Forecasting Stock Returns using Evolutionary Artificial Neural Networks[1]

Prisadarng Skolpadungket, Keshav Dahal, Napat Harnpornchai

MOSAIC Research Group, University of Bradford, Great Horton Road,
Bradford, BD7 1DP, Great Britain.
CAMT, Chiang Mai University, Chiang Mai, Thailand.
{p.skolpadungket, k.p.dahal}@bradford.ac.uk, tomnapat@camt.info

Abstract. Several models and techniques have been used to forecast stock returns. Some previous researches have applied Evolutionary Artificial Neural Networks to predict stocks prices. The most of previous researches have been concentrated on either predict stock indexes or trends rather than individual stock returns. In this paper, we use the adaptive EANNs to predict individual stock returns based on multivariate time series (AR with state variables) models. Comparing with the traditional Linear Regressions, the ANNs show promising results and our proposed EANNs can improve the performances of the ANNs as we expected even the improvements are slights

1 Introduction

For stock market investors and portfolio managers, it is crucial to have most accurate forecast of stock returns. However, stock returns are difficult to forecast accurately. Several models and techniques have been used to forecast stock returns. The relevant models include autoregressive models (AR), autoregressive-moving-average models (ARMA) and AR with state variables (explaining variable) models. It is reported that the AR models with state variables are superior to the rest both in short-run and long-run [1]. In developing a model with many state variables as model inputs to forecast stock returns, a crucial part is to identify input variables. There are numerous theories and models that determine the input variables ranging from technical analysis based on trading data complicated multivariate time series models. The multivariate time series models, which are based on fundamental factors, are considered more theoretically sound than those based on technical factors (e.g. trading volume, price trend, etc.)[1]. They are implied that stock prices and stock returns can be explained and thus predicted by a number of "fundamental" economic factors as proposed by Capital Market Theory (CAPM-single factor i.e. stock market index) [2,3] and Asset Pricing Theory (APT – multi-factors)[4]. The inputs suggested by an empirical research were changing in economic and financial variables such as changing in inflation, changing in yield spreads, etc [5].

The techniques that have been deploy to forecast stock returns are linear regression (time-series), artificial neural networks (ANNs), decision trees, rule inductions, Bayesian belief networks, evolutionary algorithms (EAs), classifier systems and association rules [6]. Researches found that ANNs shows better performances than most of techniques especially linear regressions [7]. However, ANNs with sub-optimal initial weights can be trapped in local minima. In dynamic environments as the nature of learning objects are always changing, the topologies of the ANNs also should be adapted accordingly. Evolutionary Algorithms can be applied to evolve ANNs at many levels e.g. connection weights, topologies both the number of hidden nodes as well as the number of hidden layers, and learning rules [8]. Some previous researches have applied Evolutionary Artificial Neural Networks (EANNs) to predict stocks prices [6,7]. In their research, Kwon et al. [6] aimed to predict stock price trends (only up or down) by using EANNs that could evolve their initial connection weights. Comparing buy-and-hold strategy, Recurrent ANNs (RANNs)and EANNs, they found that their proposed EANNs outperformed Recusrrent ANNs, while EANNs were significantly outperformed the buy-and-hold strategy. The same author in another paper [7], attempted to predict stock returns

based on stock correlations (with other stock in the same market). They proposed EANNs (they called Feature Selection Genetic Algorithm –FSGA) that could evolve set of inputs (selections of inputs). By comparing prediction performance (only stock price up or down) of buy-and-hold, Recurrent ANNs and EANNs (FSGAs), they found that the order of performance was the same i.e. EANNs then RANNs and then buy and hold. A related research by Armano et al. [9] proposed an EANN algorithm called NXCS, essentially a set of genetic classifiers designed to control feed forward ANNs' activation for performing forecasting at different particular local scopes, to predict stock indexes (rather than individual stock prices or returns also up or down only.) The prediction then results from experts' interactions in the population. The research found that the proposed methodology repeatedly outperformed buy-and-hold strategy.

