An Evolutionary Bootstrap Method for Selecting Dynamic Trading Strategies

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

March 1998

Abstract

This paper combines techniques drawn from the literature on evolutionary optimization algorithms along with bootstrap based statistical tests. Bootstrapping and cross validation are 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 space of potential network architectures. The combination of these two methods creates a network estimation and selection procedure which aims to find parsimonious network structures which generalize well. Examples are given from financial data showing how this compares to more traditional model selection methods. The bootstrap methodology also allows more general objective functions than usual least squares since it can estimate the in sample bias for any function. Some of these will be compared with traditional least squares based estimates in dynamic trading settings with foreign exchange series.

1 Introduction

Two recent techniques are gaining respect in nonlinear modeling. First, evolutionary methods have proved to a be a useful tool in network construction and estimation.[1] Second, Efron's bootstrap has been shown to be a useful tool for network diagnostics, and forecast combination.[2] This paper combines the two into a new model selection/estimation procedure. The bootstrap allows the estimation of off training set objectives for a very wide range of functions, while a population based search facilitates a search over the huge space of potential network structures. These procedures are applied to foreign exchange data with the objective of finding useful dynamic trading strategies through the combination of simple trend following strategies along with interest rate information.[3] The basic philosophy here is to take simple tools from each area and combine them into a more automatic network construction and selection system.

The next section describes in detail the objectives used, the bootstrap procedures, and the evolutionary network selection. Section 3 describes the results of some initial tests on foreign exchange data, and section 4 concludes, cautions the reader on how to interpret the results, and gives suggestions for future research.

2 Model Selection and Estimation

2.1 Ojective Functions

Several new techniques for estimating and evaluating trading models will be used in this paper. No one of these is particularly novel, but the combination provides interesting improvements in the performance of trading strategies.

The first major change comes in using an objective function inspired more by economic objectives than statistical goals.[4] A simple trading objective is used where it is assumed that the desire is to generate a signal, s_t that will be used to take a position at time t in a zero cost strategy. The payoff to such a

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

[2]For examples of bootstrap applications to neural networks see (Connor 1993), (LeBaron & Weigend 1998), (Paaß 1993), and (Tibshirani 1994).

[3]Early results show them to be an effective tool in simulated time series forecasts using Henon data (LeBaron 1997).

[4]The recent papers by (Bengio 1997), (Choey & Weigend 1997), and (Moody & Wu 1997) are clearly inspirational here. Other recent examples looking at the importance of other loss functions can be found in (Granger & Pesaran 1996).

strategy at time $t+1$ would be

$$s_t r_{t+1}, \tag{1}$$

where r_{t+1} is the payoff to the zero cost strategy. In general s_t will be constrained to be 1 or -1. However, it could in principle take on any value indicating a larger or smaller position. Taking the mean of this over a sample gives the obvious sample objective,

$$\frac{1}{T-1}\sum_{t=1}^{T-1} s_t r_{t+1}. \tag{2}$$

More traditional metrics will also be used such as a squared error loss function. In this case the objective is to forecast the value of the future return. It is estimated by,

$$\frac{1}{T-1}\sum_{t=1}^{T-1}(o_t - r_{t+1})^2, \tag{3}$$

where o_t is the output of a forecasting network at time t.

2.2 Bootstrap cross-validation

Model selection is obtained through the use of both bootstrap and cross-validation techniques. Candidate networks are evaluated according to their in sample objective functions adjusted by an estimate of the in sample bias.

In the case of the bootstrap a method known as the 632 bootstrap is used.[5] The basic idea is to 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 test set. This follows the traditional bootstrap estimators in spirit, but it turns out to have some problems. It is clear that this estimator will be biased since there is a large overlap between the training and test sets since they were drawn from the same finite population. The 632 bootstrap is one possible adjustment to account for this bias. This is a quick overview of this type of bootstrapping. Squared error is used in the following examples, but other measures could be used as well.

