Neural Network Ensembles for Time Series Forecasting

V. Landassuri-Moreno
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
V.Landassuri-Moreno@cs.bham.ac.uk

John A. Bullinaria
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
J.A.Bullinaria@cs.bham.ac.uk

ABSTRACT

This work provides an analysis of using the evolutionary algorithm EPNet to create ensembles of artificial neural networks to solve a range of forecasting tasks. Several previous studies have tested the EPNet algorithm in the classification field, taking the best individuals to solve the problem and creating ensembles to improve the performance. But no studies have analyzed the behavior of the algorithm in detail for time series forecasting, nor used ensembles to try to improve the predictions. Thus, the aim of this work is to compare the ensemble approach, using two linear combination methods to calculate the output, against the best individual found. Since there are several parameters to adjust, experiments are set up to optimize them and improve the performance of the algorithm. The algorithm is tested on 21 time series of different behaviors. The experimental results show that, for time series forecasting, it is possible to improve the performance by using the ensemble method rather than using the best individual. This demonstrates that the information contained in the EPNet population is better than the information carried by any one individual.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Connectionism and neural nets; I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods.

General Terms

Algorithms, Design, Experimentation.

Keywords

Evolutionary Programming, Evolutionary Neural Networks, Ensemble Neural Networks, Time Series Forecasting.

1. INTRODUCTION

There have been numerous successful applications of Evolutionary Algorithms (EAs) to evolve artificial neural networks (ANNs), simplifying the search for optimal network parameters, weights and architectures. Moreover, there have been diverse approaches demonstrating that ensembles of networks can improve the performance over any one individual network [17]. Assuming there is more valuable information in a whole evolutionary population than in a single best individual, it is reasonable to predict that the best network evolved will not have the best generalization compared to an ensemble based on the evolved population [15].

In this context, there have been several studies that test the performance of the evolutionary algorithm EPNet in the classification field [14, 15, 16], but it can be argued that the classification task is easier than prediction. First, extrapolation is more difficult than interpolation, because in prediction one has to assume that, for a given Time Series (TS), the trends or behaviors of the past will be maintained in the future [12]. Another reason is that in classification we generate only one discrete output to say if an input is from one class or not (or to say how close it is to a given class), rather than a regression value. If we only need to predict one step ahead, that will be similar to standard regression, but typically we need to predict n steps ahead, which is likely to introduce a bigger error at the end of the task, e.g. when using predictions already made as an inputs for predicting future values (see Sec. 2, Multiple-step forecasting).

For this reason, it is easier to find studies using the EPNet algorithm for classification than for prediction [13, 14]. To rectify this deficiency, this paper evaluates the performance of the EPNet algorithm on 21 different TS, taking the best evolved individual and comparing it against ensembles formed with two linear combination methods. It is likely that ensembles can achieve better results than the best individual found in the forecasting task (as happens in classification), but this needs to be confirmed empirically.

For this work, all the predictions were set to 30 steps ahead using the multiple step forecasting method (Sec. 2) and the Normalized Root Mean Squared Error (NRMSE) to measure the performance. Then the best individual prediction is compared against three main ensemble approaches: a) Average - the ensemble is formed from the population of networks in the last generation of the EPNet algorithm and then the output of the ensemble is calculated as the average of the outputs of each network (Sec. 5.1); b) Rank-Based Linear Combination (RBLC) - this uses the population of the last generation as in the previous method, but then applies the Rank-Based Linear Combination method (Sec. 5.2) to calculate the output of the ensemble; c) Ensemble of Best Individuals - this forms an ensemble from across independent

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

evolutionary runs, with either Average or RBLC ensemble output (Sec. 5.3). Also, because each generation of the evolutionary process typically contains both fit individuals and others that are not so good, we test the results for ensembles that take only the best individuals per independent run to form the ensemble, which is clearly rather different to taking all the individuals in the last generation.

To explore the generality of our findings, TS were studied belonging to several different dynamics: Chaotic, Quasi-periodic and Complex, and arising from several different fields: Physics, Economics, Financial, Hydrological and from forecasting competitions. In this sense, there is a sufficient range of behaviors and dynamics to test the EPNet algorithm in a detailed manner. That is motivated by the fact that in the previous literature, only two TS appear to have been used to test the algorithm in the forecasting task: Mackey-Glass and Logistic [13, 14]. Thus, among all the TS used, there are two main scenarios: well known TS (where there is previous domain knowledge, which means that we know some optimal parameters for them), and TS that are not so common or could be described as "new TS" for forecasting (as typically happens in real world scenarios), and consequently there is not as much information concerning them as the others. For this reason, in Sec. 3 we present our analysis of some of the parameter settings needed to perform the predictions.

It may be worth noting here that this study not only tests two representative combination methods (Average and RBLC) that are not computationally expensive for forming the ensemble outputs, but also aims to explore the performance of the EPNet algorithm for the TS forecasting task, understand the effect of different ensemble compositions, and provide a detailed analysis of the crucial parameters that need to be set appropriately.

The reminder of this paper is organized as follows: In Sec. 2 is presented the method used to perform the forecasting, and some related technical issues. Sec. 3 then gives an overview of the EPNet algorithm, and an analysis of some preliminary experiments to determinate appropriate parameter settings for performing successful predictions (Sec. 3.1 and 3.2). Sec. 4 presents our empirical prediction results for the best evolved individuals, and Sec. 5 gives the corresponding results for the various ensemble approaches. Finally, we provide our conclusions in Sec. 6.

