UNIVERSITY OF EXETER

**ORE** Open Research Exeter

**TITLE**

Pareto Evolutionary Neural Networks

**AUTHORS**

Fieldsend, Jonathan E.; Singh, Sameer

**JOURNAL**

IEEE Transactions on Neural Networks

**DEPOSITED IN ORE**

11 July 2013

This version available at

http://hdl.handle.net/10871/11712

**COPYRIGHT AND REUSE**

Open Research Exeter makes this work available in accordance with publisher policies.

**A NOTE ON VERSIONS**

The version presented here may differ from the published version. If citing, you are advised to consult the published version for pagination, volume/issue and date of publication

# Pareto Evolutionary Neural Networks

Jonathan E. Fieldsend*, *member IEEE*, and Sameer Singh, *member IEEE*

Department of Computer Science,
University of Exeter, Exeter EX4 4QF, UK

**Abstract**

For the purposes of forecasting (or classification) tasks neural networks (NNs) are typically trained with respect to Euclidean distance minimisation. This is commonly the case irrespective of any other end user preferences. In a number of situations, most notably time series forecasting, users may have other objectives in addition to Euclidean distance minimisation. Recent studies in the NN domain have confronted this problem by propagating a linear sum of errors. However this approach implicitly assumes *a priori* knowledge of the error surface defined by the problem, which, typically, is not the case.

This study constructs a novel methodology for implementing multi-objective optimisation within the evolutionary neural network (ENN) domain. This methodology enables the parallel evolution of a population of ENN models which exhibit estimated Pareto optimality with respect to multiple error measures. A new method is derived from this framework, the Pareto evolutionary neural network (Pareto-ENN). The Pareto-ENN evolves a population of models that may be heterogeneous in their topologies inputs and degree of connectivity, and maintains a set of the Pareto optimal ENNs that it discovers.

New generalisation methods to deal with the unique properties of multi-objective error minimisation that are not apparent in the uni-objective case are presented and compared on synthetic data, with a novel method based on bootstrapping of the training data shown to significantly improve generalisation ability. Finally experimental evidence is presented in this study demonstrating the general application potential of the framework by generating populations of ENNs for forecasting 37 different international stock indices.

**Index Terms**

*Neural networks, evolutionary computation, adaptive topologies, multiple objectives, time series forecasting.*

*Corresponding author.

# 1 INTRODUCTION

The use of neural networks (NNs) in the time series forecasting domain is now well-established. There are a number of review papers in this area (for example, Adya and Collopy [5]), as well as methodology studies [49, 53]. The main attribute which distinguishes NN time series modelling from traditional econometric methods is their ability to generate non-linear relationships between a vector of time series input variables and a dependent series, with little or no *a priori* knowledge of the form that this non-linearity should take. This is opposed to the rigid structural form of most econometric time series forecasting methods, e.g. auto-regressive (AR) models, exponential smoothing models, (generalised) auto-regressive conditional heteroskedasticity models ((G)ARCH), and auto-regressive integrated moving average models (ARIMA) [10, 29]. Apart from this important difference, the underlying approach to time series forecasting itself has remained relatively unchanged during its progression from explicit regression modelling to the non-linear generalisation approach of NNs. Both of these approaches are typically based on the concept that the most accurate forecast, if not the actual realised (target) value, is the one with the smallest Euclidean distance from the actual.

## 1.1 Multi-objective requirements

When measuring time series prediction performance practitioners often use a range of different error measures (for example, 15 commonly used error measures are reported in [8]). These error measures tend to reflect the preferences of potential end users of the forecast model. For instance in the area of financial time series forecasting, correctly predicting the directional movement of a time series (for instance of a stock price or exchange rate) is arguably more important than just minimising the forecast Euclidean error. Recent work that attempts to encapsulate multiple objectives using NNs have introduced augmentations to the traditional approaches of NN training. These have been in the form of propagating a linear sum of errors [61, 62], a product of terms [65], and penalising particular misclassifications more heavily [54]. However these approaches implicitly assume the practitioner has some knowledge of the *true Pareto error front* defined by the generating process, and the features and NN topology they are using to model it (in order to specify the ratio of various errors, or the form of penalisation). The roots of this problem are now discussed.

## 1.2 Problems with the linear combination of errors approach

Figure 1 illustrates the current approach to multi-objective training in NN regression and classification. Consider the situation where a number of errors measures (objectives) are used that lie in the range [0,1]. Given that the practitioner wishes to minimise these errors, the typical approach in linear sum back-propagation is to minimise

the composite error $\varepsilon_C$. In the $D$ error case (where there are $D$ errors to be minimised) this is:

$$\varepsilon_C = \alpha_1\varepsilon_1 + \alpha_2\varepsilon_2 + \ldots + \alpha_D\varepsilon_D \, , \, \sum_{i=1}^{D} \alpha_i = 1 \, , \, \forall i \, 0 < \alpha_i < 1 \tag{1}$$

The $D = 2$ dimensional case is illustrated in Figure 1a and 1b where the practitioner gives equal weighting to both errors, and both errors lie within the same range. This is calculated as:

$$\varepsilon_C = 0.5\varepsilon_1 + 0.5\varepsilon_2. \tag{2}$$

This approach implicitly assumes that the interaction between the two error terms is symmetric. Figure 1a illustrates the situation described, where the minimum error surface (the true Pareto front) defined by the problem is shown, with suboptimal models lying behind it denoted by circles. On its extremes it can be seen that the error combinations (0.0, 1.0) and (1.0, 0.0) are possible, which define the axial parallel hyper-boundaries of the front. On the application of Equation 2, each dashed line shown represents a set of objective combinations that are ranked as equivalent (the lines gradient reflecting the prior weightings). It is evident that if the true Pareto front is reached by the training process, then the model returned is one tangential to one of these parallel lines. In the case of Figure 1a this model is shown to have the error properties (0.25, 0.4). Figure 1b illustrates the same situation, with identical hyper-boundaries but a slightly different degree of convexity of the front. In this case the model returned is defined by the error properties (0.3, 0.5). The two models are significantly different, and in both cases, due to the shape of the Pareto error fronts (and contrary to the desires of the user), the error properties of the models returned are not equal. Although the feasible range of both error measures are the same, the interaction of the errors, as demonstrated by the shape of their true Pareto fronts, results in the return of models, that though Pareto optimal in themselves, do not represent the preferences of the practitioner. An even worse situation arises if the true Pareto front is non-convex. In this case composite error training (if the true Pareto front is reached) will only return those models that are on the extremes of the true Pareto front, as illustrated in Figure 1c. This is irrespective of the values used for $\alpha_1$ and $\alpha_2$. The model returned will always be the one that strictly minimises one of the objectives (errors). This problem with the linear weighting approach has been known for a number of years in the MOEA literature (theoretical proofs are provided in [16]), but it has not been addressed when using linear weighting to propagate multiple objectives in NN training. An alternative is to optimise with respect to one objective, with the second objective formulated as a constraint (for instance that is has a minimum or maximum value [58]). In the case where the shape of the trade-off surface is unknown, a set of models need to be trained that provide an estimate of the true Pareto surface so a final operating model can be selected. The composite weighting gradient descent or the constrained optimisation approach can be used, however $N$ runs would be needed with

different weights ($\alpha$) (Equation 1) or constraints to obtain $N$ different individuals on the Pareto front (therefore it would be subject to high computation time). However, where the true Pareto front itself is convex even this expensive option is infeasible for the composite weighting approach. An example of this draw-back is shown in [65] where a composite error term is used. [65] reports that the composite error weights were adjusted a number of times in order to find the best results on the test data, underlining the fact that the shape of the true Pareto error front was unknown.

A general framework for training NNs which is not susceptible to these problems will now be presented.

### 1.3 Need for a general framework

Given that it is likely that the error surface defined by the generating process is not known, a well defined approach to implementing multiple objective training within NNs is needed. Through the use of multi-objective evolutionary algorithms (MOEAs) it is possible to find an estimated Pareto set of the combination of parameters to multiple objective 'clean' function modelling problems [14, 17, 23, 25, 60, 69]. A Pareto set of solutions is defined such that in a set of parameter combinations $F$, no single parameter combination $F_i$ is better or equivalent on *all* other objective measures, than any other set member $F_j$. That is no parameter combination *dominates* any other parameter combinations in the set. Over the last 16 years, since the work by Schaffer [56], MOEAs have been applied to a vast number of design problems, where mathematical formulae define the multi-objective surface to be searched. These methods had not, until very recently, been applied to the noisy domain of multi-objective NN training. There are a limited number of studies using a MOEA to train a population of multi-objective NNs; those by Fieldsend and Singh [24] and Kupinski and Anastasio [40] are concerned with more than one type of forecast/classification error, whereas other previous studies using MOEAs and NNs have been concerned with trading off network complexity and a single error term [3, 2, 1, 27]. Chaiyaratana and Zalzala [13] in contrast use MOEAs with the outputs of standardly trained NNs.

The new framework proposed here is designed to use of those evolutionary computation (EC) methods which have previously been applied to uni-objective NN design, genetic algorithms (GAs), evolution strategies (ES) and particle swarm optimisation (PSO). GAs have previously be used for feature selection [12, 64] and topography selection [6, 9, 39, 45, 46, 48, 63] and ESs have been used for weight optimisation [28, 52, 55, 66], and adaptive topography selection [21, 47, 68]. The recent EC technique of PSO [34] has also proved popular as a uni-objective NN optimiser [15, 18, 19, 32, 59].

## 2 Multi-objective evolutionary neural network framework

The use of evolutionary approaches to NN training (with a single error function) has received increasing attention in recent years as these approaches have a number of intuitive advantages over gradient descent training in this domain. Application dependent error measures, for which a derivative may be extremely costly to calculate, can be easily incorporated in EC approaches to training, as derivatives are not needed. Indeed the ability of these approaches to facilitate NN training beyond the Euclidean objective was highlighted by Porto et al [52] (although not with multiple objectives), but taken no further in [52]. In addition, they benefit by training a population of evolutionary neural networks (ENNs) in one run, making them highly compatible with the concepts of population based multi-objective training from the MOEA literature. There are already a number of good reviews of MOEA methods [14, 17, 25, 60]. For the purposes of this study the processes of a MOEA will be described at a very general level, readers wishing a more in depth discourse on the issue are recommended to read any of these reviews. At the basic level the EC methods employed in MOEAs are similar to those used in the EC uni-objective optimisation field. However instead of a single *elite* member maintained by the process (the best individual found so far during the search), an archive, $F$, of mutually non-dominating solutions is maintained. Most recent work in the MOEA domain has been concerned with how the search population of the EC process and $F$ should interact, and how to evolve the search as time progresses. Recent work has also investigated efficient approaches to the storing and maintenance of $F$ as its size grows [20, 23, 51].

Here a framework for general multi-objective evolutionary neural network (MOENN) training is outlined in Figure 2, which can be viewed as a synthesis of work from the uni-objective ENN literature and the MOEA literature. In this new framework a set of estimated Pareto optimal ENNs is maintained in tandem with the training process.

The specific evolutionary operators used within the MOENN framework are determined by the optimisation process used (for instance ES, GA or PSO). In this study a new method is derived from the general framework for evaluation, called Pareto-ENN, which is now introduced.

### 2.1 The Pareto evolutionary neural network model

In the Pareto-ENN model introduced here, the parameters of ENNs are stored within *decision vectors* using representation of the *direct encoding* form [66], that will now be described.