Define the squared forecast error for an approximating function, f, and estimated parameters, $\hat{\beta}$, as,

$$\hat{e}^2 = 1/n\sum_{i=1}^{n}(y_i - f(x_i,\hat{\beta}))^2, \tag{4}$$

[5]The idea of the bootstrap originated in (Efron 1979). Details on this can be found in (Efron 1983) and (Efron & Tibshirani 1993).

where $\hat{\beta}$ is estimated on the entire sample of length, n. The 0.632 bootstrap estimates the bias adjustment to this in sample error as

$$\hat{\omega}^{(0.632)} = 0.632(\hat{\epsilon}^{(0)} - \hat{e}^2). \tag{5}$$

$\hat{\epsilon}^{(0)}$ is the estimated squared error for points not in the training set. This is defined for B bootstrap replications as,

$$\hat{\epsilon}^{(0)} = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{\#(i \notin K)} \sum_{i \notin K} (y_i - f(x_i, \beta^{(K)}))^2, \tag{6}$$

where K represents the set of points in each bootstrap draw, which are drawn with replacement. β is estimated on each draw, and the error is estimated over points not in K. The adjusted error estimate is

$$\hat{e}^{(0.632)} = \hat{e}^2 + \hat{\omega}^{(0.632)} \tag{7}$$

$$\hat{e}^{(0.632)} = 0.368\hat{e}^2 + 0.632\hat{\epsilon}^{(0)} \tag{8}$$

This gives an estimate that is a weighted average of the in sample error, and the "out of sample" error from $\hat{\epsilon}^{(0)}$. The weighting, 0.632, is derived from the probability that a given point is actually in a given bootstrap draw,

$$\alpha = 0.632 = 1 - (1 - (1/n))^n \approx 1 - e^{-1}. \tag{9}$$

We will also consider drawing less than the entire sample at times.[6] This gives a different weighting factor for equation 9. If the sample is drawn of length $m < n$ then the weighting factor becomes,

$$\alpha = 1 - (1 - (1/n))^m, \tag{10}$$

$$\hat{e}^{(0.632)} = (1 - \alpha)\hat{e}^2 + \alpha\hat{\epsilon}^{(0)} \tag{11}$$

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

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

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

[6]This idea of using less than a full sample bootstrap has been introduced in a slightly different context in (Hall 1990).

2.3 Network Evolution

Approximating functions $f()$ will be standard feedforward, single hidden layer, neural networks with hyperbolic tangent activation functions. Hidden units values are given by,

$$h_{t,j} = \tanh(\beta^x_{0,j} + \sum_{i=1}^{N_i} \beta^x_{i,j} x_{t,i}), \tag{12}$$

where $\beta^x_{0,j}$ is a bias or constant term, and $x_{t,i}$ are the inputs. For standard forecasts, the weighted sum of the hidden units is used as the output,

$$o_t = \sum_{j=1}^{N_h} \beta^h_j h_{t,j} \tag{13}$$

where β_j are the output weights on the hidden units.[7]

In the case of the dynamic strategy objectives the output is fed through a final tanh unit to constrain the output signal to $[-1, 1]$. Using the notation of section 2.1 gives,

$$s_t = \tanh(o_t). \tag{14}$$

This gives a smooth trading rule function. In final evaluation this will be mapped into a "hard" rule where s_t takes on the sign of o_t as its value. This objective, being discontinuous, causes problems for the hill climbing based estimation procedure, so the tanh function is used as a stand in. Surprisingly, the tanh results are very close to the "hard" rule, and they are not reported.

Network weights are estimated by hill climbing. However, the network architecture is determined through an evolutionary procedure. There is a large and growing literature on evolving neural networks.[8] This paper uses a very simple evolutionary structure. Obviously, improvements may be obtained using more sophisticated methods in the future.

A population of 50 network architectures is given by $(s_j, w^0_j; w_j)$ where j is the population index. 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^0_j 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. A picture of this flat binary representation and the corresponding network is given in figure 1.

[7]No aggregate bias term is used. The maximum number of active hidden units is set to 8, which turned out not to be a binding constraint.

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

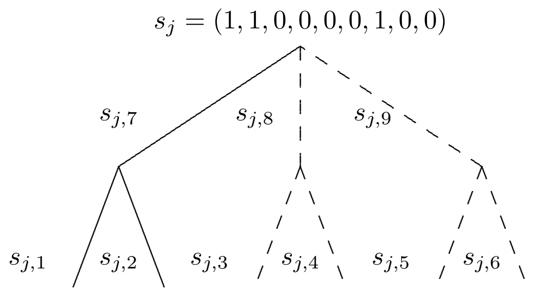

Figure 1: Single hidden unit, 2 inputs

New networks are evolved using tournament selection and two mutation operators, micro and macro mutation. A parent network is selected by choosing 10 networks at random, and then using the fittest of these in the mutation stage that follows. Micro mutation involves looking at the connections between the hidden units and inputs. The mutation operator sweeps through all input lines and with probability 0.1 it flips their bits from 0 to 1. Macro mutation involves choosing one hidden unit at random. 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, 0.9. Both types of mutations occur exclusively with probability 1/3. The remaining 1/3 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 on the full training sample.[9]