The most of previous researches have been concentrated on either predict stock indexes or trends rather than individual stock returns [1,8]. For stock trading, merely prediction on trends of stock prices or indexes would be adequate. But for portfolio optimisation especially mean-variances analysis (Markowitz) portfolio optimisation model, to construct an efficient portfolio of stocks, a portfolio manager needs to most accurately predict individual stock returns as well as their variances[11]. Our research in this paper proposes an evolutionary scheme of neural networks with evolving connection weights and "step-up" adding more hidden nodes and layers in order to search for optimal structure. We use the adaptive EANNs to predict individual stock returns based on multivariate time series (AR with state variables) models. Since the input time series are quite limited we also apply Multi-fold Cross Validation methods for the sections of the optimal ANN structures. The prediction results have been compared with those of simple regressions (Least Square Estimation) and of simple (non-evolutionary) ANNs (Backpropagation and Elman Recurrent ANNs.)

This paper is structured as follows: Section 2 gives a brief review of the time series models for explanation and prediction of stock returns. Section 3 describes the ANNs, its encoding representation for the evolutionary algorithm and the nonlinear cross-validation calculation as the objective function for evolutionary selection as well as the complete loop of evolution algorithm. Section 4 details for the dataset and experiment setting. Where as section 5 shows the results with discussion. Section 6 provides for conclusions and suggestion for future works

2 Time Series Models of Stock Returns

Generally, asset returns are the difference in prices from the beginning period (investing time) to (disinvesting time) plus dividends if any. For convenience and by assuming that either the time is quite short or dividend payouts are always reflected in asset prices, we will disregard dividends in the models. We can generalize models of asset return into 3 categories of models. In the simplest AR Model for time-variation expected returns, the expected returns follow auto-regressive (AR) processes. The second category is called the ARMA model. The logarithmic prices of assets have two components, a permanent part and a transitory part. The permanent part follows an AR process. On the other hand, the transitory part follows a moving average (MA) process. The last category is the state variable model. In this kind of model, the transitory component of price not only depends on its own past value but also on state variable (x) which are relevant financial and economic variables. The AR Model of expected returns has considerable capacity to capture the movement of stock returns over short-horizons but has a mediocre capacity to predict the expected stock return over longer-horizons. On the other hand, the ARMA model does a good job of forecasting long-horizon returns, but has no adequate flexibility to capture the pattern of expected return at short horizons. While, the State Variable model is the best in at least four aspects namely, using only the recent past returns, parsimonious, good prediction in the short-run and good prediction in the long-run[1].

A State Variable model with K state variables and N time lags can be stated as

$$r_t = \sum_{i=1}^{N} \phi_i r_{t-i} + \sum_{k}^{K} \sum_{i}^{N} \gamma_i^k \Delta x_{t-i}^k + e_t \quad (1)$$

Where,

$r_{t,}$, r_{t-i} is stock returns at the time period t and t-i respectively.

ϕ_i is stock return's autoregressive coefficient for time lag i.

x_{t-i}^k is the k^{th} state variable at previous i^{th} time period,

γ_t^k is the regression coefficient for the previous i^{th} period of the k^{th} state variable,

e_τ is the error term.

Factors that have evidences of influencing stock returns and include in our model are previous stock returns (R), unemployment (U), money supply (M), stock index (SP500), inflation (CPI), default spread (DS), term spread (TS), reference interest rate (FED), industrial production (IP), and January effect (JAN-circumstantial variable)[11]. Note that we exclude some factors that have no monthly data e.g. trade deficits, GDP etc. The model deploy in this paper can be stated as follow:

R(t=0) = R(t=-1 to -12) + U(t=-1 to -12) + M(t=-1 to -12) + SP500(t=-1 to -12) + CPI(t=-1 to -12) + DS(t=-1 to -12) + TS(t=-1 to -12) + FED(t=-1 to -12) + IP(t=-1 to -12) +JAN(t=-1 to -12) (2)

The model constitutes 1 predicting output and 120 inputs (10 kinds with 12 month lags each.) For January effect (JAN), the input is "1" if the month is January and "0" otherwise.

