An Evolutionary Bootstrap Approach to Neural Network Pruning and Generalization

Blake LeBaron
Department of Economics
University of Wisconsin - Madison
1180 Observatory Drive
Madison, WI 53706
(608) 263-2516
blebaron@ssc.wisc.edu

July 1997
Revised: August 1997

Abstract

This paper combines techniques drawn from the literature on evolutionary optimization algorithms along with bootstrap based statistical tests. Bootstrapping is used as a general framework for estimating objectives out of sample by redrawing subsets from a training sample. Evolution is used to search the large number of potential network architectures. The combination of these two methods creates a network estimation and selection procedure which finds parsimonious network structures which generalize well. The bootstrap methodology also allows for objective functions other than usual least squares, since it can estimate the in sample bias for any function. Examples are given for forecasting chaotic time series contaminated with noise.

1 Introduction

Nonlinear time series forecasting is still a difficult problem with many unanswered questions. The researcher is faced with a huge array of possible techniques as well as control parameters which are difficult to determine.[1] This paper proposes to address this problem using two recent procedures, evolutionary search techniques, and bootstrap cross-validation. They will be used in a simple example using artificial neural networks as function approximators for a chaotic time series with additive noise.

Evolutionary methods have proved to be a useful tool in network construction and estimation.[2] Also, Efron's bootstrap has been shown to be useful for network diagnostics, and forecast improvement.[3] Evolution will be used here to aid in searching the vast space of possible network structures while the bootstrap is used to estimate off training set, or out of sample, forecast error. The combination of these two procedures has not yet been explored.

A good question might be why two computationally intensive procedures need to be combined into an even more burdensome process. The basic reason put forth here is that both of these are necessary to deliver estimated function approximations able to maximize arbitrary objective functions for unknown noise distributions. Evolutionary search is used to find lean networks with many connection lines cut. For different network structures this search far exceeds what could be reasonably done enumeratively. Bootstrapping allows the general estimation of in sample bias for arbitrary model structures, noise distributions, and objective functions. This makes it a powerful tool in the search for reliable approximations. This power comes with several costs. First, it is computationally burdensome, and for some problems might quickly outstrip available computer resources. Second, the theory for bootstrap cross-validation is still not well understood, and the procedures used have no asymptotic justification.

The second section introduces both the bootstrap and evolutionary network selection systems that will be used in the later applications. The third section applies the methods to sets of simulated time series. Tests are peformed to check the reliability of this model identification system, and comparisons with some other more traditional identification systems are performed. Finally, in the last part of the third section some comparisons are made showing where the bootstrap/evolutionary method may have significant advantages. The final section summarizes and concludes.

2 Model Selection and Estimation

2.1 Bootstrap Cross-validation

In the model selection procedure candidate networks will be evaluated according to their in sample objective functions adjusted by an estimate of the in sample bias. This estimate is obtained using the bootstrap (Efron 1979). Specifically, a method called the 0.632 bootstrap is used. Details on this can be found in (Efron 1983) and (Efron & Tibshirani 1993). The basic idea is to try and estimate the in sample bias by drawing a new sample from the original sample with replacement, and using this as a training set with the original sample taking the role of a "clean" test set. This estimate of in sample bias will clearly underestimate the true bias because of the large overlap between the original sample and the bootstrapped training samples. The 0.632 bootstrap attempts to provide some correction for this sampling overlap.

[1] A very comprehensive survey on the area of nonlinear forecasting is (Weigend & Gershenfeld 1993).

[2] See examples in (Balakrishnan & Honavar 1995), (Mitchell 1996), and (Yao 1996).

[3] For examples of bootstrap applications to neural networks see (Breiman 1994), (Connor 1993), (LeBaron & Weigend 1994), (Paaß 1993), and (Tibshirani 1994).

This is a quick overview of this method. Squared error is used in the following examples, but other fitness measures could be used as well. Define the squared forecast error for an approximating function, f, and estimated parameters, $\hat{\beta}$, as,

$$\hat{e}^2(n) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i, \hat{\beta}))^2, \tag{1}$$

where $\hat{\beta}$ is estimated on the entire sample of length, n. A bias adjustment, ω_n, is equal to the difference between the in sample estimate, and the true expectation over the population,

$$\omega_n = E((Y - f(X, \hat{\beta}))^2) - E(\hat{e}^2(n)). \tag{2}$$

An estimate of ω_n would allow estimation of true off training set error, and selection of models based on that error. The bootstrap gives one possibility for estimating ω_n.

Begin the bootstrap procedure by drawing B samples of length n from the original sample indexed by b. Define K_b as the set of points in bootstrap b. Define $\hat{\epsilon}^{(0)}$ as the estimated squared error for points not in the training set. This is defined for B bootstrap replications as,

$$\hat{\epsilon}^{(0)} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{\#(i \notin K_b)}\sum_{i \notin K}(y_i - f(x_i, \beta^{(K_b)}))^2, \tag{3}$$

where K_b represents the set of points in each bootstrap draw. β^{K_b} is estimated on K_b, and the error is estimated over points not in K_b.

