Thus each model generates a suggested class and an accuracy figure. The model which claims the highest accuracy is chosen for the final prediction.

3.3 Model combination using stacked generalization

The third method of model combination used in this work is the technique of "stacked generalization," which is a way of combining multiple models that have been learned for a classification task by using a learning algorithm to arbitrate between their predictions (Wolpert, 1992; Ting & Witten, 1997). In our case the models are formed from the individual data batches; again, we conduct separate tests using different learning algorithms.

The first step in stacked generalization is to collect the output of each model into a new data set. For each instance in the entire data set, we seek every model's prediction on that instance. The only snag is that the instance will lie in a batch of data from which one of the models was formed, and the prediction of its class by that particular model is suspect because it was one of the examples used to form the model. To eliminate this bias, a cross-validation procedure is used for the instances that lie in the batch from which the model was formed. For these instances, the batch is divided into k folds and when finding predictions from instances in the ith fold, a model formed from the other $k-1$ folds is used. For the remaining instances—ones lying outside the batch from which this model was formed—there is no need for cross-validation; the model's output is used directly.

The result of the first step is a predicted output from each model for every instance. In conventional stacked generalization this is used as data for the second step, which is treated as another learning problem. However, in previous work we have found that rather than using categorical predictions from the models as input to the second step, it is better to use probabilistic outputs instead (Ting & Witten, 1997). Thus the output of a particular model on a particular instance is not the predicted class as described above, but a vector containing the prediction probability for *each* class. In the case of NB*, the Naive Bayes learner, the prediction probability for a class is simply the posterior probability for that class. For IB1*, the instance-based learner, we take an average value for the class over p nearest neighbors (we use $p = 3$ in the experiments), weighted by distance in instance space. Denote the nearest neighbors of instance x by $\{(y_s, x_s), s = 1, \ldots, p\}$. Then the prediction probability for the class ℓ is

$$P_\ell(x) = \frac{\sum_{s=1}^{p} f(y_s)/d(x, x_s)}{\sum_{s=1}^{p} 1/d(x, x_s)},$$

where $f(y_s) = 1$ if $\ell = y_s$ and 0 otherwise, and d is the Euclidean distance function.

Thus the final result of the first step is a vector of class probabilities, for each model and for each instance in the entire data set. For a particular instance, if there are r models and I classes, the vector will contain rI real-valued probabilities. Since it is training data, it will also contain the true class, a categorical value.

Now consider the second step: the higher-level learning task, whose input is the $(rI + 1)$-component vectors just described, one for each instance. For this task we use a

multi-response linear regression algorithm, MLR. This is an adaptation of a least-squares regression algorithm that Breiman (1996d) used for stacked generalization in regression settings. Any classification problem with real-valued attributes can be transformed into a multi-response regression problem. If the original classification problem has I classes, it is converted into I separate regression problems, where the problem for class ℓ has responses equal to one for instances with class ℓ, and zero otherwise.

During training, for each possible class, rI regression coefficients are chosen to minimize the squared error for that class on the training data. The coefficients are constrained to be non-negative, following Breiman's (1996d) discovery that this is necessary for the successful application of stacked generalization to regression problems. The non-negative-coefficient least-squares algorithm described by Lawson & Hanson (1995) is employed to derive the linear regression for each class. During testing, an instance of unknown class is used to derive weighted linear sums for each class, and the class for which the sum is greatest is chosen as the predicted one.

A formal statement of the stacked generalization procedure is given in the Appendix.

4 Hypothesis and experimental design

The central hypothesis of this paper is this: *the relative performance of model and data combination can be predicted from the operating point on the algorithm's learning curve as follows: data combination is superior in the early part of the learning curve and model combination in the later part.*

Consider first the data combination method. At the beginning of the learning curve, when the training data size is small, the curve is steep and so this method will give a large gain in performance. As the curve approaches the asymptote, however, additional data only improves performance marginally. One can roughly define the near-asymptotic region as the region where doubling the training data size yields only a small performance gain (further quantification of this notion is not useful for our present purposes). The effect of doubling the training data size in two different regions of a learning curve is shown graphically in Figure 3.

The reverse is true for model combination. When little data is available, models have very high variance (we quantify this notion precisely in Section 5.3). Combining models together cannot reduce the variance enough to match the performance gain achieved by data combination. However, as the training data size increases, model combination reduces the variance faster than does data combination. Thus with large amounts of data, model combination will achieve better performance gain than data combination.

We have designed experiments to illustrate the learning curves of different learning algorithms and model combination methods for a selection of standard datasets.

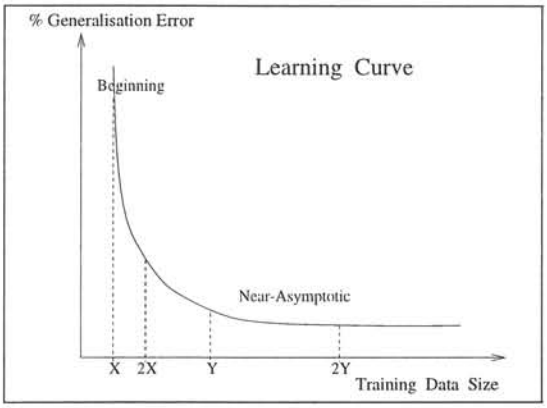


Figure 3: Performance gain that results from doubling the training data in two different regions of a learning curve

Number of models We chose to use the minimum viable number of models, three for the majority vote method and two for the other methods, in order to investigate the simplest possible situation. Our intention is only to provide an understanding of the basic phenomenon, not to quantify the trade-offs involved when data is split into different numbers of batches. Moreover, some empirical studies indicate that, other things being equal, a combination of two classifiers outperforms combinations of larger numbers (Chan & Stolfo, 1995; 1997)—a result that also holds when the combination models are stacked up to form a tree, i.e., binary trees are better than higher-order trees (Chan & Stolfo; 1997).

Model combination methods We employ the three model combination methods reviewed in Section 3. For comparison we also use an "oracle" combination method which always makes the correct prediction from the constituent models if one exists. For noisy datasets, the oracle can perform better than the best possible realizable model combination method. For example, given a random dataset with two equally probable classes, the oracle can achieve an accuracy of 0.75 in the limit; while the optimal Bayes accuracy for this data set is 0.5.

Learning algorithms We use three inductive learning algorithms in our experiments: IB1* and NB* (Ting, 1994; 1997) and C4.5 (Quinlan, 1993). IB1* is a variant of IB1 (Aha, Kibler & Albert, 1991) that incorporates the modified value-difference metric (Cost & Salzberg, 1993). NB* is an implementation of the Naive Bayes algorithm (Cestnik, 1990). These two algorithms include a method for discretizing continuous-valued attributes during a preprocessing stage (Fayyad & Irani, 1993), a procedure that improves their performance in most of the continuous-valued attribute domains studied by Ting (1994). In IB1* we use the class of the nearest neighbor as the predicted class, and the

default parameter settings are the same as those used by Aha's implementation.[1] No parameter settings are required for NB*. C4.5 is used with default settings.