Given a maximum size for a four layer feed-forward multi-layer perceptron (MLP) of $I$ input units (features), $H_1$ and $H_2$ hidden units in the first and second hidden layers, and $O$ output units, the decision vector length used to represent this network within an MOEA is of size $S$, where

$$S = (I + 1) \cdot H_1 + (H_1 + 1) \cdot H_2 + (H_2 + 1) \cdot O + I + H_1 + H_2 + D \tag{3}$$

The first $(I + 1) \cdot H_1 + (H_1 + 1) \cdot H_2 + (H_2 + 1) \cdot O$ genes are floating point and store the weight parameters (including biases) of the ENN, the next $I + H_1 + H_2$ are bit represented genes, whose value (0 or 1) denotes the presence of a unit or otherwise (in the two hidden layers and input layer). The next $D$ genes are again floating point and these are used to hold the $D$ error values associated with the network on the training data.

#### 2.1.1 Topology/feature selection through node addition/deletion

Topography and input feature selection is implemented within the Pareto-ENN model by bit mutation of the section of the decision vector representing the ENN architecture. This is facilitated by first determining a superset of input features and maximum hidden layer sizes. Once this is determined, any decision vector has a fixed maximum representation capability. Manipulation of structure is stochastic. By randomly bit flipping members of the first $I$ genes of the binary section of the chromosome the set of input features used by the ENN is adjusted. By bit flipping the genes in the subsequent binary section of the decision vector the hidden ENN topography is manipulated.

#### 2.1.2 Weights adjustment

In the Pareto-ENN the weight space of a network is perturbed by a set of values drawn at each epoch (generation) from a known distribution (Gaussian, Laplacian, etc.), as shown in Equation 4.

$$w_{i,n}^{k+1} = w_{i,n}^{k} + p \cdot \gamma \cdot \Theta \tag{4}$$

where $w_{i,k}^{n}$ is the $i^{th}$ weight of the $n^{th}$ network in the population at the $k^{th}$ epoch of training, $\Theta$ is a sample drawn from a user selected distribution, $\gamma$ is some multiplier and $p$ takes a value $(0, 1)$ with some probability.

This can be seen as similar to the common approach taken in uni-objective ENN optimisers for weight adjustment [28, 52, 55, 66], and was the only means of parameter adjustment used in [24, 40].

#### 2.1.3 Weight addition/deletion

Finally connectivity (and therefore complexity) is adjusted within the Pareto-ENN model by a GA type bit mutator, where at each generation every weight has a small probability of being severed. For this disconnected state the weight has the potential to be re-connected through the weight adjustment process described above.

The Pareto-ENN used in this study is driven by a (1+1)-ES process, where a single ENN is selected from the archive at $F^t$, evolved, and then compared to $F^t$ - which is subsequently updated if necessary at $t+1$. The archive

selection method used is the partitioned quasi-random selection (PQRS) method introduced by Fieldsend *et al.* [23]. An algorithmic description of the Pareto-ENN is shown in Algorithm 1.

**Algorithm 1** Implementation of the Pareto-ENN (standard training approach), $\mathcal{M}_S$.

| | |
|---|---|
| Inputs: | $M$, size of initial random population of solutions. Each solution chromosome $X_0, \ldots, X_m, \ldots, X_M$ representing the weights and topology of a ENN model.<br>$T$, maximum number of algorithm iterations (generations). |
| Output: | A non-dominated set of ENN models that are an estimate of the true Pareto front defined by the data generation process (represented by the training data), and the ENN genus. |
| 1: | **Initialisation:** Generate random NN population of size $M$, such that each parameter (weight) of the ENNs $\sim N(0, \alpha)$, and the binary part of the chromosome is either initialised at 1 or $\sim U(0, 1)$. Generate the empty frontal (non-dominated) set $F^0 = \emptyset$. Update $F^0$ with the non-dominated solutions from the random population, with respect to the chosen error terms. Initialise generation counter $t := 0$. |
| 2: | **Frontal Representatives**: Use PQRS [23] to select ENN representative from $F^t$, $F_R^t$. Create replica ENN of $F_R^t$, $\mathbf{x}^t$. |
| 3: | **Parameter Adjustment**: Adjust weights, topology, connectivity and inputs of ENN $\mathbf{x}^t$ using EC techniques. |
| 4: | **Fitness Assignment**: Evaluate the ENN $\mathbf{x}^t$ with respect to the user determined error measures on the training data presented. If $F^t \npreceq \mathbf{x}^t$ go to 5 otherwise delete $\mathbf{x}^t$ and go to 6. |
| 5: | **Update Archive:**<br>a) Insert ENN chromosome(s) $\mathbf{x}^t$ into $F^t$ if it is not dominated by individuals in $F^t$.<br>b) Remove ENN chromosome(s) from $F^t$ which are dominated by the $\mathbf{x}^t$. |
| 6: | **Loop**: Iterate epoch count, $t := t + 1$. If $t = T$ then go to 7, else go to 2. |
| 7: | Terminate algorithm and save members of $F^T$ for evaluation on test data. |

As discussed earlier in this section an archive $F$ is maintained of the non-dominated solutions found during the search process, which is initialised at line 1 in Algorithm 1. Decision vectors are subsequently replicated from this archive at each generation $t$ ($F_R^t$), before evolutionary adjustment and subsequent evaluation. In the context of the specific model used in this study the weights/topology/connectivity are adjusted simultaneously. The reason for taking this approach instead of doing each process separately (at different generations) is that performing them separately can prevent certain jumps in the decision space, and therefore can detrimentally constrain the search. The simple (1+1)-ES MOEA has performed well in comparison to many more complex approaches from the literature [35, 36], however, as the decision vector to be perturbed is drawn from $F$, the child must 'jump' to a non-dominating or better objective space position with respect to $F$ if it is to survive. As this is the case, all possible movements in decision space must have some probability >0 to allow escape from (frontal) local optima.

## 2.2 The problem of generalisation with multiple error measures

In an earlier work by the authors, [24], it was found that only a minority of the Pareto optimal set of ENNs fitted to training were Pareto optimal when evaluated with respect to test data; meaning that the estimated Pareto set of models on the training data were not necessarily a good estimate of the general Pareto set of models for the generating process. This clearly shows the as the approach to training by a simple separation of the data into a

training and test samples used in [24] and [40] is suboptimal, as it may lead to *overfitting*.

In order to aid generalisation in uni-objective problems, a commonly used technique is to separate time series data into consecutive training, validation and test sets. The forecast model is then trained on the first set until the measured error begins to increase on the second set, with the final generalisation error calculated on the third set. By stopping training when the observed validation error begins to rise the practitioner aims to prevent overfitting of the model on the training data. The use of this approach of NNs training to the multi-error situation is however problematic. This is because when a set of ENNs is being used, some members may exhibit falling error values on training and validation data, whilst others may exhibit some validation errors rising and some other validation errors falling. It is difficult therefore to ascertain when a set of Pareto-ENNs may indeed be over-fitting if the approach is directly transferred to MOENN training from ENN training domain. Instead two new methods are now introduced that are designed to increase the generalisation ability of models derived from the MOENN framework, which will be empirically compared in the Pareto-ENN method.

### 2.2.1 Proposed method $\mathcal{M}_V$, validation set training in MOENNs

The first new method to improve generalisation is inspired by the traditional validation set approach. As stated above, this approach cannot be transferred directly to the MOENN domain, however the main thrust of the approach can be recreated. As in the traditional approach, the data set is partitioned so that a portion of the data is separated as an 'unseen' test set on which the generalisation ability of the model(s) will be evaluated. The remaining data is split again to provide a training set and a validation set. A potential solution (ENN) is evaluated with regard to the training set. If this solution is found to be non-dominated by the current archive a copy is created and saved in the archive $F^t$ (line 5 of Algorithm 2). A second archive however is also maintained in $\mathcal{M}^V$, which maintains a non-dominated set with respect to the validation data. Potential solutions are only considered for insertion to this validation archive, $V^t$, if (and only if) it has initially been found to be non-dominating with regard to the training data at that generation. This process continues until algorithm termination, and the set of models returned are those residing in the validation archive, $V^T$, and not the training archive $F^T$ (when $T$ is the final evaluated generation).

By only comparing solutions to the validation archive (if they have been accepted previously to the training archive) this method attempts to prevent overfitting to the validation data. As solutions that are not non-dominating with respect to the validation set are not selected (even if there are non-dominating with the training set), this method also aims to prevent overfitting on the training data. A description of this training approach is shown in Algorithm 2.

**Algorithm 2** Implementation of $\mathcal{M}_V$ in the Pareto-ENN.

| | |
|---|---|
| Inputs: | As in Algorithm 1. |
| Output: | A non-dominated set of ENN models that are an estimate of the true Pareto front defined by the data generation process (represented by the training and validation data), and the ENN genus. |
| 1: | **Initialisation:** Generate random NN population of size $M$, such that each parameter (weight) of the ENNs $\sim N(0, \alpha)$, and the binary part of the chromosome is either initialised at 1 or $\sim U(0, 1)$. Generate the empty frontal (non-dominated) set $F^0 = \emptyset$, and the empty validation archive $V^0 = \emptyset$. Update $F^0$ and $V^0$ with the non-dominated solutions from the random population, with respect to the chosen error terms. Initialise generation counter $t := 0$. |
| 2: | **Frontal Representatives**: As in $\mathcal{M}_S$. |
| 3: | **Parameter Adjustment**: As in $\mathcal{M}_S$. |
| 4: | **Fitness Assignment**: As in $\mathcal{M}_S$. |
| 5: | **Update of Training Archive:** a) Insert ENN chromosome $\mathbf{x}^t$ into $F^t$ if it is not dominated by individuals in $F^t$. b) Remove ENN chromosome(s) from $F^t$ which are dominated by $\mathbf{x}^t$. |
| 6: | **Update of Validation Archive:** a) Insert ENN chromosome $\mathbf{x}^t$ into $V^t$ if it is not dominated by individuals in $F^t \bigwedge V^t$. b) Remove ENN chromosome(s) from $V^t$ which are dominated by the $\mathbf{x}^t$. |
| 7: | **Loop**: Iterate epoch count, $t := t + 1$. If $t = T$ go to 8, else go to 2. |
| 8: | Terminate algorithm and save members of $V^T$ for evaluation on test data. |

#### 2.2.2 Proposed method $\mathcal{M}_B$, bootstrap training in MOENNs

The second new method to improve generalisation is based on bootstrap techniques. The data is partitioned as in $\mathcal{M}_S$ into a training and test set. The training set is then bootstrap sampled to create $n$ data subsets, on which the ENNs are evaluated during the training process. Potential solution networks produced by the MOEA are initially evaluated with respect to all of the bootstrap sets. Initially this will lead to $nD$ fitness's associated with a solution (the number of bootstraps multiplied by the number of error terms to be optimised). The final $D$ fitness values attached to a decision vector for the archiving processes are the worst $D$ objective values recorded over the $n$ bootstrap sets.[1] This training method is designed to prevent overfitting on a particular subset of the training data, and also to prevent general overfitting to the training data itself. A description of this training approach is shown in Algorithm 3.

## 3 Experimental evaluation

In this section the three training methods applied in the Pareto-ENN model, $\mathcal{M}_S$, $\mathcal{M}_V$ & $\mathcal{M}_B$ (Algorithms 1-3) are compared on synthetic test data from the multi-objective NN literature, whose generating properties are known. The best performing training model is then applied to the real world application problem of financial forecasting.

[1] If bootstrapping is performed without replacement, it is evident that the size of these sets must be smaller than the original training set, otherwise the approach mimics the standard training method.

