Efficient Algorithms for Minimizing Cross Validation Error

Andrew W. Moore
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213
awm@cs.cmu.edu

Mary S. Lee
6413 Howe Street
Pittsburgh, PA 15206
mary@amoore.sp.cs.cmu.edu

Abstract

Model selection is important in many areas of supervised learning. Given a dataset and a set of models for predicting with that dataset, we must choose the model which is expected to best predict future data. In some situations, such as online learning for control of robots or factories, data is cheap and human expertise costly. Cross validation can then be a highly effective method for automatic model selection. Large scale cross validation search can, however, be computationally expensive. This paper introduces new algorithms to reduce the computational burden of such searches. We show how experimental design methods can achieve this, using a technique similar to a Bayesian version of Kaelbling's Interval Estimation. Several improvements are then given, including (1) the use of blocking to quickly spot near-identical models, and (2) schemata search: a new method for quickly finding families of relevant features. Experiments are presented for robot data and noisy synthetic datasets. The new algorithms speed up computation without sacrificing reliability, and in some cases are more reliable than conventional techniques.

1 INTRODUCTION

Model selection is an important aspect of many areas of supervised learning. The computer is given a dataset and a set of models for predicting with that dataset, and must determine the best model. The models might be a collection of rival learning algorithms, or different sets of high-level learning parameters, or a collection of alternative architectures for a given learning algorithm (e.g. different neural-net architectures). The best model is the one which is expected to give the most accurate predictions for future data.

Cross validation is a much used, powerful method for model selection. It is also general purpose: it has few prior assumptions and is rarely tied to particular internal features of an algorithm.

There are two prices to be paid for the generality of cross validation.

- **A data penalty.** Firstly, unless leave-one-out cross validation is used, some data must be reserved for use as a test set. Secondly, a naive intensive use of cross validation, perhaps over many thousands of models, may produce a deceptively good lowest-error model, in a manner similar to overfitting of data. Deciding how much data is required to justify searching over a given number of models is extremely difficult, and to our knowledge no general-purpose decision procedure exists. In the absence of further assumptions, the danger of searching with insufficient data must be guarded against by leaving out yet another portion of the data, and using this to check the basic cross validation search.
- **A computational penalty.** With medium or large datasets, computing the cross validation error of each model can take too long. For instance, computing the leave-one-out error for the 10-nearest neighbor algorithm on a 1000 datapoint, 12 input dataset takes 20 seconds on a SPARC. Performing search over thousands of models may take many hours, which is impractical for some applications. Some large datasets may have sufficient data to theoretically support a search across millions of models, yet the computational cost of a cross validation search would be prohibitive.

We are interested in applying cross validation to control tasks in robotics and manufacturing. These often involve decisions based on streams of information from sensors and actuators, where data is plentiful. There have been several such cases where cross validation was helpful in selecting (1) relevant sets of visual features [Moore *et al.*, 1992], (2) the relevance and weightings of robot joint torques [Atkeson, 1990], (3) smoothing kernel sizes [Schaal and Atkeson, 1994, Moore, 1992], and (4) determining neural network architectures [Moody and Utans, 1992]. In this paper we describe new algorithms for speeding up such cross validation decisions.

1.1 CROSS VALIDATION

This paper uses leave-one-out cross validation (**LOOCV**) applied to memory-based (or instance-based) learning.

Given N_{models} models and a dataset with $N_{\text{datapoints}}$ training points, define the **LOOCV** error of the ith point using model M_j as the difference between the true output of the ith point and the predicted output when model j is trained on all members of the dataset *except* for the ith point. Call this error $e_j(i)$.

$$e_j(i) = |\text{output}_i - \text{Predict}(M_j, \text{Dataset} - i\text{th point})| \quad (1)$$

Then the **LOOCV** error of model M_j is the mean

$$e_j^* = \frac{1}{N_{\text{datapoints}}} \sum_i e_j(i) \quad (2)$$

The search problem is to find which M_j minimizes e_j^*.

1.2 THIS PAPER

First we describe earlier work [Maron and Moore, 1994], where we use statistical tests to decide whether a partially completed cross validation has already produced a clear favorite, rendering the rest of the computation unnecessary. For this we appeal to statistical techniques known as experimental design methods ([Box *et al.*, 1978]). To our knowledge, this has not been used before for accelerating model selection or feature selection in function approximation problems. In a number of other applications, similar techniques have been used. One of the first such examples in the AI literature is Kaelbling's Interval Estimation method [Kaelbling, 1990], which uses confidence testing to alleviate the "exploration versus exploitation" dilemma in reinforcement learning. [Greiner and Jurisica, 1992] also use a similar technique for decision-making in knowledge-based systems. [Gratch *et al.*, 1993] uses a related method for choosing appropriate search rules in very large scheduling domains.