[9]Optimization uses the BFGS algorithm described in (Press, Flannery, Teukolsky & Vetterling 1986) with numeric derivatives. For the case of squared error loss it would probably be more efficient to use the Levenberg-Marquardt algorithm, but this would make comparisons difficult, so the same hill climbing algorithm will be used in all cases. All the algorithms have been hand coded in Matlab to maintain control, but they have been compared with the Matlab optimization toolbox routines for testing. Also, starting values use a technique suggested in (Kuan & White 1994) where the initial weights on the hidden unit outputs are set using an OLS regression on the target value. This accepts that the initial weights guide the general direction of the sensitivity of the hidden units to different patterns, and the output

Ten new networks are chosen in this fashion. 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. This is similar to $\mu + \lambda$ selection described in (Bäck 1996), and the election operator of (Arifovic 1994). In other words, after the child fitness values are estimated from the bootstraps, the best 50 out of the current pool of 60 networks will become the new population. This selection method has the property of slowing down evolution as the population improves which allows it to concentrate on getting good bootstrap bias estimates for the current set of networks. New networks are started at randomly drawn initial weights.

Bias estimates proceed according to the bootstrap or cross validation methods described earlier. However, it is important to be clear just what the timing is. Bootstrapping is a computationally intensive algorithm, and it would be infeasible to perform many bootstraps on many population members. Therefore, a single bootstrap iteration is performed on the entire population at each evolutionary time step. 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 is drawn, and β is estimated through hill climbing using randomly drawn starting values.[10]

A serious problem for 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.[11] 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 constraints, 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

should be quickly moved to reflect this. Many experiments have indicated that the only real impact of this is a very significant speed improvement.

[10]It is important to remember that the candidate network weights from the full training set are used in out of sample forecasting. The estimated weights from the bootstrap draws and corresponding estimates of in sample bias are used in model selection fitness only.

[11]A small experiment with the networks and data used in this paper showed that this variability could be significant. Objective functions varying by as much as 150 percent about their mean for different starting values in the weight vector were common. Also, for some starting values the objective changed sign.

often stop at a local minimum on the training or cross validation sample.[12]

All networks are initialized with 70 percent of their connections activated. This amounts to roughly 50 percent of the connections on since many of these bits may be connected to deactivated hidden units.[13] The starting weights for the inputs are all set to uniform values on $[-0.05, 0.05]$. This is used for all starting value initializations. The results have been generally robust to changes in this distribution. Finally, evolution is run for 30 generations. New networks are not allowed during the first and last 5 generation to get better bootstrap estimates during these periods.

3 Foreign exchange example

3.1 Data Preliminaries

This section gives a quick example of this procedure applied to foreign exchange data. See (LeBaron 1997) for more extensive applications to simulated test time series.

The series used is the British pound from January 1979 to October 25, 1993, daily with 3500 points.[14] Forecasting is done in a rolling manner in order to capture potential nonstationarities in the series. The first 1000 points will be used to estimate and evaluate networks which will be used on the next 500 out of sample data points. After these forecasts are completed the blocks will be rolled forward by 500, and the procedure is repeated. This is done five times to give a total forecast time series length of 2500.

The objective is to forecast the one day continuously compounded return corrected for the interest differential. P_t is the foreign exchange price in dollars per pound.

$$r_{t+1} = \log P_{t+1} - \log P_t - (r_t^{US} - r_t^{UK}) \tag{15}$$

Given uncovered interest parity this should be unforecastable given time t information. The strategies estimated will report

$$\frac{1}{T-1} \sum_{t=1}^{T-1} s_t r_{t+1}, \tag{16}$$

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

[13]Everytime a new network structure is set up it must be run through a specification check to make sure that it is a valid network. This eliminates hidden unit inputs for hidden units that are not active, and makes sure that active hidden units have at least one input.

[14]The sample was chosen to enable the rolling sample lengths to come out even.

an estimate of the dynamic trading strategies reported earlier. This would be a strategy of borrowing in one currency, and then investing in the other one. Given covered parity conditions hold this is also the payoff to a strategy of forward speculation where a forward position long or short is taken at t and then unwound at time $t+1$.