**Algorithm 3** Implementation of $\mathcal{M}_B$ in the Pareto-ENN.

| | |
|---|---|
| Inputs: | As in Algorithm 1, plus:<br>$n$, the number of bootstrap subsets generated from the original training sets.<br>$s$, the size of the bootstrap subsets. |
| Output: | As in Algorithm 1. |
| 1: | **Initialisation:** Generate $n$ bootstrap subsets of the training data of size $s$. Generate random ENN population of size $M$, such that each parameter (weight) of the ENNs $\sim N(0, \alpha)$, and the binary part of the chromosome is either initialised at 1 or $\sim U(0, 1)$. Generate the empty frontal (non-dominated) set $F^0 = \emptyset$. Update $F^0$ with the non-dominated solutions from the random population, with respect to the chosen error terms (using a solution's worst $D$ terms over the $n$ subsets).<br>Initialise generation counter $t := 0$. |
| 2: | **Frontal Representatives**: As in $\mathcal{M}_S$. |
| 3: | **Parameter Adjustment**: As in $\mathcal{M}_S$. |
| 4: | **Fitness Assignment**: Evaluate the ENN $\mathbf{x}^t$ with respect to the user determined error measures on the training bootstrap subsets presented. If $F^t \not\preceq \mathbf{x}^t$ go to 5 otherwise go to 6. |
| 5: | **Update of Archive:**<br>a) Insert ENN chromosome $\mathbf{x}^t$ into $F^t$ if it is not dominated by individuals in $F^t$.<br>b) Remove ENN chromosome(s) from $F^t$ which are dominated by the $\mathbf{x}^t$. |
| 6: | **Looping**: Iterate epoch count, $t := t + 1$. If $t = T$ go to 8, else go to 2. |
| 7: | Terminate algorithm and save members of $F^T$ for evaluation on test data. |

## 3.1 Synthetic data

In order to compare the three generalisation methods, as applied to the Pareto-ENN, data is used from a test function described in [43]. This test function is designed to demonstrate the trade-off between Euclidean fitting and robust fitting for the multi-objective training of NNs.

### 3.1.1 Data properties

In Euclidean minimisation the objective is to minimise

$$E_{\text{Euclidean}}(\mathbf{x}) = (y - \hat{y})^2 \tag{5}$$

where $y$ is the observed data and $\hat{y}$ its forecast. Their robust error term was defined as

$$E_{\text{Robust}}(\mathbf{x}) = \exp(\lambda |y - \hat{y}|^p) \tag{6}$$

where $\lambda$ and $p$ are both user defined. With $p = 2$ and $\lambda = 1$ this criterion is equivalent to Euclidean minimisation, and as $\lambda \to \infty$ it approaches the minimax criterion [42].

The test function from [43] used in this part of the study is a noisy sinusoid, with a random phase shift. It is defined for input $a$ as:

$$f(a) = \sin(a + c) + \epsilon \tag{7}$$

where $a \in [0, 2\pi]$, c has a 75% probability of being 0, and a 25% probability of being equal to 2. $\epsilon \sim N\left(0, 0.0706^2\right)$. A plot of 6000 input-output pair generated from this function by uniformly sampling across the range of $a$ is provided in Figure 3.

### 3.1.2 Experimental details

A single NN is trained from 5000 epochs using backprop (learning rate = 0.05, momemtum = 0.5), in order to find a good general starting point in decision space. The archive is initialised with the non-dominated ENNs from 1000 random perturbations of this initial NN, whose maximum topological representation is identical to those used in Lo & Bassu [43] (one hidden layer with 5 hidden logistic sigmoidal activation functions). The algorithm parameters are shown in Table 1. The training set was 300 patterns (or 200/100 with train/validation) for the test function. The bootstrap size was 240 patterns (80% of the training data), and a total of number of 10 bootstraps sets were taken. The test set size was 3000 patterns for both test problems and the Pareto-ENNs were trained for 5000, 15000, 25000 and 50000 generations.[2] Each algorithm was run 50 times with different initialisation vectors, different random seeds and different training data (drawn from the generating process defined in Equation 7). The two training errors to be minimised are the Euclidean distance and the criterion described in Equation 6 ($\lambda$ set at 20). The evaluated errors of the trained models are the average Euclidean error and maximum Euclidean error. The training size is kept deliberately small and the number of generations used were large to help quantify the effect of overfitting when attempting to estimate the Pareto error generating process. The three training methods used are $\mathcal{M}_S$ (as used in [24, 40]), $\mathcal{M}_V$ and $\mathcal{M}_B$, and each is started with identical states (initial archive) for each fold, with identical training and test sets.

The $\widetilde{C}$ measure [23] is used to compare the different model sets returned by the MOENNs. It counts the proportion of points in one set that are dominated by points in another (e.g. $\widetilde{C}(A, B)$ counts the proportion of individuals in set $B$ that are dominated by individuals in set $A$.) In this study it is used to pairwise compare the set of models from the three methods evaluated on test data.

The exact calculation of both of this method is described in the Appendix.

### 3.1.3 Results and interpretation

Figure 4 shows boxplots of the pairwise $\widetilde{\mathcal{C}}$ measure between the training methods for the 4 different generation lengths. $\mathcal{M}_B$, can be seen to be markedly superior to both of the other approaches. After 50000 generations forecast models trained using it are is seen to dominate, on average, 77% of models trained using $\mathcal{V}$ and 76% of models trained using $\mathcal{B}$. By 50000 generations $\mathcal{M}_B$ models can seen to be converging in the Figure 5 on this test

[2] The number of generations may seem large, but the Pareto-ENN used here is a (1+1) method. 50000 (1+1)-ES evaluations is equivalent in ENN evaluations to 500 generations of a GA with a population of 100.

problem, although the archive size is continually rising as the *resolution* of the front is increased (Figure 5 shows how $|F|$ increases with generation length).

These results indicate that the generalisation from MOENN training can be improved beyond the approach currently taken in the literature [24, 40], by using the proposed bootstrap training method, $\mathcal{M}_B$. This new generalisation technique will now be applied in the main empirical section of this study, for the forecasting of 37 international stock indices.

## 3.2 Financial data

An illustration of the interaction of multiple objectives in a problem, where a set of models is desired for collective use (as opposed to comparison) can be shown by analogy to the capital asset pricing model (CAPM) from finance [11]. The CAPM describes the relationship between risk and return in an optimum portfolio of stocks, where risk is to be minimised and return maximised. In Figure 7a the front $FF$ represents the Pareto optimal portfolios (called *efficient* portfolios in CAPM), with examples of other sub-optimal portfolios lying beneath $FF$ also marked. Line $SS$ is the security market line, with point $Rf$, where the security market line intersects the y-axis, representing the level of 'risk free' return available in the market place to the individual (e.g. through treasury bonds). The security market line is tangential to the efficient portfolio front, the point where it touches the front at '$a$' being the optimal *market* portfolio. In the simple illustration shown in Figure 7a, by investing in the market portfolio at point '$a$', and lending or borrowing at the risk free rate $Rf$, it is possible to operate on the security market line, gaining a higher rate of return for any level of risk than that possible by investing in an efficient portfolio of stocks. More complex interactions can also be modelled within the CAPM framework. For example where there are two different zero-risk rates in the market (one available to the user when borrowing, and another available from government bonds) the situation illustrated in Figure 7b occurs. Here the rate of return demanded by lenders is $Rf''$, whereas the 'risk free' rate of return for investors in bonds is lower at $Rf'$. The two tangential lines generated are $S'S'$ and $S''S''$, with the kinked Security Market Line itself a combination of the two (represented by a solid line). The central section of this line is described by the efficient portfolio front between portfolios '$a$' and '$b$'. In this situation the user therefore desires to know the portfolios described by points '$a$' and '$b$', and all those in between on the efficient portfolio frontier. The rates of risk and return described by the security market line to the left of '$a$' can be accessed by distributing the individual's wealth between government bonds (and gaining $Rf'$ return at zero risk) and portfolio '$a$' (and potentially gaining $Ra$ return at a risk of $Sa$). The risk and return levels described by the security market line to the right of '$b$' can be accessed by the individual borrowing from the market at the rate $Rf''$ and investing this, and all their other wealth in portfolio '$b$'.

An analogy can be drawn with the prediction of stock market prices. The Euclidean error of a model can be seen

as a proxy for a forecast model's risk, and a trading strategy (based around the direction success error for instance) as a measurement of the expected return of a model, as used in [24]. However in this study the measure of risk is the standard deviation of returns from the trading strategy used, a more accurate transference of economic theory [11]. The front $FF$ therefore represents the Pareto optimal set of regression models, with models '$a$' to '$b$' being the final models desired by the practitioner (to enable operation on the security market line). In addition, given that different individuals may experience differing $Rf$s (due to differing costs of borrowing and lending available to different individuals and institutions in the economy), points '$a$' and '$b$' will vary across individuals.

### 3.2.1 Data properties

The second empirical part of the this study is concerned with applying the model Pareto-ENN to a group of financial forecasting problems. The ENN are trained in order to forecast a transform of international stock indices. The data itself encompasses thirty-seven international indices, with the series varying in length, the longest being 4845 data series (over 19 years) and the shortest 416 data samples (under 2 years), all series are daily, containing open, high, low and close, and run until 7th February 2003, and were obtained from `http://uk.finance.yahoo.com/`. A description of these series is provided in Table 2.

The concern in this process is the optimisation of two measures, *risk* (minimised) and *return* (maximised), based on forecasting transforms of these series and using a trading strategy. Therefore the final set of archived Pareto optimal members, $F^T$, should provide an estimate of the trade-off of the risk/return defined by the generating process and trading strategy.

Financial forecasting (modelling the generating process of a financial time series, or process) is a popular application of NNs [5, 7, 24, 26, 30, 31, 38, 44, 49, 50, 53, 54, 57, 65]. However, in a number of studies misleading claims are made (or inferred) with regards to the actually efficiency of the models presented. Typically the accuracy of a model is described for some data set (usually in terms of Euclidean error), and an estimate of the profit generated by using the model forecasts and a trading strategy is provided. However, often the cost of trading (transaction costs) are not factored into this calculation. These addition costs, typically trading commission plus any taxation that may be relevant (e.g. stamp duty in the UK) can have a significant impact on realised profits [22]. As such the approach used in this paper is to include transaction costs in the training and final evaluation of ENN forecast models.

Due to the difficulty in predicting raw market time series for profitable day trading when transaction costs are taken into consideration, this application combines a number of time series in a novel transformation in order to make the forecasting task easier. This data transformation is an application specific one (determined by the trading strategy to be used), but has the additional benefit of creating a stationary time series for forecasting (other studies have also used data specific transformations, e.g. [54] and [41]).

### 3.2.2 Trading strategy

The trading strategy is dependent on the market/stock level falling during the day by at least 0.5% before buying (trading-in) - as described in Algorithm 4. The open value is therefore multiplied by 0.995 to obtain the realised purchase cost if buying due to the strategy occurs. The return measure is calculated using a simple trading strategy based upon transaction costs calculated at 0.1% of price (defined as a reasonable level in [57]), and therefore a minimum increase in price from buy to sell of 0.2% is needed before any profits can be realised. In addition, the trading strategy is designed such that a trade into the market will only take place if estimated profits beyond transaction costs after selling the next day equal approximately 1.5%. The measure is formally described in Algorithm 4.

The strategy is therefore to trade into the market on day $t$ when the level drops by 0.5% of the open value and sell the following day when the level rises 1.7% above the purchase level (1.2% above the open of the previous day), accruing 1.5% profit (including transaction costs of 0.1% each way). If the level does not fall by at least 0.5% then the initial trade does not occur and the capital is invested overnight in a 'risk-free' asset (i.e. a bank deposit earning 0.016% - equivalent to 4% per annum). If the initial trade has occurred and the level does not rise 1.7% above the purchase level the following day, trade out of the market occurs at the market close of day $t+1$ and profit/loss (including transaction costs) occurs. The predicted series $\varsigma^t$ is a composite series based upon this trading strategy, and is described below:

$$\varsigma^t = \left( \frac{b_h^t}{0.995 b_o^{t-1}} \right) \tag{8}$$

$$\text{if } b_l^{t-1} > 0.995 b_o^{t-1},\ \varsigma^t := 1.00016 \tag{9}$$

$$\text{else if } \varsigma^t \geq 1.017,\ \varsigma^t := 1.017 \tag{10}$$

$$\text{else } \varsigma^t := \left( \frac{b_c^t}{0.995 b_o^{t-1}} \right) \tag{11}$$

where $b_o^t$ is the open level of the market at day $t$, $b_h^t$ is the market high at day $t$, $b_l^t$ is the market low at day $t$, and $b_c^t$ is the market close at day $t$. Thus $\varsigma^t$ exactly encapsulates the profit/loss of this trading strategy (excluding the transaction costs).