2. TIME SERIES FORECASTING

For the Time Series (TS) forecasting/prediction task, it is common to try to use a small subset of the recent TS information to perform the prediction. This method is called lagged variables, or shift registers, or tapped delay line. If we use this approach, we say that we have an Autoregressive Model and the input space is called an *Embedding Space*. In this case, the TS is transformed into a reconstructed state space using a delay space embedding [1, 8]. This means that we are aiming to obtain accurate predictions using only a finite segment of previous values up to the point to be predicted. Thus we have:

$$x_{t+1} = F[x_t, x_{t-k}, x_{t-2k}, \ldots, x_{t-(d-1)k}] \qquad (1)$$

where d is the number of inputs and k is the time delay. There is a condition that needs to be satisfied: given an attractor of dimension D, we must have $d \geq 2D + 1$ [1]. But because we do not generally know D nor the delay, we need to calculate them, e.g. using Average Mutual Information for the time delay, and False Nearest Neighbour for the embedded dimension. In this work we try both techniques to calculate them, and perform some experimental analysis as discussed in Sec. 3.2. Those techniques were obtained from the package Visual Recurrent Analysis (VRA) [1].

Since there are various different behaviors to test, the VRA could not obtain accurate values for the inputs and delays for all TS. For this reason, in Sec. 3.2 is presented a brief analysis and discussion of some experiments performed to determine whether it is better to evolve the inputs, or leave them fixed as in the original EPNet algorithm [14].

The method used to perform the forecasting in this work is called multiple-step ahead or closed-loop forecasting. The TS X is $[x_1, x_2, \ldots, x_t]$, the number of points ahead to predict is n, the test set is $[x_{t+1}, x_{t+2}, \ldots, x_{t+n}]$, and the forecast in the same interval is $[y_{t+1}, y_{t+2}, \ldots, y_{t+n}]$. Table 1 shows a simple example in which the network has three consecutive inputs and the lapse n to predict is set to four.

Table 1: Multiple-step or closed-loop forecasting

Forecast	*Inputs*
y_{t+1}	x_t, x_{t-1}, x_{t-2}
y_{t+2}	y_{t+1}, x_t, x_{t-1}
y_{t+3}	y_{t+2}, y_{t+1}, x_t
y_{t+4}	$y_{t+3}, y_{t+2}, y_{t+1}$

3. THE ALGORITHM AND PARAMETERS

The EPNet algorithm [13, 14] is based upon the standard *Evolutionary Programming* approach, aimed at evolving ANN architectures and weights at the same time as obtaining smaller network topologies. It does not have a crossover operator, nor a genotype to represent the individuals. Instead it carries out the evolutionary process by performing only five different mutation operations directly on the phenotype: (1) hybrid training composed of training with the Modified Back Propagation (MBP) algorithm and Simulated Annealing (SA); (2) node deletion; (3) connection deletion; (4) connection addition; and (5) node addition. The algorithm performs only one such mutation on the selected individual in each generation.

The first mutation tested is always the partial training (MBP or SA), whereby the algorithm tries to reduce the error "considerably" (see Sec. 3.1). If the error can be reduced considerably, then the training is marked as successful (*successful training*) and the individual is passed to the next generation. This is the first mutation attempted because a change in the network's architecture can produce large changes in the ANN's behavior. If the error is not significantly reduced, then the other mutation operators take part in the process, in order starting from (2) node deletion, and finishing with (5) node addition. Thus the algorithm always attempts to delete nodes or connections before adding them, so it encourages the search for smaller architectures.

The training in the EPNet algorithm is only a partial training, i.e. the networks are not trained until they converge. This is motivated by computationally efficiency, which lets the evolution advance faster, with the individuals improving their fitness through the generations. For a more detailed description of the EPNet algorithm see [14].

There are some common parameters that were fixed for

the experiments throughout this study: population size 20, generations of evolution 300, initial connection density 70%, initial learning rate 0.2, minimum learning rate 0.1, epochs for learning rate adaptation 5, number of mutated hidden nodes 1, number of mutated connections 1-3, temperatures in SA 5, iterations per temperature in SA 100, 2000 epochs of training inside the EPNet, and 2000 of further training at the end of the algorithm. The only stopping criteria was the number of generations. For all the experiments, 30 independent runs were performed to ensure statistical validity of the results. All these parameters were set at convenient traditional values and are not intended to be optimal.

The size of the TS was limited to 2000 values and split into four sub-sets: the first being the "training set" that is used to perform the learning task with MBP or SA; then there is a "validation set" that is used to ensure that there is no over-fitting of the learning, then a "test set inside EPNet" to simulate a real prediction (multiple step ahead prediction) and obtain the fitness of the networks, and finally there is the "final test set", that is only applied after the whole evolutionary process has been completed, to evaluate the final individuals and the ensemble methods.

In this study we used 21 different TS from various origins: *Henon, Lorenz, QP2, QP3*, and *Rossler* from [11]; *Ikeda*, *Dow Jones* and *Logistic* from [1]; *Mackey-Glass* from [10]; Number of daily *Births in Quebec*, Daily closing price of *IBM Stock*, *SP500*, Monthly Flows *Colorado River*, Monthly *Lake Erie* Levels, Daily morning *Gold Prices*, *Equipment temperature* (degree Celsius of equipment used for radioactive measurement), Seismograph (vertical acceleration, nm/sq.sec) of the *Kobe* earthquake, Daily brightness of a variable *Star* and Monthly means of daily relative *Sunspot* numbers from [3]; Santa Fe Competition: *D1* and *Laser* from [7].

