Chapter 4

Adaptive Statistical Evaluation Tools for Equity Ranking Models

Problem presented by: Brad Bondy (Genus Capital Management)

Mentors: Tony Ware (University of Calgary), Len Bos (University of Calgary)

Student Participants: Amir Amiraslani (University of Western Ontario), Thomas Holloway (University of Alberta), Hua Li (University of Calgary), Maryam Mizani (University of Victoria), Mahyar Mohajer (University of Calgary), Comron Nouri (University of Western Ontario), Gergeli Orosi (University of Calgary), Nargol Rezvani (University of Western Ontario), Liang Xu (University of Washington), Oulu Xu (York University)

Report prepared by: Tony Ware (`aware@ucalgary.ca`)

4.1 Introduction

Based in Vancouver, Genus Capital Management (`http://www.genuscap.com`) is an independent investment provider who for over 15 years have been managing assets for private individuals and families, trusts, foundations and pension funds from across Canada. Their investments span a range of equity and fixed income, and they manage assets to the value of about $1.3-billion.

Genus offer investment portfolios with a variety of flavours. Each portfolio consists of investments in stocks from some given 'universe'. Competitive advantage comes from consistently out-performing competing funds. One of the approaches that Genus use to achieve competitive advantage is to make use of auxiliary information about the stocks in the universe in deciding how to adjust their portfolio month by month. The problem that Genus brought to the IPSW was concerned with how to improve the way this information is utilised.

4.2 Problem Description

A major challenge in the investment management business is to identify which stocks are likely to outperform in the future, and which are likely to perform relatively poorly. To this end the strategy

adopted by Genus is to identify *factors* (auxiliary information about the stock such as earnings-to-price ratio or dividend yield) that they believe are associated with future out-performance (i.e. factors that have predictive ability). The best of these factors are then combined (Genus use a weighted average) into a model which is used to rank the universe of stocks month-by-month. This ranking is then used to as the input to a trading strategy, resulting in a modified portfolio.

A critical step in this process is to identify the 'good' factors and to determine how heavily each should be weighted in the final model. Currently Genus use in-sample back-tests in which they simulate a trading strategy of, for example, selling stocks that drop below the 40th percentile (of the ranking) and reinvesting the proceeds in stocks in the top 20th percentile. These simulations are run over a wide range of potential weighting schemes or models, and the model with the most attractive attributes is identified. Attributes that Genus typically look for in a model include:

- High *spread* between returns for top and bottom quintiles.
- High *information ratio*: the ratio of the mean and standard deviation of the excess monthly return on portfolio versus the index (a portfolio in which all the stocks in the universe are equally-weighted).
- High *hit ratio*: the percentage of months in which the portfolio outperforms the index.

The in-sample back-tests are typically run over a 10-year period, excluding the most recent 2 years. Once a model is identified, out-of-sample tests of the model over the most recent 2 years are run in order to ensure that the model works out-of-sample.

A major issue for Genus is the inflexibility that results from the 2-year delay for out-of-sample testing. Factors tend to lose their predictive ability as other market participants incorporate them, and useful new factors seem to be getting cleaned out faster and faster. With the 2-year delay Genus run the risk of delaying introduction of a new factor until the market has already cleaned it out. On the other hand, if the out-of-sample tests are eliminated, they run the risk of over-fitting the data.

The challenge posed for the workshop team was:

1. to recommend adaptive statistical evaluation tools (alternatives to out-of-sample tests) that could be used to improve confidence in a model and to help decide in a timely fashion if a new factor should be added to a model, or if an existing factor should be removed;
2. to suggest algorithms that could be used to dynamically update the models with a view to exploring a dynamic strategy in which model factors and weights are updated monthly based on the evaluation measures.

4.3 Rising to the Challenge

It was clear from the outset that if the team were to be able to address either of these challenges realistically they would need to build tools that would enable them to implement the various components of Genus' portfolio management strategy. As the week progressed, it became apparent that a major stumbling block would be implementation of an efficient procedure for finding a 'good' model for ranking the stocks. This became the focus of the team's efforts.