The 0.632 bootstrap estimates the bias adjustment to this in sample error as[4]

$$\hat{\omega}^{(0.632)} = 0.632(\hat{\epsilon}^{(0)} - \hat{e}^2). \tag{4}$$

The adjusted error estimate is

$$\hat{e}^{(0.632)} = \hat{e}^2 + \hat{\omega}^{(0.632)} \tag{5}$$

$$\hat{e}^{(0.632)} = 0.368\hat{e}^2 + 0.632\hat{\epsilon}^{(0)} \tag{6}$$

This gives an estimate that is a weighted average of the in sample error, and the "out of sample" error from $\hat{\epsilon}^{(0)}$.[5]

A second technique, which is also used, is monte-carlo cross-validation. This is standard K-fold cross-validation where the subsets are drawn at random. The estimated error is again the mean over a set of montecarlo simulations.

$$\hat{\epsilon}^{MC} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{\#(i \notin K_b)}\sum_{i \notin K_b}(y_i - f(x_i, \beta^{(K_b)}))^2. \tag{7}$$

However, in this case the sets K_b, are drawn without replacement from the underlying distribution. Further descriptions, and some examples of performance can be found in (Shao 1993).

[4]To ease in notation, the n has been dropped from $\hat{e}^2$.

[5]The weighting, 0.632, is derived from the probability that a given point is actually in a given bootstrap draw, $1 - (1 - (1/n))^n \approx 1 - e^{-1} = 0.632$. The 0.632 bootstrap approximates the "distance" between the bootstrap samples and the original series, and the previously mentioned probability is the crucial adjustment factor. Details are given in (Efron 1983).

2.2 Network Evolution

Approximating functions $f()$ will be standard feedforward, single hidden layer, sigmoidal neural networks. These can be written as,

$$f(x, \beta) = \theta_0 + \sum_{j=1}^{h} \theta_j G(x\gamma_j), \tag{8}$$

$$G(a) = \frac{1}{1 + e^{-a}}, \tag{9}$$

where θ and γ are both in the parameter vector β. Each γ_j is a vector of parameters that is used to linearly weight the input vector, x. Hill climbing is used to estimate network weights, and the network architecture is determined through an evolutionary procedure.[6] There is a large and growing literature on evolving neural networks.[7].[8] This paper uses a very simple evolutionary structure. Obviously, improvements may be obtained using more sophisticated methods in the future.

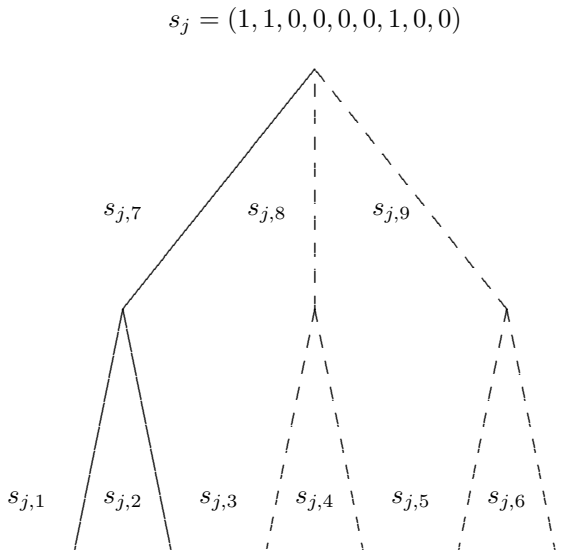
A population of network architectures is given by $(s_j, w_j^0; w_j)$ where j is the population index, and s_j is a vector of binary variables representing each connection in a network. Let L be the number of nonzero entries in s_j. w_j^0 is a real vector of length L that gives the starting values used in hill climbing, and w_j is a real vector of length L representing the final "optimized" network weights. Figure one demonstrates how the binary coding is mapped into network architectures. Each element of the vecture s_j corresponds to a connection in the network. Dashed lines refer to connections that are not active.

[6] See (Hart & Belew 1996) for another example of a hybrid technique similar to this one.

[7] Useful surveys on this field can be found in (Balakrishnan & Honavar 1995) and (Yao 1996).

[8] Related work can be found in the literature on pruning. A few examples of this are (Cun, Denker & Solla 1990), (Finnoff, Hergert & Zimmermann 1993), and (Hassibi & Storck 1993).

Figure 1: Single hidden unit, 2 inputs



New networks are evolved using tournament selection and two mutation operators, micro and macro mutation. A parent network is selected by choosing 5 networks at random (from a population of 50), and then using the fittest of these. Micro mutation involves looking at the connections between the hidden units and inputs. A hidden unit is chosen at random, and then one of the bits controlling the inputs to that unit is chosen. This bit is then flipped from 0 to 1, or 1 to 0. Macro mutation involves choosing a hidden unit. Its connection to the output is then flipped from 0 to 1, or 1 to 0. If it is activated, then each of its input lines are activated with probability, 0.5. Both types of mutations occur with probability 0.4. The remaining 20 percent of the time, the network structure is simply copied to create a new network with the same structure, but new starting values will be chosen for hill climbing. Once the child's architecture is decided through mutation, and the starting values are drawn, its final weights, w_j, are set using a hill climbing algorithm.[9]

Bias estimates proceed according to the bootstrap or cross-validation methods described earlier. In a world with infinite computing time available, the procedure would perform a large number of bootstrap iterations on all new rules entering the population. Unfortunately, this is infeasible, so bootstrap iterations