Some important preliminary experiments designed to optimize the algorithm are now discussed, though full results for them cannot be presented due to lack of space. After all these preliminary experiments were performed (Sec. 3.1 and 3.2), we could set all the required parameters and proceed for each TS to find the best individual results (Sec. 4) and explore the various ensemble approaches (Sec. 5).

3.1 Setting the successful training parameter

As mentioned above, there is a crucial parameter in the EPNet algorithm that determines what is called *successful training*. We have "success" if the training error is decreased "considerably", or "failure" if it is not. In the literature, this parameter is never discussed in detail (i.e. how much is "considerably"?), and it can easily be set with incorrect/inappropriate values. The EPNet algorithm proves to be much more robust with regard to its other parameters.

Consequently, our study began by running several experiments with different values for the *successful training* parameter: 30%, 50% and 70%. For example, for the value 30% the training is marked as a "successful" if the error is reduced by 70% or more (a strict value), and for the value 50% it is marked as a "successful" if the error is reduced by half or more (a more relaxed value). It was found that this parameter has a large impact on the performance of the algorithm, because if we use a too relaxed value (e.g. 70%, with the error only needing to be decreased by 30%) the networks enter the training stage and easily achieve a sufficient reduction of the training error (leading it to be marked as successful), and thus pass directly to the next generation,

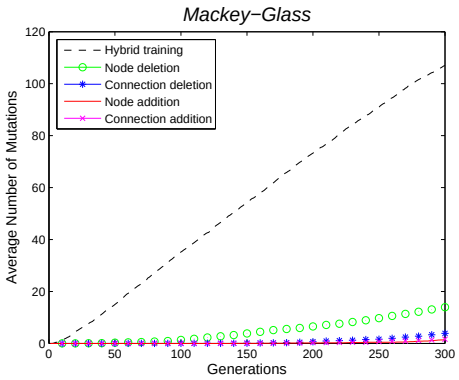

Figure 1: Average Mutations for Mackey-Glass TS. *Successful Training* parameter set to 70%

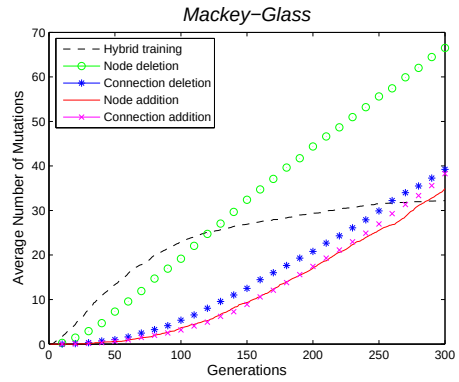

Figure 2: Average Mutations for Mackey-Glass TS. *Successful Training* parameter set to 30%

without allowing the architectural mutations to take part in the evolutionary process. That produces networks with a poor performance at the end of the evolution (i.e. bigger prediction errors are obtained).

In Fig. 1 is presented the average mutation rates over the entire evolution (set to 300 generations) for the *Mackey-Glass* TS with a relaxed value in the *successful training* parameter (set to 70%). There it can be seen that the hybrid training dominates the evolutionary process, with the other mutations used only a few times. Conversely, in Fig. 2 it can be seen how the other mutations are used more frequently if a strict parameter value is set (i.e. 30%), which shows that there are more modiffications in the architectures than with a relaxed value. That is clearly a desirable behaviour if we want to look for more solutions in the search space.

Analysing this issue for the other TS revealed that the TS behavior has a big effect on the evolutionary process. For example, for the *Logistic* TS the average mutation rates arising for the three different parameter values (30, 50 and 70%) were similar, using all the mutations in the evolutionary process, similar to the pattern of Fig. 2. In other words, for this TS, this parameter was not so crucial.

On the other hand, for the *QP2* and *QP3* TS, the average mutation rates for the three parameter values used were similar to Fig. 1, where the hybrid training was the mutation most used in the evolution. Interestingly, for those cases the average error per generation were decreased continuously in all the evolutionary process, which suggests that maybe for

these TS more epochs are required in the partial training to find suitable weights and then give the opportunity for other mutations to be applied. But even though the behavior was similar across the three values, the best performance found was with a value of 30%, i.e. when the training is marked as successful if the error is decreased by 70%. That demonstrates the importance of this value and the effect that the dynamic of the TS has over the evolution.

Summarizing the importance of the "successful training" parameter (evaluated for TS forecasting), it can be said that a strict value produces better networks that can reach the smallest prediction errors, allowing more architecture modifications and consequently looking for an optimal solution across a bigger search space.

3.2 Whether to evolve the inputs

For some TS there is enough previous domain knowledge to know many of the different optimal parameters needed to classify or predict them, e.g. for the *Mackey-Glass* and *Logistic* TS [4, 5, 13, 14] or for the *Lorenz* TS [2]. But even then, there are other studies that differ from the standard parameters, such as [6] for the *Mackey-Glass* TS. In this study, we are interested in the optimal number of inputs and delays to evolve the network's architectures. But it is not always possible to obtain this information from the literature, e.g. to predict a new TS that has never been studied before and for which there is no previous information (as is likely to be the case for many real world scenarios). Therefore a method is required to determine the inputs for the networks and the delays between them to perform the forecasting (Eq. 1). Other researchers prefer to evolve the inputs and delays, such as [9] for the *SP500* TS.