Genus had provided the team with sample data, consisting of just over 12 years worth of monthly returns on a universe of 60 stocks, along with time series of 34 factors for each of the stocks. Using these data, the approach was to build software (MATLAB) models for:

- ranking the stocks based on factor information;
- implementing a trading strategy based on a stock ranking and assessing the performance of a given trading strategy by looking at measures such as hit ratio, information ratio and spread.

The IPSW team implemented a simplified trading strategy of selling the entire portfolio each month, and using the proceeds to invest equally in the top 20% of stocks as given by the computed ranking. They also implemented the following measures of portfolio performance: excess return, hit ratio and information ratio.

4.3.1 Ranking

As noted above, determining an an effective procedure for ranking the stocks is a critical step, but the design of such a procedure cannot be separated from the other tasks, and in particular the choice of trading strategy, or performance assessment measures.

A simplified description of the approach taken by Genus is the following. Given a set of factor values $f_{i,j}(t)$ for the jth factor corresponding to ith stock at time t, and given a linear weighting vector $\vec{w} = (w_j)$, they produce a ranking at each time t from the score vector

$$S_i(t; w) := \sum_j w_j f_{i,j}(t).$$

A trading strategy, such as buying the top quintile of stocks (those obtaining the top 20% scores) in equal measure, is assumed (the actual strategy adopted by Genus is rather more complicated than this, but the details were not available to the IPSW team). The trading strategy takes the weighting vector $\vec{w}$ and the factor values for a given date and produces a set of portfolio weights $\alpha_i(t)$. The actual stock returns $R_i(t)$ can then be used to compute the portfolio return

$$\sum_i \alpha_i(t) R_i(t).$$

Given such a strategy, the performance of the model can be assessed by measuring the resulting excess portfolio returns, the information ratio, the hit ratio, and other parameters.

In order to determine a 'good' weighting vector $\vec{w}$, Genus select a training period (typically a ten-year period of time excluding the most recent two years). They construct a grid of possible weighting vectors, spanning a subset of the total set of possibilities; for each vector in the grid they produce rankings for each date in the training period, and (using the trading strategy) compute the corresponding portfolio excess returns, hit ratios, information ratios, etc. They then select the best-performing weighting vector from this sample and test it against the most recent two-year period before putting it to use.

The determination of a ranking model is an optimisation problem, with a highly nonlinear objective function (irrespective of which of the portfolio measures is used). The IPSW team decided that there could be significant advantages to be gained from exploring alternative approaches to the design of optimal ranking algorithms. They considered three distinct techniques: a genetic optimisation algorithm, a neural network, and a constrained least-squares approach.

4.3.2 Genetic Optimization

Genetic optimization is an evolutionary search technique (see [2] for more information), in which a population of abstract representations of candidate solutions (*individuals*) evolves towards better solutions. The evolution begins with a population of randomly selected individuals and proceeds in *generations*. The transition from generation to generation involves evaluating the *fitness* of the current population; individuals are randomly selected for *reproduction* (with a probability based on their fitness). Individuals selected for reproduction are randomly selected for *mating* (each pair produces two offspring, each of which has some combination of the features of their parents). In addition, each feature of the offspring can mutate independently with a given probability.

Parameters and Results

Genetic optimisation was used to search for an optimum weight vector $\vec{w}$ for use in a linear ranking model as described above. The first 113 months of factor and return data were used to construct the objective function, which took a weighting vector as input and gave the resulting excess returns as output. The populations consisted of 30 individuals, with a 0.01 probability of mutation of the offspring features, and the algorithm was allowed to evolve for 1000 generations before being terminated. The resulting weight vector $\vec{w}_0$ is shown in Figure 4.1.

In Figure 4.2, the objective function in the neighbourhood of $\vec{w}_0$ is illustrated. Each of the graphs in the figure depicts a 'slice' of the objective function in the direction of one of the 34 coordinate directions, centred at $\vec{w}_0$. The non-smooth nature of the objective function can be readily seen, as is the fact that the genetic algorithm has attained something close to, but not exactly at, a maximum.