This paper then extends these ideas to several new algorithms, and demonstrates them on cross validation search. The new algorithms should also be applicable elsewhere in machine learning.

2 RACING THE CROSS VALIDATION ERRORS

In [Maron and Moore, 1994] Hoeffding bounds were used to accelerate cross validation. Hoeffding bounds are a non-parametric method which, given a sequence of independent observations of a random variable, permit one to place a confidence interval on the underlying mean value of the random variable. With probability $(1-\delta)$ the true underlying mean lies within distance ϵ of the sample mean where

$$\epsilon = \sqrt{\frac{B^2 \ln(2/\delta)}{2n}} \quad (3)$$

where B is the (known a priori) maximum range of the random variables and n is the number of samples. The advantage of Hoeffding bounds is the lack of an assumption about the distribution of the random variable. Its disadvantage is its resulting conservatism. If, instead, we make stronger assumptions then our confidence intervals will be able to shrink faster.

In this paper we start by considering a variant which we will here call the RACE algorithm, where the Hoeffding bounds are replaced by a Bayesian approach. Estimates of the mean **LOOCV** error of each model are built up in parallel, in a kind of race. During the race, if we predict a model is highly unlikely to eventually have the lowest **LOOCV** error (i.e. is unlikely to "win") then that model is eliminated.

The RACE algorithm first randomizes the order of the datapoints. We will make the (rather strong) assumption that the leave-one-out errors are distributed normally. For each model M_j they have a priori unknown mean e_j^* and variance σ_j^*. Remember that e_j^* is the true mean **LOOCV** of all the datapoints in the dataset using model M_j, and the goal of the computation is to find the model which minimizes this value.

As evidence accumulates, the uncertainty of e_j^* decreases. Let $\hat{\mu}_{kj}$ and $\hat{\sigma}_{kj}^2$ be the sample mean and variance of model j's **LOOCV** errors up to the kth iteration, when all surviving models have been evaluated on k datapoints.

$$\hat{\mu}_{kj} = \frac{1}{k} \sum_{i=1}^{k} e_j(i) \quad (4)$$

$$\hat{\sigma}_{kj}^2 = \frac{1}{k-1} \sum_{i=1}^{k} \left(e_j(i) - \hat{\mu}_{kj}\right)^2 \quad (5)$$

(remembering that $e_j(i)$ is the leave-one-out error when model M_j is used to predict the ith datapoint).

For each model M_j we use Bayesian statistics, along with the values $\hat{\mu}_{kj}$, $\hat{\sigma}_{kj}^2$, and k, to put a probability distribution on e_j^*. This is a relatively elementary process[1] which is described in, for example, [Schmitt, 1969], and is summarized in this paper's Appendix. Figure 1 gives an example. From these distributions one can compute (using the Welch approximation [Welch, 1937]) for any pair of models, M_j and $M_{j'}$, how likely it is that e_j^* is less than $e_{j'}^*$. In Figure 1 it would be very probable that e_2^* is lower than e_4^*, but only fairly probable that e_1^* is lower than e_4^*.

The race algorithm proceeds by eliminating any model, M_j, for which there exists another model $M_{j'}$ that is almost certainly better or indistiguishable. This statement needs

[1]The priors for the mean and variance are uninformative.

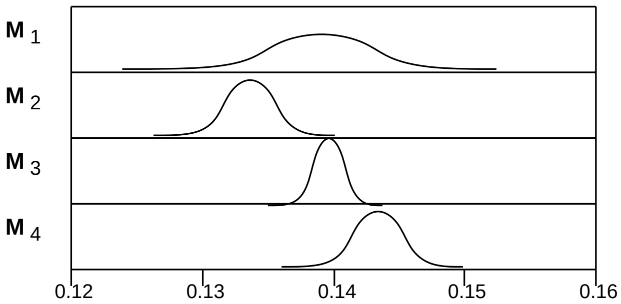

Figure 1: The posterior distributions on the **LOOCV** errors of four models involved in a race. The lower the **LOOCV** error the better, so it seems very unlikely that M_4 will turn out to be better than M_2 at the end of the race. Thus M_4 is eliminated at this point.

some clarification. We wish to eliminate any model which we are confident is worse than some other model. The confidence required will be denoted by δ. For example, if $\delta = 0.001$ that means we are willing to risk a one in a thousand chance of making an error on this test. We also wish to stop a race between two models which we believe with high confidence to be extremely similar. This can be achieved by defining a threshold γ (which is also a small positive number) and eliminating any model which we are confident has a mean value within distance γ of another model.

These rules can be combined into one rule: eliminate any model M_j for which there exists some other model $M_{j'}$ for which

$$\text{Prob}(e^*_j < e^*_{j'} - \gamma \,|\, e_j(1), \ldots e_j(k), e_{j'}(1), \ldots e_{j'}(k)) \quad (6)$$

is less than δ. Statistics are gathered and models are eliminated until only one model remains, or we run out of datapoints, whereupon we select the model with the lowest $\hat{\mu}_{kj}$.

