Multi-Layer Allocated Learning Based Neural Network for Resource Allocation Optimization

P. C. Ko[1], P. C. Lin[2], J. A. You[2] , Y. J. Tien[1], H. C. Wang[1]

[1]Department of Information Management
[2]Institute of Finance and Information
[1,2]National Kaohsiung University of Applied Sciences, Taiwan ROC.

Abstract

The investor's asset allocation choice deeply depends on the trade-off between risk and return. The well-known mean variance method requires predetermined risk and expected return to calculate optimal investment weights of portfolio. The artificial neural network (ANN) with nonlinear capability is proven to solve large-scale complex problem effectively. However, the traditional ANN model cannot guarantee to produce reasonable investment allocation, because the summation of investment weight may not preserve 100% in output layer. This article introduces a multi-layer allocated learning based neural network model and takes financial portfolio as an example to optimize assets allocation weights. This model dynamically adjusts the investment weight as a basis of 100% of summing all of asset weights in the portfolio. The experimental results demonstrate the feasibility of optimal investment weights and superiority of ROI of buy-and-hold trading strategy compared with benchmark TSE (Taiwan Stock Exchange).

Keywords: Resource allocation, neural network, allocated learning based NN, portfolio, investment weight.

1. Introduction

Most investment professionals or investors consider asset allocation as the most important part of portfolio construction. Asset allocation of portfolio is concerned with the percentage of the overall portfolio value allocated to each portfolio individual. The mean variance model for the portfolio asset allocation is one of the best known models. This model requires satisfying the two conflicting optimization criteria which minimizes risk with predetermined the expected return of portfolio. Besides, the mean variance model may be desirable to restrict the number of assets in a portfolio and the percentage of the portfolio attends to any specific asset. The searching of efficient set frontier became much difficult, if that restriction exit.

In portfolio applications, using ANNs to portfolio management has gained interest in recent years. Hung, Liang and Liu [11] integrate the arbitrage pricing theory (APT) and ANNs to extracting risk factors and generating individual in portfolio. The empirical results indicate the integrated method beats the benchmark and ARIMA model. Chapados [9] demonstrated the success of ANNs with asset allocation framework according to a Value-at-Risk adjusted profit criterion for making asset allocation decisions. Both the forecasting and decision models are significantly outperforming the benchmark market performance. Eakins and Stansell [12] examine whether superior investment returns can be earned by using ANNs to perform forecasts based on a set of financial ratio to determine the intrinsic value of assets to enter the property portfolio. They find that the value ratio provides useful information that permits the selection of portfolio s that provide investment returns superior to the DJIA and S&P500. Hung, Cheung and Xu [5] present an extended adaptive supervised learning decision EASLD trading system to enhance portfolio management. Their researches take a balance between the expected returns and risks. Plikynas, Salalauskas and Poliakova [3] using ANNs to control nonlinear dynamics f heterogeneous foreign investment impact on national capitalization structure. The results of their research show better than multidimensional linear regression forecasting performance. Ellis and Wilson [2] applied ANNs to the Australian property sector stocks to construct a variety of value portfolios. Their risk-adjusted performances show the value portfolios outperform the benchmark by as much as 7.14%.

Unfortunately, the traditional ANN model cannot produce reasonable allocation ratios, i.e. the summation of produced allocation ratios cannot retain 100%, during the learning process. The aim of this paper is to introduce an allocated learning based neural network model to optimize the assets allocation in portfolio that will outperform the market. This approach suggests using quadratic programming to obtain the expected maximum variance of risk factor to risk assets returns of portfolio, and proposes a dynamic weight modification as a basis of 100% of summing all of allocation ratios in the portfolio. Through the experimental results, the complex portfolio asset allocation management would be solved effectively by our model.

2. Multi-Layer Allocated Learning (MLAL) based Neural Network

In this section, an allocated learning algorithm applied to multi-layer neural network (MLALNN) with n inputs, one hidden-layer with h neurons and m outputs shown in Fig 1 is introduced. Let $a^I = [a_1^I \; a_1^I \cdots a_i^I \cdots a_n^I]^T$ denote the input signal, $a^H = [a_1^H \; a_1^H \cdots a_r^H \cdots a_l^H]^T$ denote the hidden signal and $y = [y_1 \; y_1 \cdots y_j \cdots y_m]^T$ denote the output signals. w^I and w^H represents the weight matrix shown in Equation (1) and Equation (2), where $w_{r,i}^I$ refers to the synaptic weight connecting the hidden layer of neuron r to the input layer of neuron i, $w_{j,r}^H$ refers to the synaptic weight connecting the output layer of neuron j to the hidden layer of neuron r . $\varphi(\cdot)$, b_r^H , b_j^O and y_j are the activation function, the bias applied to neuron r in hidden layer, the bias applied to neuron j in output layer and the output signals, respectively. a_r^H and v_j is denoted as the weighted sum of all synaptic inputs plus bias.