4.3.3 Neural Network

An artificial neural network (see [3] for more information) is a computational model for information processing, represented by a nonlinear function ν that maps a vector of input values $\vec{x}$ to a vector of output values $\vec{y}$. The nonlinear function has a particular form given by a network structure together with weights associated with the network connections. A feed forward network typically has an input layer, with a node for each element of $\vec{x}$, an output layer, with a node for each element of $\vec{y}$, and one or more hidden layers in between. Each node in a given layer can be connected to each node in the next layer.

The nodes in the input layer do no processing: they simply pass on the input values to the nodes in the next layer to which they are connected.

Nodes in the hidden layers receive values from the incoming connections; they process these values in some way before determining a value to pass on to the nodes in the next layer. A common choice for computing the output value is to compute a weighted linear combination of the inputs (the weights being those associated with the incoming connections) and to apply a logistic function, for example $\psi(s) = \frac{1}{1+\mathrm{e}^{-s}}$, to the result. Nodes in the output layer compute the elements of the vector $\vec{y}$ by computing the weighted combination of their inputs. A network of this sort is an example of a *multilayer perceptron*.

Even with just one hidden layer, such a network has a universal approximation property, and can—with an appropriate choice of connection weights—reproduce arbitrarily closely the action of any continuous function that maps intervals of real numbers to intervals of real numbers [1].