2.1 BLOCKING

We will see that the race algorithm can greatly reduce the computational effort when compared with fully computing the **LOOCV** errors of all the models. Sometimes, however, its behavior can be ruined when two or more models produce nearly identical predictions. Such near-identical models will have to run the race through every point in the dataset. This problem can be reduced by a statistical technique called Blocking [Box *et al.*, 1978].

For a given datapoint i, the $e_j(i)$ are unlikely to be independent for different models. For instance, in a noisy dataset imagine that datapoint 15 is particularly noisy and has a larger output value than its neighbors. Then all models are likely to have a large leave-one-out error for it. Some of the spread of the distributions in Figure 1 is due to such factors, and they are not independent between models. This spread can be eliminated by blocking.

The trick is to estimate the values $h^*_{jj'}$ defined by:

$$h^*_{jj'} = e^*_j - e^*_{j'} \quad (7)$$

During the race we maintain the following statistics:

$$\hat{\mu}^h_{jj'}(k) = \frac{1}{k}\sum_{i=1}^{k}\left(e_j(i) - e_{j'}(i)\right) \quad (8)$$

$$\hat{\sigma}^h_{jj'}(k) = \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}\left[\left(e_j(i) - e_{j'}(i)\right) - \hat{\mu}^h_{jj'}\right]^2} \quad (9)$$

As before, we can use Bayesian statistics to maintain, for all j, j', probability distributions on $h^*_{jj'}$. But this time dependencies between models will have been removed. If M_6 and M_7 both have a large variation of errors but M_6's are always fractionally smaller than M_7's, the blocking statistics will reveal this much more quickly than would the basic race.

As before, model M_j is eliminated when for some j', $\text{Prob}(e^*_j < e^*_{j'} - \gamma)$ becomes very small; this is detected by testing for

$$\text{Prob}(h^*_{jj'} < -\gamma) < \delta. \quad (10)$$

The simplest example of blocking having a beneficial effect is the case where two models in the race are identical. Then the RACE algorithm would have to race for a long time. The racers would have the same mean at each step of the race, but RACE would only end when they both had so many samples that the confidence intervals on its measures of the mean error of each model were both very close to γ (its indifference parameter). In contrast, the BRACE (blocking race) algorithm would maintain, at each step, the difference between the leave-one-out errors on each datapoint. Because the models are the same, this difference would always be zero and it would only require a very small number of statistics (perhaps less than ten, depending upon the parameters) before one of the models was eliminated.

The example of identical models is extreme, but in the more common case of near-identical models there can also be a large reduction in the time to elimination.

2.2 RESULTS: RACING AND BLOCKING

Table 1 shows some results comparing both the RACE algorithm and the BRACE algorithm with the standard algorithm that computes all the models exhaustively. In the results, "Evals" denotes the number of individual **LOOCV** computations needed by the search, "Correct?" asks whether the model selected by the search had the true minimum **LOOCV**, and the "Relative evals" column gives the fraction of the cost of the search relative to the exhaustive search.

The algorithms were tested on datasets from a billiards robot (3 inputs, 1 output, 253 points) and a juggling robot (12 inputs, 1 output, 972 points). For these experiments, $\delta = 0.001$ and $\gamma = 0.001$. The δ value means we have to be 99.9% ($= 100(1-\delta)$) sure before eliminating a model from a race. The γ value means we are prepared to accidently eliminate models which are very slightly better (by a factor $\gamma = 0.001$) than the race winner. There were the same set of twenty models in each race. Ten models were local weighted averaging (also known as kernel regression) and ten models were locally weighted regression [Cleveland and Delvin, 1988]. Each group of ten models had different kernel smoothing parameters from the set $\{2^{-9}, 2^{-8}, \ldots 2^{-1}, 2^0\}$.

In both these experiments RACE improved on the exhaustive method, and BRACE in turn improved upon RACE. The proportional improvement was greater for the juggling example, mainly because the juggling dataset is larger. A larger dataset means the exhaustive method has to perform proportionally many extra **LOOCV** evaluations, whereas the racing methods can stop before all datapoints are evaluated. In three of the four races in Table 1 there was only one model left at the end of the race. For the RACE algorithm on the billiards dataset, 8 of the 20 models survived to the end.

2.3 COMMENTS ON RACE AND BRACE

Here we will briefly discuss several issues regarding RACE and BRACE. The first concerns our decision to use Bayesian statistics rather than the looser non-parametric Hoeffding bounds used in [Greiner and Jurisica, 1992] and [Maron and Moore, 1994]. For a given value of δ (defining the confidence level at which we are prepared to cut off a competitor from the race) the Bayesian approach cuts off far earlier than the Hoeffding approach. The Bayesian approach achieves its superiority by making stronger assumptions about the distribution of the errors, and so might be expected to be less robust than the almost assumptionless Hoeffding approach. In the experiments we have performed to date the Bayesian method does not seem to converge to the wrong model more easily than the Hoeffding approach, but this is an empirical observation which may not be true for all datasets.