Data sets We use standard data sets from the UCI repository (Merz & Murphy, 1996). Two artificial domains are chosen, *waveform* and *LED24*, along with four real-world datasets, *euthyroid*, *nettalk(stress)*, *splice junction*, and *protein coding*.

The artificial domains were introduced by Breiman *et al.* (1984) and exhibit controlled amounts of noise. *Waveform* has three uniformly distributed classes, and each instance contains twenty-one relevant and nineteen irrelevant continuous-valued attributes. Each class consists of a random convex combination of two of the three waveforms, with Gaussian noise added. The class in *LED24* is one of the ten digits and there are seven boolean attributes indicating whether or not each segment of a seven-segment display is illuminated. Each attribute value is inverted with a probability of 0.1.

The *euthyroid* dataset is one of the sets of thyroid examples from the Garvan Institute of Medical Research in Sydney described by Quinlan *et al.* (1987). It contains 3163 case data and diagnoses for euthyroidism, one of the many thyroid disorders. Eighteen binary attributes and seven continuous-valued attributes are involved. The task is to predict whether a patient suffers euthyroid or not.

The goal of the NETtalk task is to learn to pronounce English words by studying a dictionary of correct pronunciations (Sejnowski & Rosenberg, 1987). Each letter is presented to the classifier together with a context of three preceding and three succeeding letters. The goal is to produce the phoneme and stress values that constitute the pronunciation of the letter. The *nettalk(stress)* dataset of 5438 instances is for the prediction of one of five levels of stress, and forms part of the NETtalk corpus of the 1000 most common English words.

The *splice junction* dataset contains 3177 instances of sixty sequential DNA nucleotide positions (Towell *et al.* 1990). Each position can have one of four base values. The task is to recognize for a particular a DNA sequence two types of the splice junction or neither.

The *protein coding* dataset also contains DNA nucleotide sequences; here the classification task is to differentiate the coding sequences from the non-coding ones (Craven & Shavlik, 1993). Each sequence has fifteen nucleotides with four different values each. This dataset contains 20,000 sequences.

[1] IB1 stores all training instances. It uses maximum differences for attributes that have missing values, and computes the Euclidean distance between any two instances.

5 Experimental results

In order not to become overwhelmed with experimental results we focus attention on one of the three methods of model combination introduced in Section 3, model combination using a measure of characterization (Section 3.2). (We have chosen this because it is the one that we have done most work on ourselves.) We begin by reporting the experiments using the artificial domains. Next we show that the same phenomena are observed with the real-world datasets. Following that we examine the difference between data and model combination in terms of the bias and variance of the individual classifiers. Finally we look more briefly at the two other methods of model combination: majority vote and stacked generalization. These exhibit the same characteristics as before. For the sake of variety we present bias and variance results for majority vote, and learning curves for stacked generalization. Moreover, for these two methods we also show the effect of using C4.5 as the base method, as well as IB1* and NB*.

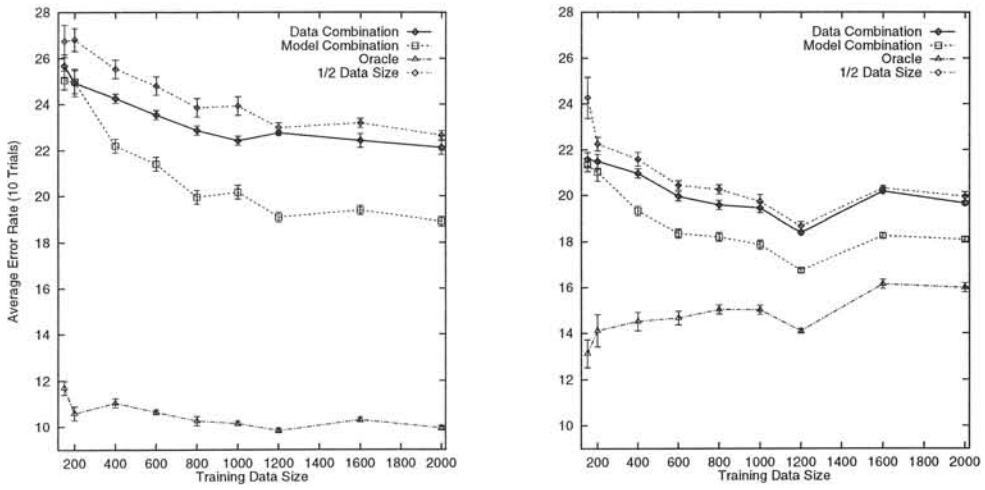
5.1 Artificial domains

To simulate different batches of data, different seeds were used to generate the data in the artificial domains. We use two equal batches for training, varying the batch size from 200 to 2000 in order to operate at different points of the learning curve. A fixed number of 5000 independently-generated instances are used for testing.

For each batch size, the training data is used to generate models for both IB1* and NB*. With model combination, two models are produced, one for each data batch. With data combination, the batches are concatenated into one training set, which is used to produce a single model. The procedure is repeated ten times, using different random number seeds, for each batch size; and the average error rate is reported along with its standard error.

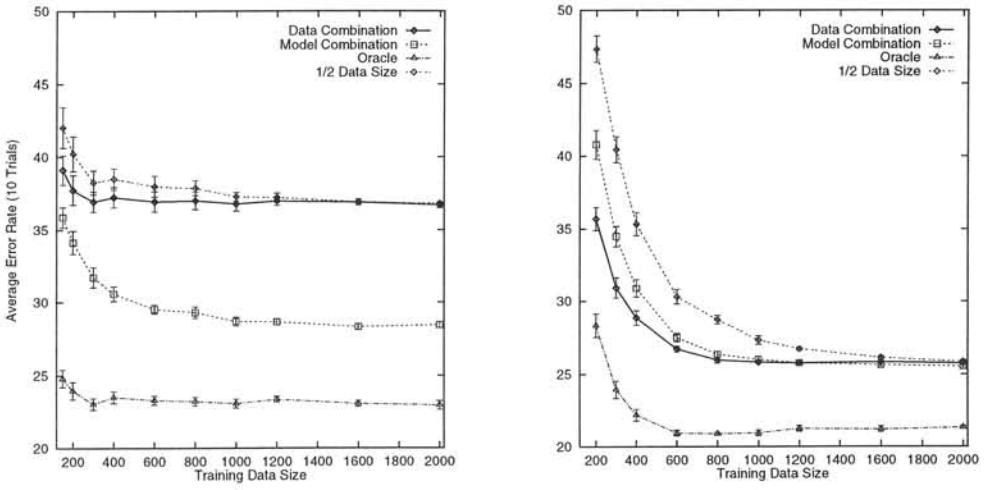
Figures 4 and 5 show the learning curves that result from these experiments. Plots (a) and (b) in each figure show the results using IB1* and NB* respectively. As noted above, the model combination method used is based on the measure of characterization as discussed in Section 3.2. Note that in these (and other) graphs, the vertical scales are chosen to maximize use of space and are different in each figure: care should be taken when comparing the graphs with each other.

Along the horizontal axis is the size of the training data. The single model induced from a batch containing half this training data is designated as "1/2 Data Size"—there are, of course, two such models, but they perform so similarly that just one is chosen for display. The single model induced from the whole training data is labeled "Data combination". The result of the oracle, which makes an incorrect prediction only if both models predict incorrectly, is also shown.