The inputs for forecasting are a sequence of price moving average ratios. Inputs are constructed from a commonly used set of predictor variables. These are listed in table 1.[15] Ideally, this table would be much bigger to test out the

Table 1: Information Variables

1.	Two price/moving average ratios. (150 and 5 day)
2.	One lag of the interest adjusted return series defined in equation 15.
3.	A local average of squared returns (10 days) as a volatility estimate.
4.	The interest differential at time t, $r_t^{US} - r_t^{UK}$.
5.	The interest rate differential less its 30 week moving average.

evolutionary system's ability to weed out irrelevant predictors, but this hasn't been done. These have not been individually tested for relevance, so several of these may be useless.

There is some preprocessing performed before networks are estimated. All preprocessing uses only information in the training set. All the inputs are divided by the training set standard deviation for each value. The volatility and interest rate series are demeaned using the training set. Finally, the output target, or future return is demeaned in the training set alone using the mean from the training set. This is done to keep the trading rule objective from trying to just "fit the mean" of the sample. If the sample mean was positive then the trading rule might find it optimal to go long all the time. Setting this mean to zero makes sure that it goes after a dynamic switching strategy.

The data are then formed into pairs, (r_{t+1}, x_t), where x_t is the vector of prediction variables. It is this set of pairs that will serve as the pool for drawing new values during bootstrapping. The x_t are mapped into s_t using the fitted networks, and a hyperbolic tangent in the estimation stage. This is converted to a hard $[-1, 1]$ function for the actual reported results.[16]

[15]The technical predictors have been used in many papers including, (Dooley & Shafer 1983), (Sweeney 1986), (LeBaron 1998), (Levich & Thomas 1993), and (Taylor 1992). Volatility based correlation forecasts were introduced in (Bilson 1990) and (LeBaron 1992). The inclusion of the interest differential reflects the large literature on the forward bias in foreign exchange markets, (Engel 1996) and (Lewis 1995). The interest rate differential moving average is inspired by recent work in (Eichenbaum & Evans 1995) that indicates that the impact of large interest rate differentials may drop off over time.

[16]Fitted tanh objectives turned out to be very close to the hard trading rule objective,

3.2 Results

Table 2 presents the out of sample forecast and trading results. These are the values of the dynamic trading strategies evaluated out of sample on each of the 500 test periods, and averaged over all 5 of these. These test periods are used for running the strategies, and play no roll in model selection or estimation. Several numbers are reported giving estimates of the usefulness of the dynamic strategy, and its significance.

The first row reports the bootstrap results using a half sample selection rule. 500 points are drawn from the 1000 length training sample with replacement, and this is used as a bootstrap draw in the bias adjustment procedure.[17] In the first row of the table, the mean percentage return is given as 0.067 percent with a standard error of 0.015 which is clearly significantly different from zero.[18] Zero is what would be expected in the case where uncovered parity held, or foreign exchange markets were unpredictable. Also, reported is the fraction of correct sign predictions which is also significantly different from 0.5, but not too far from it in magnitude at 0.54. Finally, the annual Sharpe ratio is given. This is the mean return divided by the standard deviation of the return. It is not a dimensionless number, but under independent returns it scales as $\sqrt{n}$ or the length of the period. It is reported for the annual horizon here which is common. The reported value of 1.39 is very large relative to other strategies and portfolios available to investors.[19] Also, reported are the 95% confidence bounds for the bootstrapped Sharpe ratio distribution. These indicate that its large value is quite significant, but also reminds us how imprecise the estimated Sharpe ratios are.

Moving to the next row of the table, labeled Bootstrap 1.0, we see the results for the traditional 632 bootstrap where the full sample length, 1000, is used as the number for the training set. While the numbers here are still significantly large, they appear to be smaller than the ones for the 0.5 sample sizes.

$[-1, 1]$ in the results they produced. This indicates that the tanh function output is very close to -1 or 1 in most cases. This finding is relevant to the problem of whether a an objective that looked at other criteria, such as the Sharpe ratio, would have changed the results. If the tanh function is giving -1 or $+1$ as output, and if there is no forecastibility in the conditional variance, then maximizing expected returns will maximize the Sharpe ratio, since the denominator of the Sharpe ratio is minimized by maximizing the expected return. To see this remember that the standard deviation is $\sqrt{E(s_t r_{t+1}^2) - E(s_t r_{t+1})^2}$.See (Skouras 1997) for more information on this.

[17]The weightings are adjusted as in equation 10.