A visual example of this is given in Figure 8, which shows the open level of the Japanese Nikkei 225 index over the past 17 years, and its corresponding $\varsigma^t$ transformation.

### 3.2.3 Experimental details

In order to use the trading strategy introduced, a model is needed to produce a prediction of the time series $\varsigma^t$, $\widehat{\varsigma}^t$.

A completely adaptive topology is used in this section, allowing heterogeneous topologies to be maintained by

**Algorithm 4** Trading strategy (return objective).

1: $t$, current time step (day).
2: $\widehat{\varsigma}^t$, the model forecast at day $t$.
3: $q_c^t = \frac{b_c^t}{0.995 b_o^{t-1}}$, where $b_c^t$ is the market close on day $t$.
4: $\varepsilon_{Rt}^t$, Return value at time $t$ (as a percentage of capital at $t-1$).
5: Set $t := 1$, first trading day of train (or test) set instance.
6: If $\left(\widehat{\varsigma}^{t+1} \geq 1.017\right) \bigwedge \left(x_l^{t-1}/x_o^{t-1} \leq 0.995\right)$ shift capital from risk free deposit into market at the point where the market price falls to 99.5% of open (incurring transaction costs), go to 7, otherwise go to 8.
7: $t := t+1$, Calculate profit/loss.
8: If $(\varsigma^t \geq 1.017)$, sell when market reaches the level 101.7% of that when entered, accrue return minus transaction cost, go to 2. Else:
9: If $(\varsigma^t < 1.017)$, sell at the end of day, $\varepsilon_{Rt}^t = (q_c^t - 1) - (0.1 + 0.1 q_c^t)$, go to 6.
10: Calculate nominal risk free interest accrued on assets, $\varepsilon_{Rt}^t = 0.0016$ (compound equivalent to 4% p.a.) , $t := t+1$, go to 6.
10: Halt process when end of train (or test) set is reached.

the estimated Pareto set of NNs. The model inputs are defined as follows:

$$v_t^{1,\ldots,10} = \varsigma_{t-2}, \ldots, \varsigma_{t-11} \tag{12}$$

variables 1 to 10 contain the last 10 lagged realised values of $y^t$ (2 weeks of trading). $\varsigma^{t-1}$ cannot be used as it incorporates information that will not be available at the start of day at $t-1$ (the index high of day $t-1$).

The fist 80% of each of the data sets was used for training ENNs for that financial index, and the final 20% for testing. The $\mathcal{M}_B$ of the previous section was used as the training algorithm.

Each run was initialised with 200 random ENNs. The algorithm parameters are shown in Table 1. The number of evaluated generations was 25000. Maximum ENN representation was a 10:10:1 network.

### 3.2.4 Results and interpretation

Graphical examples of estimated Pareto fronts defined by the archive of ENNs on the test data are provided in Figures 9 - 11. Figure 9 shows the estimated Pareto front generated for a market from the Americas group (the S&P 500 index), Figure 9 shows the estimated Pareto front generated for a market from the Asia/Pacific group (the Nikkei 225 index), and Figure 9 shows the estimated Pareto front generated for a market from the European group (the FTSE 100 index).

These figures show that, although there is some degree of noise present, the general shape and properties of ENN models fitted on the training data is consistent with their performance on the training data.

Figure 12 illustrates the risk and return experienced by different models on the Pareto front and their corresponding performance on the test data. Three models are taken as exemplars from the archives, those that lie on either extreme of the front, and the mid set member, for each of the 37 different stock indices. The performance of

the profit maximising extreme ENN is denoted by plus signs in Figure 12, joined by dots. The performance of the mid set members are denoted by triangles and the performance of the risk minimising ENNs by circles. The market performance is shown with a solid line, and the random walk model performance is shown with a dashed line (in this case the random walk model uses the trading strategy based on its prediction that $\varsigma^{t+1} = \varsigma^{t-1}$)[3]. The profit maximiser model consistently produces higher rate of return than the other models across both training data and test data, and the market rate experiences the highest volatility of return across the data. The relative performance of the five different models can be more clearly seen in Figure 13. Here boxplots of the results of the five models over training and test sets for the two objective measures are shown. Using the Wilcoxon signed ranks test (two tailed at the 0.05 significance level), the higher return rates of model 'A' are found to be significantly better than all of the other models, with the next best model being the random walk, followed by model 'B'. Over the training data the market return is higher than the MOENN operating point 'C', however on the test data, 'C' (effectively keeping the capital in the bank and not trading) is significantly better than investing in the markets (which isn't surprising given recent market performance).

- The volatility of the market return was found to be significantly higher than all the other models, followed by 'A', the random walk and 'B'.
- The three different operating points of the MOENN results were consistent across the test data, with 'A' having higher return followed by 'B' and then 'C', and 'C' having the lowest volatility, followed by 'B' and 'A'.
- Model 'A' was the best performing model (in terms of return) out of the five used, and produced consistent positive returns on all markets (as shown by the box-plots), even when some of the markets were experiencing significant downward trends. Mean values and standard deviations are shown in Table 3.

The fact that consistent profits can be observed using the random walk approach is interesting. This may be due to the trading strategy itself restricting the random walk model to trades where the market is volatile (which is its aim). However it is conceivable that the transaction costs modelled may be too low for certain markets. If this second case is true, it should be noted that even with transactions cost raised in order to make the random walk model unprofitable, model 'A' will still experience significant positive returns beyond both the market and bank returns.

Figure 14 illustrates the wide range of ENN architectures generated in the non-dominated set of models returned by the Pareto-ENN process. Figure 14a shows a Hinton plot of the weights of the 84 ENNs residing in the archive of the Nikkei 225 Pareto-ENN, each column representing a different ENN (each white square denoting an active

[3]The random walk model of $\varsigma^{t+1} = \varsigma^{t}$ cannot be used as $\varsigma^{t}$ is unknown at the start of day $t$ and can only be calculated at the close of trade on day $t$ (as it uses the high at day $t$).

weight, and each empty square denoting a disconnected weight). The weights are ordered such that $w_1, \ldots, w11$ represent the weights from the 10 inputs nodes and the bias to the first hidden node, $w_{12}, \ldots, w23$ represent the same for the second hidden node and so on. $w_{110}, \ldots, w121$ represent the weights between the hidden layer and the output node, and the output bias. As it can be seen, a wide range of weights and degrees of connectivity are used, from 75 active weights (the 81st ENN) to 121 active weights (the 83rd ENN). Figure 14b shows which inputs were used by each of the ENNs (with some ENNs using all the available inputs, and others using as few as 6) and Figure 14c shows which hidden units when used by each of the ENN set members (with between 7 and 10 hidden units used).

### 3.3 Further efficiency tests: Comparison with single objective ES runs

As described earlier, another potential solution to this problem is to run a single objective optimiser many times on a problem keeping the other objective(s) 'fixed' (i.e. constraining acceptable solutions to have the other objective(s) equal to or below/above a pre-defined value). It has been argued previously that this is inefficient in time resources, however this statement will be empirically investigated here. To facilitate this a uni- objective optimiser for the previous finance problem was formulated. Here the optimiser (again a (1+1)-ES) is solely concerned with optimising return, for a maximum acceptable risk level. The perturbation techniques and probabilities are identical to those used in the MOENN previously, and again the bootstrap generalisation method is used to prevent over-fitting. For each of the 37 finance problems five single objective ENNs were trained, with the constraint of risk being no higher than 0.2, 0.4, 0.6, 0.8 and 1.0, for a maximum of 25000 generations (meaning 185 separate optimiser runs). The optimiser is described in Algorithm 5.

Each run was initialised with 200 random networks. The algorithm parameters were:

- Probability of weight perturbation = 0.2
- Probability of individual weight elimination = 0.02
- Probability of individual node elimination = 0.02
- Perturbation $\sim N(0, 0.1)$
- Initial weights $\sim N(0, 0.1)$
- Generations = 25,000

The progress of the elite ENN in objective space, $E_t$ in Algorithm 5, was recorded every 250 generations and compared to the set of ENNs discovered by the MOENN after 25000 generations, the generation at which the single objective ENN is no longer dominated by member(s) of the saved front $F$ being marked. For each test problem,

**Algorithm 5** Implementation of the uni-objective ES NN optimiser (bootstrap training approach), for the finance problem.

| | |
|---|---|
| Input: | $M$, size of initial random population of solutions. Each solution chromosome $\mathbf{x}$ representing the weights and topology of a NN model.<br>$n$, the number of bootstrap subsets generated from the original training sets.<br>$s$, the size of the bootstrap subsets.<br>$r$, the maximum risk allowed by a model. |
| Output: | A single NN which estimates the maximum return possible given $r$. |
| 1: | **Initialisation:** Generate $n$ bootstrap subsets of the training data of size $s$. Generate random NN population of size $M$, such that each parameter (weight) of the ENNs $\sim N(0, \alpha)$, and the binary part of the chromosome is either initialised at 1 or $\sim U(0, 1)$. Generate the empty elite individual $E_0 = \emptyset$. Update $E_0$ with the fittest solution from the random population, with respect to the chosen error term (using a solution's worst term over the $n$ subsets), constrained that the highest risk on the subsets is lower than $r$. Initialise generation counter $t := 0$. |
| 2: | $X_t^* = E_t$. |
| 3: | **Genetic Recombination**: As in $\mathcal{M}_B$. |
| 4: | **Fitness Assignment**: Evaluate the ENN(s) $X_t^*$ with respect to return on the training subsets presented. If the return is greater than that of $E_t$, and risk is less than $r$, go to 5, otherwise go to 6. |
| 5: | $E_{t+1} := X_t^*$ |
| 6: | **Looping**: Iterate epoch count, $t := t + 1$. If stopping criteria have not been met then go to 2, else terminate algorithm and save $E_t$ for evaluation on test data. |
| 7: | end |

if the average number of generations needed for the single objective optimiser to train an ENN that is not *behind* $F$ is less than $25000/|F|$ then there is no efficiency benefit to using the MOENN training regime. If however the average number of generations needed is greater than $25000/|F|$ there are tangible efficiency gains, to which a value can even be assigned.

### 3.3.1 Results

Results from the different runs are shown in Table 4

On all bar two of the 37 test problems the MOES is shown to be more efficient than the uni-objective optimiser - performing 22 times better on average (i.e. for the uni-objective optimiser to find the same points in $|F|$ it would need to perform 22 times more function evaluations). This indeed may even be an underestimate - as nearly 40% of the 185 uni-objective optimiser runs did not reach the front found by the MOES within 25000 generations (an example of this is shown in Figure 15).

The implication of this is that the formulation of nominally uni-objective problem as a multi-objective problem can actually improve the search process - as indeed has been previously postulated (e.g. Knowles *et al.* [37], Jensen [33], and Abbass & Deb [4]). This is most likely due to the synergies present in multi-objective search. The evolution of a point in one area of objective space, through its decision space parameters, may lead to its shifting

to a different area of objective space. In multi-objective optimisation this solution is stored if it is non-dominating with other solutions in the area it has moved to. In the case of uni-objective optimisation these kind or fortuitous movements are not easily sustained as a movement in decision space is purely defined as better or worse - therefore the uni-objective formulation may be more likely to be caught in local optima.