(a) IB1* (b) NB*

Figure 4: Learning curves (average error rate) for the *waveform* domain



(a) IB1* (b) NB*

Figure 5: Learning curves for the *LED24* domain

We summarize the results depicted in Figures 4 and 5 as follows. When using IB1*, model combination performs significantly better than data combination in both the *waveform* and *LED24* domains at almost all the training data sizes used. We regard two average error rates as being "significantly" different if they differ by more than two standard errors (that is, with $\geq 95\%$ confidence). Similar performance is observed for NB* in the *waveform* domain. Note that in these three cases, the performance improvement of data combination over the single model induced from half of the training data is small.

We expect the performance improvement of model combination over data combination to increase as the asymptotic region of the learning curve is approached. For the one situation in which data combination sometimes outperforms model combination, namely NB* in the *LED24* domain (Figure 5(b)), data combination performs significantly better than model combination when using small amounts of training data, and becomes marginally worse as the data size increases. In all cases, both model combination and data combination uniformly outperform the constituent models used for model combination. The oracle bounds the optimal performance for model combination and is always by far the best of the four curves.

It is interesting that in the *LED24* domain IB1* seems to reach near-asymptotic performance from 800 training instances onwards (Figure 5(a)). But combining models learned from half this much data can nevertheless significantly improve performance. Results for the oracle show that the two models differ significantly, because the oracle improves performance over its constituent models by as much as 15%. We will return to this point in Section 6.

5.2 Real-world datasets

For each of the four real-world datasets, two different batches are simulated by randomly sampling the training data into disjoint subsets of equal size. The size of the training data is varied from 10% to 90% of the entire dataset. Experiments for each data size are repeated over 20 trials, except in the case of the *protein coding* dataset for which only 10 trials are used because of its enormous size. The testing data is the remaining part of the dataset that is not used for training.

Figures 6 to 9 show the results for each dataset. In the first three, *euthyroid*, *nettalk(stress)* and *splice junction*, the performance trends of model combination and data combination generally accord with the results observed in the artificial domains. In all three datasets, model combination performs worse than data combination at the beginning of the learning curve. As the asymptotic region of the *euthyroid* dataset is approached, model combination performs better using NB* and comparably using IB1*; as the asymptotic region of the other two datasets is approached, model combination performs better using IB1* and

comparably using NB*.

For the *protein coding* dataset using IB1*, shown in Figure 9(a), the trends seem to suggest that the near-asymptotic region of the learning curve does not appear in the figure because even the biggest batches are insufficiently large. Model combination is significantly better than data combination when the training set contains between 10% to 50% of the available data, a trend that is reversed when the training data sizes are 80% and 90%. Although this appears to contradict our central hypothesis, we attribute it to initial transient behaviour: insufficient data is available to reach the asymptotic region. When NB* is used (Figure 9(b)), data combination outperforms model combination at the beginning of the curve, and they perform comparably in the near-asymptotic region.

5.3 Error decomposition

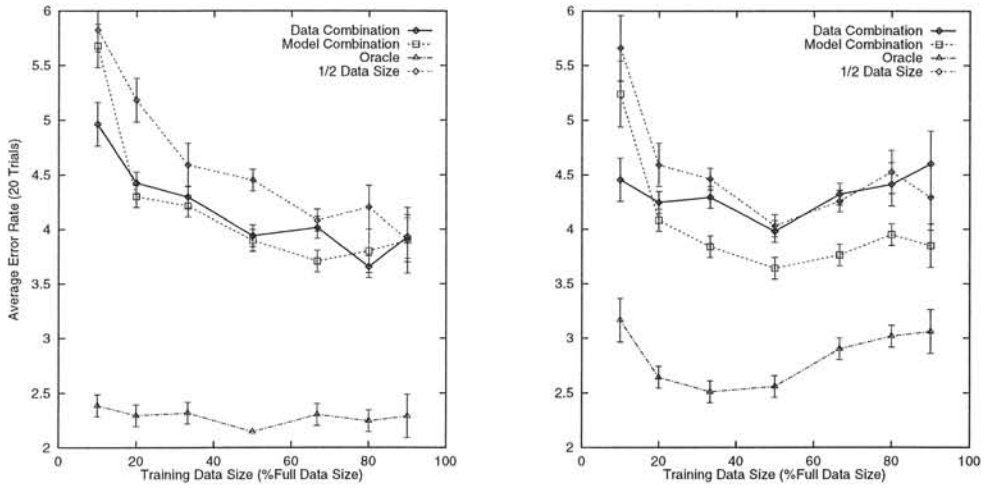
This section examines the phenomenon uncovered in the previous two subsections in terms of the bias and variance of the individual classifiers. Breiman (1996b) has shown how the expected prediction error of a classifier C can be decomposed into three parts: the expected prediction error of an ideal classifier C^*, $PE(C)$, the *bias* of the classifier C, and the *variance* of C:

$$PE(C) = PE(C^*) + Bias(C) + Var(C). \tag{1}$$

The ideal classifier C^* is chosen to be the optimal Bayes classifier. This decomposition is prompted by a well-known decomposition of prediction error in regression, extended to encompass classification problems.

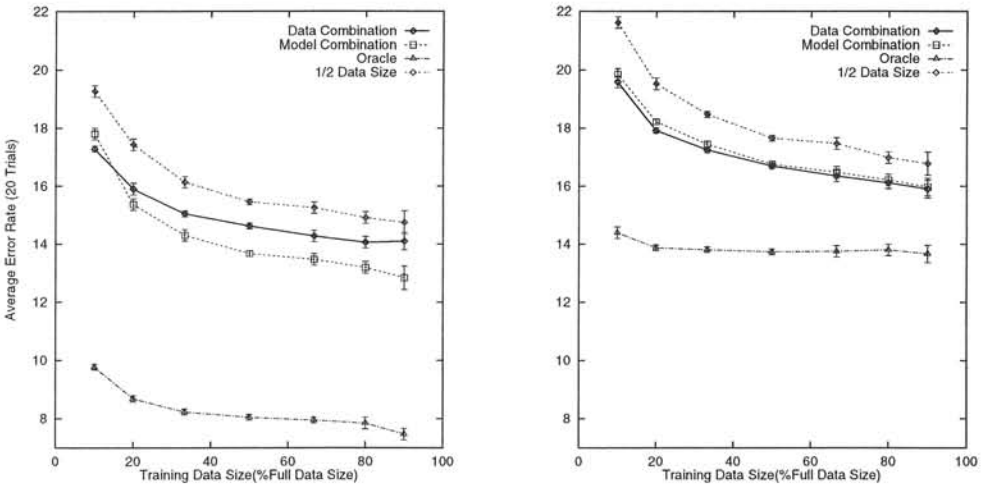
For present purposes, the key notions are the definition of bias and variance for classifiers that lead to the decomposition above. In order to explain these, we first imagine that the training set contains instances (y, x), where x is a vector of attribute values and y is the classification of that instance, that are independent, identically distributed, samples of some underlying probability distribution. We denote by X and Y the random variables that represent attribute vectors and classifications respectively. For the purposes of this discussion, a "classifier" is a function C that takes a training set $\mathcal{L}$ and a vector x of attribute values and produces a predicted class value $y = C(x, \mathcal{L})$. We are really interested in results that are independent of the particular training set chosen, and this is achieved by taking the expectation over all training sets $\mathcal{L}$.