Another issue concerns our choice to assume a normal distribution for the errors. In future work it might be more sensible to use a different distribution, such as an exponential or chi-squared. The normal distribution does not take into account the possibility of occasional highly aberrant datapoints (outliers) in the dataset which might have errors very many times greater than the root-mean-square error.

It is also worth considering the computational penalty of maintaining the statistics and making cut-off decisions during the race. This cost is $O(N_{\text{models}}(k)^2)$ for the kth step of the race, when $N_{\text{models}}(k)$ models remain. In our examples, this cost is quite small in comparison with the cost of computing the leave-one-out errors. In the case of memory-based methods such as locally weighted regression, for one model, the cost of computing the leave-one-out error of one datapoint is $O(N_{\text{datapoints}} D^2 + D^3)$ where D is the number of input variables in the dataset. Each step of the race requires the computation of a leave-one-out error for $N_{\text{models}}(k)$ models. Thus provided that $O(N_{\text{models}}(k)^2)$ is small compared with $O(N_{\text{models}}(k)(N_{\text{datapoints}} D^2 + D^3))$, requiring

$$N_{\text{models}}(k) << N_{\text{datapoints}} D^2, \qquad (11)$$

then the computational penalty is small. Furthermore, if we ever need to reduce the computational penalty further it might not be difficult to invent racing schemes which require less computation than $O(N_{\text{models}}(k)^2)$ per step by means of only testing a carefully chosen subset of all pairs of models.

3 SEARCHING FOR SETS OF GOOD FEATURES

A particularly promising use of cross validation is to automatically choose relevant inputs from a wider set of possible inputs. An obvious benefit is in accelerating the learning rate of algorithms which suffer in the face of irrelevant inputs. Other benefits include helping select relevant visual features for visually controlled robot tasks [Moore *et al.*, 1992], and selecting sets of time windows in time series predictions. This problem, known as "subset selection" is a well-known problem in statistics, surveyed thoroughly in [Miller, 1990] and is rapidly gaining attention in the Machine Learning community. In this proceedings, three other papers addressing subset selection for various machine learning algorithms are: [Caruana and Freitag, 1994, John *et al.*, 1994, Skalak, 1994].

Given D inputs there are 2^D possible input sets, and so performing an exhaustive cross validation search over all of them soon becomes impractical as D rises, even assuming adequate data support to justify searching so many models. Hill climbing is clearly a sensible alternative. In this section we provide several hill climbing versions of BRACE; these aim to both speed up the computation and also to reduce the danger of becoming trapped at local maxima.

Sets of inputs can be represented as binary strings. Given four possible inputs 0101 would denote "ignore inputs 1 and 3, use inputs 2 and 4." The standard non-racing hill climbing algorithm begins with a start string (e.g. 0000) then makes all possible 1-feature changes to it (1000, 0100, 0010, 0001) and exhaustively finds which minimizes the **LOOCV** error. It then uses this best string (say 0100) as a new base point, generates all its 1-feature successors (1100, 0000, 0110, 0101), and determines the best. It continues in this way until no single-feature change improves it. The special case of starting with all zeroes is termed forward selection (FOR-SEL), and that of starting with all ones is termed backward elimination (BACK-EL). Forward selection is better if only a few features are expected to be relevant and backward elimination is better if only a few features are expected to be irrelevant. Unfortunately, such prior knowledge may not be available at the start of the search.

Table 1: Experiments Described In The Text

	Billiards			Juggling		
Method	Evals	Correct?	Relative Evals to EXH	Evals	Correct?	Relative Evals to EXH
EXH	5060	Yes	1.000	19440	Yes	1.000
RACE	2464	Yes	0.487	2558	Yes	0.132
BRACE	1049	Yes	0.207	882	Yes	0.045

The racing counterparts to these algorithms are straightforward: from the base string generate all 1-feature changes and race them. Proceed until the winner of a race does not improve on its base. In the experiments described later two versions are tested, FOR-BRACE and BACK-BRACE, which start at all zeroes and all ones respectively.

There is, however, an objection to this simple application of BRACE. Imagine that inputs 4, 5, and 6 are all relevant and independently provide a reduction in the LOOCV error. If we start at string 0000000 and successors 0000100, 0000010, and 0000001 are all good, it will be a shame to run through three separate hill climbing iterations to switch them all on. This motivates the next algorithm, a Gauss-Seidel version of hill climbing:

- Begin with a predefined start string (e.g. 00000)
- Race between the current string and the current string with the first bit flipped (00000 versus 10000)
- Select the winner of that race as the new current string (e.g. 10000)
- Now race between the current string and the current string with the second bit flipped (10000 versus 11000)

$$\vdots$$

...until all bits have been raced. Then return to the first bit and proceed until an entire pass through the current string fails to produce an improvement.