# 4 Discussion

As described in Section 1, a single composite error term cannot be meaningfully propagated through a NN in a multi-objective application. The hybrid ENN methods used for example in [67] and highlighted in [66], where individual networks are trained at each generation using a gradient descent technique in addition to their evolutionary manipulation, are also infeasible for these reasons. Instead a general MOENN framework has been introduced in this study where ENN parameters are entirely adjusted by EC methods, and a model derived, the Pareto-ENN, which has demonstrated a number of significant results. The first set of results were on the area of multi-objective ENN generalisation. Training in [40] and [24] was simply terminated after a fixed number of epochs. Problems of network generalisation, under/over-fitting and validation were not addressed. In this study these concerns have been addressed through the introduction of two new techniques for MOENN training, one of which, bootstrap training, has been shown to improve MOENN generalisation performance significantly.

The Pareto-ENN model has also been applied in the domain of regression analysis, where multiple objectives are especially of interest. In one general training procedure a population of ENN models have been, through genetic recombination and mutation, manipulated with respect to their weights, topographies, connectivities and inputs. At algorithm termination a set of heterogeneous MOENNs are available for use, that represent an estimate of the true error trade-off (the Pareto front) defined by the data generating process, as described by the training data.

The performance on 37 real-world data sets showed that a set of ENNs training using the described methodology can perform in a consist manner on unseen test data. This has been manifest in both the visual inspection of the trade-off fronts produced, and through statistical comparison of different operating points on training data, and their relative positions on test data. The technique is also shown to generate significant results when compared to other models in the financial domain, and to produce significant returns even when considering transaction costs. New methods to improve generalisation in MOENNs have also been introduced and compared on test sets from the multi-objective NN literature, with the approach based on the bootstrapping of training data found to be significantly better than the models compared.

Once a set of MOENNs has been generated, that lie upon an estimate of the Pareto surface in the error space, a practitioner not only gains knowledge with respect to the error interactions of their problem, but they also have an opportunity to select an individual model that represents their error trade-off preferences, or a group of models

if so desired.

ACKNOWLEDGEMENTS

Jonathan Fieldsend gratefully acknowledges support from the EPSRC, grant GR/R24357/01.

Table 1: Parameters of the Pareto-ENN used in the two empirical sections.

| | |
|---|---|
| Probability of weight perturbation | 0.2 |
| Probability of node deletion/addition | 0.02 |
| Probability of weight deletion | 0.02 |
| Perturbation | $\sim N(0,1) \times 0.1$ |
| Initial weights | $\sim N(0,1)$ |

Table 2: Stock index descriptions

| Country | Index | From | Until | Samples |
|---|---|---|---|---|
| Americas | | | | |
| Argentina | MerVol | 24-10-1996 | 07-02-2003 | 1538 |
| Brazil | Bovespa | 13-05-1993 | 07-02-2003 | 2408 |
| Canada | S&P TSX Composite | 22-08-1984 | 07-02-2003 | 4647 |
| Mexico | IPC | 29-05-2001 | 07-02-2003 | 416 |
| Peru | Lima General | 15-05-1998 | 07-02-2003 | 1169 |
| USA | S&P 500 | 25-11-1983 | 07-02-2003 | 4845 |
| Asia & Pacific | | | | |
| Australia | All Ordinaries | 28-08-1984 | 07-02-2003 | 4659 |
| China | Shanghai Composite | 21-07-1997 | 07-02-2003 | 1329 |
| Hong Kong | Hang Seng | 18-12-1990 | 07-02-2003 | 3000 |
| India | BSE 30 | 17-07-1997 | 07-02-2003 | 1368 |
| Indonesia | Jakarta Composite | 18-07-1997 | 07-02-2003 | 1336 |
| Japan | Nikkei 225 | 19-09-1986 | 07-02-2003 | 4035 |
| Malaysia | KLSE Composite | 21-12-1993 | 07-02-2003 | 2247 |
| New Zealand | NZSE 40 | 06-10-1992 | 07-02-2003 | 2590 |
| Pakistan | Karachi 100 | 23-07-1997 | 07-02-2003 | 1322 |
| Philippines | PSE Composite | 18-07-1997 | 07-02-2003 | 1376 |
| Singapore | Straits Times | 10-07-1997 | 07-02-2003 | 2651 |
| South Korea | Seoul Composite | 18-07-1997 | 07-02-2003 | 1357 |
| Sri Lanka | All Share | 15-10-1998 | 07-02-2003 | 1032 |
| Thailand | SET | 18-07-1997 | 07-02-2003 | 1362 |
| Taiwan | Taiwan Weighted | 18-07-1997 | 07-02-2003 | 1351 |
| Europe | | | | |
| Austria | ATX | 27-11-1992 | 07-02-2003 | 2517 |
| Belgium | BEL-20 | 01-07-1992 | 07-02-2003 | 2587 |
| Czech Republic | PX50 | 16-07-1999 | 07-02-2003 | 875 |
| Denmark | KFX | 11-02-1993 | 07-02-2003 | 2499 |
| France | CAC 40 | 19-03-1990 | 07-02-2003 | 3231 |
| Germany | DAX | 12-12-1990 | 07-02-2003 | 3048 |
| Greece | General Share | 13-05-1998 | 07-02-2003 | 1182 |
| Italy | MIBTel | 04-08-1993 | 07-02-2003 | 2391 |
| Netherlands | AEX General | 28-10-1992 | 07-02-2003 | 2604 |
| Russia | Moscow Times | 15-05-1998 | 07-02-2003 | 1125 |
| Spain | Madrid General | 27-05-1999 | 07-02-2003 | 857 |
| Sweden | Stockholm General | 17-01-2001 | 07-02-2003 | 505 |
| Switzerland | Swiss Market | 27-11-1990 | 07-02-2003 | 3031 |
| Turkey | ISE National 100 | 18-07-1997 | 07-02-2003 | 11321 |
| U.K. | FTSE 100 | 18-04-1984 | 07-02-2003 | 4750 |
| Africa and Middle East | | | | |
| Israel | TA-100 | 25-05-1998 | 07-02-2003 | 931 |

Table 3: Mean risk and return over the 37 international indices. Results shown for the extreme and mid archived ENNs the market performance and from the random walk model (standard deviations in parenthesis).

| | Train | | Test | |
|---|---|---|---|---|
| | Risk | % Ret | Risk | % Ret |
| Risk Averse (archive ENN) | 0.0000 (0.0000) | 0.0160 (0.000) | 0.0000 (0.0000) | 0.0160 (0.000) |
| Middle Way (archive ENN) | 0.1923 (0.0882) | 0.0475 (0.0285) | 0.2949 (0.2025) | 0.0366 (0.0330) |
| Profit Maximiser (archive ENN) | 0.6570 (0.3526) | 0.1494 (0.1317) | 0.7573 (0.3551) | 0.1142 (0.1375) |
| Market | 1.7533 (0.7917) | 0.0416 (0.0941) | 1.6207 (0.6864) | -0.0176 (0.1061) |
| Random Walk | 0.5142 (0.2782) | 0.0544 (0.0711) | 0.5137 (0.2634) | 0.0733 (0.1037) |

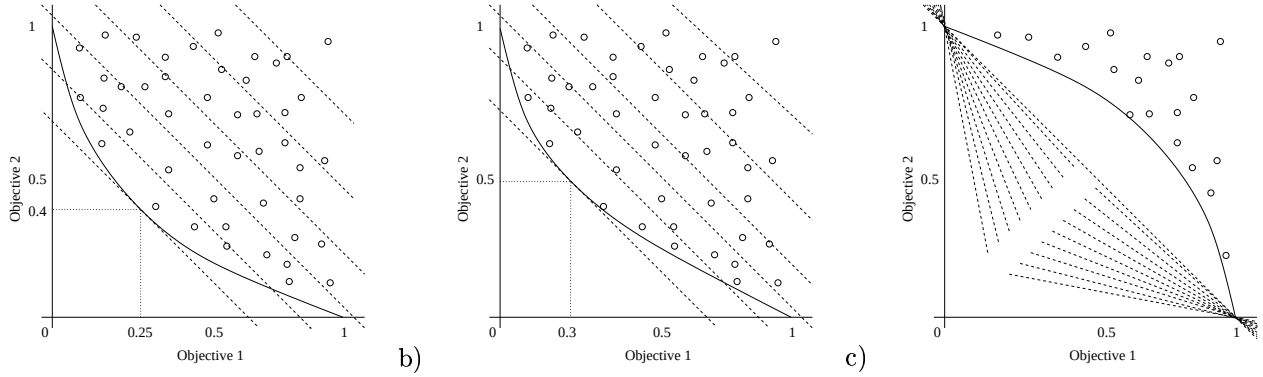


Figure 1: **Left**: Two dimensional error surface 1. Suboptimal models denoted by circles. The optimal model returned by equal weighting of the errors highlighted at the tangent point. **Middle**: Two dimensional error surface 2. **Right**: Example the effect of composite weighting when the front is convex with respect to the origin. Irrespective of weights given to the respective errors, the optimal model returned will only be one of the extreme optimal solutions.

Table 4: Results comparing uni-objective and multi-objective optimiser. $G^U$ is the average number of generations taken by the uni-objective to reach the estimated Pareto front found by the MOES. $G^M$ is the number of generations per non-dominated point on the estimated Pareto front found by the MOES. The ratio of the two values, $G^U/G^M$, gives an approximation as to how much more efficient the MOES is at finding estimated Pareto solutions from a given set than the uni-objective ES.

| Financial Index | $G^U$ | $G^M$ | $G^U/G^M$ |
|---|---|---|---|
| MerVol | 10650 | 892.9 | 11.9 |
| Bovespa | 1250 | 781.3 | 1.6 |
| S&P TSX Composite | 4200 | 675.7 | 6.2 |
| IPC | 16850 | 263.2 | 64.0 |
| Lima General | 25000 | 1388.9 | 18.0 |
| S&P 500 | 1900 | 342.5 | 5.5 |
| All Ordinaries | 25000 | 925.9 | 27.0 |
| Shanghai Composite | 11700 | 384.6 | 30.4 |
| Hang Seng | 3200 | 471.7 | 6.7 |
| BSE 30 | 15950 | 396.8 | 40.2 |
| Jakarta Composite | 14350 | 609.8 | 23.5 |
| Nikkei 225 | 1950 | 625.0 | 3.1 |
| KLSE Composite | 2950 | 581.3 | 5.0 |
| NZSE 40 | 500 | 925.9 | 0.5 |
| Karachi 100 | 13300 | 431.0 | 30.8 |
| PSE Composite | 7950 | 568.1 | 14.0 |
| Straits Times | 24350 | 1086.9 | 22.4 |
| Seoul Composite | 1150 | 595.2 | 1.9 |
| All Share | 25000 | 1388.9 | 18.0 |
| SET | 17100 | 500.0 | 34.2 |
| Taiwan Weighted | 10950 | 675.7 | 16.2 |
| ATX | 13600 | 757.6 | 18.0 |
| BEL-20 | 25000 | 581.4 | 43.0 |
| PX50 | 20100 | 1086.9 | 18.5 |
| KFX | 25000 | 806.5 | 31.0 |
| CAC 40 | 16750 | 294.1 | 57.0 |
| DAX | 25000 | 531.9 | 47.0 |
| General Share | 10600 | 609.8 | 17.4 |
| MIBTel | 18050 | 657.9 | 27.4 |
| AEX General | 15000 | 333.4 | 45.0 |
| Moscow Times | 750 | 1562.5 | 0.5 |
| Madrid General | 11150 | 735.3 | 15.2 |
| Stockholm General | 7150 | 609.8 | 11.7 |
| Swiss Market | 20550 | 362.3 | 36.7 |
| ISE National 100 | 9700 | 347.2 | 27.9 |
| FTSE 100 | 2350 | 925.9 | 2.5 |
| TA-100 | 11700 | 454.5 | 25.7 |