Thus, in this section is presented some preliminary experimental results that are focused on determining if it is better (or not) to evolve the inputs as another parameter inside the EPNet algorithm. To do this, two experiments were set up: in the first the inputs were evolved, and in the second the inputs were fixed during evolution and calculated using the VRA package mentioned in Sec. 2, with random connections between the inputs and hidden nodes. The experiments analysed in this section were run to 2000 generations to provide detailed results for the algorithm, i.e. to see if the parameters converge when allowed more generations. The rest of the parameters were set as described in Sec. 3

The inputs here were treated as another node in the network, so the mutation of add node or delete node could be applied to them. Similarly, the connections from inputs to hidden nodes were treated as another connection between hidden nodes, so the mutation add or delete connection could be applied to them. With this configuration, the delays are implicit in the representation.

From the preliminary experiments it was determined that for around half of the TS it was useful to evolve the inputs, but the other half gave good predictions if the inputs were calculated and fixed. Our results suggest that more experimental results would be beneficial here, and that this topic really needs to be addressed by a more complete research program, which is beyond the scope of this paper. Nevertheless, it can be said that calculating the inputs and delays (with the False Nearest Neighbour and Average Mutual Information) does not always give appropriate values for finding accurate ANNs, and therefore it was generally better to evolve the inputs and delays.

From those experiments it was also observed that if the inputs are fixed, the evolution of hidden nodes or connections advances faster in some TS, consistent with the fact that there are fewer parameters to evolve. On the other hand, if the inputs are evolved, it was found that in some cases the algorithm found accurate networks faster, even though they required more computational processing to evolve the increased number of parameters.

Since architectural modifications generally produce large changes in evolved networks' behavior, it might be expected that the addition or deletion of inputs could have the same effect on the network's performance. Sometimes the addition or deletion of inputs results in significant variation in the performance of the networks, however those variations do not usually have a big impact on the evolution, probably because after addition or deletion the network passes to the partial training phase which could correct any undesirable deviation in the networks' learning. Again, the dynamic of the TS influences the behavior of the algorithm. The most important conclusion found was that it is better to evolve the inputs if there is no previous domain knowledge of the given TS. Thus, the rest of the experiments performed in this study were set up evolving the inputs, even for TS where there is previous information.

4. BEST INDIVIDUAL RESULTS

In this section is presented the results from a set of experiments developed to obtain the best evolved individual predictions, for 30 independent runs for each TS. The configuration used was that determined in Sec. 3, with the "successful training" parameter set to 30%, and the inputs evolved rather than calculated. A robust metric to measure performance is clearly required. One obvious choice is the Normalized Root Mean Square Error (NRMSE) defined as:

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{N} (x_i - o_i)^2}{\sum_{i=1}^{N} (o_i - \bar{o})^2}} \qquad (2)$$

where x_i is the prediction, o_i is the actual value and $\bar{o}$ is the mean of the actual values. Other measures, such as "*accuracy*" as a percentage, were also tested, but found to be less informative. For example, for some TS a percentage accuracy might say that a predictions was close to 100%, when in reality the NRMSE was high (around or over 0.5) and the prediction was only following the trend of the original data. Therefore NRMSE was used to perform all the main comparisons in this work.

Table 2 (columns 2-5) shows the best individual NRMSE results obtained with the independent test set for each TS. The column "$Mean$" shows the average of the best individual NRMSE results from each of the 30 independent runs, and "*Std Dev*" shows the corresponding variance across runs. The column "Min" shows the NRMSE of the best individual overall, and the column "Max" shows the worst of the best individuals from the 30 runs. The TS are arranged according to their dynamics: Chaotic: from *Henon* to *Rossler*; Demographic: *Births in Quebec*; Economical/Financial: from *Dow Jones* to *SP500*; Hydrological: *Colorado River* and *Lake Eriel*; Physics: *Equipment temperature* to *Sunspots*; and the last two form the Santa Fe competition.

The actual prediction errors can be best visualized by comparing the denormalized predictions with the actual target values. Thus, the predictions of the best individuals

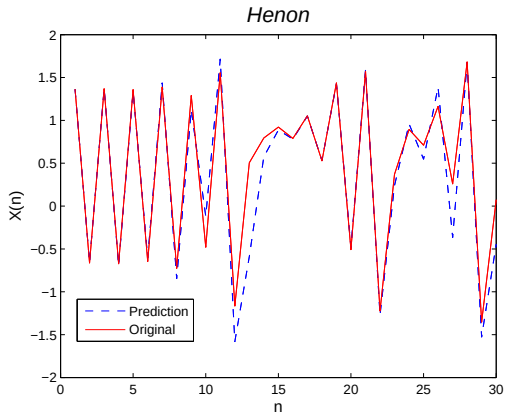

Figure 3: Prediction to 30 steps ahead for the Henon TS. Best individual found over 30 runs