Now we can turn to the bias and variance of an arbitrary classifier C. First we define the *bias set* B to be the set of instances x for which C is biased, that is, for which an aggregated classifier C_A gives a prediction that differs from that of the optimal Bayes classifier, $C_A(x) \neq C^*(x)$. C_A is aggregated by voting over 50 classifiers C in our experiments. U, the complement of B, is the set of instances x for which the classifier C



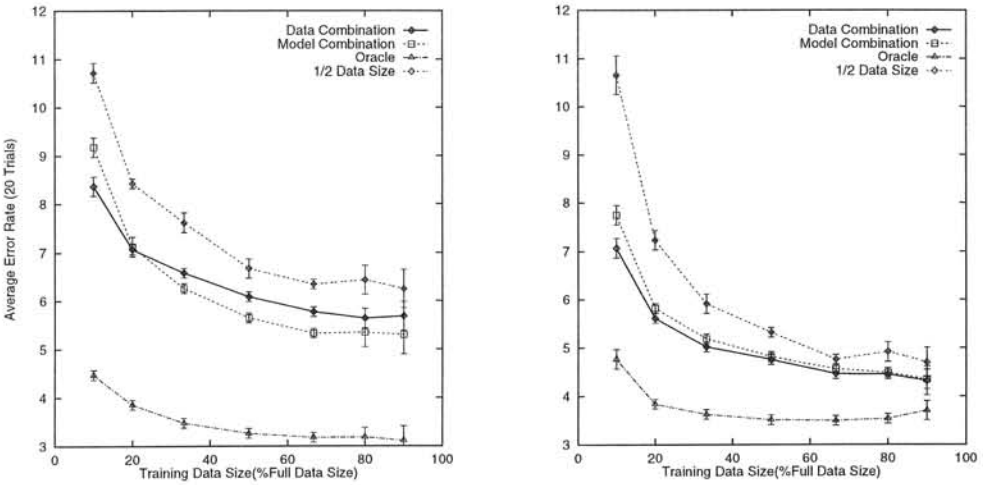
(a) IB1* (b) NB*

Figure 6: Learning curves (average error rate) for the *euthyroid* dataset



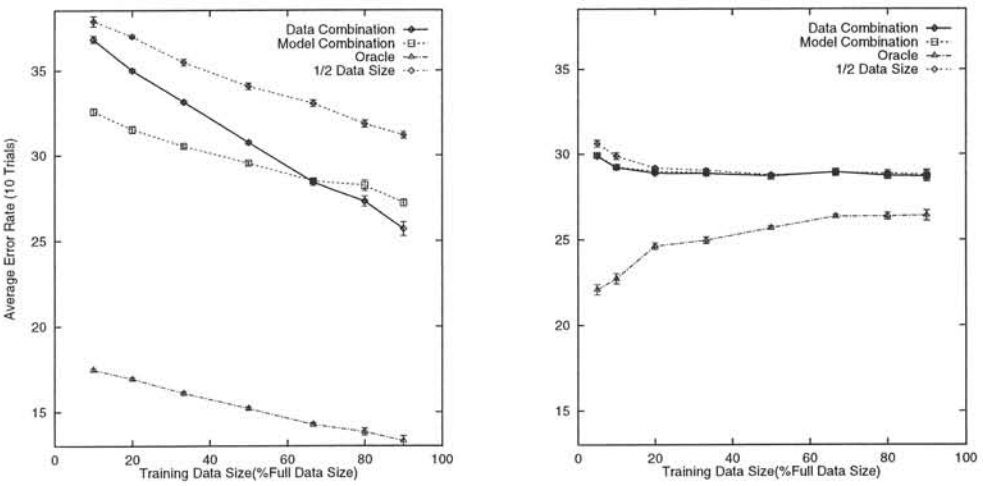
(a) IB1* (b) NB*

Figure 7: Learning curves for the *nettalk(stress)* dataset



(a) IB1* (b) NB*

Figure 8: Learning curves for the *splice junction* dataset



(a) IB1* (b) NB*

Figure 9: Learning curves for the *protein coding* dataset

is unbiased, that is, $C_A(x) = C^*(x)$. Now the bias and variance of C are defined as

$$Bias(C) = P(C^*(X) = Y, X \epsilon B) - E[P(C(X, \mathcal{L}) = Y, X \epsilon B)],$$

$$Var(C) = P(C^*(X) = Y, X \epsilon U) - E[P(C(X, \mathcal{L}) = Y, X \epsilon U)].$$

The probabilities P in each expression average over the distributions of X and Y, while the expectations E average over training sets $\mathcal{L}$. Breiman (1996b) contains a more extensive discussion of these notions; our aim here is to employ them to provide insight into the behavior of the model and data combination methods.

In our experiments, we compare the error decomposition for data and model combination for different amounts of training data. As in Section 5.1, the two models IB1* and NB* are used for the model combination method, and as before we show results for the combined model using the measure of characterization scheme. We employ the two artificial datasets *waveform* and *LED24*, for which the optimal Bayes error rates are estimated to be 14.86% and 24.88% respectively, using 5000 testing instances. Each experiment is repeated over 50 trials in order to produce a more accurate estimate of the bias and variance.

Tables 1 and 2 show the error decomposition for data and model combination using IB1* or NB* in the two domains, for different training set sizes. In each case the data combination method uses a training set of the given size N, while the model combination method combines two models each derived from a training set of size $N/2$.

Examination of the trend for $PE(C)$ in these tables shows that while both data combination and model combination improve as N increases, model combination improves more. For example, in the top half of Table 1, $PE(C)$ decreases from 26.20% to 21.97% for data combination, and from 27.82% to 18.60% for model combination. Examination of the $Bias(C)$ and $Var(C)$ figures shows that, in general, it is improvement in variance rather than bias that accounts for the superior performance of model combination as N increases. Using the same example, the main contribution to the decrease in $PE(C)$ for IB1* is $Var(C)$ in both cases, which decreases from 10.90% to 6.90% for data combination and from 11.78% to 3.57% for model combination. The same trends can be observed in the other results in Tables 1 and 2.

The results can be summarized as follows. When little data is available, model combination has higher bias and higher variance than data combination, and, in this situation, a higher error rate. However, model combination achieves a larger variance reduction than data combination as the training data size increases. As a result, given sufficient data model combination achieves a lower error rate than data combination.