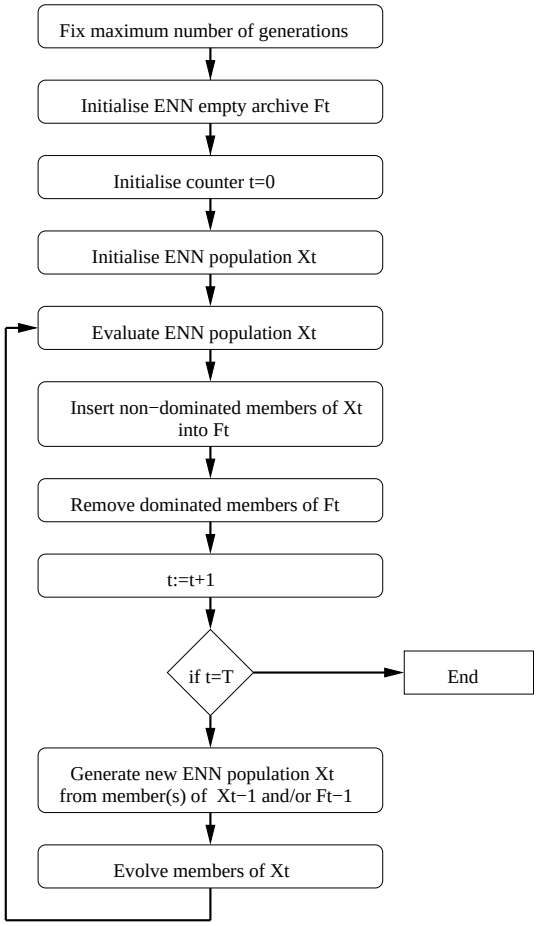


Figure 2: Flow diagram of the general MOENN framework.

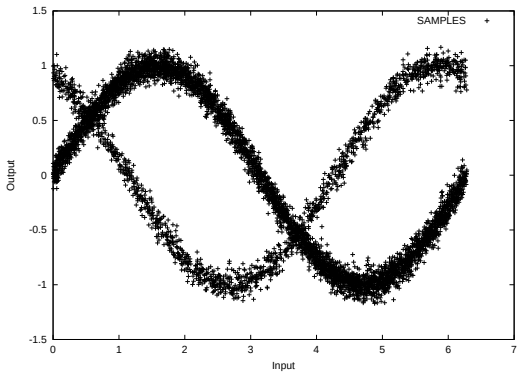


Figure 3: Lo and Bassu's test function.

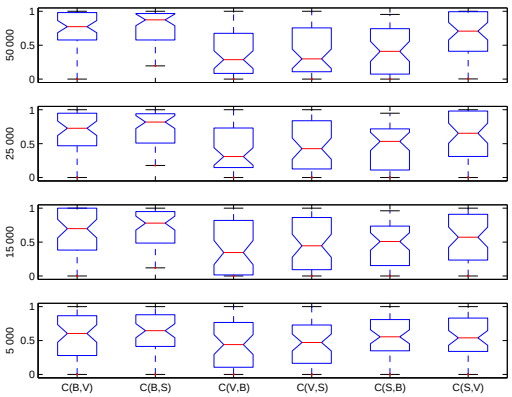


Figure 4: Boxplot of the pairwise $\tilde{C}$ measure between the three training methods, for 5000, 15000, 25000 and 50000 generations.

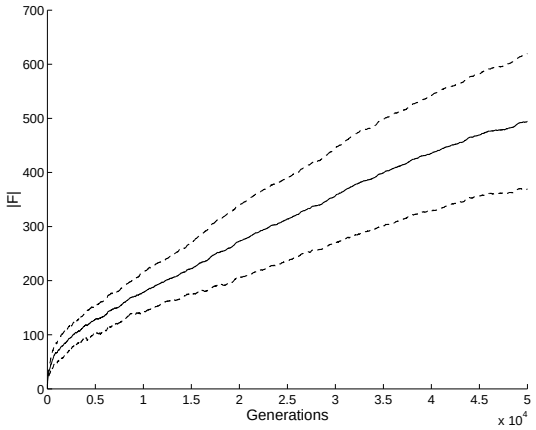


Figure 5: Plot of the mean |F| of the bootstrap training method across the 50 folds. The dashed lines denote this vale +/- 1 standard deviation.

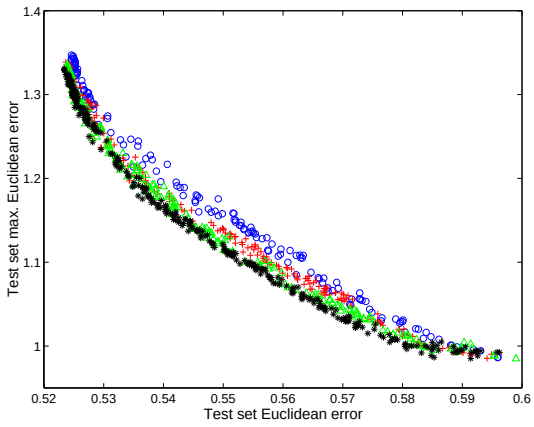


Figure 6: Plot of the archive of models $F$, evaluated on the test data from the bootstrap training method for a sigle fold. $F$ after 5000 generations is denoted by circles, after 15000 by pluses, after 25000 by triangles and after 50000 by stars. The difference between 25000 and 50000 is small, indicating that the search process by that point converging.

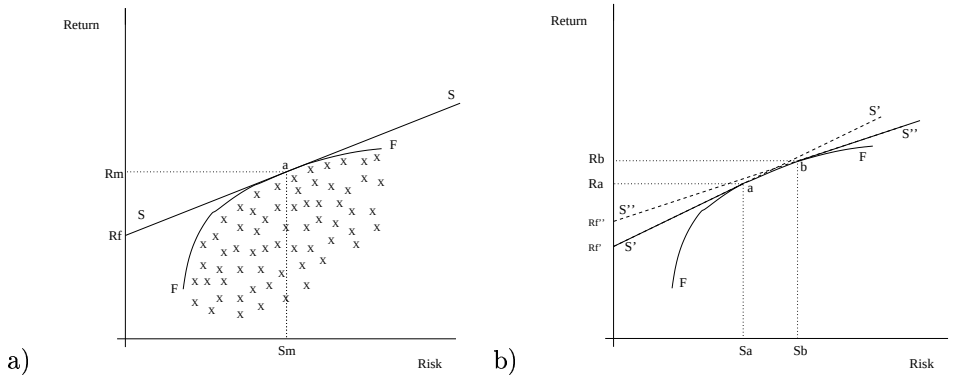


Figure 7: **Left**: The CAPM model. Pareto front defining trade-off between Profit and Risk in a Portfolio of stocks, and also in relation to a prediction model genus with various model parameters. **Right**: Two risk free rates of interest in the CAPM model (and forecast model analogy).

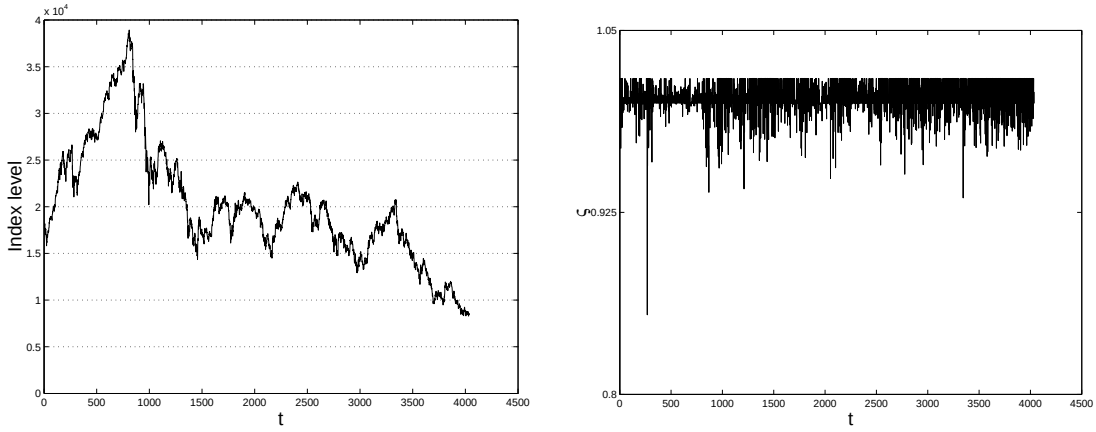


Figure 8: **Left**: The Nikkei 225 index (open level). **Right**: The $\varsigma_t$ transformation of the Nikkei 225, as described in Equations 8-11.

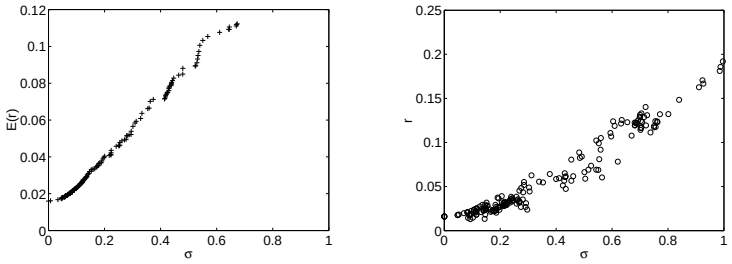


Figure 9: Risk and return on the S&P 500 index (80% train, 20% test). **Left:** Training Pareto front. **Right:** Testing estimated Pareto front.

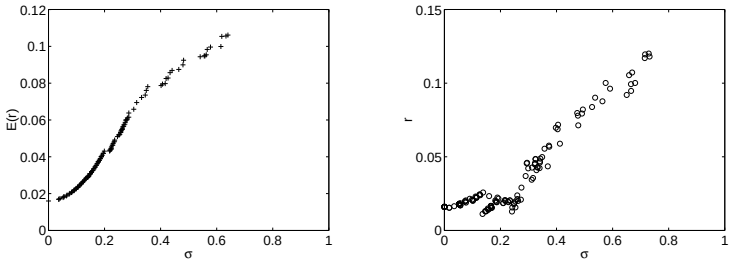


Figure 10: Risk and return on the Nikkei 225 index (80% train, 20% test). **Left:** Training Pareto front. **Right:** Testing estimated Pareto front.

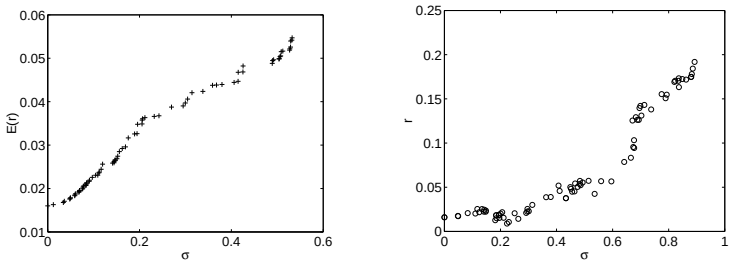


Figure 11: Risk and return on the FTSE 100 index (80% train, 20% test). **Left:** Training Pareto front. **Right:** Testing estimated Pareto front.