[9]For squared error estimates a Levenberg-Marquardt method is used, and for other objectives, a simplex method is used. The former uses analytic derivatives.

are built up as the population evolves. A new bootstrap iteration is performed for all members of the population at each generation. This means that the bias estimates are continuously being updated over time, and the stochastic fitness values will be changing. For each bootstrap bias estimate, a new subsample, K_b is drawn, and β is estimated through hill climbing using the stored network starting values, w_j^0.[10]

New networks have an initial startup number of bootstrap simulations of 5. This prevents many simply lucky, but not very good, networks from invading the population. Also, new networks must be better than the parents they replace.[11] The population size is set to 50, and 15 potential new networks are created at each generation. The selection method has the property of slowing down evolution as the population improves, allowing it to concentrate on getting good bootstrap bias estimates for the current set of networks since networks will spend more time in the population. New networks are started at randomly drawn initial weights.[12] These starting values are stored in w_j^0 so they can be used again for each bootstrap sample. The final estimated weights are stored in w_j.

A summary of the evolution bootstrap procedure is given in the following list:

1. "Select" 15 candidate networks to be parents for the next generation.
2. Mutate each of these according to macro, micro, or copy operators.
3. Choose new random starting values for each new child.
4. Hill climb each of these on the entire training set to get final weights, ω.
5. Perform 5 bootstrap bias estimates on randomly drawn subsamples of the data for each new network to estimate its fitness.
6. Merge the parent and child populations, and keep only the 50 best networks from both.
7. For each network in the population, perform another bootstrap bias estimate, b, and get a new estimate for $\hat{e}^{(0)}$. Use this to update the adjusted fitness.
8. Go to 1 until last generation.

A serious problem for sigmoidal neural networks is that of local minima. The possibility of having candidate solutions that have stopped at a local, and not a global minimum adds a troubling aspect of imprecision. An attempt is made to avoid these local minimum solutions in the population. New networks can be direct copies of old network structures subject only to a change in the startup vector, w^0. This new candidate network is simply a retry of an old network structure at new starting weights. If this network out performs its parent in the population in terms of in sample objectives, then it shows that the parent was at a local minimum in terms of that objective. When this happens this child directly replaces the parent. In this way the system strives for networks that, subject to their architecture, minimize in sample objectives. Architecture decisions are made according to the bias adjustments previously described. This is in contrast to techniques such as "stop training" which may often stop at a local minimum on the training or cross validation sample.[13]

[10]The population based search is effectively faced with a very computationally costly objective function which can be better estimated with additional iterations. The population allocates the scarce computer resources to the most promising solutions.

[11]This is similar to $\mu + \lambda$ selection described in (Bäck 1996), and the election operator of (Arifovic 1994).

[12]The weights are drawn uniformly from $[-1/m, 1/m]$ where m corresponds to the number of inputs.

[13]The concern for avoiding local minimum is approached in another bootstrapping context by (Tibshirani & Knight 1995).

3 Hénon Map Time Series Forecasting

3.1 The Hénon Map

The Hénon map is one of the simpler chaotic maps, and provides a useful testing framework for the forecasting methods proposed in this paper.[14] It is given by the following set of difference equations,

$$\begin{aligned} x_{t+1} &= 1 - 1.4x_t^2 + y_t \\ y_{t+1} &= 0.3x_t. \end{aligned} \tag{10}$$

Here, attention will be focused on forecasting x_t alone from lagged x values. The Takens embedding theorem, (Takens 1983), states that if the underlying system is a diffeomorphism, then an embedding, or vector of lags $(x_t, \ldots, x_{t-m})$, for large enough m, can completely characterize x_{t+1}. More specifically, there exists a function $g()$ such that

$$x_{t+1} = g(x_t, \ldots, x_{t-m}) \tag{11}$$

Define a noise contaminated series as,

$$z_t = x_t + \epsilon_t, \tag{12}$$

where ϵ_t is independent, identically distributed noise. Forecasts will be made on future values of z_t using lagged values of the "clean" series, x_t. In other words the prediction problem is to approximate $f()$ in the following nonlinear regression problem,

$$z_{t+k} = f(x_t, \ldots, x_{t-m}) + \epsilon_k. \tag{13}$$

This problem differs slightly from the more traditional approaches of adding noise to dynamical systems. In some cases the laws of motion are made stochastic by introducing noise into the dynamical system itself. In other cases, "read out" noise is added to the underlying series and this series is considered the time series of interest. In this example this would imply that lagged values of z_t would be used in the above regression. The method used here provides a better benchmark test for several reasons. First, the best one can do in terms in terms of forecasting is clear. If $f()$ were known, or approximated perfectly, then the the distribution of ϵ_t would characterize the forecast error. Second, the use of noisy explanatory variables introduces the difficult problem of errors in variables.[15][16]

3.2 Mean Squared Error Prediction

This section presents experiments using simulated Hénon data. Predictions are made 2 periods into the future so the following function is being estimated,

$$z_{t+2} = f(x_t, \ldots, x_{t-m}; \beta) + \epsilon_{t+2}. \tag{14}$$

Gaussian noise is added to the series with variance equal to 1/2 the variance of x_t. All estimation and validation takes place in a sample of length 100 which is taken after the system has been run for 800 time

[14] For other results using ANN's and the Hénon map see (Gençay 1994). Also, papers and references to other work on the large area of forecasting chaotic time series can be found in (Eubank & Casdagli 1991) and (Weigend & Gershenfeld 1993).