[18]Standard errors are estimated using a bootstrap procedure where the dynamic returns themselves are redrawn with replacement. The block bootstrap was also used to simulate dependence in the series, but the results were the same, so they are not reported.

[19]A well diversified aggregate stock portfolio has a Sharpe ratio of between 0.3 and 0.4 depending on the securities and time period.

Table 2: Comparisons

Description	Mean Payoff (percent)	Sign Probablity	Sharpe	Sharpe (0.025)	Sharpe (0.975)
Bootstrap 0.5	0.067 (0.015)	0.540 (0.009)	1.39 (0.31)	0.80	2.02
Bootstrap 1.0	0.042 (0.016)	0.526 (0.009)	0.86 (0.32)	0.20	1.47
Monte-carlo Cross Validation	0.047 (0.015)	0.529 (0.009)	0.98 (0.32)	0.37	1.62
No Adjust	0.044 (0.014)	0.527 (0.009)	0.91 (0.30)	0.28	1.45
No adjust Full Connection	0.021 (0.015)	0.512 (0.010)	0.43 (0.32)	-0.19	1.04
MSE	0.027 (0.015)	0.511 (0.009)	0.55 (0.31)	-0.10	1.15
MSE BIC	0.020 (0.015)	0.513 (0.009)	0.41 (0.31)	-0.19	1.02
MA(150)	0.030 (0.015)	0.527 (0.010)	0.61 (0.32)	-0.02	1.20

Dynamic trading strategy results. All numbers are estimated using the 5, 500 length testing periods. These were not used for model estimation or selection. Mean payoff is the mean payout on the dynamic strategy over the out sample period, see equation 16. Sign Probability is the probability of a correct sign prediction, and Sharpe is the annual Sharpe ratio. Numbers in parenthesis are standard errors from a 1000 length bootstrap. Sharpe (0.025) and (0.975) are the respective quantiles of the bootstrap distribution.

The next few rows present several other methods for comparison. The first is monte-carlo cross validation. For this a training set is drawn without replacement from the sample, and the remaining data is used for testing in network selection. The numbers are reported using a 50/50 training/testing split. They are generally not very different from the bootstrap using the full sample.

The next row, labeled "no adjust", presents a very crucial comparison. In these runs the bootstrap adjustment is turned off completely, and the system evolves toward the network which fits best in the entire training set. This should be very sensitive to over fitting problems. Interestingly, it doesn't do all that badly. It is comparable to the monte-carlo cross validation and the full sample bootstrap. Also, it should be noted that the mean return is not significantly different from the 0.5 bootstrap either.[20] The row labeled, "Full Connection", evaluates the usefulness of lean networks. Here a fully connected network is used. The best network over 1000 different starting values is chosen for out of sample forecasting, with no bias adjustment on the in sample fitness. The performance of the fully connected nets drops dramatically to 0.0208 which is significantly different from the bootstrap 0.5 forecasts.

The next two rows report numbers using traditional MSE estimates. These networks are then used as trading rules by taking the sign of the forecast. In the first case no adjustment is made in fitting. This gives a much smaller mean return of 0.0267 which is significanly different from the bootstrap case with a t-test value of 1.89 which is significant at the 5 percent level using a one tailed test.[21] The Sharpe ratio of 0.55 is less than half that for the first bootstrap. The next row adds a standard BIC (Schwarz 1978) penalty to the network fitness value. This penalizes the network for extra parameters. It is clear that this is over penalizing since the estimated networks do not even give a significantly positive return or Sharpe ratio. With a link percentage of only 0.03 these networks cannot be doing very much.

The final row presents a comparison with a simple dynamic trading strategy using a 150 day moving average. For this strategy the signal s_t is set to 1 if the price is above a 150 day moving average and -1 otherwise. This simple strategy is one of the inputs to the neural network, but it is often used on its own. The eyeball test again sees a big difference. The Sharpe ratio for this strategy is only 0.61 which compares to 1.38 for the bootstrap. The means are again significantly different with a t-value of 1.79.[22]

The dynamic trading strategies cannot really be evaluated as to whether they should be used until some account is taken of transaction costs. Here, a relatively conservative estimate of 0.05 percent for a one way trade will be used. This actually means that each time a strategy changes sign it imposes a 0.1 percent cost since changing positions involves both unwinding the foreign position, and setting up a new domestic position, or the reverse.