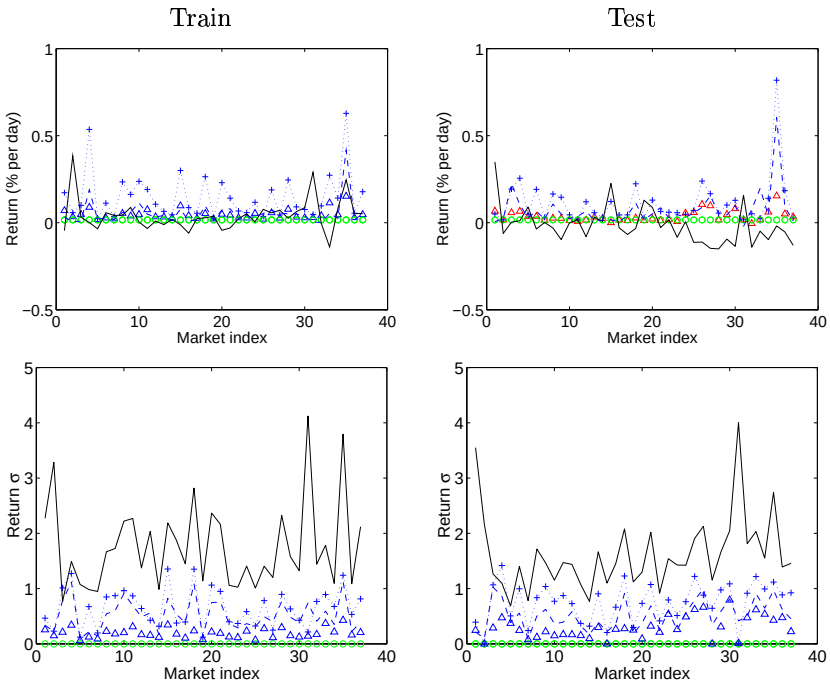


Figure 12: Risk and return for 3 different exemplar members of the archived ENNs for the 37 international indices (indices ordered as in Table 2). Market performance and performance of the random walk model using the same trading strategy are also shown.

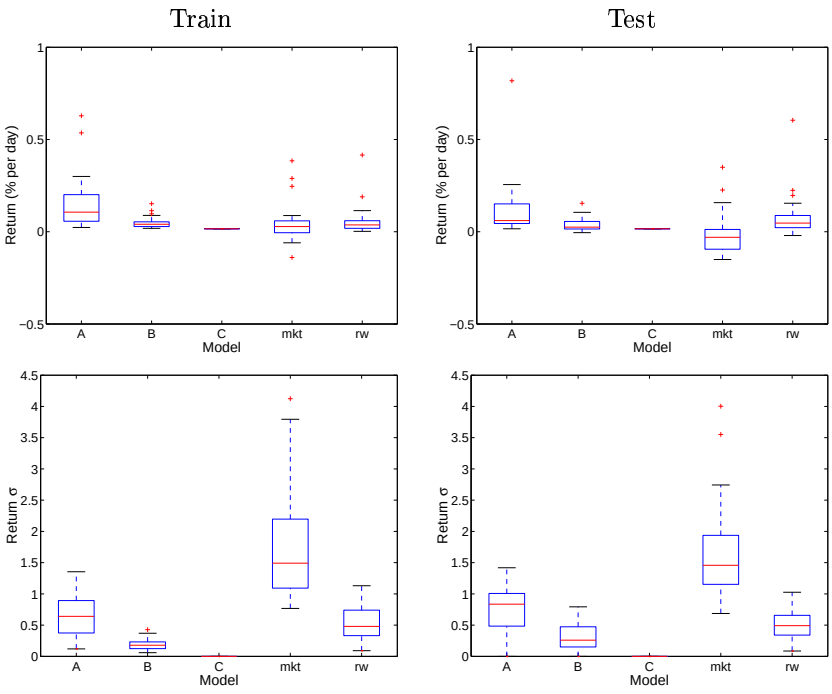


Figure 13: Boxplots of the realised risk and return for 3 different exemplar members of the archived ENNs across the 37 international indices. Market performance and performance of the random walk model using the same trading strategy are also shown.

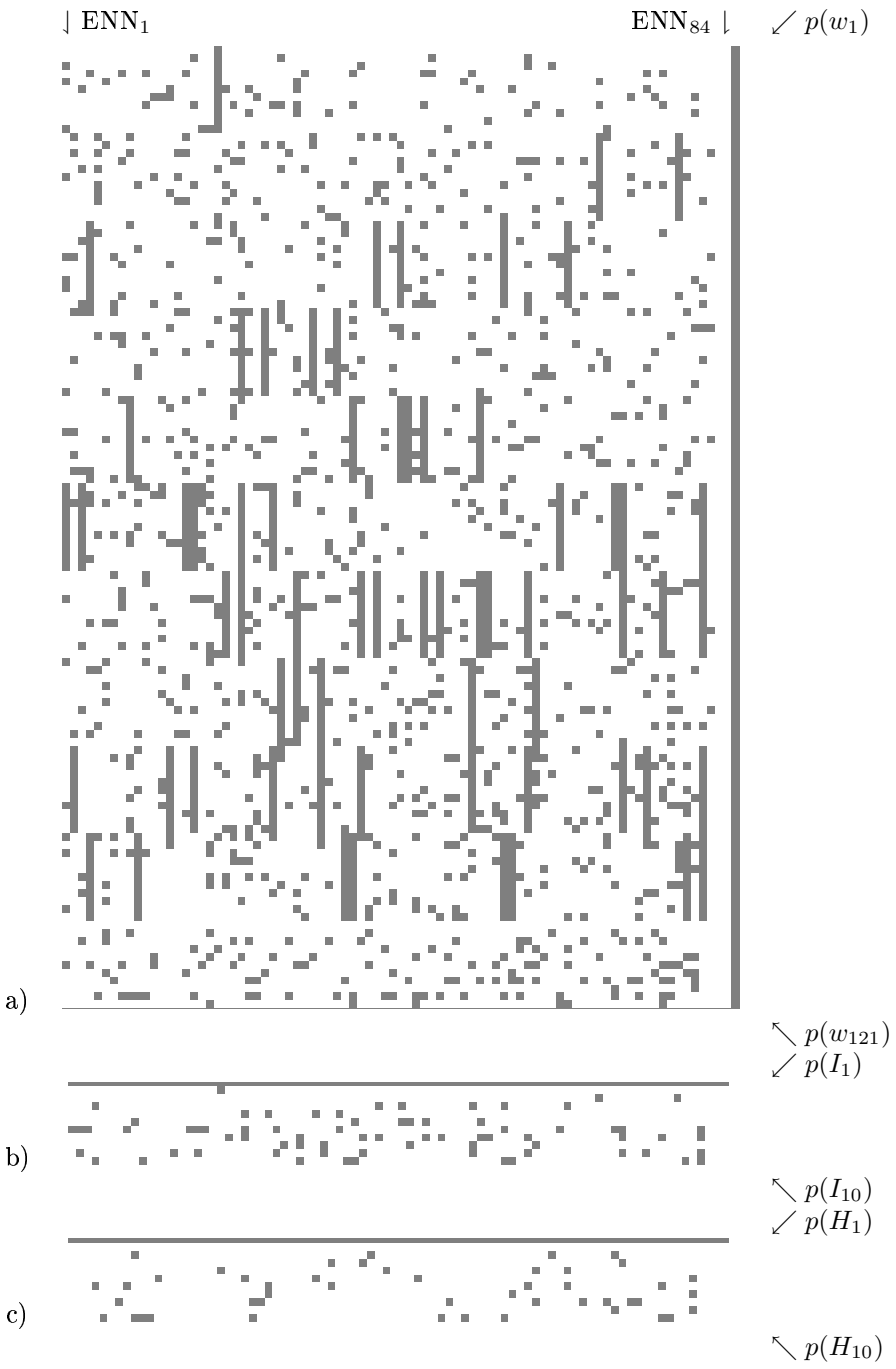


Figure 14: Example of the range of ENN topographies on an estimated Pareto front. Hinton plots are shown for the ENN weights (a), input topography (b) and hidden topography (c) of the 84 ENNs lying on the estimated Pareto error surface for the Nikkei 225 data.

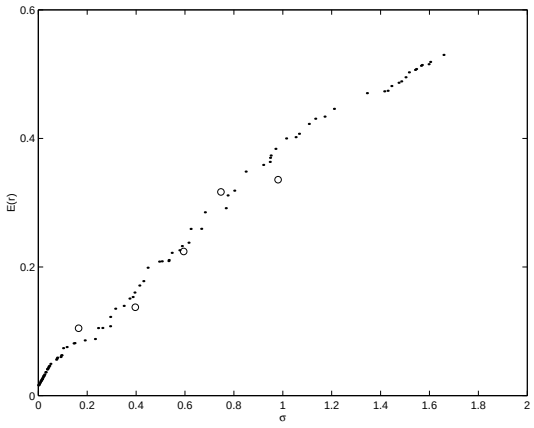


Figure 15: Risk and return on the Lima General index training data, the models found by the MOES after 25000 generations plotted as points, the models found by the uni-objective ES with 5 different risk maximums plotted as circles, again after 25000 generations.

# References


[1] H. Abbass and R. Sarker. Simultaneous evolution of architectures and connection weights in anns. In *Artificial Neural Networks and Expert Systems Conference*, pages 16–21, Dunedin, New Zealand, 2001.

[2] H.A. Abbass. A Memetic Pareto Evolutionary Approach to Artificial Neural Networks. In *The Australian Joint Conference on Artificial Intelligence*, pages 1–12. Springer, 2001.

[3] H.A. Abbass. An evolutionary artificial neural networks approach for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 25(3):265–281, 2002.

[4] H.A. Abbass and K. Deb. Searching under multi-evolutionary pressures. In *Proceedings of the 2003 Evolutionary Multiobjective Optimization Conference (EMO03)*, pages 391–404. Springer-Verlag, 2003.

[5] M. Adya and F. Collopy. How Effective are Neural Networks at Forecasting and Prediction? A Review and Evalution. *International Journal of Forecasting*, 17:481–495, 1998.

[6] E. Alba, J.F. Aldana, and J.M. Troyla. Full Automatic ANN Design: A Genetic Approach. *Lecture Notes in Computer Science*, 686:399–404, 1993.

[7] U. Anders, O. Korn, and C. Schmitt. Improving the Pricing of Options: A Neural Network Approach. *Journal of Forecasting*, 17:369–388, 1998.

[8] J.S. Armstrong and F. Collopy. Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, 8(1):69–80, 1992.

[9] S. Baluja. Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller. *IEEE Transactions on Systems Man and Cybernetics - Part B: Cybernetics*, 26(3):450–463, 1996.

[10] T. Bera and L. Higgins. ARCH Models: Properties, Estimation and Testing. *Journal of Economic Surveys*, 7(4):305–362, 1993.

[11] R.A. Brealey and S.C. Myers. *Principles of Corporate Finance.* McGraw-Hill, 5th edition, 1996.

[12] F. Brill, D. Brown, and W. Martin. Fast Genetic Selection of Features for Neural Network Classifiers. *IEEE Transactions on Neural Networks*, 3(2):324–328, 1992.

[13] N. Chaiyaratana and A.M.S. Zalzala. Hybidisation of neural networks and genetic algorithms for time-optimal control. In *Proceedings of the Congress on Evolutionary Computation*, pages 389–396, 1999.

[14] C.A.C Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, 1999.

[15] A.V.E. Conradie, R. Miikkulainen, and C. Aldrich. Adaptive Control utilising Neural Swarming. In *Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA*, pages 60–67, 2002.

[16] I. Das and J.Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1997.

[17] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[18] W. Duch. Alternatives to gradient-based neural training. In *Fourth Conference on Neural Networks and Their Applications, Zakopane, Poland*, pages 59–64, 1999.

[19] A. Engelbrecht and A. Ismail. Training product unit neural networks. *Stability and Control: Theory and Applications*, 2(1-2):59–74, 1999.

[20] R.M. Everson, J.E. Fieldsend, and S. Singh. Full Elite Sets for Multi-Objective Optimisation. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture V*, pages 343–354. Springer, 2002.

[21] J. Fang and Y. Xi. Neural Network design based on evolutionary programming. *Artificial Intelligence in Engineering*, 11:155–161, 1997.

[22] J.E. Fieldsend. *Novel Algorithms for Multi-Objective Search and their application in Multi-Objective Evolutionary Neural Network Training*. PhD thesis, Department of Computer Science, University of Exeter, June 2003.