Table 1: Error decomposition in the *waveform* domain (all figures in Tables 1, 2, and 3 are percentages)

	N	$PE(C) = PE(C^*) + Bias(C) + Var(C)$	
		Data combination	Model combination
IB1*	100	26.20 = 14.86 + 0.45 + 10.90	27.82 = 14.86 + 1.18 + 11.78
	200	24.37 = 14.86 + 0.23 + 9.28	23.84 = 14.86 + 0.49 + 8.49
	400	23.40 = 14.86 + 0.19 + 8.34	21.40 = 14.86 + 0.22 + 6.32
	600	22.93 = 14.86 + 0.27 + 7.79	20.40 = 14.86 + 0.20 + 5.34
	800	22.71 = 14.86 + 0.18 + 7.67	19.76 = 14.86 + 0.13 + 4.77
	1000	22.49 = 14.86 + 0.25 + 7.38	19.54 = 14.86 + 0.14 + 4.54
	1200	22.45 = 14.86 + 0.38 + 7.20	19.08 = 14.86 + 0.21 + 4.01
	1600	22.19 = 14.86 + 0.00 + 7.33	18.91 = 14.86 + 0.08 + 3.97
	2000	21.97 = 14.86 + 0.21 + 6.90	18.60 = 14.86 + 0.17 + 3.57
NB*	100	22.75 = 14.86 + 2.47 + 5.41	23.33 = 14.86 + 2.95 + 5.51
	200	20.76 = 14.86 + 2.46 + 3.43	20.02 = 14.86 + 2.21 + 2.95
	400	19.93 = 14.86 + 2.57 + 2.50	18.48 = 14.86 + 1.99 + 1.63
	600	19.52 = 14.86 + 2.51 + 2.15	17.96 = 14.86 + 1.78 + 1.32
	800	19.43 = 14.86 + 2.57 + 2.00	17.82 = 14.86 + 1.88 + 1.09
	1200	19.17 = 14.86 + 2.64 + 1.68	17.54 = 14.86 + 1.92 + 0.76
	1600	19.20 = 14.86 + 2.92 + 1.42	17.57 = 14.86 + 1.98 + 0.74
	2000	19.14 = 14.86 + 3.06 + 1.21	17.57 = 14.86 + 2.01 + 0.70

Table 2: Error decomposition in the *LED24* domain

	N	$PE(C) = PE(C^*) + Bias(C) + Var(C)$	
		Data combination	Model combination
IB1*	100	38.52 = 24.88 + 1.00 + 12.65	40.55 = 24.88 + 1.03 + 14.64
	200	37.59 = 24.88 + 0.98 + 11.74	33.41 = 24.88 + 0.81 + 7.72
	400	36.77 = 24.88 + 0.97 + 10.92	30.43 = 24.88 + 0.73 + 4.81
	600	36.55 = 24.88 + 0.93 + 10.74	29.51 = 24.88 + 0.83 + 3.80
	800	36.65 = 24.88 + 0.81 + 10.96	29.00 = 24.88 + 0.68 + 3.43
	1000	36.64 = 24.88 + 1.03 + 10.73	28.68 = 24.88 + 0.56 + 3.24
	1200	36.58 = 24.88 + 0.85 + 10.85	28.54 = 24.88 + 0.60 + 3.06
	1600	36.52 = 24.88 + 0.70 + 10.93	28.40 = 24.88 + 0.46 + 3.06
	2000	36.45 = 24.88 + 0.66 + 10.90	28.27 = 24.88 + 0.58 + 2.81
NB*	100	46.01 = 24.88 + 1.24 + 19.88	53.42 = 24.88 + 1.52 + 27.01
	200	35.18 = 24.88 + 0.87 + 9.43	38.29 = 24.88 + 0.98 + 12.43
	400	28.68 = 24.88 + 0.41 + 3.39	30.06 = 24.88 + 0.63 + 4.54
	600	26.81 = 24.88 + 0.64 + 1.29	27.35 = 24.88 + 0.45 + 2.02
	800	26.07 = 24.88 + 0.43 + 0.76	26.40 = 24.88 + 0.29 + 1.23
	1000	25.77 = 24.88 + 0.51 + 0.39	25.91 = 24.88 + 0.24 + 0.78
	1200	25.67 = 24.88 + 0.44 + 0.34	25.69 = 24.88 + 0.56 + 0.25
	1600	25.66 = 24.88 + 0.50 + 0.27	25.54 = 24.88 + 0.36 + 0.30
	2000	25.63 = 24.88 + 0.49 + 0.27	25.48 = 24.88 + 0.43 + 0.17

Table 3: Error decomposition using C4.5 combined with majority vote

	N	$PE(C) = PE(C^*) + Bias(C) + Var(C)$	
		Data combination	Model combination
waveform	50	37.73 = 14.86 + 1.37 + 21.50	37.87 = 14.86 + 3.39 + 19.62
	150	33.05 = 14.86 + 1.27 + 16.92	31.24 = 14.86 + 1.33 + 15.04
	450	29.50 = 14.86 + 0.44 + 14.20	26.49 = 14.86 + 1.26 + 10.37
	1350	26.99 = 14.86 + 0.58 + 11.55	23.99 = 14.86 + 0.42 + 8.71
	4050	25.04 = 14.86 + 0.38 + 9.80	22.13 = 14.86 + 0.56 + 6.72
	12150	23.88 = 14.86 + 0.27 + 8.74	20.84 = 14.86 + 0.39 + 5.59
LED24	50	47.07 = 24.88 + 1.97 + 20.02	61.59 = 24.88 + 3.33 + 33.17
	150	37.33 = 24.88 + 1.47 + 10.78	39.51 = 24.88 + 1.76 + 12.67
	450	30.86 = 24.88 + 0.96 + 4.82	31.93 = 24.88 + 1.35 + 5.50
	1350	28.02 = 24.88 + 1.10 + 1.83	27.96 = 24.88 + 0.77 + 2.11
	4050	27.04 = 24.88 + 0.85 + 1.11	26.64 = 24.88 + 0.97 + 0.59
	12150	26.42 = 24.88 + 0.48 + 0.86	26.22 = 24.88 + 0.68 + 0.46

5.4 Model combination using majority vote

We now examine whether model combination using majority vote (Section 3.1) also exhibits the phenomenon described above. Table 3 shows, for the two artificial domains, the average error rate, bias and variance of data combination, and majority vote using three models derived by C4.5. In all other respects the experimental conditions are as before.

The trends described above can be observed in these results. In both domains model combination eventually outperforms data combination (only very marginally for *LED24*), although it takes a much larger number of instances for this to become apparent. And in both domains it is the reduction in variance, rather than bias, that is principally responsible for the edge that model combination achieves as N becomes large.