Adjusted returns and Sharpe ratios are given in table 3. It is clear that for most of the rules there is an impact from transaction costs. For the bootstrap they fall from 0.0667 to 0.0513, but the Sharpe ratios that are commanded by

[20] A rough judge of significance here would be a t-test on the means. Given their estimated standard errors the standard error on the difference would be $\sqrt{(s_1^2 + s_2^2)}$ which is 0.02 for these showing that the difference between any of the means reported so far is not significantly different from zero.

[21] Again caution should be used in interpreting this test since the difference is being measured off the best performing rule.

[22] These tests would be an interesting application for the Reality Check of (White 1997). Without a framework like that it is still difficult to interperet the difference between values that may have been preselected.

Table 3: Transaction Cost Adjustment

Description	Mean Payoff (percent)	Trade Probability	Sharpe
Bootstrap 0.5	0.0513	0.1536	1.0682
MCV 0.5	0.0327	0.1448	0.6787
No adjust	0.0247	0.1904	0.5127
MA(150)	0.0263	0.0328	0.5462

Dynamic strategy figures adjusted for transaction costs. MCV stands for monte-carlo cross-validation. Trade probability is the estimated probability of the strategy trading.

the best strategies are still in a range that would be thought of as interesting. This table also reports the probability of trade. It is clear that the network strategies trade more frequently then the simple moving average trading rule which only trades about 3 percent of the time.[23]

Table 4: Subsamples

Period	Training	Test	Test	Test
	Boot 0.5	Boot 0.5	Boot 1.0	No adjust
831003-851008	0.0629	0.0947	0.1025	0.0330
851009-871009	0.0948	0.0911	0.0190	0.0429
871012-891004	0.0914	0.0447	0.0132	0.0348
891005-911007	0.0428	0.0376	0.0295	0.0318
911008-931025	0.0546	0.0654	0.0439	0.0762

Table 4 presents some results from subsamples to show how the network fitting and trading rule profitability changes over subsamples. The values are taken from the 0.5 bootstrap. The range is pretty big with some periods showing dramatic one day returns of nearly 0.1 percent, but other periods drop off to only 0.037 percent. This is reflective of the large noise in financial data, and in the trading rule objective. The table also gives a comparison across several of the methods. Interestingly, the 0.5 bootstrap performs the best in 3 of the 5 subsamples, and second best in the other two. This is supportive of constistently good performance of this method, and also that it might have been able to have been selected using past data.

Table 5 displays the time variation in the fitted networks when applied to further distant future subsamples. Each entry shows the initial objective from

[23]It should be noted that the inclusion of transaction costs eliminates the significant difference between the bootstrap and MA(150) rules. This is due to the fact that the first strategy incurs larger costs from more frequent trading.

the 500 days following the training period, the test set, along with the objective evaluated further out in the future. This gives a view of how time varying the predictability is. Most of the network trading objectives do decay as the horizon is moved farther out in the future.

Table 5: Future Subsamples

Period	831003-851008	851009-871009	871012-891004	891005-911007	911008-931025
831003-851008	0.0947	0.0889	0.0034	0.0091	0.0209
851009-871009		0.0911	0.0158	0.0206	0.0570
871012-891004			0.0447	0.0312	0.0571
891005-911007				0.0376	0.0070

Subsample variation is also displayed in figure 2. This picture shows the evolution in network structure for the bootstrap and no adjustment cases. Link fraction refers to the number of connections that are active divided by the total number possible, or the overall connectedness of the networks. It is clear in the bootstrap case the networks are getting leaner as they evolve. This is not necessarily the case for no adjustments. Some networks are getting leaner, and others are expanding. It is interesting that any network would be eliminating connections in the no adjustment case. However, it is possible that the difficulty of estimating this objective function may get larger as nodes are added making it more unlikely that they will hit a set of starting values that leads to good networks. This puts some pressure towards leaner networks even without an explicit bias adjustment.

4 Conclusions and Future Work

These results show promise for some of the methods introduced here. As mentioned in the introduction the combination of several things, trading objectives, bootstrap bias adjustment, and evolutionary network selection, appears to be helpful in forecasting experiments. The results should still be viewed with some caution as this was one problem and one time period of observation. Also, while the method clearly dominates some simpler forecasting methods, and random walk guessing, statistically significant differences across several methods was not seen.

