Regression Error Characteristic Optimisation of Non-Linear Models

Jonathan E. Fieldsend

School of Engineering, Computer Science and Mathematics,
University of Exeter, Exeter, UK, EX4 4QF
J.E.Fieldsend@exeter.ac.uk

Summary. In this chapter recent research in the area of multi-objective optimisation of regression models is presented and combined. Evolutionary multi-objective optimisation techniques are described for training a population of regression models to optimise the recently defined Regression Error Characteristic Curves (REC). A method which meaningfully compares across regressors and against benchmark models (i.e. 'random walk' and maximum a posteriori approaches) for varying error rates. Through bootstrapping training data, degrees of confident out-performance are also highlighted.

This approach is then extending to encapsulate the complexity of the model as a third objective to minimise. Results are shown for a number of data sets, using multi-layer perceptron neural networks.

1 Introduction

When forecasting a time series there are often different measurements of the *quality* of the signal prediction. These are numerous and often problem specific [1]. Recent advances in regressor comparison has been concerned not solely with the chosen error measure itself, but also with the distributional properties of a regressor's error – this has been formulated in a methodology called Regression Error Characteristic (REC) curve, introduced by Bi & Bennett [2]. This curve traces out the proportion of residual errors of a model which lie below a certain error threshold, allowing the comparison of models given different error property preferences.

This chapter proceeds with the introduction of evolutionary computation techniques as a process for generating REC curves for regressor families, instead of for individual regressors, allowing the visualisation of the potential prediction properties for an entire class of method for a problem.

This approach is then augmented with the simultaneous optimisation of model complexity, with a similar framework to that introduced in [13]. The highlighting of regions of the REC curve on which we can confidently out-

perform the maximum *a posteriori* (MAP) trained model is also introduced, through bootstrapping the optimised REC curve.

The chapter proceeds as follows: REC curves are formally defined and there properties discussed in Section 2. A general model for multi-objective REC optimisation is introduced in Section 3, and the regression models to be optimised and compared are presented in Section 4.3. Empirical results are presented on a range of problems in Section 4, and the methodology is extended to include complexity trade-off in Section 5. The chapter ends with a final discussion of results in Section 6.

2 Regression Error Characteristic Curves

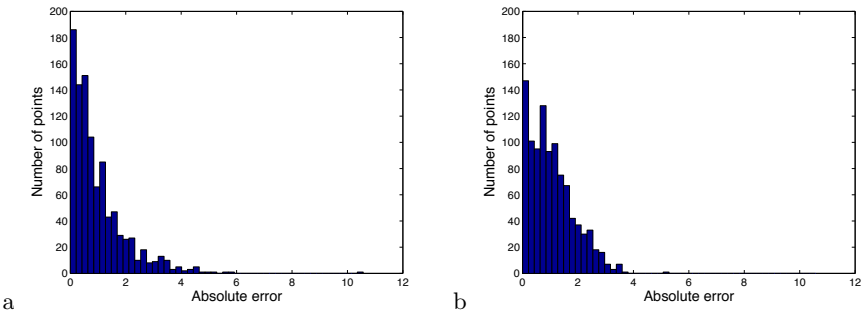
Receiver Operating Characteristic (ROC) curves have proved useful for a number of years as a method to compare classifiers when the costs of misclassification are *a priori* unknown. In the binary classification case it plots the rates of correct classification of one class against the misclassification rates of the other class, typically derived through changing the threshold/cost of a particular parameterised classifier. This formulation allows the user to see what range of classifications they can obtain from a model, and also allows them to compare models where misclassification costs are unknown by using such measures as the area under the curve (AUC) and the Gini coefficient. A more in depth discussion of ROC curves can be found in the chapter on ROC optimisation in this book.

Inspired by this, Bi & Bennett [2] developed the Regression Error Characteristic curve methodology to represent the properties of regression models. In a regression problem, the task is to generate a good estimate of a signal y_i (called the dependent variable), from a transformation of one or more input signals $\mathbf{x}_i$ (called independent variables), such that

$$\hat{y}_i = f(\mathbf{x}_i, \mathbf{u}). \tag{1}$$

$\hat{y}_i$ is the regression model prediction of y_i, given the data $\mathbf{x}_i$ and the model parameters $\mathbf{u}$. The optimisation process of regression models typically takes the form of varying $\mathbf{u}$ in order to make $\hat{y}_i$ as close to y_i as possible, where *closeness* is calculated through some error term (like Euclidean distance or absolute error). This error is calculated for all the n training data points used, $\boldsymbol{\xi} = error(\hat{\mathbf{y}}, \mathbf{y})$, and the average error, $\bar{\xi}$ typically used as the scaler evaluation of the parameters $\mathbf{u}$.

In REC, instead of dealing purely with the average error of a regressor, the entire error distribution is of interest. The proportion of points forecast below a certain error threshold are plotted against the error threshold, for a range of error thresholds (from zero to the maximum obtained error for a null regressor on a single point). This effectively traces out an estimate of the cumulative distribution function of the error experienced by a regressor, and



a b

Fig. 1. Distribution of residual errors of two regression models, *a: A* and *b: B*.

can be created by ordering $\boldsymbol{\xi}$ in ascending order, and plotting this against the element index divided by n.

There are many useful properties of this representation, for instance the area over the curve (AOC) is a good estimate of the expected error of the model. However, probably the most useful contribution of the REC formulation is the easy visualisation of error information about a model across the range of data used to train or test it. This gives information to the user which may lead them to select a model other than the one with the lowest average error.

2.1 Illustration

Let us start with a toy example where we have two regression models available to us, A and B, which are predicting some time series (for instance the demand of a product in the next month). If model A experiences, on average, an absolute error of 0.99 and model B an absolute error of 1.06, then without any additional information one would typically choose to use model A as it exhibits a lower mean error. Knowing the distributional properties of this error, however, may lead to a different decision.