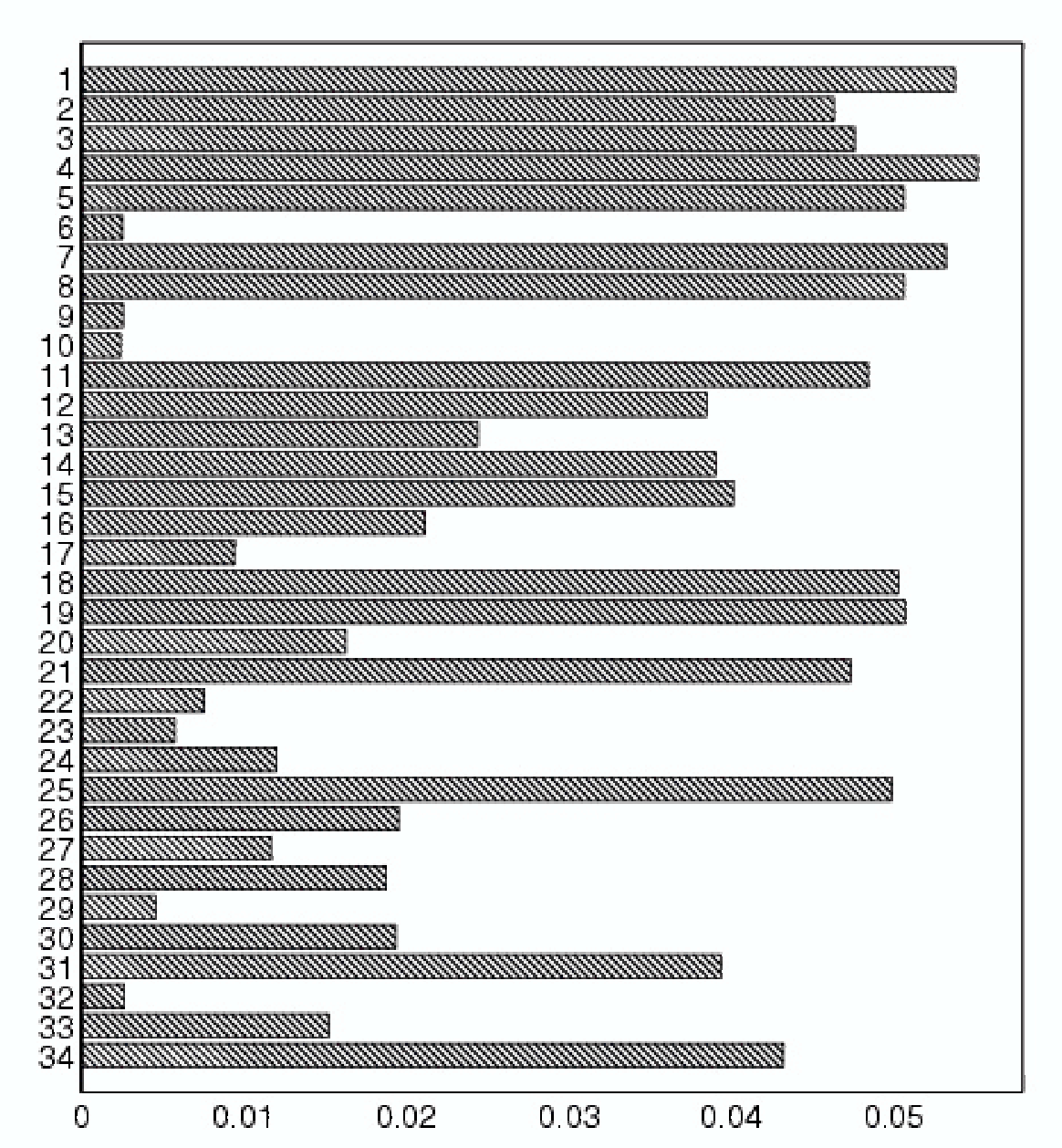

Figure 4.1: Factor weights from the genetic algorithm.