3 Evolutionary ANN Design

3.1 The ANN and EA Encoding

In this paper, BPNs and Elman RANs are used to predict stock return for next T period ahead. The proposed encodings for the BPNs and RANs for evolving proposes uses direct encoding for connection weights and indirect encoding for the number of hidden nodes and the number of layers. There are two set of evolving encoding gene namely weight matrices and layer specification. A weight matrix describes weights of connections for each node (hidden and output nodes) to other hidden nodes in the immediate previous layer (may be the input layer for hidden nodes in the first hidden layer.) The weight values are in between -1 and 1. In the first generation the weights are randomly set up. And also when the structure of an ANN is changing i.e. adding a hidden node or adding a hidden layer, the weights are also randomly re-assigned. In evolutionary process, each weight mutates to its current value plus random number between -0.5 and 0.5. If the mutated values exceed 1 or below -1, the weight will be set to 1 or -1 respectively. A layer specification is a vector of integer describing the number of hidden nodes in each layer excluding input layers thus the length of layer specification is equal to the number of hidden node plus one of output layer. In evolutionary process, after a mutation on connection weights cannot improves the performance of an ANN, a hidden node is added into the first hidden layer with all connection weights are randomly reassigned. If the inclusion of a hidden node in the first hidden layer also cannot improve the performance of the ANN, a new hidden layer with 2 hidden nodes is put before the first hidden layer (then become the fist hidden layer.)