5.5 Model combination using stacked generalization

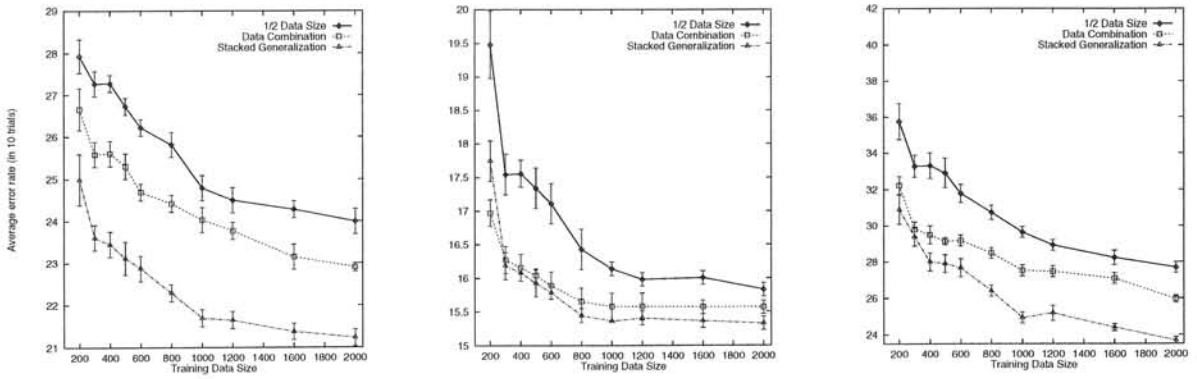
To assess results for the third method of model combination, stacked generalization (Section 3.3), Figures 10 and 11 show learning curves for IB1*, NB, and C4.5 for the two artificial domains. Experimental conditions are exactly as in Section 5.1. Each graph contains three learning curves. The first shows the behavior of the model derived from a single batch of half the data size. The second shows the model derived from data combination, that is, using the full data size. The third shows the model derived by using stacked generalization on two models, each formed from half the data.

The results are just as reported in Section 5.1 for model combination using the measure of characterization. With IB1*, model combination using stacked generalization performs significantly better than data combination in both domains. Similar performance is observed for NB* in the waveform domain, while for the *LED24* domain the two methods converge to the same near-asymptotic performance.

6 Discussion

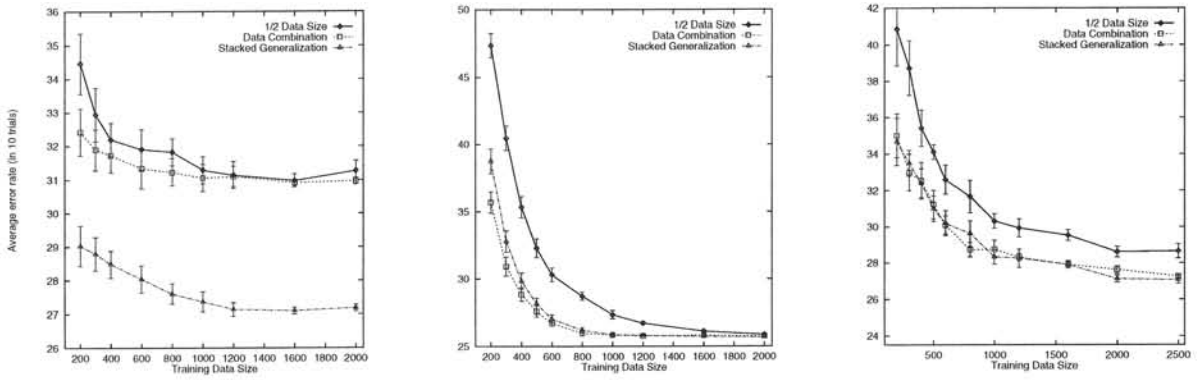
We have observed in the experiments that the relative performance of model combination and data combination can be predicted from the position on the learning curve of the learning algorithm used, which is exactly the hypothesis of Section 4. If combining the data yields only a small performance improvement, model combination generally gives results which are comparable to or better than those of data combination. This is a common feature of all the learning curves shown in Section 5. As Section 5.3 also reveals, this effect is caused by the faster variance reduction of model combination when compared with data combination, as the training data size increases.

Our empirical findings on the expected performance of model combination seem to be even more striking than a theoretical result based on the same working assumption



(a) IB1* (b) NB* (c) C4.5

Figure 10: Learning curves for the *waveform* domain when model combination is done using stacked generalization



(a) IB1* (b) NB* (c) C4.5

Figure 11: Learning curves for the *LED24* domain when model combination is done using stacked generalization

(Kearns & Seung, 1995). This theoretical work investigates "... the possibility of somehow combining the independent hypotheses in a way that considerably outperforms any single hypothesis," where this "single hypothesis" refers to any the constituent models that are combined. Our work indicates that model combination can significantly outperform not only the constituent models, but the model learned from aggregating the available data. This is even more surprising considering that we have only investigated the base-line behavior of model combination, i.e. combinations of just two models.

One might assume that, because they exhibit the same level of performance, models learned in the near-asymptotic region would be very similar to each other. However, our experimental findings suggest that two models learned from separate data batches are substantially different even though they lie in the near-asymptotic region and demonstrate the same performance. This occurs in most of the experimental datasets, because the oracle performs significantly better than its constituent models, even in the near-asymptotic region. This indicates that the assumption is incorrect.

What surprises us is that *near-asymptotic performance can be improved significantly by combining multiple models generated by the same learning algorithm.* This can only occur because the constituent models are substantially different and they capture different regularities that can be exploited by combination methods. Figure 5(a) shows an example of performance improvement as a result of model combination's exploitation in the near-asymptotic region. This empirical evidence of further significant improvement on the near-asymptotic performance of a learning algorithm using multiple models is new. Previous work (e.g., Hansen & Salamon, 1990; Perrone & Cooper, 1993; Oliver & Hand, 1995; Chan & Stolfo, 1995; Breiman, 1996a, 1996b; Freund & Schapire, 1996; Ali & Pazzani, 1996) shows only that performance can be improved using multiple models for some datasets, without considering the (near-)asymptotic performance.

One might assume that the above phenomenon is due to the fact that model combination increases the representation power of the base model, enabling it to perform better in the (near-)asymptotic region. For example, the expressive power of a learning system that can only induce a single conjunctive rule can be increased by combining several rules with disjunction. However, this explanation does not apply to the learning algorithms and model combination methods employed here. For example, combining several decision trees using majority vote is equivalent to a single, more complex, decision tree that subsumes them all. In this case, the combination adds no extra representation power. Our explanation is rather different: the whole process of combining multiple models is tantamount to another algorithm that has a different *search bias* than the one that generates the base models. The new bias allows more exploration in a search space that consists of multiple models, instead of a single search exploration to derive the base model. Thus it is not surprising that model combination and data combination display different learning

behavior.

In a regression setting, Meir (1994) undertakes a theoretical analysis of the effect of linearly combining several least-squares linear estimators according to their expected performance, under the same working assumption as ours. While his general result agrees with ours—model combination can significantly outperform data combination in some situations—the two results differ significantly in detail. Meir finds that model combination outperforms data combination even for small training sets, and that it can be worse for intermediate sample sizes. This result is based on analyzing the bias/variance decomposition; however, this decomposition is quite different in classification tasks (Kohavi & Wolpert, 1996). It also assumes, for tractability, that the data batches are independent; but our empirical result has no such assumption. Relaxing the non-overlapping and independence assumptions, Sollich and Krogh (1996) show analytically that model combination can improve substantially on data combination by optimizing the weights of the linear combination for intermediate sample sizes. This latter result is based on a slightly different working assumption from ours.

7 Other issues

When more than two batches are available, one possibility is to stack up the combinations in a binary tree structure, as investigated by Chan and Stolfo (1997). Nevertheless, more evidence is required to show that this is generally applicable. On the other hand, one need not necessarily use the same learning algorithm for all batches of data. A model selection technique such as cross-validation (Schaffer, 1993) may be applied to each batch to choose between several learning algorithms, prior to model combination. However, this incurs multiple folds of computation.