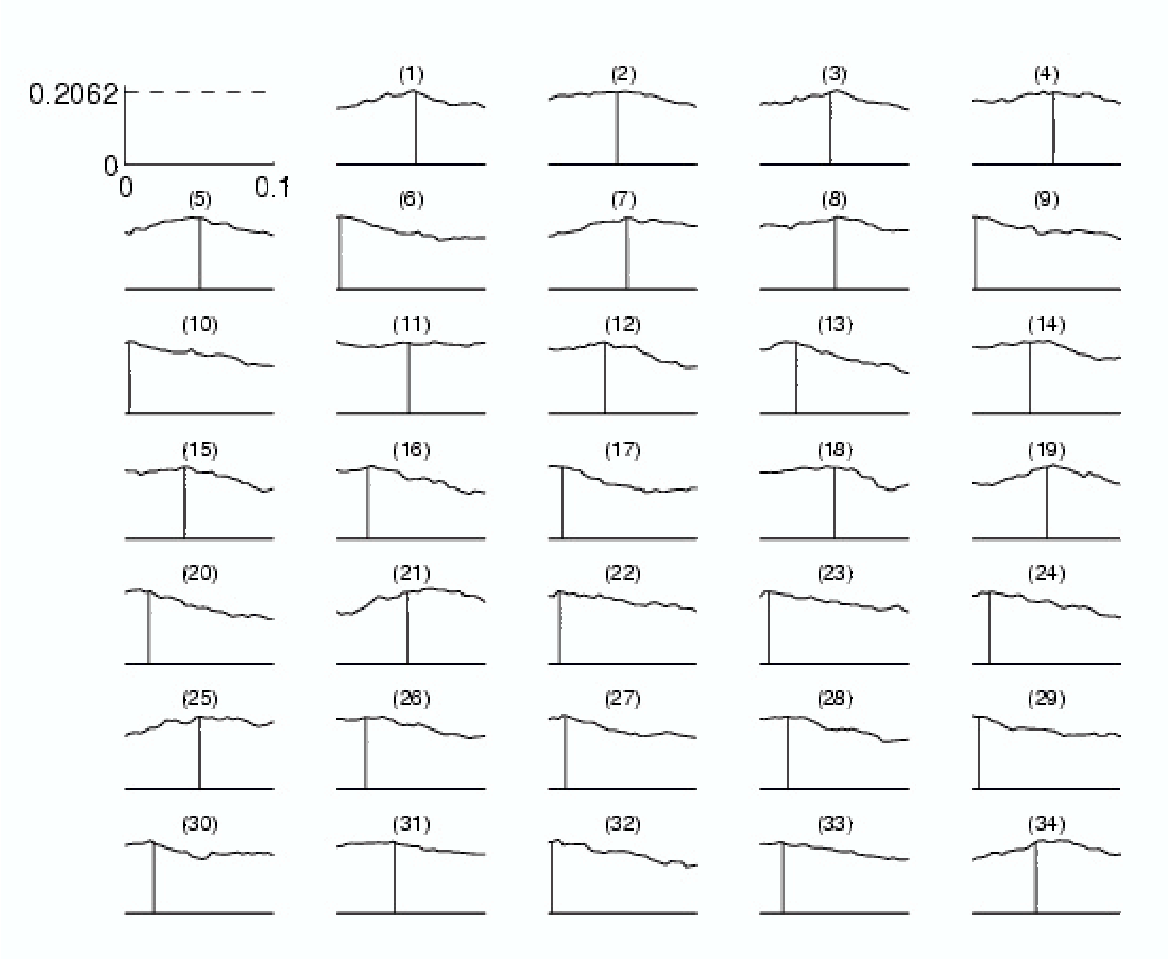

Figure 4.2: Slice view of the 34-dimensional objective function near the chosen maximum for the genetic optimisation. The first graph shows the limits of the weight values, as well as the value of the objective function at $\vec{w}_0$. In each of the other graphs the vertical line denotes the value of the corresponding component of $\vec{w}_0$.

In order for the network to be of any practical use, the connection weights need to be determined. A training set of inputs and corresponding outputs is used for this purpose. One popular choice for adapting the network to the training set is *back-propagation*, which is an implementation of a gradient-descent approach to minimising the sum of the squares of the differences between the given outputs and the outputs from the network.

Parameters

A single-layer perceptron was employed, with the hidden layer having just two nodes. The network was allowed to have six inputs (six of the the possible thirty-four factors were selected) and one output: the stock return. The input factors were chosen by determining which of the factors were most highly correlated with the subsequent months' returns.

The training data set was selected to be the first 113 periods. Back-propagation, with approximately 1000 iterations, and a learning rate of 0.1 (the distance moved in the direction of the gradient vector at each iteration), was used to determine the weights.

Once the network was trained it was used to produce forecasted returns on stocks for each of the remaining 36 months. In each month, the forecast returns were used to rank the stocks, and the trading strategy of investing in the top 20% used to construct a portfolio for that month.

4.3.4 Constrained Least Squares Optimization

A third approach was also used; as with the genetic optimisation, an optimal weight vector $\vec{w}$ was sought, with the entries constrained to be positive. Subject to this constraint, the weights were chosen to minimise the sum of squared discrepancies between the individual stock returns[1] and the corresponding score vector for each time t in the first 113 months:

$$\sum_{t=1}^{113} \sum_{i} (S_i(t; w) - R_i(t))^2.$$

The positive weight vector that minimised this error term was computed using MATLAB's `quadprog` routine, which will compute the minimum of a quadratic function subject to bounds on the input variables [4]. The resulting weight vector is shown in Figure 4.3.

4.4 Results

Each of the resulting models were applied to the most recent 36 months of data (i.e. not the data used in determining the models). The performances of their portfolios are illustrated in Figure 4.4, where the portfolio returns are shown in raw form in the upper graph, along with the performance of an index portfolio. In the lower graph, the excess returns, relative to the index, are shown. Also, in Figure 4.5, the values of several other measures of portfolio performance are shown.

[1] Other objectives could be designed: for example, the ranking of the stock returns could be used.