Figure 2.1 shows an (illustrative) error distribution of the absolute errors of models A and B, Figure 2.1 in turn traces out the REC curves for A (solid line) and B (dashed line). This shows that although model A has a lower average error than model B, its largest errors (the top 15%) are proportionally bigger than that of model B – meaning it makes more extreme errors than model B. Given that the cost of extreme errors to the user of the forecast may be proportionally greater than small errors (small under predictions of demand can be taken up by inventoried goods, large under predictions may result in turning customers away), B may actually be preferred.[1]

[1] It should be noted in the original work by Bi and Bennett they recommended ranking models by the AOC – i.e. an estimate of the mean expected error. However this assumes proportional costs of error, which in many situations is not the case.

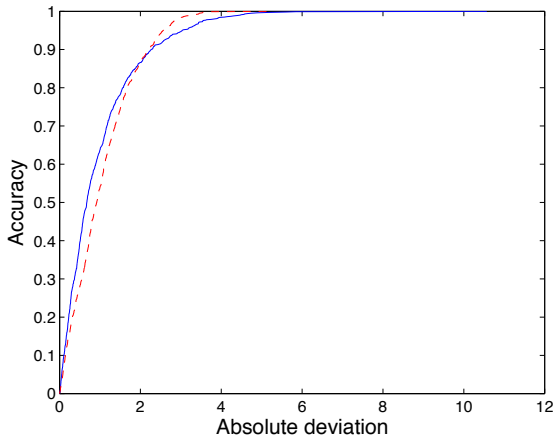


Fig. 2. REC curves of two regression models, A (solid) and B (dashed).

The final choice of model will depend upon the preferences and costs of the user of the model which may be difficult to incorporate in an optimisation algorithm when there is no *a priori* knowledge of the *shape* of the REC curves possible [5].

[2] used REC to compare different regressors trained to minimise the average error, and so were effectively interested in those models which minimised the AOC. When we are interested in a particular region of the REC curve (distributional property of our residual errors), minimising the AOC of a single model will not necessarily lead us to the best model given our preferences. However minimising the AOC of a *set* of models can.

Using the previous illustration, if we merge the two REC curves of models A and B, taking only the portions which are in front, we can create an REC curve which illustrates the possible error/accuracy combinations given the available models. The illustration in Figure 2.1 shows this composite REC curve along with the REC curves of two new models, C and D. As the REC of model C lies completely below the composite REC curve, we can see that for any possible error/accuracy combination models A and/or B are better than model C. Model D however is slightly in front of the composite REC for a small range of accuracy, and so would be useful to retain and offer to the end user as a possible model.

3 Multi-Objective Evolutionary Computation for REC

The generation of a composite REC curve to describe the possible error/accuracy combinations for a problem (given a model family or families)

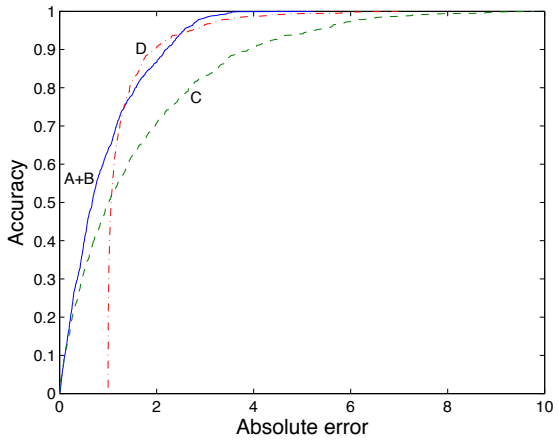


Fig. 3. Composite REC of models A (solid) and B (dashed), and REC curves of models C and D.

is easily cast in terms of multi-objective optimisation problem. However the casting itself can be in two different forms, which affect both the representation and the complexity of the optimisation process.

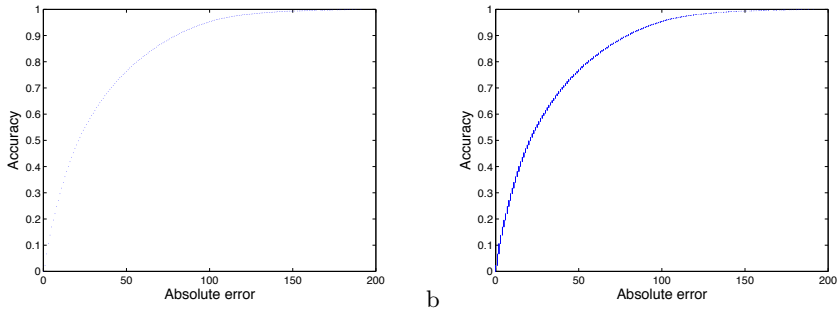
3.1 Casting as a 2 objective problem

The obvious representation of the REC optimisation problem is as a 2-objective problem, the first objective (error threshold) to be minimised and the second objective (accuracy) to be maximised. In this case a single parameterisation, $\mathbf{u}$, results in n error/accuracy pairs, which need to be compared to the current best estimate of the composite REC curve. Any values on the current best estimate which have higher error threshold for the same accuracy as $\mathbf{u}$ need to be removed as they are *dominated* and replaced by relevant the pair(s) from the evaluation of $\mathbf{u}$. Formulated in this fashion, $\mathcal{O}(n \log n)$ domination comparisons are needed for each parameter vector compared to the current best estimate of the Pareto front/composite REC curve.

We can instead however represent the problem as an n objective problem, which actually turns out to be faster.

3.2 Casting as an n objective problem

On calculating the error, $\boldsymbol{\xi}$, of two parameterisations $\mathbf{u}$ and $\mathbf{v}$, and arranging them in ascending order, we can use these n dimensional errors as the fitness vectors for $\mathbf{u}$ and $\mathbf{v}$. The accuracy term (the second objective in the previous formulation) always takes on the value of the index (over n) of the elements



a b

Fig. 4. Composite REC curves from casting the REC optimisation problem as a *a:* 2-objective or a *b: n*-objective one. Note that all the points in *(a)*, are included in *(b)* .

of $\boldsymbol{\xi}$, so each element of the ordered $\boldsymbol{\xi}$ for $\mathbf{u}$ is coupled with the corresponding element of the ordered $\boldsymbol{\xi}$ of $\mathbf{v}$. If all the error thresholds at each index for the regressor with decision vector (parameters) $\mathbf{u}$ are no larger than the error thresholds at the corresponding index for regressor $\mathbf{v}$ and at least one threshold is lower, then the regressor parameterised by $\mathbf{u}$ is said to *strictly dominate* that parameterised by $\mathbf{v}$ ($\mathbf{u} \succ \mathbf{v}$).