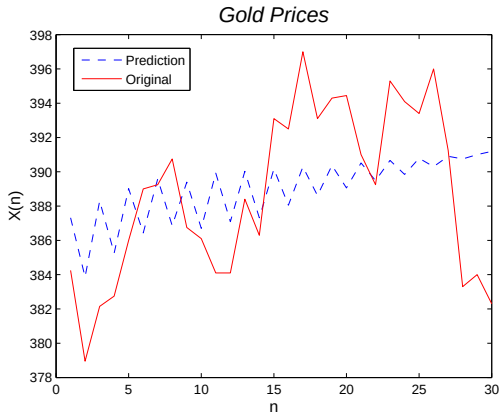

Figure 5: Prediction to 30 steps ahead for Gold Prices TS. Best individual found over 30 runs

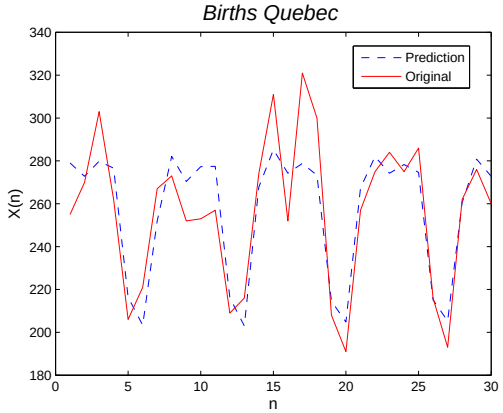

Figure 4: Prediction to 30 steps ahead for Births in Quebec TS. Best individual found over 30 runs

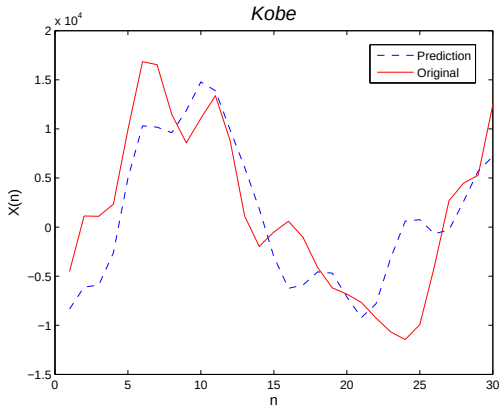

Figure 6: Prediction to 30 steps ahead for Kobe TS. Best individual found over 30 runs

over the 30 independent runs for four representative TS are shown in Figs 3 to 6 using the denormalized predictions and actual target values. Summarizing the full set of results: Fairly good predictions are achieved for the *Henon* TS (Fig. 3) and *Births in Quebec* TS (Fig. 4). Conversely, for the *Dow Jones*, *IBM Stock*, *SP500* and *Gold Prices* TS, we do not achieve good predictions at all. These are well known to be difficult TS to predict, given that they come from complex economic systems. The predictions there only try to follow the trend and accurate predictions were not possible, as is clear from the *Gold Prices* TS graph (Fig. 5). However, if it is only required to know the trend, such predictions could still be useful in a real world scenario, e.g. to know if the trend will continue upwards or downwards. The remainder of the predictions were acceptable, in the sense that in the majority of cases they did obtain a reasonably accurate prediction. Even the *Kobe* TS (Fig. 6), that gave a NRMSE of 0.618348 (for best individual over 30 independent runs, Table 2 column 4), still has an accurate prediction in the sense that it could be useful in a real world scenario.

5. ENSEMBLE RESULTS

In this section are presented the results from a series of experiments designed to determine whether ensembles formed from the last population of individuals evolved with the EPNet algorithm for the TS forecasting task are better, or worse, than the best individuals found (Sec. 4). Two different linear combination methods to compute the ensemble outputs are tested: taking the Average and using the Rank-Based Linear Combination (RBLC) method. The two ensemble approaches are compared against each other, and against the Best individuals. Since 30 independent runs were performed to test the results statistically, another ensemble formed of the best individual per run was used to see if the prediction could be improved further (and to compare that against the other methods).

When the ensembles are created using all the individuals from the last generation of the EPNet Algorithm, there is a danger that the best possible performance will not be obtained because the last generation will always contain a mixture of fit individuals and others that are not so good. It is possible that the overall performance will be seriously reduced by large errors being introduced into the ensemble output by the worst individuals. Since we are always working with a fitness sorted population, where the first individuals are better than the last members of the population, is it straightforward to form ensembles using only the best evolved individuals. In this way, we carried out experiments which showed that, overall, the best ensembles with both the Average and RBLC methods were those created with only the fittest half of the population. For this reason, the