In terms of methodology this work is pushing closer to handing a larger part of the network selection, and parameter choice problems over to a mechanized system. The objective is obviously to take more of the guess work out of

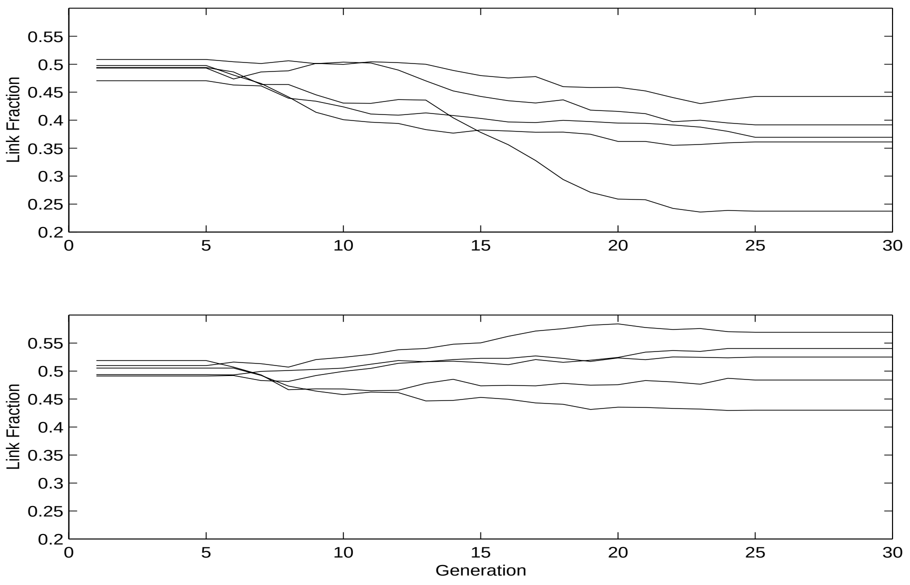

Figure 2: Linkage Fraction: Bootstrap (top) and No adjustment (bottom), 30 Generations each run

nonlinear model selection and fitting. There is still a long ways to go, and this is far from a fully automatic black box forecasting system. The system analyzed here brings along some new problems of its own. First, it introduces many new parameters for which theory gives little guidance on how they should be set. Second, the bootstrap procedure is very computationally burdensome. On current levels of machinery it remains a constraint on what can be done. The use of relatively small simple networks, small populations, and input lists was a constraint imposed by computational size.

Since the philosophy was to take the simplest components from each area the possibilities for improvement are quite large. As mentioned earlier the evolutionary structure for the networks is crude. Crossover like operations are not used, and the actual data structure of the nets as a flat bitstring could certainly be changed. A network structure that was better suited to evolution would be useful. Also, the value added of bootstrapping and crossvalidation is still not entirely clear here. This is the area of greatest uncertainty in terms of structure and parameter choice. A good question that remains is why did the half sample bootstrap do so much better than the full sample approach advocated in the bootstrapping literature. This remains a serious mystery for future research, and bigger computers. It is always possible that financial series may just be too noisy to give a firm answer here. Finally, direct application of greater computing horsepower is also an obvious area for improvement. It would be easy to simply extend population sizes and generation lengths. Also, a very delicate issue that is not directly addressed is the rate of evolution versus the bootstrap. Since evolution is occurring over an essentially noisy fitness value it is possible that the system can be evolving over noise in some cases. This can be lessened by increasing the amount of bootstrapping relative to evolution, by reducing the number of new rules coming in, or running several bootstrap iterations for every population step. Another method for increasing the precision of the bootstrap at each step is to reduce the in sample draw on the bootstrap. This method appeared successful here in that the best performing bootstrap used a 50 percent sample draw.

Another methodological question that this paper asks concerns the value of population based search. Several arguments for evolutionary search have been offered including, combinations of building blocks and endogenous annealing schedules. This paper offers a more direct reason for using population based search. In the case where the computational time for estimating an objective is very costly, and noisy objectives can be slowly improved through extra function evaluations, the use of a population based search may helpful. It allows the search to concentrate scarce computing resources on the most likely useful candidate solutions. It would be impossible to cover the entire sample computationally, but the population defines a subset where the searcher should work hard to get better fitness values from the bootstrap iterations. This is a much more direct and practical benefit to population based search than the previously mentioned benefits. However, it does not negate their possible contribution in the search process. A second aspect of populations that is not used here, is that the final forecasting rule could be a combination of several of the best rules, or the trading signal could be a vote from several nets. These methods have been tried by many others in different contexts, and they might prove useful here. One complication, is that it is a little difficult to judge the bootstrap bias estimates if the actual forecast is going to be a function of several networks.