[23] J.E. Fieldsend, R.M. Everson, and S. Singh. Using Unconstrained Elite Archives for Multi-Objective Optimisation. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323, 2003.

[24] J.E. Fieldsend and S. Singh. Pareto Multi-Objective Non-Linear Regression Modelling to Aid CAPM Analogous Forecasting. In *Proceedings of the 2002 IEEE International Joint Conference on Neural Networks, part of the 2002 IEEE World Congress on Computational Intelligence*, pages 388–393, Hawaii, May 12-17, 2002. IEEE Press.

[25] C.M. Fonseca and P.J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[26] D. Geigle and J. Aronson. An Artificial Neural Network Approach to the Valuation of Options and Forecasting of Volatility. *Journal of Computational Intelligence in Finance*, 7(6):19–25, 1999.

[27] J. González, I. Rojas, J. Ortega, H. Pomares, F.J. Fernández, and A.F. Díaz. Multiobjective evolutionary optimization of the size, shape and position parameters of radial basis function networks for function approximation. *IEEE Transactions on Neural Networks*, 14(6):1478–1498, 2003.

[28] G.W. Greenwoood. Training Multiple-Layer Perceptrons to Recognise Attractors. *IEEE Transactions on Evolutionary Computation*, 1(4):244–248, 1997.

[29] D. Gujarati. *Essentials of Econometrics*. McGraw-Hill, 1992.

[30] T. Hann and E. Steurer. Much ado about nothing? Exchange rate forecasting: Neural networks vs. linear models using monthly and weekly data. *Neurocomputing*, 10(4):323–340, 1996.

[31] P. Harrald and M. Kamstra. Evolving Artificial Neural Networks to Combine Financial Forecasts. *IEEE Transactions on Evolutionary Computation*, 1(1), 1997.

[32] A. Ismail and A. Engelbrecht. Training Product Units in Feedforward Neural Networks using Particle Swarm Optimization. In *Proceedings of the International Conference on Articial Intelligence, Durban, South Africa*, pages 36–40, 1999.

[33] M.T. Jensen. Guiding Single-Objective Optimization Using Multi-objective Methods. In S. Cagnoni, J.J. Romero Cardalda, D.W. Corne, J. Gottlieb, A. Guillot, E. Hart, C.G. Johnson, E. Marchiori, J.-A. Meyer, M. Middendorf, and G.R. Raidl, editors, *Applications of Evolutionary Computing: EvoWorkshops 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM, Lecture Notes in Computer Science*, number 2611, pages 268–276, Essex, UK, April 2003. Springer.

[34] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of the Fourth IEEE International Conference on Neural Networks*, pages 1942–1948, Perth, Australia, 1995. IEEE Service Center.

[35] J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 98–105, Piscataway, NJ, 1999. IEEE Service Center.

[36] J.D. Knowles and D. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[37] J.D. Knowles, R.A. Watson, and D.W. Corne. Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science*, number 1993, pages 269–283, Zurich, Switzerland, March 2001. Springer-Verlag.

[38] N. Kohzadi, M. Boyd, B. Kermanshahi, and I. Kaastrs. A Comparison of artificial neural network and time series models for forecasting commodity prices. *Neurocomputing*, 10:169–181, 1996.

[39] J.R. Koza and J.P. Rice. Genetic generation of both the weights and architecture for a neural network. In *Proceedings of IJCNN'92, Seattle IEEE/INNS*, volume II, pages 397–404, 1992.

[40] M.A. Kupinski and M.A. Anastasio. Multiobjective Genetic Optimization of Diagnostic Classifiers with Implications for Generating Receiver Operating Characterisitic Curves. *IEEE Transactions on Medical Imaging*, 18(8):675–685, 1999.

[41] B. LeBaron and A. Weigend. A bootstrap Evaluation of the Effect of Data Splitting on Financial Time Series. *IEEE Transactions on Neural Networks*, 9(1):213–220, 1998.

[42] J.T. Lo. Minimization through Convexitization in Training Neural Networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks, part of the IEEE World Congress on Computational Intelligence*, pages 1889–1894, Hawaii, May 12-17, 2002. IEEE Press.

[43] J.T. Lo and D. Bassu. Robust Approximation of Uncertain Functions where Adaptation is Impossible. In *Proceedings of the IEEE International Joint Conference on Neural Networks, part of the IEEE World Congress on Computational Intelligence*, pages 1956–1961, Hawaii, May 12-17, 2002. IEEE Press.

[44] M. Malliaris and L. Salchenberger. Using neural networks to forecast the S l& P 100 implied volatility. *Neurocomputing*, 10(2):183–196, 1996.

[45] V. Maniezzo. Genetic Evolution of the Topology and Weight Distribution of Neural Networks. *IEEE Transactions on Neural Networks*, 5(1):39–53, 1994.

[46] F.J. Marin and F. Sandoval. Genetic Synthesis of Discrete-Time Recurrent Neural Network. *Lecture Notes in Computer Science*, 686:179–184, 1993.

[47] J. McDonnell and D. Waagen. Evolving Recurrent Perceptrons for Time-Series Modeling. *IEEE Transactions on Neural Networks*, 5(1):24–38, 1994.

[48] J.J. Merelo, M. Paton, A. Canas, A. Prieto, and F. Moran. Optimization of a competitive learning neural network by Genetic Algorithms. *Lecture Notes In Computer Science*, 686:185–192, 1993.

[49] J. Moody. Forecasting the Economy with Neural Nets: A survey of Challenges and Solutions. In G.B. Orr and K-R Mueller, editors, *Neural Networks: Tricks of the Trade*, pages 347–371. Berlin: Springer, 1998.

[50] S. Moshiri and N. Cameron. Neural Network Versus Econometric Models in Forecasting Inflation. *Journal of Forecasting*, 19:201–217, 2000.

[51] S. Mostaghim, J. Teich, and A. Tyagi. Comparison of Data Structures for Storing Pareto-sets in MOEAs. In *Proceedings of the 2002 Congess on Evolutionary Computation, part of the 2002 IEEE World Congress on Computational Intelligence*, Hawaii, May 12-17, 2002. IEEE Press.

[52] V.W. Porto, D.B. Fogel, and L.J. Fogel. Alternative Neural Network Training Methods. *IEEE Expert*, 10(3):16–22, 1995.

[53] A-P.N. Refenes, A.N. Burgess, and Y. Bentz. Neural Networks in Financial Engineering: A Study in Methodology. *IEEE Transactions on Neural Networks*, 8(6):1222–1267, 1997.

[54] E.W. Saad, D.V. Prokhorov, and D.C. Wunsch. Comparitive Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks. *IEEE Transactions on Neural Networks*, 9(6):1456–1470, 1998.

[55] N. Saravanan and D.B. Fogel. Evolving Neural Control Systems. *IEEE Expert*, 10(3):23–27, 1998.

[56] J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 99–100, 1985.

[57] C. Schittenkopf, P. Tino, and G. Dorffner. The profitability of trading volatility using real-valued and symbolic models. In *IEEE/IAFE/INFORMS 2000 Conference on Compuational Intelligence for Financial Engineering (CIFEr)*, pages 8–11, 2000.

[58] Y. Shimizu. Multi-objective optimization for site location problems through hybrid genetic algorithm with neural networks. *Journal of Chemical Engineering of Japan*, 32(1):51–58, 1999.

[59] F. van den Bergh and A. Engelbrecht. Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal*, (26):84–90, 2000.

[60] D. Van Veldhuizen and G. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.

[61] Y. Wang and F.M. Wahl. Multiobjective neural network for image reconstruction. *IEE Proceedings - Vision, Image and Signal Processing*, 144(4):233–236, 1997.

[62] C-G. Wen and C-S Lee. A neural network approach to multiobjective optimization for water quality management in a river basin. *Water Resources Research*, 34(3):427–436, 1998.

[63] D. White. GANNet: A Genetic Algorithm for Optimizing Topology and Weights in Neural Network Design. *Lecture Notes in Computer Science*, 686:322–327, 1993.

[64] S.M. Yamany, K.J. Khiani, and A.A. Farag. Application of neural networks and genetic algorithms in the classification of endothelial cells. *Pattern Recognition Letters*, 18(11-13):1205–1210, 1997.

[65] J. Yao and C.L. Tan. Time dependant Directional Profit Model for Financial Time Series Forecasting. In *IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 2000.

[66] X. Yao. Evolving Artificial Neural Networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[67] X. Yao and Y. Liu. A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, 1997.

[68] X. Yao and Y. Liu. Making Use of Population Information in Evolutionary Neural Networks. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 28(3):417–425, 1998.

[69] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

# 5 Appendix

## 5.1 Pareto Optimality

The multi-objective optimisation problem seeks to simultaneously extremise $D$ objectives:

$$y_i = f_i(\mathbf{x}), \qquad i = 1, \ldots, D \tag{13}$$

where each objective depends upon a vector $\mathbf{x}$ of $n$ parameters or decision variables (in the case of regression modelling, these may represent the weights/topologies of a NN). Without loss of generality it is assumed that these objectives (also referred to as model errors in this study) are to be minimised. As such the problem can be stated as:

$$\text{Minimise} \qquad \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_D(\mathbf{x}))\,, \tag{14}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_D)$.

When faced with only a single error measure, an optimal solution is the one which minimises the error given the model constraints. However, when there is more than one non-commensurable error term to be minimised, it is clear that solutions exist for which performance on one error cannot be improved without sacrificing performance on at least one other. Such solutions are said to be *Pareto optimal* [60] and the set of all Pareto optimal solutions are said to form the Pareto front.

The notion of *dominance* may be used to make Pareto optimality more precise. A decision vector $\mathbf{u}$ (vector of model parameters) is said to *strictly dominate* another $\mathbf{v}$ (denoted $\mathbf{u} \prec \mathbf{v}$) if

$$f_i(\mathbf{u}) \leq f_i(\mathbf{v}) \quad \forall i = 1, \ldots, D \quad \text{and} \quad f_i(\mathbf{u}) < f_i(\mathbf{v}) \quad \text{for some } i. \tag{15}$$

Less stringently, $\mathbf{u}$ *weakly dominates* $\mathbf{v}$ (denoted $\mathbf{u} \preceq \mathbf{v}$) if

$$f_i(\mathbf{u}) \leq f_i(\mathbf{v}) \quad \forall i = 1, \ldots, D \tag{16}$$

A set of $M$ decision vectors $\{W_1, W_2, \ldots, W_M\}$ is said to be a *non-dominated set* (an estimate of the Pareto front) if no member of the set is dominated by any other member:

$$W_k \not\prec W_j \quad \forall j, k = 1, \ldots, M \tag{17}$$

## 5.2 Front comparison measures

The comparison of estimated Pareto fronts is difficult as there are several ways in which a front can be inferior or superior to another.

In this work the following modified version of the $\mathcal{C}$ measure is used:

$$\tilde{\mathcal{C}}\,(A, B) = \frac{|\{\mathbf{b} \in B : \exists\, \mathbf{a} \in A\,,\, \mathbf{a} \prec \mathbf{b}\}|}{|B|} \tag{18}$$

$\tilde{\mathcal{C}}$ measures the fraction of members of $B$ which are strictly dominated by members of $A$. As such it measures the quality of $A$ with respect to $B$. When $\mathcal{C}(A, B) = 1.0$, then all of the individuals in $B$ are dominated by solutions in $A$; when $\mathcal{C}(A, B) = 0.0$ none of the individuals in $B$ are dominated by any member of $A$. $\tilde{\mathcal{C}}(A, A) = 0$ and, in addition, it measures two mutually non-dominating sets as equivalent, i.e. if $A \subseteq W$ and $B \subseteq W$ are each subsets of a non-dominating set $W$, then $\tilde{\mathcal{C}}(A, B) = 0$.