Traditionally, a set F of decision vectors is said to be *non-dominated* if no member of the set is dominated by any other member:

$$\mathbf{u} \not\prec \mathbf{v} \quad \forall \mathbf{u}, \mathbf{v} \in F, \tag{2}$$

and this formulation is used to store the archive of the best estimate of the *Pareto front* found by an optimisation process. In the REC optimisation situation, because the set F itself traces out the composite REC, our decision to add or remove an element to/from F is not quite so straightforward as that in Equation 2. If we define $REC(F)$ as the composite curve generated by the elements in F, then we actually want to maintain F such that:

$$\mathbf{u} \not\prec REC(F \setminus \mathbf{u}) \quad \forall \mathbf{u} \in F. \tag{3}$$

For this formulation a single domination comparison is needed for each parameter vector compared to the current best estimate of the Pareto front/ composite REC curve (albeit a comparison across n objective as opposed to 2). Additionally we known that at most n unique model parameterisations will describe the REC composite curve.

Interestingly the points returned by casting the problem as a 2-objective or as a n-objective problem are slightly different – the n objective optimisation of the curve returning an attainment surface representation of the REC curve (along the error axis) [32, 26], whereas the 2-objective optimisation returns a

Algorithm 1 REC optimising MOEA.

```
1:  F := initialise()                                  Initial front estimate
2:  n := 0
3:  while n < N :                                      Loop
4:      u := evolve(select(F))                         Copy and evolve from F
5:      REC(u)                                         Evaluate
6:      if u ⋠ REC(F)                                  If non-dominated
7:          F := F ∪ u                                 Insert in archive
8:          F := {v ∈ F | v ⊀ REC(F \ v)}               Remove any dominated
9:      end
10:     n := n + 1
11: end
```

strictly non-dominated front representation (and therefore may have less than n elements, as illustrated in Figure 3.2

4 Empirical illustration

In this section a simple multi-objective evolutionary algorithm (MOEA) is introduced to show the generation of REC curves for a number of different well-known autoregressive problems from the literature, namely the Santa Fe competition suite of problems [29]. Note that any recent MOEA from the literature could equally be used [4, 6, 7, 17, 18, 28, 33] – however they would need augmentation to compensate for the problem specific archive update shown in Equation 3.

4.1 The optimiser

The MOEA used here is based on a simple (1+1)-evolutionary algorithm, a model which has been used extensively in the literature [9, 15, 14, 12, 10, 18, 19, 21, 20]. An overview is provided in Algorithm 1. The process commences with the generation of an initial estimate of the REC curve for the problem (Algorithm 1, line 1). This can typically be provided by the generation of random parameterisations of the model and/or the optimisation of the model with traditional scaler optimisers concerned with the mean error (i.e. back-propagation or scaled conjugate gradient for a neural network model [24]). These decision vector(s) are stored in F. The algorithm continues by iterating through a number of generations (line 3), and at each generation creating a new model parameterisation, $\mathbf{u}$, through mutation and/or crossover of elements of F (line 4). $\mathbf{u}$ is compared at each generation to F (line 6). If it is non-dominated by the composite front defined by F, it is inserted into F (line

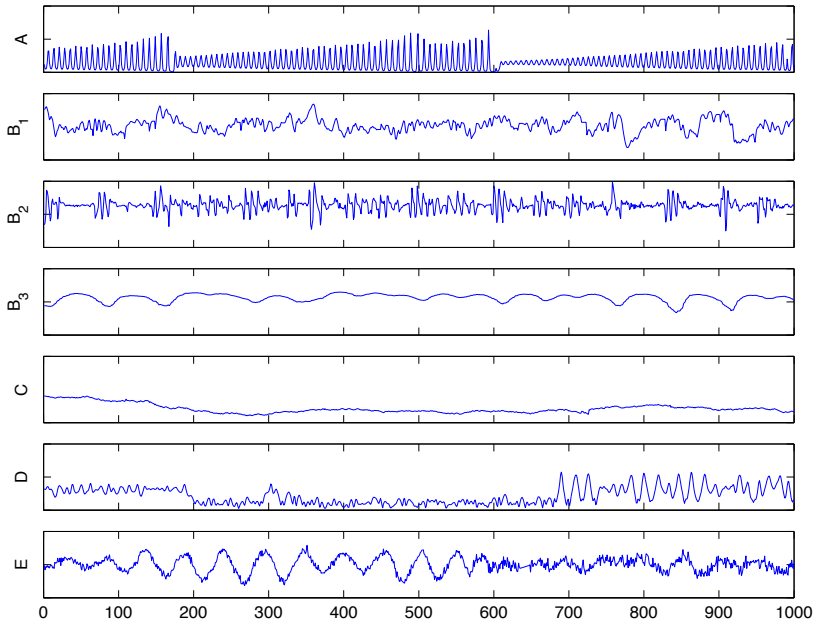


Fig. 5. Santa Fe competition series. 1000 sequential points of each used for training.

7), and any elements in F which no longer contribute to the composite front are removed (line 8).

An equivalent formulation would be to simply update F to minimise its AOC, and remove any elements that do not contribute to its minimisation – effectively a set based scalar optimisation. However as later in this chapter the additional imposition of a complexity minimisation objective will be introduced, the nominally multi-objective formulation will be adhered to here.

The evolve(select(F)) methods (line 4) either generates a new solution by single point crossover of two members from F and perturbing the weights, or by copying a single solution from F and perturbing its weights. Crossover probability was 0.5, weight perturbation probability 0.8 and weight perturbation multiplier 0.01 (perturbation values themselves were drawn from a Laplacian distribution). Parameters used are encoded as real values and crossover occurred at the transform unit level. Different representations and crossover/perturbation/selection methods could equally be used, and an excellent review of those concerned with NN training can be found in Yao [31].