Table 2: Prediction performances measured as NRMSE on the independent test set

Time Series	Best individual				Ensemble Average method				Ensemble RBLC method			
	Mean	*Std Dev*	*Min*	*Max*	*Mean*	*Std Dev*	*Min*	*Max*	*Mean*	*Std Dev*	*Min*	*Max*
Henon	0.636499	0.204336	0.302114	0.956291	0.554292	0.077605	0.405555	0.697547	0.482403	0.125600	0.248148	0.725689
Ikeda	0.909702	0.055751	0.759410	0.992052	0.887642	0.050243	0.764676	0.993704	0.852636	0.069143	0.685461	0.965083
Logistic	7.31E-04	3.01E-04	2.74E-04	1.58E-03	0.000842	0.000451	0.000223	0.002147	0.000590	0.000229	0.000198	0.001055
Lorenz	0.020447	0.012247	2.49E-03	0.062434	0.034556	0.024151	0.004911	0.101607	0.015360	0.012058	0.002231	0.063757
Mackey-Glass	5.03E-03	0.001839	1.88E-03	1.00E-02	0.003825	0.001298	0.002084	0.006607	0.003293	0.000992	0.001891	0.005612
Qp2	0.080269	0.042100	0.018826	0.252037	0.097080	0.041825	0.036019	0.198994	0.063854	0.031575	0.022730	0.138016
Qp3	0.404471	0.128390	0.136128	0.663789	0.536005	0.091588	0.319728	0.733362	0.390987	0.117870	0.116490	0.601360
Rossler	3.92E-03	0.004671	5.51E-04	2.65E-02	0.054942	0.157863	0.000529	0.649472	0.005612	0.014313	0.000457	0.079038
Births in Quebec	0.512206	0.015515	0.487339	0.546592	0.512350	0.013755	0.490407	0.550965	0.500604	0.015007	0.470685	0.534231
Dow Jones	1.091639	0.044362	0.992117	1.161880	1.132410	0.050727	1.075047	1.276407	1.092141	0.040172	1.006429	1.164714
Gold Prices	1.061642	0.120833	0.895633	1.463080	0.667389	0.124511	0.504391	1.082530	0.607098	0.128760	0.437117	1.115892
IBM Stock	0.869855	0.061650	0.784294	1.068950	1.137256	0.171837	0.893250	1.603258	1.021421	0.087460	0.897422	1.219439
SP500	0.642662	0.040994	0.570208	0.733510	0.907062	0.060131	0.829057	1.074574	0.865357	0.055375	0.809707	1.034101
Colorado River	0.821921	0.089596	0.639491	0.998298	0.669257	0.043266	0.603953	0.808359	0.635859	0.027518	0.580866	0.697312
Lake Eriel	0.697431	0.130573	0.548006	1.138210	0.834949	0.097537	0.677836	1.076927	0.781777	0.079763	0.637406	0.946411
Equipment Temp.	0.803416	0.108539	0.540734	1.057860	0.873240	0.117517	0.708384	1.100276	0.766290	0.098556	0.559897	0.965825
Kobe	0.794816	0.110117	0.618348	1.019470	0.907487	0.107310	0.723403	1.105762	0.785307	0.105720	0.607867	0.989435
Star	0.024374	0.005112	0.016734	0.036160	0.024645	0.006535	0.012370	0.036240	0.021938	0.005182	0.012842	0.033448
Sunspot	0.683762	0.063064	0.533456	0.798159	0.758341	0.067267	0.619330	0.921413	0.680211	0.062338	0.491676	0.765263
D1	1.180132	0.207090	0.731355	1.532210	1.129695	0.122540	0.849833	1.351434	1.025955	0.183861	0.648052	1.443262
Laser	0.086133	0.036005	0.042059	0.202223	0.077413	0.033201	0.034756	0.164771	0.066214	0.031229	0.030088	0.146956

ensembles presented in the next two sections (Sec. 5.1 and 5.2) were created with only the half of the population (the best individuals of the final generation). Taking exactly half the population does not represent the optimal ensemble size for each problem instance, but if we want to optimize the correct number of networks in each ensemble, the computational cost will be increased, e.g. using the evolutionary algorithm to find the best combination of networks to form the appropriate ensemble for each problem.

5.1 Ensemble with Average

In this section, the output of the ensembles is calculated as the average of the outputs from each constituent network. This method may not prove to be optimal, but it is the simplest way to calculate the output of the ensemble, and was used as an initial test. The results presented here correspond to the NRMSE on the final independent test set. As noted above, the ensembles were created using only the fittest half of the population of the final generation, so any very poor performance individuals did not affect the outcome.

Table 2 (columns 6-9) shows the prediction results for the ensembles formed by the Average method. Comparing the corresponding mean values in Table 2 (i.e. columns 2 and 6) suggests that the ensemble with the Average method performs better than using the Best individual for 7 of the 21 TS. Statistical significance of these differences was tested using the standard *t-test* (two-tailed with unequal variances). Table 3 shows, for each TS, the *t-test* p values for the three main comparisons of this study: comparing the Best individual to the ensemble formed with the Average method, comparing the Best individual with the RBLC ensemble (next section), and comparing the two ensemble methods against each other. At 0.1 level of significance, the Average method ensemble is significantly better than the Best individual for 3 of the 21 TS, is significantly worse for 9 TS, which means that the best individual gave better results in more cases than the Average method. For the rest of the cases (9 TS) there were not significance in the results.

5.2 Ensemble with Rank-Based Linear Combination

In this section are presented the results for ensembles formed from the fittest half of the population using the Rank-Based Linear Combination (RBLC) method [15]. The main new aspect here is the calculation of a weight w_i for the ith individual network in a population sorted by fitness (with the best-fitness individual at the top, i.e. at $i = 1$):

$$w_i = \frac{exp(\beta(N + 1 - i))}{\sum_{j=1}^{N} exp(\beta j)} \quad (3)$$

which is used to give more importance to better individuals. This is achieved by calculating the ensemble output O as the weighted average of the network outputs o_i:

$$O = \sum_{j=1}^{N} w_j o_j \quad (4)$$

The β parameter was chosen after some preliminary experiments trying different values (0.1, 0.25, 0.5 and 0.75) for each TS. On average it was found that a value of 0.25 was better for the *Mackey-Glass* TS; 0.5 for *Henon*, *Lake Eriel* and *Laser* TS; and the rest had a better performance with a value of 0.75. Thus, the β parameters of this section and Sec. 5.3 were set up with those values to calculate the networks' weights in the ensemble approach. Note that the β parameter was optimized for each TS using standard model selection techniques because it was expected that different values would be appropriate for different TS. If the size of the ensemble or the constituent individuals change, it is possible that those optimized values will need changing.