$$w^I = \begin{bmatrix} w_{1,1}^I & w_{1,2}^I & \cdots & w_{1,n}^I \\ w_{2,1}^I & w_{2,2}^I & \cdots & w_{2,n}^I \\ \vdots & \vdots & \vdots & \vdots \\ w_{l,1}^I & w_{l,2}^I & \cdots & w_{l,n}^I \end{bmatrix} \tag{1}$$

$$w^H = \begin{bmatrix} w_{1,1}^H & w_{1,2}^H & \cdots & w_{1,l}^H \\ w_{2,1}^H & w_{2,2}^H & \cdots & w_{2,l}^H \\ \vdots & \vdots & \vdots & \vdots \\ w_{m,1}^H & w_{m,2}^H & \cdots & w_{m,l}^H \end{bmatrix} \tag{2}$$

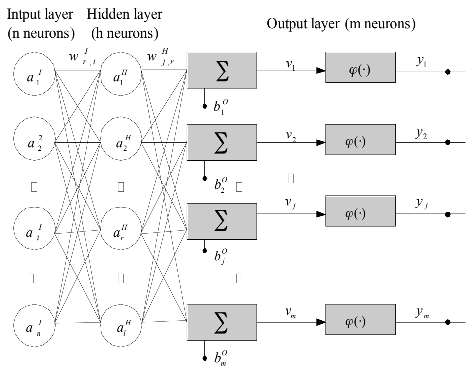


Fig 1. Multi-Layer Combinatorial Learning based Neural Network

Assume $d = \left[d\text{¡B}d\text{¡B}\cdot\text{¡B}d\text{¡B}\cdots\text{¡B}d_m \right]^T$ and $e = \left[e\text{¡B}e\text{¡B}\cdot\text{¡B}e\text{¡B}\cdots\text{¡B}e_m \right]^T$ represents the desired responses and error signal at the output layer. The instantaneous value *ξ(k)* of the total error energy at iteration k is defined as the summation over all neurons in the output layer in Equation (3).

$$\xi(k) = \frac{1}{2}\|\mathbf{d} - \mathbf{y}\|^2 = \frac{1}{2}\sum_{j=1}^{m} e_j^2(k) \tag{3}$$
$$where \; \mathbf{e} = \mathbf{d} - \mathbf{y}$$

To do the minimization, a manner similar to the *Least Mean Square* (LMS) algorithm is adopted to apply a correction $\Delta w_{j,r}^H(k)$ shown in Equation (4) to its corresponding synaptic weights $w_{j,r}^H(k)$, where $\eta_{j,r}^H$ is the learning rate. To obtain better assets allocation in a portfolio problem, the summation of all output signals must be 100% (i.e. $\sum_{j=1}^{m} y_j = 1$) in learning epoch. $\eta_{j,r}^H$ is not a fixed value commonly used in the conventional neural network. Especially, $\eta_{j,r}^H$ is a variable calculated to maintain the summation of outputs always equal to 100% in our allocated learning based algorithm.

$$\Delta w_{j,r}^H(k) = -\eta_{j,r}^H \frac{\partial \xi(k)}{\partial w_{j,r}^H(k)} \tag{4}$$

It means

$$w_{j,r}^H(k+1) = w_{j,r}^H(k) + \eta_{j,r}^H \frac{\partial \xi(k)}{\partial w_{j,r}^H(k)} \tag{5}$$