Model combination has three advantages over data combination: learning can be faster as discussed below, less memory is required because less data is used for each learned model, and the process can run in parallel on multiple processors. These advantages make model combination well suited to large datasets. Catlett (1991) studied a variety of sampling techniques to extract a subset from a large dataset for decision tree learning, but concluded that they do not solve the problem of scaling up to very large datasets. Our experiments with real-world datasets indicate that sampling together with parallel processing is a promising way to approach this problem.

The extra computation that model combination requires differs from one method to the other, and with the type of learning algorithm used. For the principal method used in our experiments, combination using a measure of characterization, the main additional computational load is the estimation of predictive accuracy during training. As Section 3.2 described, this involves a cross-validation step on the training data. Assume that the

learner has linear time complexity (as NB* does) of $T(n,m) = n + km$ where n and m are the training and testing data sizes and k is a constant that reflects the different costs of training and testing. Because three-fold cross-validation is used, model combination induces a single model in time $T_s(n,m) = 4n + km$. Data combination, on the other hand, requires $T_d(2n,m) = 2n + km$. For example, in the *LED24* domain NB* requires $T_s(n,m) = 6$ seconds and $T_d(2n,m) = 2.5$ seconds on a Sun SPARCserver 1000 machine, where $n = 1000$ and $m = 5000$. Clearly model combination is not going to win for a linear-time learner. However, in the nonlinear case the situation is different. For example, IB1*, whose learning time is $O(mn^2)$, requires $T_s(n,m) = 6.5$ minutes and $T_d(2n,m) = 12$ minutes for the same example. Parallel processing can indeed speed up the computation in such cases, even with this relatively expensive model combination method. Using a simpler and less expensive model combination method such as majority vote guarantees that a speed-up will be obtained from parallel processing. In our experiment reported in Section 5.4, for models induced from C4.5, $T_s(n,m) = 0.8$ seconds and $T_d(3n,m) = 1.2$ seconds, where $n = 450$ and $m = 5000$.

8 Conclusions

A comparison of the base-line behavior of model combination and data combination using randomly-drawn disjoint data batches reveals that model combination compares favorably when the performance gain achieved by increasing the training set size is small, which occurs in the near-asymptotic region of the learning curve. This is because model combination achieves a larger variance reduction than data combination as the training data increases. The empirical evidence presented in this paper shows that this result is not sensitive to the particular learning algorithm or model combination method employed.

Moreover, we have shown empirically that the near-asymptotic performance of a learning algorithm can often be improved significantly by combining multiple models generated by the same algorithm.

References

Aha, D.W., D. Kibler & M.K. Albert (1991), Instance-Based Learning Algorithms, *Machine Learning, 6*, pp. 37-66.

Ali, K.M. & M.J. Pazzani (1996), Error Reduction through Learning Multiple Descriptions, *Machine Learning*, Vol. 24, No. 3, pp. 173-206.

Baxt, W.G. (1992), Improving the Accuracy of an Artificial Neural Network using Multiple Differently Trained Networks, *Neural Computation*, Vol. 4, No. 5, pp. 772-780.

Brazdil,P. & Torgo,L. (1990), Knowledge Acquisition via Knowledge Integration. In *Current Trends in Knowledge Acquisition*, Wielinga, B. et al.(eds.).

Breiman, L. (1996a), Bagging Predictors, *Machine Learning*, Vol. 24, No. 2, pp. 123-140.

Breiman, L. (1996b), Bias, Variance, and Arcing Classifiers, *Technical Report 460*, Department of Statistics, University of California, Berkeley, CA.

Breiman, L. (1996c), Pasting Bites Together for Prediction in Large Data Sets and On-Line, [ftp.stat.berkeley.edu/users/pub/breiman/pasting.ps].

Breiman, L. (1996d), Stacked Regressions, *Machine Learning*, Vol. 24, pp. 49-64.

Breiman, L., J.H. Friedman, R.A. Olshen & C.J. Stone (1984), *Classification And Regression Trees*, Belmont, CA: Wadsworth.

Brodley, C.E. (1993), Addressing the Selective Superiority Problem: Automatic Algorithm/Model Class Selection, in *Proceedings of the Tenth International Conference on Machine Learning*, pp. 17-24.

Buntine, W. (1991), Classifiers: A Theoretical and Empirical Study, in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 638-644, Morgan-Kaufmann.

Cestnik, B. (1990), Estimating Probabilities: A Crucial Task in Machine Learning, in *Proceedings of the European Conference on Artificial Intelligence*, pp. 147-149.

Chan, P.K. & S.J. Stolfo (1995), A Comparative Evaluation of Voting and Meta-learning on Partitioned Data, in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 90-98, Morgan Kaufmann.

Chan, P.K. & S.J. Stolfo (1997), On the Accuracy of Meta-learning for Scalable Data Mining, in *Journal of Intelligent Systems,*, 8, pp. 5-28.

Cost, S & S. Salzberg (1993), A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features, *Machine Learning, 10*, pp. 57-78.

Craven, M.W. & J.W. Shavlik (1993), Learning to Represent Codons: A Challenge Problem for Constructive Induction, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1319-1324.

Fayyad, U.M. & K.B. Irani (1993), Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, in *Proceedings of 13th International Joint Conference on Artificial Intelligence*, pp. 1022-1027.

Freund, Y. & R.E. Schapire (1996), Experiments with a New Boosting Algorithm, in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148-156, Morgan Kaufmann.

Hansen, L.K. & P. Salamon (1990), Neural Network Ensembles, in *IEEE Transactions of Pattern Analysis and Machine Intelligence, 12*, pp. 993-1001.

Kearns, M. & H.S. Seung (1995), Learning from a Population of Hypotheses, *Machine Learning, 18*, pp. 255-276, Kluwer Academic Publishers.

Kohavi, R. & D.H. Wolpert (1996), Bias Plus Variance Decomposition for Zero-One Loss Functions, in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 275-283, Morgan Kaufmann.

Kononenko, I. & M. Kovačič (1992), Learning as Optimization: Stochastic Generation of Multiple Knowledge, in *Proceedings of the Ninth International Conference on Machine Learning*, pp. 257-262, Morgan Kaufmann.

Krogh, A. & J. Vedelsby (1995), Neural Network Ensembles, Cross Validation, and Active Learning, in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D.S. Touretsky & T.K. Leen (Editors), pp. 231-238.

Kwok, S. & C. Carter (1990), Multiple Decision Trees, *Uncertainty in Artificial Intelligence 4*, R. Shachter, T. Levitt, L. Kanal and J. Lemmer (Editors), pp. 327-335, North-Holland.

Lawson C.L. & R.J. Hanson (1995), *Solving Least Squares Problems*, SIAM Publications.

Meir, R. (1994), Bias, Variance and the Combination of Estimators: The Case of Linear Least Squares, *Technical Report No. 922*, Department of Electrical Engineering, Technion, Haifa, Israel.

Merz, C.J. & Murphy, P.M. (1996), *UCI Repository of machine learning data-bases* [http:// www.ics.uci.edu/ mlearn/MLRepository.html].