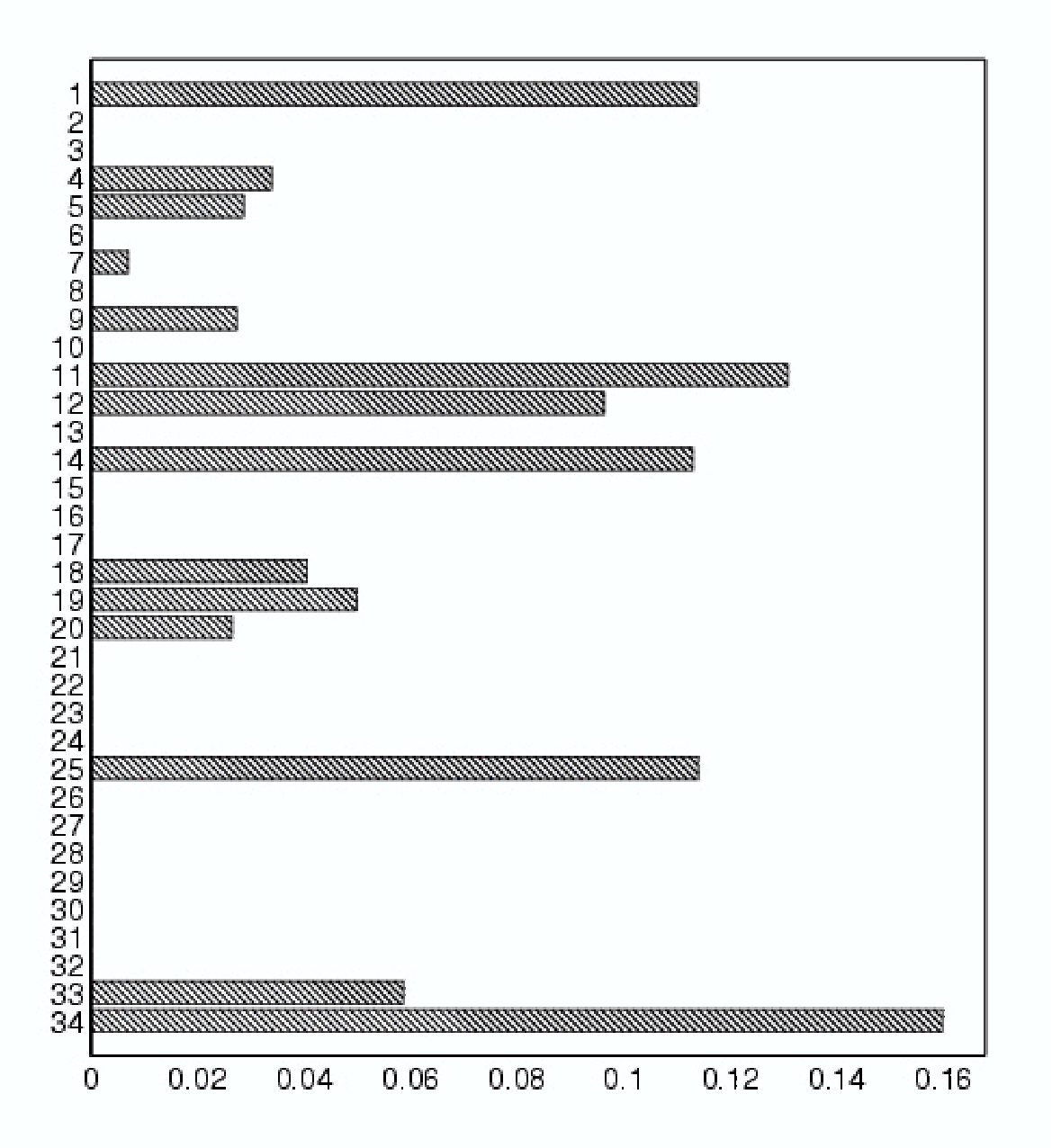

Figure 4.3: Factor weights from the constrained least squares optimisation.