According to the chain rule of calculus, we may express this gradient as

$$\frac{\partial \xi(k)}{\partial w^H_{j,r}(k)} = \frac{\partial \xi(k)}{\partial e_j(k)} \cdot \frac{\partial e_j(k)}{\partial y_j(k)} \cdot \frac{\partial y_j(k)}{\partial v_j(k)} \cdot \frac{\partial v_j(k)}{\partial w^H_{j,r}(k)} \tag{6}$$

where

$$\frac{\partial \xi(k)}{\partial e_j(k)} = \frac{\partial(\frac{1}{2}\|d - y\|^2)}{\partial e_j(k)} = e_j(k) \tag{7}$$

$$\frac{\partial e_j(k)}{\partial y_j(k)} = -1 \tag{8}$$

$$\frac{\partial y_j(k)}{\partial v_j(k)} = \varphi'(v_j(k)) \tag{9}$$

$$\frac{\partial v_j(k)}{\partial w^H_{j,r}(k)} = \frac{\partial(\sum_{r=1}^{l} w_{j,r}^H a_r^H + b_j^O)}{\partial w^H_{j,r}} = a_r^H \tag{10}$$

Let $\delta_j(k)$ denote the local gradient and be defined in Equation (11)

$$\delta_j(k) = -\frac{\partial \xi(k)}{\partial v_j(k)} = e_j(k) \cdot \varphi'(v_j(k)) \tag{11}$$

Equation (4) should be rewritten as Equation (12)

$$\Delta w^H_{j,r}(k) = \eta^H_{j,r} \cdot \delta_j \cdot a^H_r \tag{12}$$

After adjusting each weight $w_{j,i}$, the summation of output signals would be 100%. It means that.

$$\sum_{j=1}^{m} \sum_{r=0}^{l} \varphi(w_{j,r}^H(k+1) \cdot a_r^H) = 1 \tag{13}$$
$$where \; w_{j,0} = b_j \; and \; a_0^H = 1$$

Substituting Equation (5), (12) into Equation (13), we obtain

$$\sum_{j=1}^{m} \sum_{r=0}^{l} \varphi((w_{j,r}^H(k) + \eta^H_{j,r} \cdot \delta_j \cdot a^H_r) \cdot a_r^H) = 1 \tag{14}$$

Multiplying $\eta^H_{row,col}$ (*row=1, 2, ..., m and col=1, 2, ..., l*) with Equation (14) independently, we obtain following general equation form.

$$F(\eta^{H}_{row,col}) = \eta^{H}_{row,col} \sum_{j=1}^{m}\sum_{r=0}^{l} \varphi\left[\left(w^{H}_{j,r}(k) + \eta^{H}_{j,r}\cdot\delta_{j}\cdot a^{H}_{r}\right)\cdot a^{H}_{r}\right] - \eta^{H}_{row,col} = 0 \tag{15}$$

Partial differentiating Equation (15) with $\eta^{H}_{row,col}$, we obtain

$$\begin{aligned}\frac{\partial F(\eta^{H}_{row,col})}{\partial \eta^{H}_{row,col}} &= \varphi\left[\Delta w^{H}_{row,col}\cdot a^{H}_{col}\right] + \Delta w^{H}_{row,col}\cdot a^{H}_{col}\cdot \varphi'\left[\left(\eta^{H}_{row,col}\cdot\delta_{row}(k)\cdot a^{H}_{col}\right)\cdot a^{H}_{col}\right] \\ &+ \sum_{\substack{j=1 \\ j\neq row\ and\ r\neq col}}^{m}\sum_{r=0}^{n}\varphi\left[\Delta w^{H}_{j,r}\cdot a^{H}_{r}\right] + \sum_{j=1}^{m}\sum_{r=0}^{n}\varphi\left[w^{H}_{j,r}(k)\cdot a^{H}_{r}\right] - 1 = 0\end{aligned} \tag{16}$$

let φ(x)=x. Equation (16) would be simplified as follows:

$$2\cdot\Delta w^{H}_{row,col}\cdot a^{H}_{col} + \sum_{\substack{j=1 \\ j\neq row\ and\ r\neq col}}^{m}\sum_{r=0}^{n}\varphi\left[\Delta w^{H}_{j,r}\cdot a^{H}_{r}\right] = 1 - \sum_{j=1}^{m}\sum_{r=0}^{n} w^{H}_{j,r}(k)\cdot a^{H}_{r} \tag{17}$$