Oliver, J.J. & D.J. Hand (1995), On Pruning and Averaging Decision Trees, in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 430-437. Morgan Kaufmann.

Perrone, M.P. & L.N. Cooper (1993), When Networks Disagree: Ensemble Methods for Hybrid Neural Networks, in *Artificial Neural Networks for Speech and Vision*, R.J. Mammone (Editor), Chapman-Hall.

Provost, F.J. & D.N. Hennessy (1996), Scaling Up: Distributed Machine Learning with Cooperation, in *Proceedings of the Thirteen National Conference on Artificial Intelligence*, pp. 74-79, Menlo Park, CA: AAAI Press.

Quinlan, J.R. (1993), *C4.5: Program for machine learning*, Morgan Kaufmann.

Quinlan, J.R. (1996), Boosting, Bagging, and C4.5, in *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 725-730, AAAI Press.

Quinlan, J.R., P.J. Compton, K.A. Horn & L. Lazarus (1987), Inductive Knowledge Acquisition: A Case Study, in *Applications of Expert Systems*, J.R. Quinlan (Editor). Turing Institute Press with Addison Wesley.

Schapire, R.E. (1990), The Strength of Weak Learnability, *Machine Learning, 5*, pp. 197-227, Kluwer Academic Publishers.

Sejnowski, T.J. & C.R. Rosenberg (1987), Parallel networks that learn to pronounce English text, *Complex Systems*, 1, pp. 145-168.

Sollich, P. & A. Krogh (1996), Learning with ensembles: How overfitting can be useful, in *Advances in Neural Information Processing Systems 8*, D.S. Touretzky, M.C. Mozer & M.E. Hasselmo (Editors) , MIT Press.

Tcheng, D., B. Lambert, C-Y. Lu & L. Rendell (1989), Building Robust Learning Systems by Combining Induction and Optimization, in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 806-812.

Ting, K.M. (1994), Discretization of Continuous-Valued Attributes and Instance-Based Learning, *TR 491*, Basser Department of Computer Science, University of Sydney.

Ting, K.M. (1996), The Characterisation of Predictive Accuracy and Decision Combination, in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 498-506, Morgan Kaufmann.

Ting, K.M. (1997), Discretisation in Lazy Learning Algorithms, *Journal of Artificial Intelligence Review*, Special issue on Lazy Learning, Vol. 11, pp. 157-174, Kluwer Academic Publishers.

Ting, K.M. & I. H. Witten (1997), Stacked Generalization: when does it work?, to appear in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence.* A long version appears as *Working Paper 97/3*, Dept of Computer Science, University of Waikato. [*http://www.cs.waikato.ac.nz/cs/Staff/kaiming.html*].

Towell, G., J. Shavlik & M. Noordewier (1990), Refinement of Approximate Domain Theories by Knowledge-Based Artificial Neural Networks, in *Proceedings of the Eighth National Conference on Artificial Intelligence.*

Utgoff, P.E. (1989), Perceptron Trees: A case study in hybrid concept representations, *Connection Science, 1*, pp. 337-391.

Wettschereck, D. (1994), A Hybrid Nearest-Neighbor and Nearest-Hyperrectangle Algorithm, in *Proceedings of the Seventh European Conference on Machine Learning, LNAI-784*, pp. 323-335, Springer Verlag.

Wolpert, D.H. (1992), Stacked Generalization, *Neural Networks*, Vol. 5, pp. 241-259, Pergamon Press.

Appendix: The stacked generalization procedure

The stacked generalization process for the two-batch case is as follows. The original data set $\mathcal{L}$ consists of two batches $\mathcal{L}A = \{(y_a, x_a), a = 1, \ldots, N\}$ and $\mathcal{L}B = \{(y_b, x_b), b = 1, \ldots, M\}$. Randomly split the first batch into J almost equal parts $\mathcal{L}A_1, \ldots, \mathcal{L}A_J$, and define $\mathcal{L}A_j$ and $\mathcal{L}A^{(-j)} = \mathcal{L}A - \mathcal{L}A_j$ to be the test and training sets respectively for the jth fold of cross-validation. Using the data in the training set, induce a model $\mathcal{M}_A^{(-j)}$ using some learning algorithm. Use the same learning algorithm on all of the data in $\mathcal{L}B$ to derive a single model, which we call $\mathcal{M}_B$.

Given a data instance x_a in $\mathcal{L}A_j$, denote by z_{Aia} the probability that $\mathcal{M}_A^{(-j)}$ assigns to the ith class. For the same instance, denote by z_{Bia} the probability that the model $\mathcal{M}_B$

assigns to the ith class. Using this procedure, form one set of "level-1" data as

$$\mathcal{L}A_{CV} = \{(y_a, z_{A1a}, \ldots, z_{AIa}, z_{B1a}, \ldots, z_{BIa}),\ a = 1, \ldots, N\},$$

where I is the number of classes.

A second set of level-1 data is obtained by randomly splitting $\mathcal{L}B$ into J almost equal parts and deriving a cross-validation set of models $\mathcal{M}_B^{(-j)}$ for use on the test sets $\mathcal{L}B_j$ and a single model $\mathcal{M}_A$ from $\mathcal{L}A$:

$$\mathcal{L}B_{CV} = \{(y_b, z_{A1b}, \ldots, z_{AIb}, z_{B1b}, \ldots, z_{BIb}),\ b = 1, \ldots, M\}.$$

Now concatenate the two data sets $\mathcal{L}A_{CV}$ and $\mathcal{L}B_{CV}$ to form the joint level-1 data:

$$\mathcal{L}_{CV} = \{(y_c, z_{A1c}, \ldots, z_{AIc}, z_{B1c}, \ldots, z_{BIc}),\ c = 1, \ldots, N + M\}.$$

We now use the MLR learning strategy on this training data to give a "level-1" model $\tilde{\mathcal{M}}$. The linear regression for class ℓ is simply

$$LR_\ell(x) = \sum_i^I (\alpha_{Ai\ell} z_{Ai} + \alpha_{Bi\ell} z_{Bi}).$$

The level-1 model is formed by choosing the linear regression coefficients α to minimize

$$\sum_{c=1}^{N+M} \left[(y_c - \sum_i^I (\alpha_{Ai\ell} z_{Aic} + \alpha_{Bi\ell} z_{Bic}) \right]^2,$$

for each class ℓ, constraining the coefficients α to be non-negative.

The final classification process uses models $\mathcal{M}_A$, $\mathcal{M}_B$ and $\tilde{\mathcal{M}}$ to classify an instance of unknown class. First, the instance is presented to models $\mathcal{M}_A$ and $\mathcal{M}_B$. Each model yields I class probabilities that reflect the probability it assigns to the instance being in each of the I possible classes. These probabilities are assembled into a vector which constitutes the level-1 data for the unknown instance. This vector is presented to $\tilde{\mathcal{M}}$, which comprises a set of I linear regression models LR_ℓ. The value of $LR_\ell(x)$ is computed for all I classes and the instance is assigned to that class ℓ with greatest value:

$$LR_\ell(x) > LR_{\ell'}(x) \text{ for all } \ell' \neq \ell.$$