[15] Much is known about this problem for the case of linear regression, (Judge, Griffiths, Hill & Lee 1980). Work on the nonlinear case is only just beginning (Weigend, Zimmermann & Neuneier 1996).

[16] It is assumed that the network can do a reasonable job at approximating the chaotic series. This is not unreasonable given the approximation theory that has been proved for several types of networks structures. Much of this theory is surveyed in (Kuan & White 1994).

steps to make sure that it has settled into a stationary attractor. The initial value of y is set to zero, and the intial value of x is chosen uniformly on $(0, 1)$. The added noise for this set of experiments will be Gausian. Finally, a test sample of length 1000 is used to report out of sample prediction. This sample is not used in the estimation or model selection procedures. These parameters have been chosen with several constraints in mind. First the series should be able to be approximated to within machine precision in the no noise case. This shows that in the case of clean data the search and estimation procedure can find good approximators for $g()$. The second constraint was that overfitting must be a relevant problem. For the simple Hénon case, as the sample size gets long most of the methods used here get arbitrarily close to $g()$. The problem of overfitting appears in short noisy series, where the definition of short and noisy depends on the underlying complexity of the system.

The network takes 5 lags of x_t from the map as inputs along with a constant set to 1, or bias term which gives 6 possible inputs to each hidden unit. There is also a constant term for the entire network. Networks are limited to 6 hidden units as a maximum. This was found not to be a binding constraint for most cases.

The function, f, and parameter vector, β are estimated using the evolutionary procedure outlined above, and a mean squared error objective, MSE,

$$\hat{e}^2(n) = \frac{1}{n}\sum_{t=1}^{n}(z_{t+2} - f(x_t, \ldots, x_{t-m}; \beta))^2. \tag{15}$$

Various adjustments will be added to this objective in attempting to correct for overfitting biases. The two previously mentioned adjustments, the bootstrap, and monte-carlo cross validation adjustments will be augmented with two more traditional penalty terms, the Schwarz information criterion (Schwarz 1978), or BIC, and the Akaike information criterion, or AIC (Akaike 1973). BIC uses the following objective function,

$$\log(\hat{e}^2(n)) - \frac{\log n}{n}(p), \tag{16}$$

and AIC uses,

$$\log(\hat{e}^2(n)) - \frac{2}{n}(p). \tag{17}$$

In each case p is the number of parameters, or weights, used in the given network structure.

Table 1 presents summary statistics over sets of 30 runs of the Hénon map for each of these different methods. The first line, labeled none, refers to no bias adjustment. The first two columns show the MSE in the training sample and the test sample. The increase from 0.12 to 0.55 shows a good example of overfitting. The variance of the noise of 0.25 is the minimum forecast MSE that is possible with perfect forecasts. It is clear that this case does poorly relative to this benchmark. The table also shows the standard error for the test MSE, σ(MSE Test). It also shows the mean of the ratio of the adjusted training MSE divided by the test MSE across the 30 runs. This is an indication of how well the adjusted MSE did in predicting the true test sample objective. In the case of no adjustment this is just the mean ratio of training to test sample forecast variances. The small value of 0.25 repeats the results of the first two columns. The last two columns describe the types of networks fit. The first, labeled hidden, shows the mean number of hidden units (out of a maximum of 6) that were used, and the second, labeled connection fraction, shows the fraction of connections in the network that are activated. Both these numbers are highest for the zero adjustment case, emphasizing that bias adjustment has an important impact on network parsimony.

The next two rows in the table, labeled Bootstrap, and MC Cross Val., refer to the two adjustment methods described in the second section. In both cases we see a dramatic reduction in overfitting, which is revealed in a significant reduction in out of sample MSE. Both methods do moderately well at forecasting the

Table 1: Mean Squared Error Objective

Adjustment	MSE Train	MSE Test	σ(MSE Test)	Adjusted Train MSE/ Test MSE	Hidden Units	Connection Fraction
None	0.1287	0.5478	0.0222	0.2452	6.000	0.6992
Bootstrap	0.2375	0.3443	0.0087	0.9463	3.682	0.3326
MC Cross Val.	0.2248	0.3515	0.0101	0.9073	3.732	0.3372
BIC	0.2373	0.3495	0.0089	1.1753	3.067	0.2736
AIC	0.2045	0.4190	0.0181	0.7388	4.166	0.4302
True Cross Val.	0.2413	0.3098	0.0056	0.9697	3.934	0.3752

Mean squared error estimates for length 100 training, and length 1000 test set experiments. Numbers are means over 30 different runs. σ is the standard error of the mean. Hidden is the mean number of hidden units, and connection is the fraction of connections set. The best possible forecast given the simulated noise would give a MSE of 0.25.

out of sample MSE with Adjusted Train/Test ratios of 0.95 and 0.91, respectively. They also fit much leaner networks using only slightly more than half the available hidden units, and about a third of the possible connections.