Expanding Equation (17) respect to $\eta^{H}_{row,col}$, where *row=1, 2, ..., m* and *col=1, 2, ..., l.* we obtain

$$\begin{bmatrix} 2a_1^{H\,2}\delta_{1,1} & a_2^{H\,2}\delta_{1,2} & \cdots & a_r^{H\,2}\delta_{1,r} & \cdots & a_r^{H\,2}\delta_{m,l} \\ a_1^{H\,2}\delta_{1,1} & 2a_r^{H\,2}\delta_{1,2} & \cdots & a_r^{H\,2}\delta_{1,r} & \cdots & 2a_r^{H\,2}\delta_{m,l} \\ \vdots & a_r^{H\,2}\delta_{1,2} & \ddots & \vdots & \cdots & \vdots \\ a_1^{H\,2}\delta_{1,1} & \cdots & & 2a_r^{H\,2}\delta_{1,r} & & 2a_r^{H\,2}\delta_{m,l} \\ \vdots & \cdots & \cdots & \vdots & \ddots & \vdots \\ a_1^{H\,2}\delta_{1,1} & a_t^{H\,2}\delta_{1,2} & \cdots & a_r^{H\,2}\delta_{1,r} & \cdots & 2a_r^{H\,2}\delta_{m,l} \end{bmatrix} \cdot \begin{bmatrix} \eta^{H}_{1,1} \\ \eta^{H}_{1,2} \\ \vdots \\ \eta^{H}_{j,r} \\ \vdots \\ \eta^{H}_{m,l} \end{bmatrix} = \begin{bmatrix} 1-\sum_{j=1}^{m}\sum_{r=0}^{l} w^{H}_{j,r}(k)\cdot a^{H}_{r} \\ 1-\sum_{j=1}^{m}\sum_{r=0}^{l} w^{H}_{j,r}(k)\cdot a^{H}_{r} \\ \vdots \\ 1-\sum_{j=1}^{m}\sum_{r=0}^{l} w^{H}_{j,r}(k)\cdot a^{H}_{r} \\ \vdots \\ 1-\sum_{j=1}^{m}\sum_{r=0}^{l} w^{H}_{j,r}(k)\cdot a^{H}_{r} \end{bmatrix} \tag{18}$$

Let

$$H = \begin{bmatrix} 2a_1^{H\,2}\delta_{1,1} & a_2^{H\,2}\delta_{1,2} & \cdots & a_r^{H\,2}\delta_{1,r} & \cdots & a_r^{H\,2}\delta_{m,l} \\ a_1^{H\,2}\delta_{1,1} & 2a_r^{H\,2}\delta_{1,2} & \cdots & a_r^{H\,2}\delta_{1,r} & \cdots & 2a_r^{H\,2}\delta_{m,l} \\ \vdots & a_r^{H\,2}\delta_{1,2} & \ddots & \vdots & \cdots & \vdots \\ a_1^{H\,2}\delta_{1,1} & \cdots & & 2a_r^{H\,2}\delta_{1,r} & & 2a_r^{H\,2}\delta_{m,l} \\ \vdots & \cdots & \cdots & \vdots & \ddots & \vdots \\ a_1^{H\,2}\delta_{1,1} & a_t^{H\,2}\delta_{1,2} & \cdots & a_r^{H\,2}\delta_{1,r} & \cdots & 2a_r^{H\,2}\delta_{m,l} \end{bmatrix}$$

$$R = \begin{bmatrix} 1-\sum_{j=1}^{m}\sum_{r=0}^{l} w^{H}_{j,r}(k)\cdot a^{H}_{r} \\ 1-\sum_{j=1}^{m}\sum_{r=0}^{l} w^{H}_{j,r}(k)\cdot a^{H}_{r} \\ \vdots \\ 1-\sum_{j=1}^{m}\sum_{r=0}^{l} w^{H}_{j,r}(k)\cdot a^{H}_{r} \\ \vdots \\ 1-\sum_{j=1}^{m}\sum_{r=0}^{l} w^{H}_{j,r}(k)\cdot a^{H}_{r} \end{bmatrix},\quad \eta^{H} = \left[\eta^{H}_{1,1}\ \ \eta^{H}_{1,2}\ \ \cdots\ \ \eta^{H}_{j,r}\ \ \cdots\ \ \eta^{H}_{m,l}\right]^{T}$$

Then,

$$\eta^{H} = H^{-1}\cdot R \tag{19}$$

3. Experimental Results

We specify the sampling data, sliding window process, parameters setting and analysis of results in this section. This allocated learning based ANN model was written in Borland C++ Builder 6.0 and run in Microsoft Window XP environment.

3.1 Data and Sample

Twenty one companies are selected to be our testing targets from Taiwan 50 Index Constituents. The descriptive statistics of our target companies shown in Table 1 must be listed and traded in the Taiwan Stock Exchange (TSE). The relevant data are collected from Taiwan Information Time Plus.