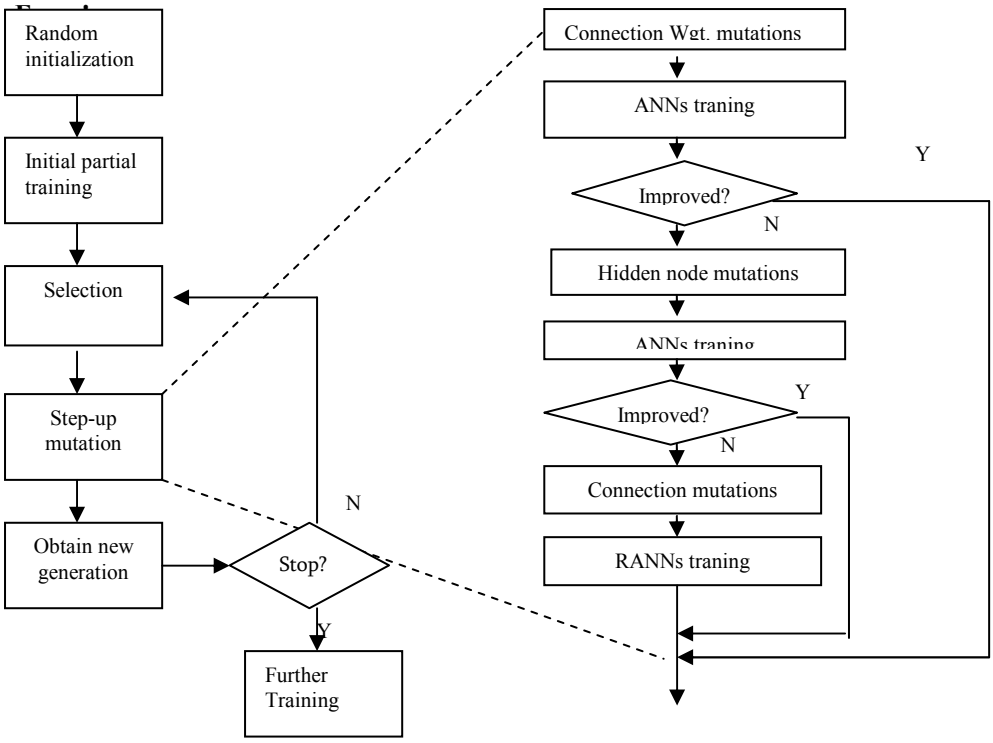

Fig 1: The Stepwise Mutation EANN Algorithm

3.2 Evolutionary Algorithm

The proposed Evolutionary Algorithm used in this paper is a modified EP Net Algorithm from [2]. The proposed EA is named Step-up Mutation Evolutionary ANN. As the name suggests, the algorithm begins with a mutation that has least effect on the structure of ANNs then "step-up" to which have more effects if the previous mutation fails to improve the performance of the ANNs. On the other hand, if the mutation can improve the performance, the mutant will be selected while its parent will be discarded (dual tournament with its parent) and the loop will continue to the next iteration. The main loop is shown in Figure 1. The Algorithm begins with random initialization a set of ANN encoding genetic boxes then creates the corresponding ANNs. All of the ANNs are initially trained to collect their preliminary fitness values. The selected ANNs then goes under the "step-up" mutation with 3 conditional sub-steps, namely, connection weight mutations, hidden node mutations and connection mutations. The mutations are conditionally step-up in such a way that it will step at a sub-step if the trained corresponding ANN's fitness value (in this case, MCV value as described in the next section) is improved. The process repeats until the pre-specified round count is met. The algorithm does not have mating operator but based the evolution solely on the mutation operators.

3.3 The EA Objective

Multifold Cross-Validation (MCV) is a method that makes efficient use of the available data. It is a sample re-used method to estimate prediction risk. Our EA objective is to minimize prediction risk of the RANN. MCV is essentially a perturbation refinement of Cross-Validation (CV) methods. The method can be described as follows:

Let the data set D be divided into m randomly chosen disjoint subsets D_j of roughly equal size.

$$\bigcup_{j=1}^{m} D_j = D, \qquad D_i \cap D_j = \phi \quad for\ \forall\, i \neq j \tag{3}$$

For each disjoint set j, CV is defined as

$$CV_{D_j}(\lambda) = \frac{1}{N_j} \sum_{(x_k, t_k) \in D_j} (t_k - \hat{\mu}_\lambda (D_j, x_k))^2 \tag{4}$$

Where,
$\mu_\lambda(D_j , x_k)$ is an estimator trained on all data except (x, t) ε D_j ,
t_k is the realized (actual) output,
x_k is the vector of all inputs,
N_j is the number of observation in subset D_j.

Cross-Validation (CV) for all available data set of an ANN is a non parametric estimation of the prediction risk.

$$CV(\lambda) = \frac{1}{m} \sum_j CV_{D_j}(\lambda) \tag{5}$$

A refinement is required for CV to become MCV. An ANN is trained on the entire set of data D to obtain estimates $\mu_\lambda(D, x_k)$ with set of weights W_0. The weights W_0 are used as starting point m-fold cross validation production procedure. Each subset D_j is removed from the training data in turn. The ANN is then retrained using the remaining data (starting at W_0, not random initial weights) assuming that deleting a subset from training data set does not lead to a significant different in the locally-optima weights. These perturbed retraining from W_0 yield W_i (i = 1 to m.) The MCV error is calculated for each "perturbed model" by the sum $(t_k - \mu_\lambda(D_j, x_k))$ as an estimation of prediction risk of the model with W_0 [3.]