Versions of this algorithm, FOR-GS-BRACE and BACK-GS-BRACE, are tested below. On some occasions they do indeed help, but on others their performance is poor. A further new algorithm does Gauss-Seidel's job better, and also solves another problem.

3.1 SCHEMATA SEARCH

Schemata search is another new algorithm which aims to solve the same problem that the Gauss-Seidel method addresses—the problem of forward selection taking a long time if many features are relevant, or similarly backward elimination taking a long time if many features are irrelevant. It will also help with a second problem. Suppose there is a family of three features which must all be on simultaneously for any reduction in the **LOOCV** error. If any family member is ignored then the **LOOCV** error is just as bad as if all family members were ignored. This can happen quite easily, for example if the features are distributed between -1 and 1 and the function being learned is their product. Forward selection would be very likely to miss this family and to converge on something suboptimal. Backward elimination would not have this problem, but if many features are irrelevant then it can become stuck itself (because in the early stages of hill climbing the removal of one irrelevant attribute among many does not improve the **LOOCV** error).

Schemata search searches over the space of schemata strings, which have 0's, 1's and $\star$'s in them. A $\star$ denotes a fifty percent chance of the attribute being ignored, and a fifty percent chance of it being used. The **LOOCV** error of such a string is the expected **LOOCV** error of a binary string generated from the schemata string according to these random rules, for example

$$\begin{aligned}\textbf{LOOCVE}(101\star\star) \quad = \quad & \frac{1}{4}\textbf{LOOCVE}(10100) + \\ & \frac{1}{4}\textbf{LOOCVE}(10101) + \\ & \frac{1}{4}\textbf{LOOCVE}(10110) + \\ & \frac{1}{4}\textbf{LOOCVE}(10111)\end{aligned}$$

Now a simple algorithm is to begin with all stars (e.g. $\star\star\star\star\star$) , and then to find out (by racing) whether it is better to have the first field as a 1 or a 0 (i.e. we would race $1\star\star\star\star$ and $0\star\star\star\star$). Having finished the first race, we could determine the second field by another race, and so on until the entire string is filled with 1's and 0's.

In practice, we can do better than this. Instead of beginning by racing the first field, we can race all fields against each other in parallel:

$1\star\star\star\star$	races against	$0\star\star\star\star$
$\star1\star\star\star$	races against	$\star0\star\star\star$
$\star\star1\star\star$	races against	$\star\star0\star\star$
$\star\star\star1\star$	races against	$\star\star\star0\star$
$\star\star\star\star1$	races against	$\star\star\star\star0$

Thus, given D inputs, we have D races occurring in parallel, and we stop all races when any one race produces a winner (to confidence level δ). On each step of the race a random binary string is generated, and then the **LOOCV** error of one randomly chosen datapoint is computed using that binary string. This statistic is added to the statistics of all the strings in the above sets of races which match the binary string. This continues until one of the pairs of racers becomes significant, i.e. when we believe with probability $1-\delta$ that one member of the significant pair beats its competitor. Then the next iteration of the race begins with a new set of racers which all have the winning field of the previous race switched on. If, in the above race, $\star 1 \star\star\star$ became significantly better than $\star 0 \star\star\star$ the next iteration would have

$11\star\star\star$	races against	$01\star\star\star$
$\star 11\star\star$	races against	$\star 10\star\star$
$\star 1\star 1\star$	races against	$\star 1\star 0\star$
$\star 1\star\star 1$	races against	$\star 1\star\star 0$

This may be preferable to our other hill climbing racers for three reasons:

- If any feature is outstandingly good, it will be detected quickly without having to wait for an entire iteration of hill climbing to take place.
- If several features are independently good then one of them will be quickly selected, without having to wait to determine which precisely is the best, which is a weakness of FOR-BRACE and BACK-BRACE.
- Small, mutually dependent, families of features that would be missed by the other hill climbers may be found. If features 1, 2 and 3 must all be on to gain any benefit, then schemata string $1\star\star\star\star\star$ will eventually win a race against $0\star\star\star\star\star$ because 25% of the strings generated from the former have features 1, 2, and 3 all on, whereas 0% of the latter do. We have performed experiments, not shown here, to test this phenomena in noisy binary optimization problems with mutually dependent families of up to size five, and schemata search is the only algorithm that finds the correct family.

4 EXPERIMENTS

We have run all these algorithms on fifty-six randomly generated synthetic datasets. The task was to find the set of features which minimized the leave-one-out cross-validation error of a 1-nearest-neighbor function approximator. The datasets all had between 4 and 12 inputs and one real-valued output which was a noisy multivariate function of a random subset of the inputs. All inputs were randomly generated uniformly in the range $-1 \leq x_i \leq 1$. The multivariate function was from the syntax in Table 2.