The next two rows present results for the two complexity penalties, BIC, and AIC. In the case of BIC, most of the results indicate that it would be difficult to distinguish between the two cross validation methods. It tends to over estimate out of sample error, but since BIC is not really designed for this, this number should not be taken too seriously. Finally, it does appear to fit even leaner networks with a connection fraction of 0.27. AIC performs very badly on this problem. It appears to be significantly worse than either BIC, or the two cross validation methods, but better than no adjustment.

To further test the performance of the evolutionary procedure without the bootstrap, a special cross validation test is performed. In this case a second cross validation series is provided for model specification testing. Specifically, an extra 1000 point series is used only for model cross validation. Networks are still trained on the 100 length training set, but their fitness in the network population is evaluated on this special cross validation set. This allows the network to maintain all the problems of fitting a network on the short noisy data, but architecture selection problems should be much better solved given this true cross validation series. Results for this test are given in the row labeled, True Cross Val., in table 1. It is clear that this run produces the lowest test MSE, and the best prediction of test error, with a training/test ratio of 0.97. This should be expected in this best case benchmark. Interestingly, it does not give the leanest networks, with a connection fraction of 0.38.

3.3 Network Pruning

This section explores the value added of network pruning on the performance of the fitted time series models. The goal is to compare the evolutionary procedures of the previous section with more commonly used techniques. The time series will be approximated with a fully connected network. Network architecture is varied only over the number of hidden units. Also, each network architecture will be started at many different random starting values. For each of the different complexity penalty methods 100 networks are estimated for 1 through 6 hidden units, for a total of 600 networks. Each network is started at different random starting weights.

Comparisons between table 2 and table 1 show large improvements from using the evolutionaty pruning algorithm. For example, under the BIC penalty, training set MSE moves from 0.51 to 0.35 when pruning is

Table 2: Fully Connectected Network Comparisons

Adjustment	MSE Train	MSE Test	σ(MSE Out)	Adjusted Train MSE/ Test MSE	Hidden Units	Connection Fraction
None	0.1627	0.6009	0.0421	0.3014	5.734	0.9566
BIC	0.3165	0.5105	0.0147	1.3981	2.433	0.4194
AIC	0.1796	0.5107	0.0330	0.7888	5.033	0.8426
True Cross Val.	0.2222	0.4085	0.0101	0.9809	5.000	0.8372

Mean squared error estimates for length 100 training, and length 1000 test set experiments. Numbers are means over 30 different runs. σ is the standard error of the mean. Hidden is the mean number of hidden units, and connection is the fraction of connections set. The best possible forecast given the simulated noise would give a MSE of 0.25.

allowed. Even in the case with the true cross validation objective the test set MSE goes from 0.41 to 0.31. All these improvements appear to be significant given the estimated standard errors.

It is clear that restricting the space to fully connected networks drives the system to less parsimonious network structures. In the case of BIC the mean fraction of connections over the 30 runs changes from 0.42 to 0.27 when pruning is allowed. Interestingly, in some cases pruning allows the system to fit a model with *more* hidden units since in the fully connected case each unit brings with it a full set of input connections. For example, in the BIC case the mean number of hidden units rises from 2.4 to 3.1 when pruning is allowed. Since this relation does not hold across all the different methods, it is not clear how often this will be the case, and how significant this difference is.

These results clearly show that all of the proposed newer estimation methods in table 1 show significant improvements over more traditional methods with fully connected networks.[17] The comparisons were not able to include bootstrap methods due to the computational burden with that many network structures evaluated in parallel.

3.4 Objective functions

It appears that the bootstrap performs well in terms of avoiding overfitting, but in the experiments shown so far it is a computationally intensive equivalent of the BIC criteria which is quite close in most of the comparisons. To see how they may differ two changes are implemented that take advantage of the limitations of the BIC. First, the error distribution is changed from Gaussian to a skewed three point distribution, and second, a non MSE objective will be used. The nongaussian noise means that the likelihood approximations for the BIC, and AIC penalties will be farther off target. The objective function change makes both penalty parameters unusable for direct estimation. Estimation will be carried out in the BIC and AIC cases for a MSE error objective, and this will be used to approximate the other objective function.[18] Criteria beyond squared error loss functions do not allow the use of these penalty functions. Both of these cases do not bother the basic assumptions behind the bootstrap, and it should still be valid.

The first of these objectives is obtained by altering the noise distribution. The sample size is dropped to 50 in the training set, and the noise is changed from Gaussian, to a 3 point distribution, $\{-1, 0, 0.5\}$ with

[17] These results are conditional the approach used for fully connected network selection. Other approaches might yield different results.

[18] This is not an unreasonable thing to do since the MSE estimation should give a consistent estimate of the conditional mean, which in the situation considered will easily transform into the control based objective. It is not clear what will happen in small samples, or whether the BIC can properly penalize networks for this task.

probabilities, $(1/6, 1/2, 1/3)$. This gives a mean zero, negatively skewed noise with zero mean, and variance equal to that in the previous tests, 0.25.