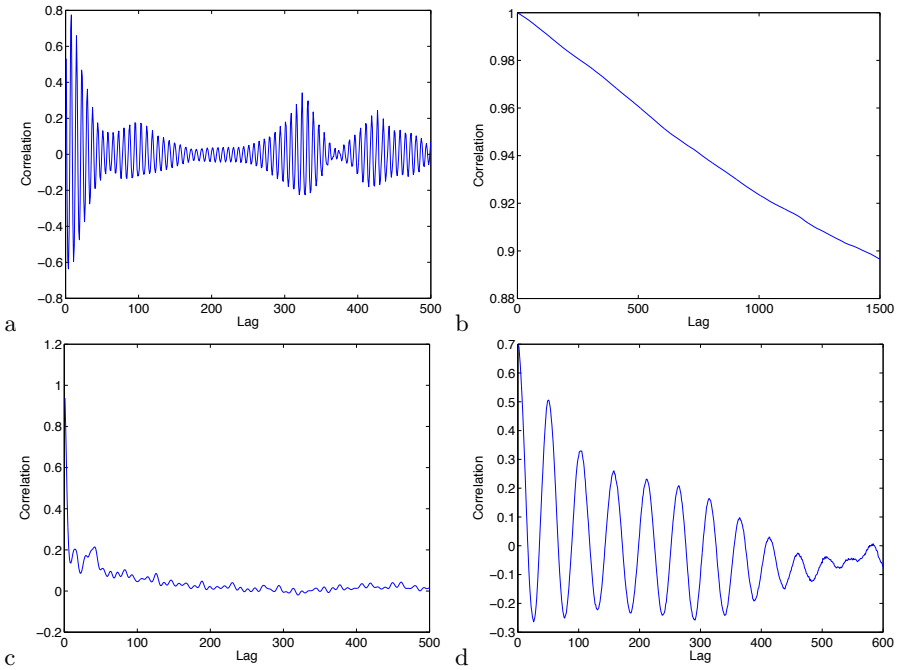


Fig. 6. Autocorrelation of series *a:* A, *b:* C, *c:* D and *d:* E.

4.2 Data

The data used in this chapter to show the properties of REC optimisation is the Santa Fe competition data [29], which is a suite of autoregressive and multi-variate data problems exhibiting different properties. The final series, Series F, is not used here as this exhibits the missing data property, which this chapter is not concerned with.

The other 5 series are:

- Series A: Laser generated data.
- Series B: Three sets of physiological data, spaced by 0.5 second intervals. The first set is the heart rate, the second is the chest volume (respiration force), and the third is the blood oxygen concentration (measured by ear oximetry).
- Series C: Tickwise bids for the exchange rate from Swiss Francs to US dollars. Recorded by a currency trading group for 12 days.
- Series D: computer generated time series.
- Series E: Astrophysical data. A set of measurements of the light curve (time variation of the intensity) of the variable white dwarf star PG1159-035 during March 1989, at 10 second intervals.

All may be obtained from `http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html`. Figure 4.2 shows the first 1000 samples of each of the series (except for series B, where the second 1000 is shown). As can be seen, the series exhibit varying degrees of oscillation and rapid change. In the prediction tasks used here, all series are standardised (the mean of the training data subtracted and divided through by the variance of the training data), however the error reported is that on the prediction transformed back into the original range.

Series A, C, D and E are autoregressive problems (i.e. past values of the same series are used to predict future values). One step ahead predictions are made, and the number of lags (past steps) to use is determined by observing the auto-correlation between steps (shown in Figure 4.2). Series B is a multivariate problem, and is formulated here as a problem of predicted one of the series at time t given the values of the two other series at t.

On inspection of the autocorrelations, 40 lags are used for series A and D, 10 lags for series E. Series C is highly correlated even with very large lags (see Figure 4.2). 5 lags were chosen for this series, but the correlation levels and *a priori* knowledge of the exchange rate market would indict results outperforming a random walk would be surprising.

4.3 Non-linear Models

In the traditional linear regression model the functional transformation takes the form of

$$\hat{y} = u_1 x_1 + u_2 x_2 + \ldots + u_{m-1} x_n + u_m \tag{4}$$

with correspondingly $m = p+1$ model parameters to fit, where p is the number of independent variables.

Here however we shall use the non-linear multi-layer perceptron (MLP) neural network regression model. In an MLP, the functional transformation takes the form of a number of parallel and sequential functional transformations. With k parallel transformation functions (known as the hidden layer), and a single hidden layer, they can be represented in their regression form as:

$$\hat{y} = f_1(\mathbf{x}, u_1, \ldots, u_l) + f_2(\mathbf{x}, u_{l+1}, \ldots, u_{2l}) + \ldots + f_k(\mathbf{x}, u_{(k-1)l}, \ldots, u_{kl}) + u_m \tag{5}$$

In the case of the MLP transfer, these units take the form of a hyperbolic tangent:

$$f(\mathbf{x}, \mathbf{q}) = \tanh\left(q_{p+1} + \sum_{i=1}^{p} q_i x_i\right) q_{p+2} \tag{6}$$

The first p elements of $\mathbf{q}$ are the weights between the inputs to the hidden unit. The $p+1$th element is the unit bias and the $p+2$th element is the weight between the unit and the output. It therefore has $m = k(p+2)+1$ parameters.

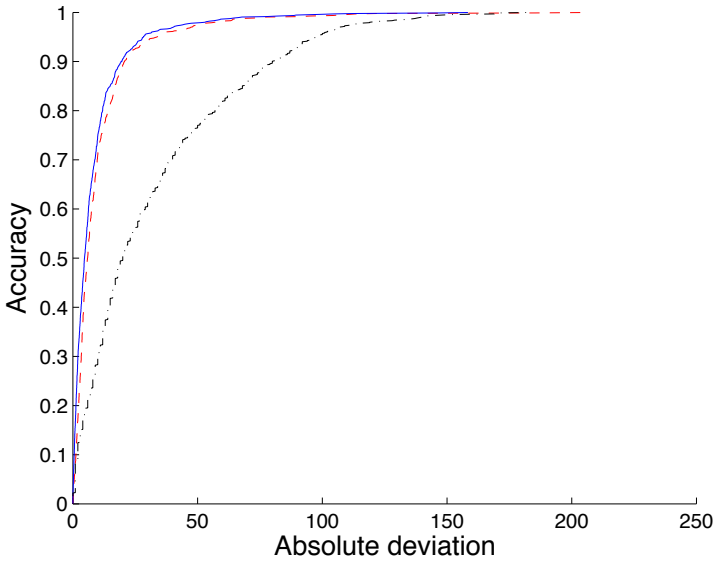


Fig. 7. REC curves of Santa Fe competition series 'A' for a 5 hidden unit MLP. Solid line evolved composite REC, dashed line optimised REC curve of scalar best model and dot-dashed line REC of null model.