The number of terms in the dataset was also randomly decided, and varied between 5 and 30. Thus some datasets were trivial, such as output $= \frac{1}{2}(x_2 + x_7)$, and others complex, such as

```
output = max( corrupt( product(
mean( g( x4 ) , g( corrupt( x2 )
) ), product( x5 , corrupt( x5 )
) ) ) , g( corrupt( g( g( g( g(
corrupt( max( x5 , corrupt( x3 )
) ) ) ) ) ) ) ) )
```

It is interesting to note that all the searchers managed to identify the precise set of relevant inputs for this complex dataset, which had 950 datapoints[2]. In all the experiments, $\delta = \gamma = 0.001$.

Figure 2 shows the performance of the forward searchers and schemata on the 56 random datasets. There are two measures of performance.

- **Accuracy.** How often do the searchers end at suboptimal solutions? This is shown by the columns
  - P_{IP} = the percentage of datasets for which the searcher produced an "imperfect" result. A result is imperfect if any other search produced a feature-set with a lower **LOOCV** error.
  - P_{FW} = the percentage of datasets for which the result was fairly wrong, i.e. had a **LOOCV** error more than 0.001 greater than the minimum found by any other search. To give this number some meaning, the minimum **LOOCV** errors found were typically in the range $0.01 - 0.2$ depending on the dataset, with a similar magnitude of variation.
  - P_{VW} = the percentage of datasets for which the result was very wrong, i.e. had a **LOOCV** error more than 0.01 greater than the minimum found by any other search.
- **Search time.** This is given by the number of individual evaluations of **LOOCV** errors. The mean figure is shown, but this is dominated by the few hard datasets which required tens of thousands of evaluations for all methods. Many other datasets required only thousands, or in some cases, hundreds of evaluations. For this reason, also shown (by scatterplots) are the distributions of the ratio of number of samples needed compared with the number of samples needed by the conventional forward selection method. As can be seen, this distribution is highly skewed, especially for the schemata searches. 50% of the schemata searches took less than a quarter of the time of the conventional search. 12 of the 56 schemata searches took over twice as long.

Figure 3 is a similar table comparing the various backward methods and the same schemata searches (which have no forward or backward biases).

The forward and backward racing methods were usually faster than the conventional methods with little loss of accuracy. The Gauss-Seidel races were similar in performance. The schemata search was also roughly equal in accuracy,

[2] The slowest method was FOR-SEL, needing 19000 evaluations, and the fastest was SCHEMATA+, described shortly, which needed 2215 evaluations

Table 2: Syntax Of Multivariate Functions In Experiments

expr ::=	X_i	the ith input
	expr $\times$ **expr**	the product of the subexpressions
	mean(**expr**, **expr**)	the mean of the subexpressions
	max(**expr**, **expr**)	the maximum of the subexpressions
	corrupt(**expr**)	gaussian random noise of ± 0.1 is added to the value of the subexpression
	g(**expr**)	where g is a non-differentiable function $g(x) = -x^2$ if $x > 0$ and $g(x) = -x^2 - 2x$ if $x \leq 0$.

except that in this set of experiments it achieved the distinction of no "very wrong" errors.

Of the twelve schemata searches which were twice as long as the conventional forward method, seven were due to the conventional method quickly becoming stuck with an inferior solution—after considerably more computation schemata search found a better result. Of the other five, four eventually found equally good solutions and one found a slightly inferior solution. Interestingly the schemata searches frequently found all the relevant features very fast, often in a tenth of the total time of their search. This produced strings with only 1's and $\star$'s in them. But they would then spend a very long time convincing themselves that they were justified in putting 0's elsewhere.

As an initial and ugly attempt to address this, we tried an additional algorithm, SCHEMATA+, which would give up and replace $\star$'s with 0's very eagerly. If 2000 iterations of one of its races produced no significant winners, it forced one of the stars to zero (using the race statistics to choose the input least likely to be relevant). SCHEMATA+ was the fastest algorithm at converging but was less reliable than SCHEMATA.

5 DISCUSSION

We would like to extend this research in several ways. Firstly, we could investigate the formal relationship between the cutoff probability, δ, the tolerance factor γ, and the overall probability that a race will produce a satisfactory answer. Secondly, we could explore the relationship between our searches and the techniques of gradient descent and genetic optimization.

Gradient descent can be used in combination with cross validation and memory-based methods [Atkeson, 1990], and can also be used to identify irrelevant features. It can select between models that are parameterized by smoothly varying continuous parameters. Gradient descent can be particularly useful for locally refining near-optimal solutions. In other cases, we suspect that it would often be more computationally expensive, and more prone to local maxima, than our discrete searches, but additional investigation is merited.