The second difference in these final tests is that the objective function will be changed. MSE minimization may be useful in many cases, but there may be instances in which other objectives are desired. The example used here is motivated by finance, with connections to other control problems. Consider the time series above given by z_t. Assume the controller at time t is interested in finding a control variable, b_t, in the set $\{-1, +1\}$ which maximizes,

$$E(b_t z_{t+1}), \tag{18}$$

where b_t is formed based on time t information. This corresponds very closely to trading rule situations where b_t would indicate a buy $(+1)$ or sell(-1) signal. This objective function may appear close to a sign prediction objective, but it is not exactly the same.[19] The expectation is approximated by the sample average,

$$L_n = \frac{1}{n}\sum_{t=1}^{n}(b_t z_{t+1}), \tag{19}$$

where b_t is found by taking the output of the network and mapping it into $\{-1, 1\}$ depending on the sign of the network output. This is equivalent to putting an signum function on the output of the previously described networks. This more complicated objective function requires a more general optimization system. Gradients are badly behaved in finite samples, so a simplex method is used.[20] Shifting to this less powerful optimization algorithm is a severe handicap for using this objective function. Future modifications may involve a smooth function in the sign mapping which would allow the use of derivative based hill climbing.[21]

Results for various approaches to this problem are compared in table 3. The sign objective L_n is given in columns 3 and 4 for both the training sample of length 50, and the test sample of length 1000, and the last column gives the standard deviation across the 30 runs for the sign test in the test sample. The first three lines of the table show results for methods directly using the L_n objective function. This includes no adjustment (none), the bootstrap, and the true cross validation case, where a separate series of length 1000 is drawn for model selection. In all three of these L_n is used for both weight estimation and model selection. The first feature that should be noted is the increase in MSE from table 1. This is due to the smaller sample size as well as the fact that MSE is not the objective function. Columns 4 and 5 report the in sample and out of sample sign objective. It is clear that in the case of no adjustment overfitting is present. The sign objective here of 0.49 is less than the bootstrap, and true cross validation values of 0.55, and 0.64, respectively. These are still far from the theoretical maximum for this objective which should be equal to the absolute value of the series, x_t.[22] This was estimated to be 0.77 for a representative sample of length 1000.

The next 3 rows examine networks that were estimated on MSE objectives, and then used to generate b_t controls to get L_n. In these cases, true objectives, and estimation objectives differ. Because MSE is used in estimation the BIC penalty can again be used as a penalty. Row 4, labeled Bootstrap $\hat{e}^2$, reports results for a bootstrap bias estimator based on MSE estimation which is the same one used in table 1. This row

[19] See (Cumby & Modest 1987) for some examples.

[20] See (Press, Flannery, Teukolsky & Vetterling 1986) for information on the simplex method. The bad behavior of the gradients is due to the fact that changing a parameter by an arbitrarily small amount may flip the sign of a b_t, changing the objective function by a relatively large amount.

[21] Actually, in all cases the network is trained for 50 iterations using the Levenberg-Marquardt algorithm and a squared error objective, and then it shifts to the simplex algorithm.

[22] $E(b_t z_{t+1}) = E(b_t x_{t+1} + b_t \epsilon_{t+1})$, where the second term is zero, so the maximum corresponds to the absolute value of x_t given that b_t predicts the sign perfectly.

Table 3: Sign Objective

Adjustment	MSE In	MSE Out	Sign Objective In	Sign Objective Out	σ(Sign Obj)
None	0.1633	0.8289	0.7596	0.4859	0.0177
Bootstrap	0.2464	0.5685	0.7064	0.5544	0.0127
True Cross Val.	0.2788	0.4396	0.6913	0.6373	0.0044
Bootstrap $\hat{e}^2$	0.2678	0.4820	0.6118	0.5158	0.0179
BIC	0.2752	0.8159	0.5988	0.4484	0.0228
True Cross Val. $\hat{e}^2$	0.2179	0.3468	0.6589	0.6240	0.0070

Sign objective functions, L_n, for length 50 training and length 1000 test set experiments. Sign objective is the mean value of L_n over the 30 runs, for both the training and test periods as indicated. The maximum value for this corresponds to the absolute value of the time series which was estimated at 0.77 for a representative run of length 1000.

provides an interesting contrast to the first row of the table. The MSE bootstrap finds a network giving a lower MSE, and a lower sign objective. In other words, it is doing better on the MSE criterion, and worse on the sign criterion. This makes sense since it is estimated on the MSE. The next row, labeled BIC, shows the BIC penalty performing worse than the bootstrap, and much worse than the sign objective bootstrap of the first row. The fact that the BIC performs worse than the MSE bootstrap indicates the effect of the error distribution since both are using the same objective function. The experiment is the same as that in table 1 except for the unusual error distribution, and smaller training set. The final row shows the true cross validation example for the MSE objective. As should be expected, this is the best performer. What is interesting is that it doesn't perform too much worse in terms of the out of sample sign objective than the true cross validation case for the sign predictor. This demonstrates that when the model selection problem is close to being solved, the MSE objective can get very close to the conditional expectation, and therefore it generates good sign forecasts.