This paper has shown the possibility for using a new evolutionary bootstrap process in selecting dynamic trading strategies. It should be reminded that this process remains somewhat preliminary, and further complete out of sample periods need to be checked before this should be taken live as a trading strategy. Also, the troubling lack of significant differences between several of the methods should also suggest proceeding with some caution as well. However, it suggests that there many be value added to this procedure and further explorations in this directions may prove fruitful.

5 Acknowledgments

The author is would like to thank Andreas Weigend, and other participants at CF97 for useful comments. Also, the Mathworks provided some of the software used in this study. Much of this research was completed while the author was visiting the Center for Biological and Computational Learning (CBCL) at MIT, and the author is grateful for the many comments and suggestions from the CBCL community. Finally, the author thanks the Sloan Foundation, and the Wisconsin Alumni Research Foundation for financial support.

References

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.

Bengio, Y. (1997), Training a neural network with a financial criterion rather than a prediction criterion, *in* 'Decision Technologies for Financial Engineering: Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets', World Scientific, Singapore, pp. 36–48.

Bilson, J. F. (1990), "Technical" currency trading, *in* L. R. Thomas, ed., 'The Currency-Hedging Debate', IFR Publishing Ltd., London.

Choey, M. & Weigend, A. S. (1997), Nonlinear trading models through sharpe ratio maximization, *in* A. S. Weigend, Y. S. Abu-Mostafa & A. P. N. Refenes, eds, 'Decision Technologies for Financial Engineering: Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets', World Scientific, Singapore, pp. 3–22.

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.

Dooley, M. P. & Shafer, J. (1983), Analysis of short-run exchange rate behavior: March 1973 to november 1981, *in* D. Bigman & T. Taya, eds, ‘Exchange Rate and Trade Instability: Causes, Consequences, and Remedies’, Ballinger, Cambridge, MA.

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.

Eichenbaum, M. & Evans, C. L. (1995), ‘Some empirical evidence on the effects of shocks to monetary policy on exchange rates’, *Quarterly Journal of Economics* **110**, 975–1009.

Engel, C. (1996), ‘The forward discount anomaly and the risk premium: A survey of recent evidence’, *Journal of Empirical Finance* **3**, 123–192.

Granger, C. & Pesaran, M. H. (1996), A decision theoretic approach to forecast evaluation, Technical report, University of California, San Diego, La Jolla, CA.

Hall, P. (1990), ‘Using the bootstrap to estimate mean squared error and select smoothing parameter in nonparametric problems’, *Journal of Multivariate Analysis* **32**, 177–203.

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

LeBaron, B. (1992), ‘Forecast improvements using a volatility index’, *Journal of Applied Econometrics* **7**, S137–S150.

LeBaron, B. (1997), An evolutionary bootstrap approach to model selection and generalization, Technical report, The University of Wisconsin - Madison, Madison, Wisconsin.

LeBaron, B. (1998), Technical trading rules and regime shifts in foreign exchange, *in* E. Acar & S. Satchell, eds, ‘Advanced Trading Rules’, Butterworth-Heinemann, pp. 5–40.

LeBaron, B. & Weigend, A. S. (1998), ‘A bootstrap evaluation of the effect of data splitting on financial time series’, *IEEE Transactions on Neural Networks* **9**(1), 213 – 220.

Levich, R. M. & Thomas, L. R. (1993), 'The significance of technical trading-rule profits in the foreign exchange market: A bootstrap approach', *Journal of International Money and Finance* **12**, 451–474.

Lewis, K. (1995), Puzzles in international financial markets, *in* G. M. Grossman & K. Rogoff, eds, 'Handbook of International Economics', Vol. 3, North Holland, Amsterdam, pp. 1913–1971.

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

Moody, J. & Wu, L. (1997), Optimization of trading systems and portfolios, *in* 'Decision Technologies for Financial Engineering: Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets', World Scientific, Singapore, pp. 23–36.

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.

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

Skouras, S. (1997), An (abridged) theory of technical analysis, Technical report, European University Institute, Florence, Italy.

Sweeney, R. J. (1986), 'Beating the foreign exchange market', *Journal of Finance* **41**, 163–182.

Taylor, S. J. (1992), 'Rewards available to currency futures speculators: Compensation for risk or evidence of inefficient pricing?', *Economic Record* **68**, 105–116.

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.

White, H. (1997), A reality check for data snooping, Technical report, University of San Diego, La Jolla, CA.

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