Genetic optimization has an evident similarity to the schemata search presented here, but in earlier work [Moore *et al.*, 1992] we saw that it was inferior to conventional hill climbing (producing poorer solutions and requiring more computation time). Indeed, the invention of schemata search was intended to yield some of the widely reported benefits of genetic optimization, while using statistical bookkeeping to minimize binary string evaluations.

In [Aha, 1990, Aha *et al.*, 1991] a method is given to permit nearest-neighbor-like learners to incrementally improve estimates of the relevance of input features. An advantage of their technique is that it operates locally, so that features that are relevant only in some parts of the input space may be ignored elsewhere. It would be interesting to try combining schemata search with this kind of local model selection.

6 CONCLUSION

This paper introduced the notion of "racing" with Bayesian statistics to accelerate model selection. It then extended these algorithms in several new ways. Some of these new search methods may have applications elsewhere in machine learning, and this merits further investigation.

APPENDIX

This appendix concerns inferences about means of normal distributions from a random sample using Bayesian statistics. The analysis is directly from [Schmitt, 1969].

Let us assume that a priori we know nothing about the distribution except that it is normal. The mean μ might be any value between $-\infty$ and $+\infty$. The variance σ^2 might be any value between 0 and $+\infty$. This ignorance can be turned into the uninformative priors:

$$p(\mu) = \text{Constant} \tag{12}$$

$$p(\sigma) = \text{Constant}/\sigma \tag{13}$$

Where $p(.)$ denotes a probability density function[3].

Then, if we observe a sample $\{x_1, x_2, \ldots, x_n\}$ we can define

[3]Note that neither of these priors is a legitimate probability density function. Such an approximation is harmless (see [Schmitt, 1969] for more details).

Method	Mean # Evals	P_{IP}	P_{FW}	P_{VW}	Mean relative evals	Median relative evals
FOR-SEL	17378	17.9	16.1	7.1	1	1
FOR-RACE	6658	19.6	17.9	5.4	0.51	0.43
FOR-GS-RACE	7960	17.9	14.3	5.4	0.67	0.53
SCHEMATA	7779	16.1	8.9	0	1.7	0.22
SCHEMATA+	4159	19.6	14.3	5.4	0.9	0.13

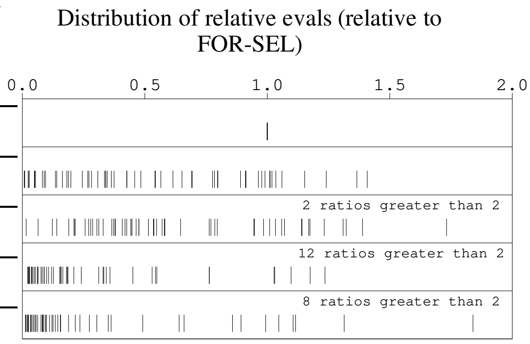

Figure 2: Comparing the conventional forward selection algorithm against its racing counterparts, and against schemata search (which has no forwards-backwards bias).

Method	Mean # Evals	P_{IP}	P_{FW}	P_{VW}	Mean relative evals	Median relative evals
BACK-EL	26358	16.1	14.3	7.1	1	1
BACK-RACE	13943	16.1	12.5	7.1	0.62	0.57
BACK-GS-RACE	5611	17.9	16.1	5.4	0.46	0.23
SCHEMATA	7779	16.1	8.9	0	2.1	0.18
SCHEMATA+	4159	19.6	14.3	5.4	1.0	0.11

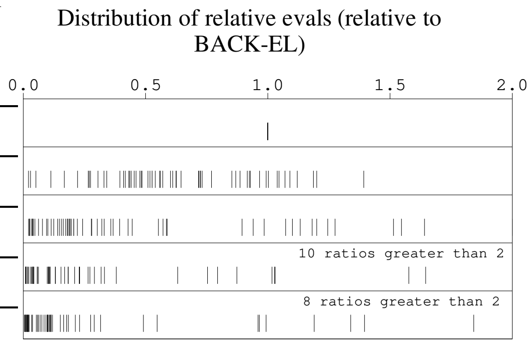

Figure 3: Comparing the conventional backward elimination algorithm against its racing counterparts, and against the same schemata search results as in Figure 2.

sample size n, sample mean

$$\bar{x} = \frac{1}{n} \sum_i x_i \tag{14}$$

and sample variance

$$s^2 = \frac{1}{n-1} \sum_i (x_i - \bar{x})^2. \tag{15}$$

The marginal posterior distribution of the mean μ is a student distribution with mean $\bar{x}$, variance s^2/n, and $n-1$ degrees of freedom. The cumulative density function of this distribution can then be used to compute the probability that the true mean μ lies within any given interval.