The next two tables perform bilateral comparisons on some of the results from the previous experiments. In table 4 the means for the out of sample sign objectives are compared using a simple t-test for the differences in means. Under the null hypothesis of equal means, these numbers are distributed $N(0,1)$.[23] It is clear from the table that the mean objective from the bootstrap is larger than that from BIC with a value of 4.06. Also, the sign objective based bootstrap is marginally better than the squared error bootstrap with a value of 1.76 which is significant at the 10% confidence level. Finally, the table indicates that the cross validation method using the large extra hold out sample generates larger objectives than both of the bootstrap methods.[24]

Comparisons of experiments using the means from simulation runs can be adversely effected by a few outliers. For this reason, a more robust distribution test is also implemented. Table 5 uses a Mann-Whitney-Wilcoxon test to compare the sign objective distributions over the different cases (Hogg & Craig 1978). This is a nonparametric test for the equality between two independent distributions. The test statistic will have a distribution that is asymptotically normal with zero mean and unit variance under the null hypothesis that the distributions are the same. In table 5 it is applied to the sign objective for the various methods using the distribution across the 30 runs performed on each in a series of bilateral comparisons. In the first row the value of 1.47 indicates that the two bootstrap methods are indistinguishable according to this test, while the bootstrap is still preferred to the BIC, but not as good as the true cross validation. Results similar to

[23]The values presented are $\frac{\mu_1-\mu_2}{\sqrt{\sigma_1^2+\sigma_2^2}}$, where 1 and 2 refer to the two comparison distributions.

[24]The minus sign indicates that the column method is larger.

Table 4: Mean Comparison: Sign Objectives

Base comparison	Bootstrap $\hat{e}^2$	BIC	True Cross Val.
Bootstrap	1.76	4.06	-4.80
Bootstrap $\hat{e}^2$		2.33	-5.64

T-tests for equality of means between different procedures. Values compare the row and column labeled tests. The test statistic is N(0,1) under the null hypothesis of mean equality. 10 and 5 percent critical values are 1.64 and 1.96 respectively for a two tailed test. A negative value indicates that the test labeled on the column is larger(better).

those in table 4 are shown in the second row. The MSE bootstrap still out performs the BIC using this more robust test.

Table 5: Distribution Comparisons: Sign Objectives

Base comparison	Bootstrap $\hat{e}^2$	BIC	True Cross Val.
Bootstrap	1.47	3.37	-4.19
Bootstrap $\hat{e}^2$		2.05	-4.95

Mann-Witney-Wilcoxon test for distribution equality. Values compare the row and column labeled tests. The test statistic is N(0,1) under the null hypothesis of distribution equality. 10 and 5 percent critical values are 1.64 and 1.96 respectively for a two tailed test. A negative value indicates that the test labeled on the column is larger(beter).

The results in this section suggest that there are possible advantages to using objectives other than MSE when they are desired, and that overfitting can be avoided in these cases using the bootstrap. The sign based bootstrap improved on the BIC, and it was marginally better than the MSE based bootstrap. The true cross validation continued to be the best performer which should be expected. It is displayed only as a comparison case to show what would happen in a situation where the model selection problem was close to being solved.

4 Conclusions and future issues

The examples in this paper have demonstrated that combining evolutionary search methods with bootstrap cross-validation is an effective tool in developing parsimonious network approximations that generalize well. Experiments showed that for certain sample sizes, methods compared well with more traditional procedures designed to penalize overly complex network structures. It was also shown that these methods have power to find robust networks subject to objectives which differ from traditional MSE and likelihood based objectives.

Several questions remain about the bootstrap/evolutionary dynamics. The most obvious is that no system trying to use training data for model selection can ever be completely free of overfitting problems. The data is "tainted" after the first generation and there might be a tendency to overfit as generations proceed. This would be a somewhat subtle type of overfitting since it would imply that networks exist which both are overfitting the sample and also all possible randomly drawn subsamples. This doesn't appear to have been a problem in the experiments performed here, but it is an issue that anyone using this method should be cautious about.

A second question concerns the actual dynamics of the bootstrap and the accuracy of the estimated bias terms. Since bias is estimated as evolution procedes it means that evolution is occcuring on progressively

better estimates, of random fitness values. If evolution moved too fast, the system would probably evolve over noise, never getting a very good bootstrap sample to estimate the true prediction error. On the other hand, slower evolution would yield good bootstrap estimates, but would slow down the search through network architectures. The appropriate balance may be a tricky issue. The crucial parameter controlling this is the number of potential new entries into the population.

A more practical question which should be considered as well is whether the bootstrap is worth the extra computer time required. It is important to keep in mind that each bootstrap iteration requires a run of the hill climbing algorithm. It seems unlikely that this can be improved upon. Therefore, there may be many problems for which the computational burden is too large, even for our ever increasing computer resources.

Finally, the reader must be cautioned that these examples were taken from one particular map with a specific noise structure. While every effort is made to make these tests as clean and precise as possible, they should be taken as a motivation for this procedure. Attempts to demonstrate its broader usefulness are currently underway.

Subject to these caveats the combination of evolution and bootstrapping appears to be a useful one, and further explorations on simulated and actual data should tell more about its effectiveness.

References

Akaike, H. 1973, Information theory and the extension of the maximum likelihood principle, *in* B. N. Petrov & F. Csaki, eds, '2nd International Symposium on Information Theory', Akailseoniai-Kiudo, Budapest, pp. 267–281.

Arifovic, J. 1994, 'Genetic algorithm learning and the cobweb model', *Journal of Economic Dynamics and Control* **18**, 3–28.

Bäck, T. 1996, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, Oxford, UK.

Balakrishnan, K. & Honavar, V. 1995, Evolutionary design of neural architectures: A preliminary taxonomy and guide to the literature, Technical report, Dept. of Computer Science, Iowa State University, Ames, Iowa.