Table 2 (columns 10-13) shows the prediction results obtained for ensembles formed using the RBLC method. In

Table 3: Significance of differences between Best individual, Average ensemble, and Rank-Based Linear Combination ensemble, indicated by t-test p values

Time Series	*Best − Ave.*	*Best − RBLC*	*Ave. − RBLC*
Henon	4.64E-02	9.57E-04	0.010384
Ikeda	0.112892	8.73E-04	0.029085
Logistic	0.267784	4.50E-02	0.009011
Lorenz	6.62E-03	1.10E-01	0.000341
Mackey-Glass	5.10E-03	0.000042	0.079705
Qp2	0.126211	9.33E-02	0.001023
Qp3	3.02E-05	0.673310	1.98E-06
Rossler	0.087325	0.542900	0.098793
Births in Quebec	0.969946	4.66E-03	0.002513
Dow Jones	1.60E-03	0.963541	0.001227
Gold Prices	5.40E-02	1.46E-01	0.002000
IBM Stock	2.13E-02	0.767294	0.007049
SP500	1.76E-02	0.453915	0.000815
Colorado River	0.592142	7.20E-02	0.024526
Lake Eriel	0.365550	9.12E-03	0.070344
Equipment Temp.	2.01E-02	0.170784	0.000336
Kobe	1.74E-04	0.734197	4.06E-05
Star	0.858921	7.19E-02	0.080948
Sunspot	4.26E-05	8.27E-01	1.87E-05
D1	0.256745	3.47E-03	0.013110
Laser	0.333520	2.58E-02	0.183671

Table 4: Performance of the ensembles composed from the best individuals from 30 independent runs. Errors on the independent test set

Time Series	*Average NRMSE*	*RBLC NRMSE*
Henon	0.312337	0.285958
Ikeda	0.767174	0.596142
Logistic	1.96E-04	8.92E-05
Lorenz	0.011636	9.69E-04
Mackey-Glass	1.42E-03	1.24E-03
Qp2	0.020087	0.019774
Qp3	0.299935	0.124260
Rossler	1.84E-03	4.05E-04
Births in Quebec	0.484020	0.476418
Dow Jones	1.064933	0.988405
Gold Prices	0.946449	0.888813
IBM Stock	0.837004	0.793824
SP500	0.620128	0.571701
Colorado River	0.701466	0.622002
Lake Eriel	0.474379	0.461984
Equipment Temp.	0.679475	0.569733
Kobe	0.713617	0.566432
Star	0.016177	0.014455
Sunspot	0.629409	0.543307
D1	0.724425	0.591681
Laser	0.026967	0.019909

this case, performance improvements are seen for 16 of the 21 TS when compared against the mean values for the Best individuals shown in Table 2 (i.e. comparing columns 2 and 10). As before, the significance of these differences are shown in Table 3 as *t-test* p values. At 0.1 level of significance, the RBLC ensembles are significantly better than the Best individuals for 10 of the 21 TS, and significantly worse only for the *Lake Eriel* TS.

Finally, we compare the two ensemble methods. The mean results in Table 2 (i.e. columns 6 and 10) and the final column in Table 3 show that the RBLC method has improved performance over the Average method on all 21 TS; though only 16 of these improvements are significant at 0.05 level, and 12 at 0.01 level.

5.3 Ensemble of best individuals from independent runs

We have seen that ensembles formed from the fittest half of the evolved populations can improve upon the performance of the best individuals in the evolved populations for some TS. It is conceivable that the more diverse ensembles generated by taking individuals from across multiple runs will provide even more performance improvements. Consequently, for each TS we created a further ensemble comprised of the best individual from each of the 30 independent runs, and tested them using both the Average and Rank-Based Linear Combination (RBLC) methods. Table 4 presents the NRMSE prediction results obtained for each method.

Note that, because the ensemble here is already using all the individual evolutionary runs, it is not possible to give the same statistics as in the Tables 2, nor perform the significance *t-tests*. However, the NRMSE columns in Table 4 can still be directly compared against the mean and best (i.e. *Min*) results from the other approaches (Table 2 columns 2, 4, 6, 8, 10 and 12), to give an indication of the power of the approach. The ensemble of best individuals with Average is seen to improve the prediction for all TS over the mean Best individual results, and is better than the *Min* Best individual results for 7 TS. It is better than the mean for 19 TS compared with the standard ensemble with Average from Table 2 column 6, and is better than the *Min* result for 14 TS. The ensemble of best individuals with RBLC shows improvement for all 21 TS compared with the mean and 16 TS against the *Min* Best individual results (Table 2 columns 2 and 4); improvement for 20 TS compared with the mean, and 18 TS compared with the *Min*, against the standard ensemble with Average (Table 2 columns 6 and 8); and improvement for all 20 TS compared with the mean, and 13 TS compared with the *Min*, against the standard ensemble with RBLC (Table 2 columns 10 and 12).

The ensemble of best individuals with RBLC also shows improvement for all 21 TS over the ensemble of best individuals with Average (Table 4). Consequently, it can be concluded that the RBLC method is better than the Average method in almost all cases considered.