In our implementation of BRACE the computation of

$$\text{Prob}(h^*_{jj'} < -\gamma) < \delta$$

(Equation 10) is achieved in this manner. For each j and j' the sample size is k, the sample mean is $\hat{\mu}^h_{jj'}(k)$ and the sample variance is $(\hat{\sigma}^h_{jj'}(k))^2$ from[4] Equations 8 and 9.

[4]It should be noted that these statistics can be updated incrementally: $\hat{\mu}^h_{jj'}(k+1)$ and $\hat{\sigma}^h_{jj'}(k+1)$ can be defined in terms of k, $\hat{\mu}^h_{jj'}(k)$, $\hat{\sigma}^h_{jj'}(k)$, and $e_j(k+1) - e_{j'}(k+1)$.

In our implementation of RACE we need to compute the probability that the means of two distributions differ by more than a certain amount (Equation 6). This is achieved by the Welch approximation to the Behrens-Fisher problem [Welch, 1937].

Given two samples with the same assumptions as before, let the first sample have size n_1, sample mean $\bar{x}_1$ and sample variance s_1^2. The corresponding values for the second sample are n_2, $\bar{x}_2$ and s_2^2. Let

$$u_1 = s_1^2/n_1, u_2 = s_2^2/n_2, \tag{16}$$

$$b = u_1/(u_1 + u_2). \tag{17}$$

Then $\mu_1 - \mu_2$ (the signed difference between the population means) has, approximately, a student distribution with mean $\bar{x}_2 - \bar{x}_1$, variance $u_1 + u_2$, and degrees of freedom

$$\left(\frac{b^2}{n_1 - 1} + \frac{(1-b)^2}{n_2 - 1} \right)^{-1}. \tag{18}$$

ACKNOWLEDGEMENTS

Thanks to Justin Boyan, Oded Maron, and the reviewers of the paper for many helpful comments. Some of this work was supported by a research gift to Andrew Moore from 3M corporation.

References

[Aha *et al.*, 1991] D. W. Aha, D. Kibler, and M. K. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6:37–66, 1991.

[Aha, 1990] D. W. Aha. A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical and Psychological Evaluations. PhD. Thesis; Technical Report No. 90-42, University of California, Irvine, November 1990.

[Atkeson, 1990] C. G. Atkeson. Memory-Based Approaches to Approximating Continuous Functions. In *Proceedings, Workshop on Nonlinear Modeling and Forecasting, Santa Fe, New Mexico*, 1990.

[Box *et al.*, 1978] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters*. Wiley, 1978.

[Caruana and Freitag, 1994] R. A. Caruana and D. Freitag. Greedy Attribute Selection. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, 1994.

[Cleveland and Delvin, 1988] W. S. Cleveland and S. J. Delvin. Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting. *Journal of the American Statistical Association*, 83(403):596–610, September 1988.

[Gratch *et al.*, 1993] J. Gratch, S. Chien, and G. DeJong. Learning Search Control Knowledge for Deep Space Network Scheduling. In *Proceedings of the 10th International Conference on Machine Learning*. Morgan Kaufman, June 1993.

[Greiner and Jurisica, 1992] R. Greiner and I. Jurisica. A statistical approach to solving the EBL utility problem. In *Proceedings of the Tenth International Conference on Artificial Intelligence (AAAI-92)*. MIT Press, 1992.

[John *et al.*, 1994] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the Subset Selection Problem. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, 1994.

[Kaelbling, 1990] L. P. Kaelbling. Learning in Embedded Systems. PhD. Thesis; Technical Report No. TR-90-04, Stanford University, Department of Computer Science, June 1990.

[Maron and Moore, 1994] O. Maron and A. Moore. Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. In *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, April 1994.

[Miller, 1990] A. J. Miller. *Subset Selection in Regression*. Chapman and Hall, 1990.

[Moody and Utans, 1992] J. E. Moody and J. Utans. Principled Architecture Selection for Neural Networks:, Application to Corporate Bond Rating Prediction. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, April 1992.

[Moore *et al.*, 1992] A. W. Moore, D. J. Hill, and M. P. Johnson. An Empirical Investigation of Brute Force to choose Features, Smoothers and Function Approximators. In S. Hanson, S. Judd, and T. Petsche, editors, *Computational Learning Theory and Natural Learning Systems, Volume 3*. MIT Press, 1992.

[Moore, 1992] A. W. Moore. Fast, Robust Adaptive Control by Learning only Forward Models. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, April 1992.

[Schaal and Atkeson, 1994] S. Schaal and C. G. Atkeson. Assessing the Quality of Local Linear Models. In *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, April 1994.

[Schmitt, 1969] S. A. Schmitt. *Measuring Uncertainty: An elementary introduction to Bayesian Statistics*. Addison-Wesley, 1969.

[Skalak, 1994] D. B. Skalak. Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann, 1994.

[Welch, 1937] B. L. Welch. The significance of the difference between two means when the population variances are unequal. *Biometrika*, 29, 1937.