Breiman, L. 1994, Bagging predictions, Technical report, University of California - Berkeley, Berkely, CA.

Connor, J. T. 1993, Bootstrap methods in neural network time series prediction, *in* J. Alspector, R. Goodman & T. X. Brown, eds, 'International Workshop on applications of neural networks to telecommunications', Erlbaum, Hillsdale, NJ, pp. 125–131.

Cumby, R. E. & Modest, D. M. 1987, 'Testing for market timing ability: A framework for forecast evaluation', *Journal of Financial Economics* **19**, 169–189.

Cun, Y. L., Denker, J. S. & Solla, S. A. 1990, Optimal brain damage, *in* D. S. Touretzky, ed., 'Advances in Neural Information Processing Systems', Morgan Kaufman, San Mateo, CA.

Efron, B. 1979, 'Bootstrap methods: Another look at the jackknife', *The Annals of Statistics* **7**, 1–26.

Efron, B. 1983, 'Estimating the error rate of a prection rule: improvement on cross validation', *Journal of the American Statistical Association* **78**, 316–331.

Efron, B. & Tibshirani, R. 1993, *An Introduction to the Bootstrap*, Chapman and Hall, New York.

Eubank, S. & Casdagli, M. 1991, *Proceedings of the 1990 NATO Workshop on Nonlinear Modeling and Forecasting*, Addison-Wesley, Redwood City, CA.

Finnoff, W., Hergert, F. & Zimmermann, H. 1993, 'Improving model selection by nonconvergent methods', *Neural Networks* **6**(6).

Gençay, R. 1994, 'Nonlinear prediction of noisy time series with feedforward networks', *Physics Letters A* **187**, 397–403.

Hart, W. E. & Belew, R. K. 1996, Optimization with genetic algorithm hybrids that use local searches, *in* R. K. Belew & M. Mitchell, eds, 'Adaptive Individuals in evolving populations: models and algorithms', Addison Wesley, Reading, MA.

Hassibi, B. & Storck, D. G. 1993, Second order derivatives for network pruning: Optimal Brain Surgeon, *in* S. J. Hanson, J. D. Cowan & C. L. Giles, eds, 'Advances in Neural Information Processing Systems', Vol. 5, Morgan Kaufman, San Mateo, CA.

Hogg, R. V. & Craig, A. T. 1978, *Introduction to Mathematical Statistics*, Macmillan, New York, NY.

Judge, G. G., Griffiths, W. E., Hill, R. C. & Lee, T.-C. 1980, *The Theory and Practice of Econometrics*, Wiley, New York, NY.

Kuan, C. M. & White, H. 1994, 'Artificial neural networks: An econometric perspective', *Econometric Reviews* **13**, 1–91.

LeBaron, B. & Weigend, A. 1994, Bootstrap evaluation of the effect of data splitting on financial time series, Technical report, The University of Wisconsin - Madison, Madison, Wisconsin.

Mitchell, M. 1996, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA.

Paaß, G. 1993, Assessing and improving neural network predictions by the bootstrap algorithm, *in* S. J. Hanson, J. D. Cowan & C. L. Giles, eds, 'Advances in Neural Information Processing Systems', Vol. 5, Morgan Kaufmann, San Mateo, CA, pp. 196–203.

Press, W. H., Flannery, B. P., Teukolsky, S. A. & Vetterling, W. T. 1986, *Numerical Recipes: The art of scientific computing*, Cambridge University Press, Cambridge, UK.

Schwarz, G. 1978, 'Estimating the dimension of a model', *Annals of Statistics* **6**, 461–464. %K %X %F.

Shao, J. 1993, 'Linear model selection by cross-validation', *Journal of the American Statistical Association* **88**, 486–494.

Takens, F. 1983, Distinguishing deterministic and random systems, *in* G. Bornblatt, G. Looss & D. Joseph, eds, 'Nonlinear Dynamics and Turbulence', Pitman Advanced Publishing Program, Boston, MA.

Tibshirani, R. 1994, A comparison of some error estimates for neural network models, Technical report, Dept. of Preventive Medicine and Biostatistics, University of Toronto, Toronto, Ontario.

Tibshirani, R. & Knight, K. 1995, Model search and inference by bootstrap "bumping", Technical report, Dept of Statistics, University of Toronto, Toronto, Ontario.

Tong, H. 1990, *Non-linear Time Series: A Dynamical Systems Approach*, Oxford University Press, Oxford, UK.

Weigend, A. S. & Gershenfeld, N. A. 1993, The future of time series: learning and understanding, *in* A. S. Weigend & N. A. Gershenfeld, eds, 'Time Series Prediction: Forecasting the Future and Understanding the Past', Addison-Wesley, Reading, MA, pp. 1–71.

Weigend, A. S., Zimmermann, H. & Neuneier, R. 1996, Clearning, *in* A.-P. N. Refenes, Y. Abu-Mostafa, J. Moody & A. Weigend, eds, 'Neural Networks in Financial Engineering (Proceedings of NNCM'95, London)', World Scientific, pp. 511–522.

Yao, X. 1996, 'A review of evolutionary artificial neural networks', *International Journal of Intelligent Systems* **8**, 539–567.