4.4 Results

The first example results are shown here for a 5 hidden unit multi-layer perceptron neural network. The model is initially trained using a scaled conjugate gradient algorithm [24], with acts as an initial F. As series A is highly oscillatory the first difference is used as the dependant variable, and the final prediction reconstructed from this. The null model, which is typically the mean of the data, in this formulation is therefore the more appropriate random walk model (which predicts that $\hat{y}_t = y_{t-1}$) – as differencing the data gives a mean of zero.

Figure 4.3 shows the composite (evolved) REC curve after 20000 generations, the REC curve of a single model optimised with the scaled conjugate gradient algorithm for 1000 epochs, and the null model. It should be noted that the training of the single neural network and the subsequent run of the MOEA took approximately the same computation time. The composite REC curve is only slightly in front of the single AOC minimising model, however it does completely dominate it. Both curves are well in front of the null model – implying there is indeed information in the series which enables a degree of prediction beyond the most simple formulation.

Figure 4.4 gives the error histograms of three different points on the composite REC curve, to better illustrate the qualitative difference between the

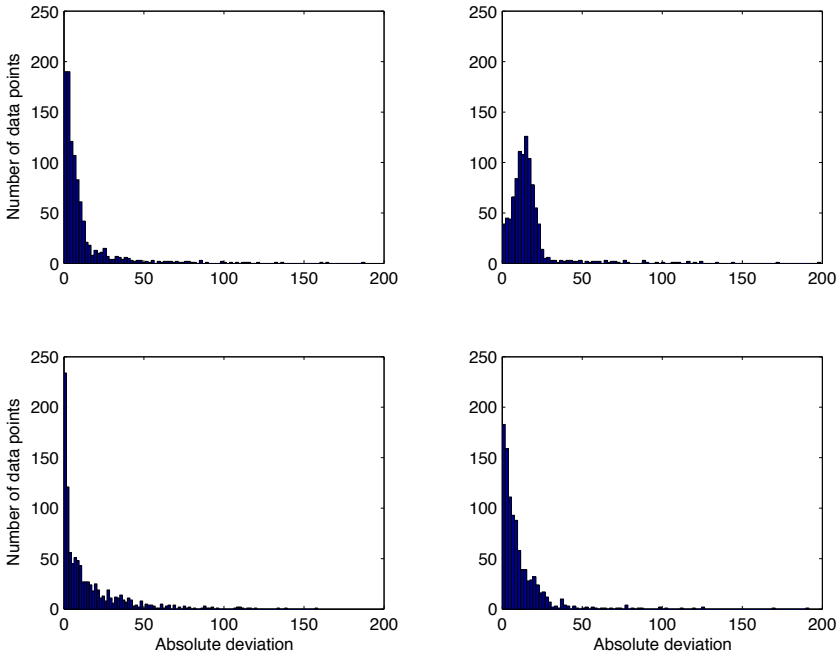


Fig. 8. Histograms of models on evolved composite REC. Bottom left, lowest threshold at 0.1 level. Top left lowest threshold at 0.5 level. Top right lowest threshold at 0.9 level. Bottom right histogram of lowest mean error model.

errors made by regressors on different points on the composite REC curve, the bottom right histogram shows that of the single minimising AOC Model (trained with the scaled conjugate gradient algorithm – the maximum *a posteriori* model). The bottom left histogram (corresponding to the model with the lowest threshold at the 0.1 level) can be seen to exhibit the greatest number of points with very low absolute error. Conversely, although the mean error of the histogram in the top right is pushed higher than the other four models shown, it exhibits fewer very high errors than the others. Figure 4.4 shows the actual errors corresponding to the histograms provided in 4.4, which shows where these errors are being made.

Uncertainty

In [2] the REC curve presented are the means of a number of different runs (on cross validation data), meaning they were an average of a number of different model parameterisations on a number of different datasets – as each model was trained in turn on the different data. What we are interested in here more specifically is the expected variation of a *single* parameterisation, as in

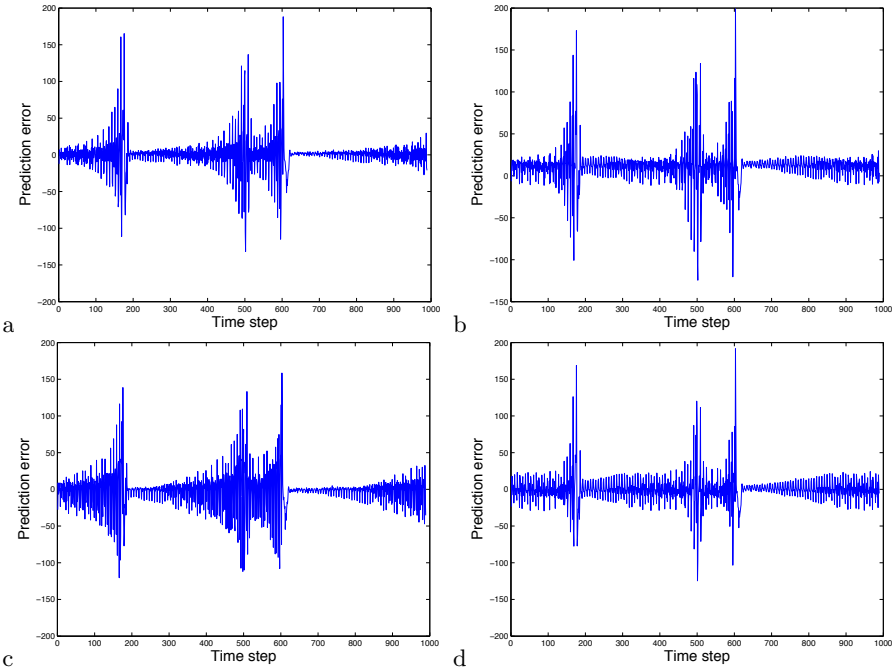


Fig. 9. Errors of models on evolved composite REC. *a:* lowest threshold at 0.5 level. *b:* lowest threshold at 0.9 level. *c:*, lowest threshold at 0.1 level. *d:* histogram of lowest mean error model.