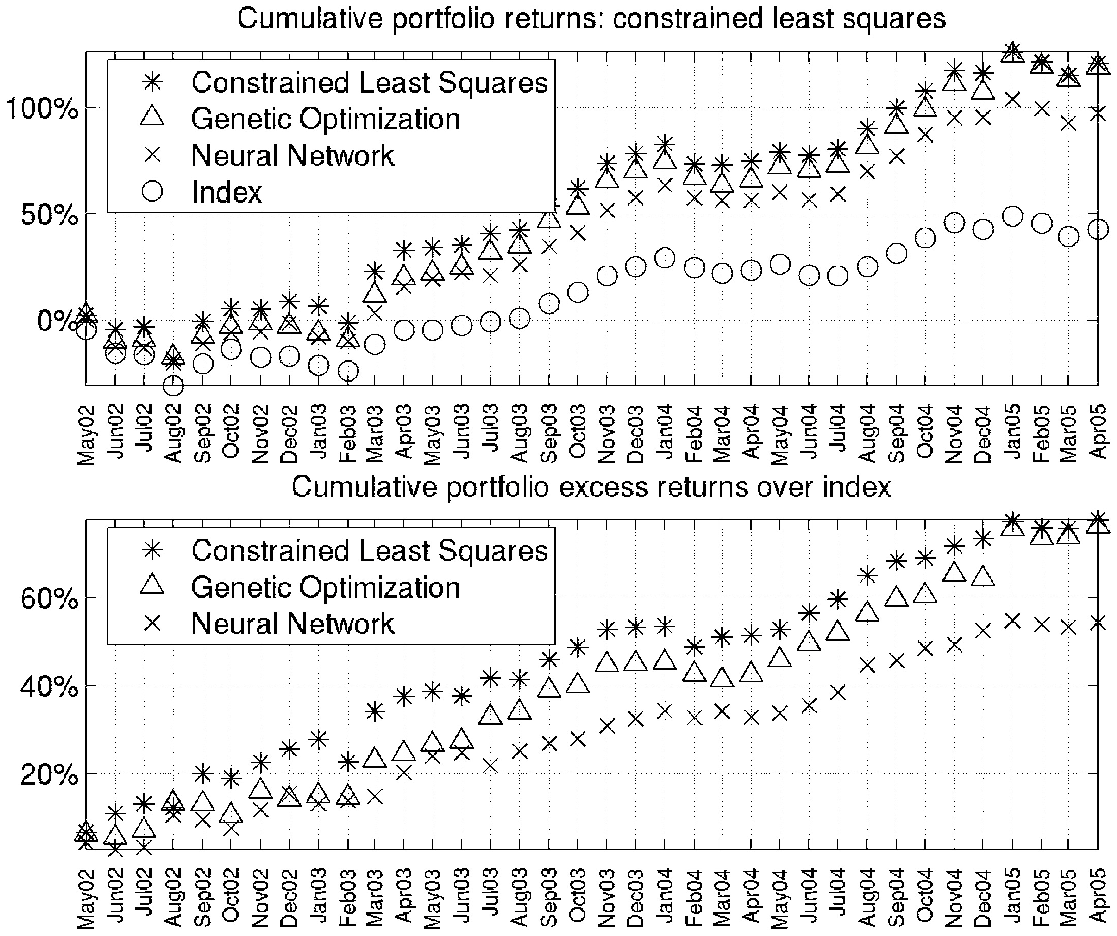

Figure 4.4: Three-year results from the constrained least-squares model, the genetic optimisation model, and the neural network.