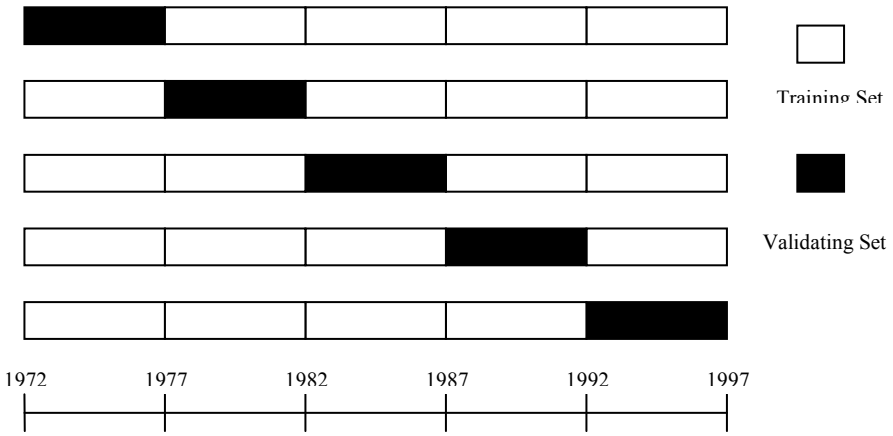

Fig 2: Multi-fold Cross Validation for Selection of Optimal ANN Structure

4 The Experiment

The forecast ANNs and the experiments were conducted with the Step-up Mutation EANN Algorithm proposed above with BPNs and Elman RANs without evolutions as well as forecasting from Linear Least Square Regression (LS) in order to compare their performance. All forecasting have been trained and tested with monthly dividend and split adjusted return series from 1971 to 2007 on 10 selected stocks in US Stock markets, namely Alcoa (AA), Boeing (BA), Caterpillar (CAT), Dupont (DD), Disney (DIS), General Electric (GE), General Motor (GM), Honeywell (HON), HP (HPQ) and IBM (IBM). The independent variables are 12 month time lag (from lag = -1 to -12) of changing on inflation (CPI), default yield spread, term yield spread, fed fund rate, indus-

trial product, money quantity (M), S&P 500, unemployment rate, January effect (dummy variable) also stock returns own lags. The sets of data are paired between a dependent variable and a set of time lags independent variables (10 nominal variables with 12 lags, in totality 120 including lags) to form pattern sets (10 stocks in consideration thus 10 pattern sets.)

For training and testing the regression model (LS), BPNs and Elman RNNs, each pattern sets are break into 16 subsets: 8 subsets for training and 8 subsets for testing (1972 – 1999 for training and 2000 for testing, 1973-2000 for training and 2001 for testing correspondingly to the eighth set 1978-2006 for training and 2007 for testing). But for training and testing the Evolutionary ANNs, each pattern set from 1972-1997 is break into 5 subsets (12 months for 5 years thus 60 patterns each.) The subsets then form 5 training sets corresponding with a validating set for training and testing to obtain MCVs' value (see Figure 2). Then the best group of genes, in which they have minimum MVCs of the last generation for each stock, is deployed to structure ANNs. The same pattern sets are used to train and test LS, BPNs and Elman RNNs.

The parameters for all BPNs, Elman RNNs and final training testing for EANNs are as follows: Epoch limit is 100. Error limit is 0.0. The learning constant is 1. The (initial for EANN) architecture is 1 hidden layer with 2 nodes. The step-up evolution algorithm was run 50 generations on population of 10 ANNs (5 BPNs and 5 Elman RANNs) with conditional mutation probability is 1 (a mutation will affect only there is an improvement).

5 The Results

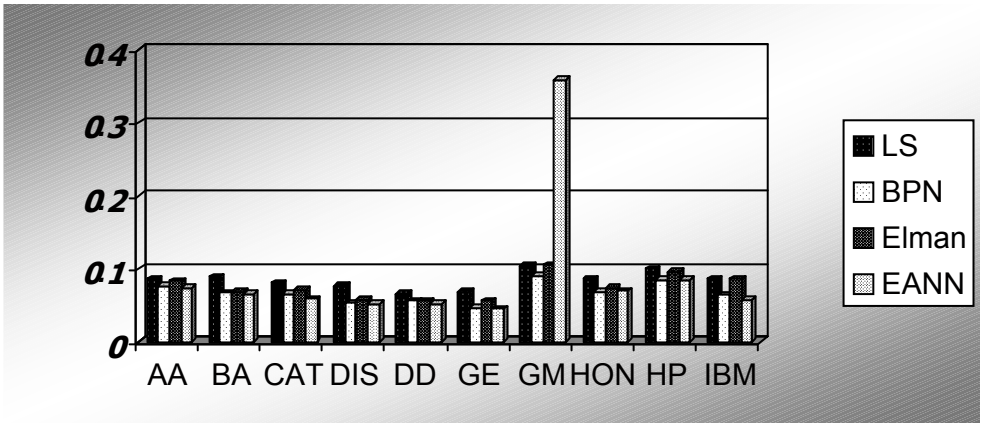

Fig 3: Comapring Average NCV values of Linear Regression (LS), Backpropagation ANN (BPN) and Elman Recurrent ANN (Elman)