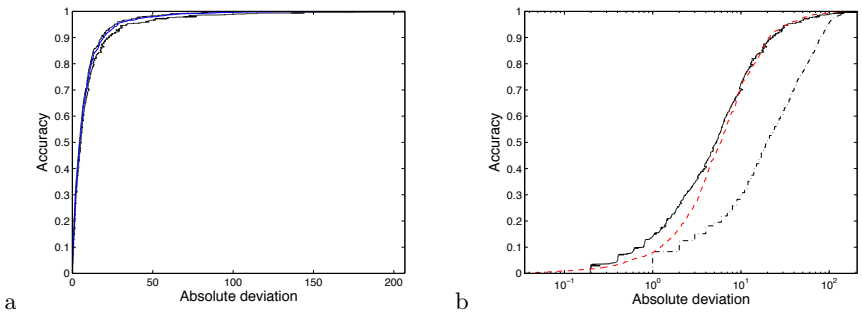
the end we have to decide upon a single model. We can do this effectively by bootstrapping our training data [8], and noting the variation in errors to generate a probability of operating in a particular objective space region [11].

More formally, by bootstrapping the data we are generating a data set of the same size, which is statistically equivalent to the original. If we generate p bootstrap replications, and evaluate our composite REC curve on these p data sets, we are provided with p error thresholds for each accuracy level. This gives us a $n \times p$ set of error values Ξ. We can calculate the probability that the regressor defining the REC curve point at accuracy level, i, will, have a lower error level than a value $\tilde{e}$ as:

$$p(\Xi_i < \tilde{e}) = \frac{1}{p} \sum_{j=1}^{p} I(\Xi_{i,j} < \tilde{e}) \tag{7}$$

where $I(\cdot)$ is the indicator function.

Figure 4.4a shows the probability contours for composite REC front shown in Figure 4.3, at the 5% and 95% levels, created from 200 bootstrap resamples of the data. Figure 4.4b in turn shows the 95% REC contour and the REC curve of the single model trained using the conjugate gradient algorithm.



a b

Fig. 10. *a:* Probability contours of REC for MLP on series A, 5% and 95% level contours shown around composite REC curve. *b:* 95% level composite REC contour and REC curve of MAP model (dashed) and null model (dot-dashed). Error in log scale to aid visualisation.

From this we can say not only (from Figure 4.3) that the single MAP curve lies completely in front of that of the single MAP model, but that we are confident (at the 95% level) that it will lie in front of the single REC model for accuracy levels from 0.02 up to 0.85 on statistically equivalent data.

Figure 4.4 shows these fronts for the other four test problems (once more using an MLP with 5 hidden units). Again the 95% composite REC contour (solid line), the REC of the MAP model (dashed line) and the null model (dot-dashed line) are plotted. (The null model is again set as the random walk model – which is a far better fit than the mean allocation model suggested in [2].) From these we can see that we are confident (at the 95% level) of the composite REC models outperforming the single MAP model on the accuracy range 0.01-0.5 for series B, 0.00-0.10 for series D and 0.00-0.30 on series E. In the case of series C, the REC of the MAP model (and for most of the null model) lies in front of the 95% composite REC contour, implying there is little or no information in the series beyond a random walk prediction.

Until this point we have been concerned with the uncertainty over the error preferences for a regression model (leading to the optimisation of REC curves), and the uncertainty over the variability of the data used (leading to the use of probability contours). It is also very likely that, although we may have a preferred regression model type, we may not know how complex a model we should use – i.e., in the case of NNs, how many hidden units, how many inputs, what level of connectivity? In the final section of this chapter the previous optimising model will be extended so that a set of REC curves can be returned, in one optimising process, which describe the error/accuracy trade-off possibilities for a range of model complexities.

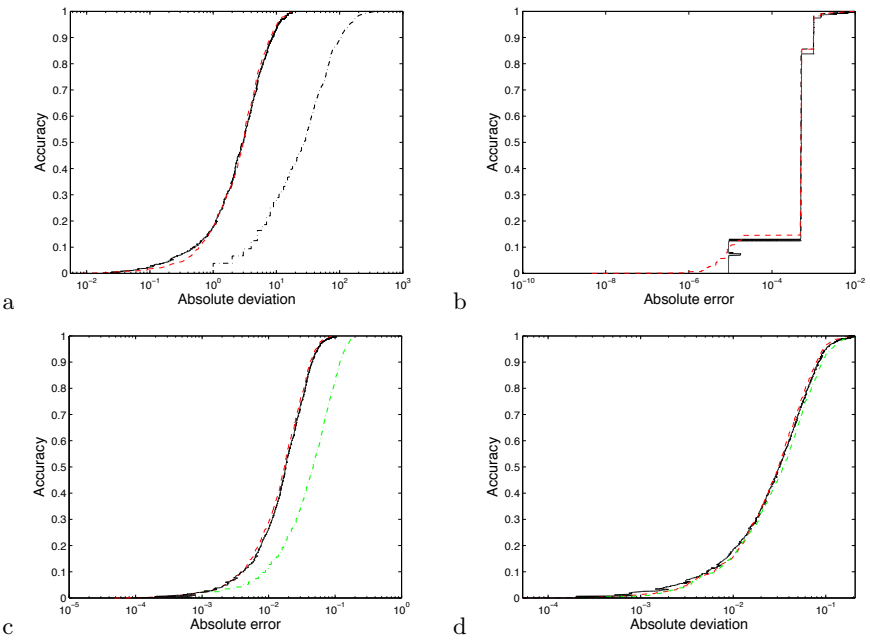


Fig. 11. 95% level composite REC contour and REC curve of MAP model (dashed) and null model (dot-dashed) for series *a:* B, *b:* C, *c:* D and *d:* E. Error in log scale to aid visualisation.