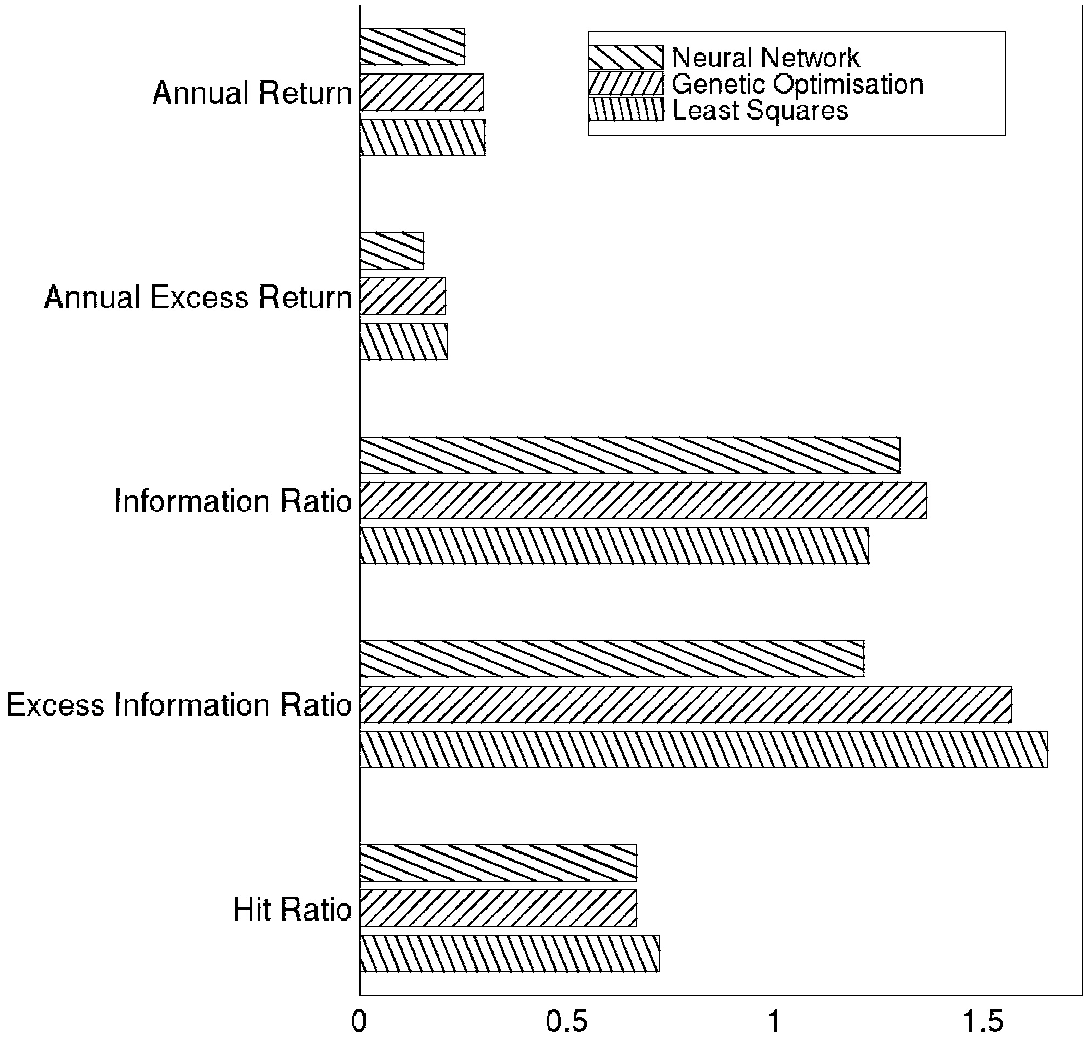

Figure 4.5: Values of Hit Ratio, Excess Information Ratio, Excess Information Ratio (using portfolio returns relative to the index), Annual Excess Return and Annual Return for the portfolios generated by the three different ranking models.

4.5 Conclusions and Directions for Further Work

The results from the three ranking models shown in the previous section are illustrative only. Each of the approaches is capable of refinement. However, the results obtained—even without refinement—compare well with the performance of Genus' current model. Each of the approaches has the potential to deliver significant improvements for Genus. Moreover, especially in the case of the constrained least-squares optimisation, they can be implemented much more quickly. Future work might involve:

- refinement of the neural network and genetic optimisation approaches: each offers a vast array of possible implementations and the ones presented here can certainly be improved upon;
- experimentation with alternative objective functions for the constrained least squares (quadratic programming) approach;
- addressing Genus' original challenge....

Bibliography

[1] Hornik, K., Stinchcombe, M. & White, H. (1989). Multi-layer feedforward networks are universal approximators, *Neural Networks*, 2, pp. 359-366.

[2] Bäck, T. (1996). Evolutionary Algorithms in Theory and Practice. Oxford University Press.

[3] Rojas, R. (1996). Neural Networks: A Systematic Introduction. Springer-Verlag.

[4] Coleman, T.F., & Li, Y. (1996). A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables, *SIAM Journal on Optimization*, 6(4), pp. 1040-1058.