6. CONCLUSIONS

This paper has explored the Time Series (TS) forecasting improvements obtainable by using ensemble approaches in conjunction with a popular evolutionary algorithm for evolving Artificial Neural Networks (ANNs). We first presented an analysis of the various parameters used in the EPNet Algorithm to evolve ANNs for TS forecasting. The algorithm was found to be not as sensitive to variations of some parameters, like the population size or initial learning rate, as others, in particular the *successful training* parameter. It was shown how important it is to set this with an appropriate value. Then, variations of the algorithm were

compared, in particular the differences in results obtained by using fixed calculated input architectures compared to when they are evolved. It was determined after some preliminary experiments that calculating the Average Mutual Information for the time delay and False Nearest Neighbour for the embedded dimension was not the best option for all TS. For this reason the main experiments presented in this work were performed evolving the inputs and delays.

After setting those details, the best individual evolved ANN results were compared against those of ensembles of evolved individuals. In some cases it seemed that the ensembles worked better, but it was found that the best results were not achieved if the entire population of the last generation of the EPNet Algorithm was used to form the ensembles. Instead it was better to only use the fittest half of the population, discarding the worst individuals in the population to avoid the introduction of unnecessary noise/error into the prediction of the ensembles.

Comparisons were then made between two approaches for combining the outputs of the ensemble constituents: a simple Average versus a Rank-Based Linear Combination (RBLC) method. It was found that, overall, the RBLC ensembles were better than Average ensembles. On the other hand the Average ensembles only improved a few TS and the RBLC improved around the half of them at 0.1 level of significance when they are compared with the best individual. Finally, when ensembles of best individuals from independent runs were tested, further improvements over nearly all the previous results were achieved.

The diversity of information in the ensembles is thus seen to provide better TS forecasting results than individual solutions, as long as appropriate individuals and output combination methods are used. There remain many further possible variations of the approaches studied in this paper. For example, more carefully optimized values for the various parameters (e.g. β), or inclusion of more than one individual from each independent run in the ensembles of best individuals approach. It is hoped that a more exhaustive study, including the evolutionary optimization of such details, will be presented in a longer future publication.

In summary, a detailed analysis of building ensembles from the evolved populations of the EPNet Algorithm was carried out on 21 TS with different dynamics and from different fields, and improved results were obtained for almost all of them. Thus, evolving ANNs and using ensembles with the EPNet algorithm for the TS forecasting task seems to be as successful as analogous ensemble EPNet approaches for classification tasks [15], demonstrating that the ensemble has more valuable information than a single individual for the TS forecasting task.

7. ACKNOWLEDGMENTS

The first author would like to thank CONACYT for the support of his graduate studies through a scholarship.

8. REFERENCES

[1] J. Belaire-Franch and D. Contreras. Recurrence plots in nonlinear time series analysis: Free software. *Journal of Statistical Software*, 7(9), 2002.

[2] R. J. Frank, N. Davey, and S. Hunt. Time series prediction and neural networks. *Journal of Intelligent and Robotic Systems*, 31:91–103, 2001.

[3] Hyndman, R.J. (n.d.). Time series data library. http://www.robhyndman.info/TSDL, Accessed on January 2009.

[4] H. A. Mayer and R. Schwaiger. Evolutionary and coevolutionary approaches to time series prediction using generalized multi-layer perceptrons. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99*, 1:275–280, 1999.

[5] R. Mikolajczak and J. Mandziuk. Comparative study of logistic map series prediction using feed-forward, partially recurrent and general regression networks. *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, 5:2364–2368 vol.5, Nov. 2002.

[6] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *ICANN '97: Proceedings of the 7th International Conference on Artificial Neural Networks*, pages 999–1004, London, UK, 1997. Springer-Verlag.

[7] Santa Fe Competition. The Santa Fe time series Competition Data. Stanford Psychology, Stanford University. http://www-psych.stanford.edu/∼andreas /Time-Series/SantaFe.html, Accessed on January, 2009.

[8] F. Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980*, volume 898, pages 366–381, Berlin, 1981. Springer.

[9] F. E. H. Tay and L. J. Cao. ϵ-descending support vector machines for financial time series forecasting. *Neural Processing Letters*, 15(2):179–195, 2002.

[10] E. A. Wan. Time series data. Department of Computer Science and Electrical Engineering. Oregon Health & Science University. http://www.cse.ogi.edu/ ∼ericwan/data.html, Accessed on 19 March, 2008.

[11] E. Weeks. Chaotic time series analysis. Physics Department, Emory University. http://www.physics. emory.edu/∼weeks/research/tseries1.html, Accessed on January, 2009.

[12] A. S. Weigend and N. A. Gershenfeld, editors. *Time series prediction: Forecasting the future and understanding the past.* Addison Wesley, 1994.

[13] X. Yao and Y. Liu. Epnet for chaotic time-series prediction. In *SEAL'96: Selected papers from the First Asia-Pacific Conference on Simulated Evolution and Learning*, pages 146–156, London, UK, 1997. Springer-Verlag.

[14] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, 1997.

[15] X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 28(3):417–425, Jun 1998.

[16] X. Yao and Y. Liu. Ensemble structure of evolutionary artificial neural networks. *Proceedings of IEEE International Conference on Evolutionary Computation, 1996*, pages 659–664, 20-22 May 1996.

[17] G. P. Zhang and V. L. Berardi. Time series forecasting with neural network ensembles: An application for exchange rate prediction. *The Journal of the Operational Research Society*, 52(6):652–664, 2001.