5 Complexity as an additional objective

An additional problem manifest when training regression models is how to specify a model that is sufficient to provide 'good' results to the task at hand without any *a priori* knowledge of how complex the function you wish to emulate actually is. Too simple a model and the performance will be worse than is actually realisable, too complex a model and one runs the risk of 'overfitting' the model and promoting misleading confidence on the actual error properties of your chosen technique. Depending upon the regressor used, various methods to tackle this problem are routinely in use. In the neural network domain these take the guise of weight decay regularisation [3, 25], pruning [22], complexity loss functions [30] and cross validation topology selection [27].

A multi-objective formulation of this problem, which recognised the implicit assumptions about the interaction of model complexity and accuracy that penalisation methods make (e.g. [23]), was recently proposed by the author [16]. This cast the problem of accuracy and complexity as an explicit trade-off, which could be traced out and visualised without imposing any complexity costs *a priori*.

Algorithm 2 REC and complexity optimising MOEA.

1:	$F :=$ initialise()	Initial front estimate
2:	$n := 0$	
3:	`while` $n < N$:	Loop
4:	$\quad \mathbf{u} :=$ evolve(select(F))	Copy and evolve from F
5:	$\quad$ REC($\mathbf{u}$)	Evaluate
6:	$\quad \tilde{F}^- := F \setminus \mathbf{v} \in F$ where $\lvert\mathbf{v}\rvert \leq \lvert\mathbf{u}\rvert$	Lower and equal complexity
7:	$\quad$ `if` $\mathbf{u} \preceq REC(\tilde{F})$	If non-dominated
8:	$\quad\quad F := F \cup \mathbf{u}$	Insert in archive
9:	$\quad\quad \tilde{F}^+ := F \setminus \mathbf{v} \in F$ where $\lvert\mathbf{v}\rvert \geq \lvert\mathbf{u}\rvert$	Higher and equal complexity
10:	$\quad\quad \tilde{F}^- := F \setminus \mathbf{v} \in F$ where $\lvert\mathbf{v}\rvert < \lvert\mathbf{u}\rvert$	Lower complexity
11:	$\quad\quad \tilde{F}^+ := \{\mathbf{v} \in \tilde{F}^+ \lvert \mathbf{v} \npreceq REC(\tilde{F}^+ \setminus \mathbf{v})\}$	Remove any dominated
12:	$\quad\quad F := \tilde{F}^+ \cup \tilde{F}^-$	
13:	$\quad$ `end`	
14:	$\quad n := n + 1$	
15:	`end`	

This method can be applied to the REC optimisation to trace out estimated optimal REC curves for different levels of model complexity. Here complexity shall be cast in terms of the number of model parameters – the larger the parameterisation of a model from a family, the more complex. In the linear regression case this is simply the number of coefficients. In the MLP case this is the number of weights and biases.

5.1 Changes to the optimiser

The evolve(select(F)) methods (line 4) of Algorithm 2 is adjusted in this extension to allow the generation and evolutionary interaction of parameterisations of differing complexity (dimensionality). The existing crossover allows the interaction of models with different dimensionality – in addition weight deletion is also incorporated with a probability of 0.1.

F now contains a set of models generating a composite REC curve for each complexity level, as such its update is also modified from Algorithm 1. Line 6 of Algorithm 2 shows the selection from F of those members with equal or lower complexity to the new parameterisation $\mathbf{u}$ into $\tilde{F}^-$. $\mathbf{u}$ is then compared to the composite REC curve defined by the members of $\tilde{F}^-$, if it is non-dominated, then it is inserted into F (line 8) and those members of F with and equal or higher complexity than $\mathbf{u}$ are then compared to $\mathbf{u}$, and any dominated members removed (line 11).

Apart from these alterations the algorithm is run as previously, this time for 50000 generations.

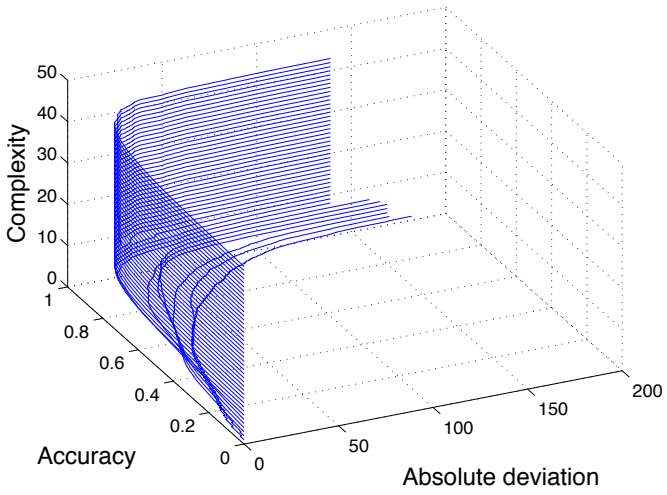


Fig. 12. REC/complexity surface for problem A.

5.2 Empirical results

Figure 5.1 shows the composite REC curves for various MLP complexities for the Series A problem previously defined. As can be see, the REC curve rapidly approaches a stable position with only 10 weights in the network.

Figure 5.2 in turn provides these plots for the other 4 problems previously described. Test problem B can also be adequately described with only 10 weights whereas the REC curves of test problem C are relatively unchanged across all complexities. There are very small adjustments throughout the range of complexities shown for problem D, although good results can be obtained with relatively low complexity. On the other hand 20 weights are needed for problem E before the improvements to the REC curve become relatively small improvements for higher complexity levels.

6 Conclusion

This chapter has introduced the use of MOEAs to optimise a composite REC curve, which describes the (estimated) best possible error/accuracy trade off a regression model can produce for a particular problem.

REC specific properties where also highlighted – the hard limit of the number of different parameterisations possible, the casting as a 2 objective, n objective or even scalar problem, and the different computational complexities of these different formulations.