In comparison between the four methods of forecasting, namely Least Square Regression (LS), Backpropagation ANN, Elman RAN and Evolutionary ANN (EANN) for ten stock returns, we found that BPNs have a better performances in all stock return forecasts than those of LS and Elman RANs. This results show that to increase complexity of ANN by introducing recurrent networks are not always improve forecasting performances. For the EANNs, they are all but one (namely GM's) at least slightly better than those of BPNs. However, the improvements are not substantial. Most of EANNs have evolved only in initial weights of BPNs (AA, BA, CAT, DD and HON) only a few have evolved both initial weights and structures of BPNs (GE, HON and HPQ). Only for GM and IBM, evolved Elman RANs are selected.

6 Conclusions and Future Work

ANNs have potentials to make a better forecasting of financial and economics time series. In this paper, we go a step further to automatically evolve both initial and structures (number of hidden nodes and number of hidden layers). Comparing with the traditional Linear Regressions, the ANNs show promising results and most of our proposed EANNs can improve the performances of the ANNs as we expected even the improvements are slights. There is an ample room for further researches. Firstly, the running time is quite long about 36 hours for each stock return forecast. This causes us, due to limited computer power, unable to experiment with many populations and many generations as we initially wish. To run the experiment in parallel high performance computer may show some more improvements. The EANN algorithm also can be modified for further improvements such as introducing more variations of ANNs or selection of inputs. To apply the EANN to other related forecasting problems such as to predict stock volatilities, exchange rates, etc. is quite a natural step to do.

References

1. C. Zhou, "Forecasting long- and short- horizon stock returns in a unified framework," *Board of Governors of the Federal Reserve System Finance and Economics Discussion Series FEDS, Paper no. 96-4,* Jan. 1996.
2. W.E. Shape, "Capital Assets Prices: A Theory of Market Equilibrium and Conditions of Risk," *The Journal of Finance, 19(3),* 1964, pp. 425-442
3. J. Lintner., "The Valuation of Risk Assets and the Selection of Risky Investments in Stock Portfolio and Capital Budgets," *Review of Economics and Statistics, 47(1), 1965*, pp. 13-37
4. S.A. Ross, "The Arbitrage Theory of Capital asset Pricing," The Journal of Economic Theory 13(3), 1967, pp. 341-360
5. R.R. Roll and S.A. Ross, "An Empirical Investigation of the Arbitrage Pricing Theory," *The Journal of Finance*, 39(5), 1980, pp. 1073-1104
6. Y. Kwon, S. Choi and B. Moon, "Stock Prediction Based on Financial Correlation," In Preceeding of GECCO 2005.
7. Y. Kwon and B. Moon, "A hybrid neurogenetic approach for stock forecasting," *IEEE Transactions on Neural Networks, vol. 18, No. 3, pp. 851-864.* May 2007.
8. X. Yao, "Evolutionary artificial neural networks," *Proceeding of the IEEE, vol. 87, No. 9, Sep. 1999.*
9. J. Moody, "Forecasting the economy with neural nets : A survey of challenges and solutions," in *Neural Networks : Tricks of the Trade,* G. B. Orr and K. Muller, Eds., Berlin, Germany : Springer-Verlag, 1998, pp. 347-371.
10. G. Armano, M. Marchesi and A. Murru, "A hybrid genetic-neural architecture for stock indexes forecasting," *Information Sciences, vol. 170,* pp. 3-33.
11. R. E. Oberuc, *Dynamic Portfolio Theory and Management,* New York, NY : McGraw-Hill, 2004.