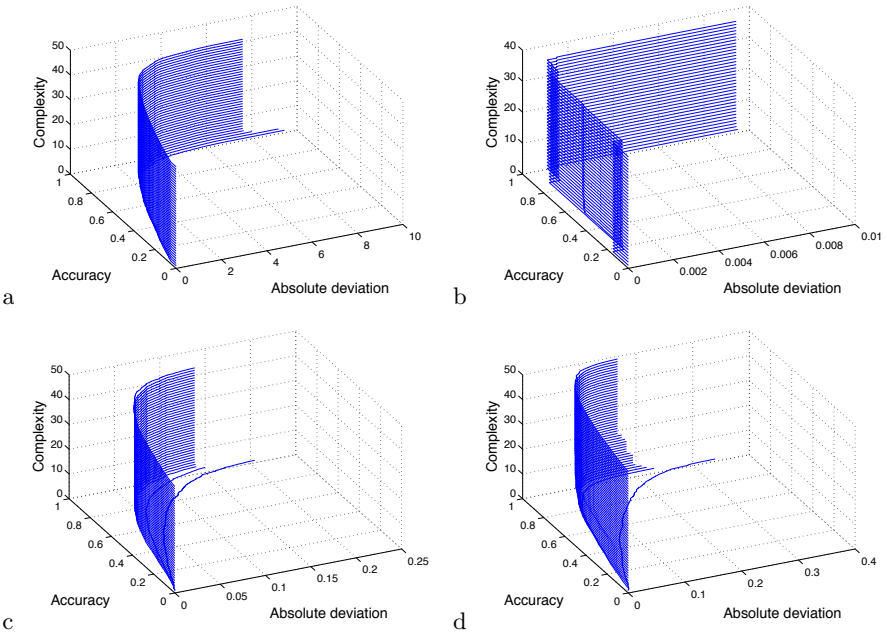


Fig. 13. REC curves of series *a:* B, *b:* C, *c:* D and *d:* E, for varying complexities.

The optimisation method was extended by the use of bootstrapping to show the probability of performance, on statistically equivalent data. The problem itself was then expanded so that the complexity of the model was also optimised, producing a REC surface, which makes it easy to identify the minimum level of complexity needed to achieve specific error/accuracy combinations for a particular model family on a particular problem.

Acknowledgements

The author gratefully acknowledges support from the EPRSC grant number GR/R24357/01 during the writing of this chapter. The NETLAB toolbox for MATLAB, available from `http://www.ncrg.aston.ac.uk/netlab/index.php`, was used as the basis of the NNs used here.

References

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

2. J. Bi and K.P. Bennett. Regression Error Characteristic Curves. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003.
3. C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1998.
4. 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.
5. 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.
6. K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, 2001.
7. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. Fast and elitist multiobjective genetic algorithm: Nsga–ii,. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
8. B. Efron and R.J. Tibshirani. *An introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Probability. Chapman & Hall, New York, 1993.
9. 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.
10. J.E. Fieldsend and R.M. Everson. ROC Optimisation of Safety Related Systems. In J. Hernández-Orallo, C. Ferri, N. Lachiche, and P. Flach, editors, *Proceedings of ROCAI 2004, part of the 16th European Conference on Artificial Intelligence (ECAI)*, pages 37–44, Valencia, Spain, 2004.
11. J.E. Fieldsend and R.M. Everson. Multi-objective Optimisation in the Presence of Uncertainty. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, 2005. Forthcoming.
12. 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.
13. J.E. Fieldsend, J. Matatko, and M. Peng. Cardinality constrained portfolio optimisation. In Z.R. Yang, R. Everson, and H. Yin, editors, *Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'04)*, number 3177 in Lecture Notes in Computer Science, pages 788–793. Springer, 2004.
14. J.E. Fieldsend and S. Singh. A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. In *Proceedings of UK Workshop on Computational Intelligence (UKCI'02)*, pages 37–44, Birmingham, UK, Sept. 2-4, 2002.
15. 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*, pages 388–393, Hawaii, May 12-17, 2002. IEEE Press.
16. J.E. Fieldsend and S. Singh. *Optimizing forecast model complexity using multi-objective evolutionary algorithms*, chapter Applications of Multi-Objective Evolutionary Algorithms, pages 675–700. World Scientific, 2005.
17. C.M. Fonseca and P.J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

18. 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.
19. J.D. Knowles and D. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
20. M. Laumanns, L. Thiele, and E. Zitzler. Running Time Analysis of Multiobjective Evolutionary Algorithms on Pseudo-Boolean Functions. *IEEE Transactions on Evolutionary Computation*, 8(2):170–182, 2004.
21. M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb. Running Time Analysis of Multi-objective Evolutionary Algorithms on a Simple Discrete Optimization Problem. In J.J. Merelo Guervós, P. Adamidis, H-G Beyer, J-L Fernández-Villacañas, and H-P Schwefel, editors, *Parallel Problem Solving from Nature—PPSN VII*, Lecture Notes in Computer Science, pages 44–53. Springer-Verlag, 2002 2002.
22. Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, pages 598–605, San Mateo, CA, 1990. Morgan Kauffman.
23. Y. Liu and X. Yao. Towards designing neural network ensembles by evolution. *Lecture Notes in Computer Science*, 1498:623–632, 1998.
24. I.T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer, 2002.
25. Y. Raviv and N. Intrator. Bootstrapping with noise: An effective regularization technique. *Connection Science*, 8:356–372, 1996.
26. K.I. Smith, R.M. Everson, and J.E. Fieldsend. Dominance Measures for Multi-Objective Simulated Annealing. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'04)*, pages 23–30, 2004.
27. J. Utans and J. Moody. Selecting neural network architectures via the prediction risk: application to corporate bond rating prediction. In *Proc. of the First Int. Conf on AI Applications on Wall Street*, pages 35–41, Los Alamos,CA, 1991. IEEE Computer Society Press.
28. D. Van Veldhuizen and G. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.
29. A. S. Weigend and N. A. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, Reading, MA, 1994.
30. D. Wolpert. On bias plus variance. *Neural Computation*, 9(6):1211–1243, 1997.
31. X. Yao. Evolving Artificial Neural Networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.
32. E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology Zurich (ETH), 1999. Diss ETH No. 13398.
33. 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.