3.2 Sliding window process

The data encompasses the entire period from January 1, 2000 to September 30, 2005. The training phase (*k*), the validation phase (*v*), and the test phase (*l*) are one period in each sliding window (*SW*) as shown in Fig 2 to cross-validate our model.

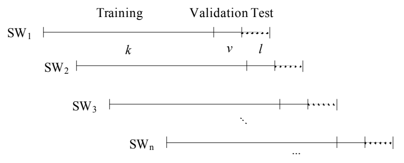


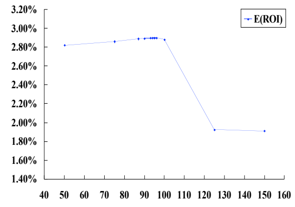
Fig 2. Sliding windows simulation process

3.3 Neural Networks Parameters

Effective neural network construction required employment of various optimizations setting. The root mean error square (RMSE) and expected return are considered as our key performance indexes in difference training iterations. RMSE is defined in Equation (20), where y_t refers to the real testing data and $\hat{y}_t$ is produced by our MLALNN model. Minor changes in these parameters seem not to have a major effect on the performance in our experimental results.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}{N}} \tag{20}$$

The suitable number of iterations is set to 95 and number of neuron in hidden layer is set to 5 shown in Fig 3 and Fig 4, respectively, where the right side and left side of y-axis denote expected return E_{ROI}.



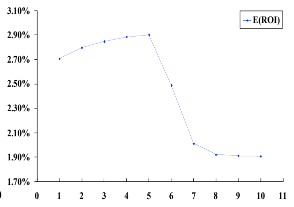


Fig 3. E_{ROI} in different training iterations.

Fig 4. E_{ROI} with different number of neuron in hidden layer.

3.4 Analysis of Results

The performances of our proposed MLALNN model are summarized into two parts: (1) the comparisons of investing returns in each sliding window, and (2) analysis of ROI contributed by different asset allocations in each sliding window. Each sliding window is done five times and their mean values are taken to obtain more general results. Let ROI_{MLALNN} and ROI_{TSE} refer to the obtained ROI by using MLALNN model or simply TSE indexes. From Table 3, it is shown that the average of ROI_{MLALNN} (14.2945%) is greater than average of ROI_{TSE} (0.5291%). Moreover, ROI_{MLALNN} is superior to ROI_{TSE} in each sliding window. Even if the ROI_{MLALNN} in SW_1(-0.9845%) and SW_2(-3.7299%) are negative, they are also higher than the corresponding ROI_{TSE} (-5.1540%) and (-9.8803%). These results are also shown in Fig 5. It demonstrates that MLALNN obtain better asset allocations effectively.

Table 3. Comparisons of ROI_{MLALNN} and ROI_{TSE} in each sliding window

Sliding Windows	ROI_{MLALNN}	ROI_{TSE}
SW_1	-0.9845%	-5.1540%
SW_2	-3.7299%	-9.8803%
SW_3	3.2852%	-7.9346%
SW_4	16.8383%	-3.5109%
SW_5	15.2256%	0.4236%
SW_6	20.8753%	6.9391%
SW_7	43.3789%	17.9721%
SW_8	11.6140%	2.9929%
SW_9	22.1479%	2.9142%
Average	14.2945%	0.5291%

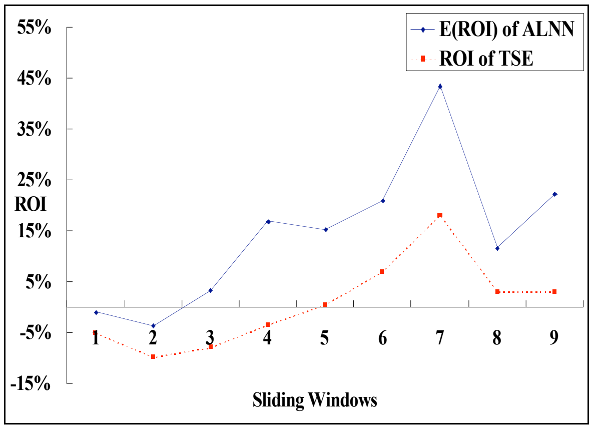


Fig 5. Comparisons of ROI_{MLALNN} and ROI_{TSE} in each sliding window

4. Conclusions

This paper introduces a novel allocated learning based neural network model to optimize investment weight of portfolio that will outperform the market. This model proposes a dynamic weight modification as a basis of 100% of summing all of asset weights in the portfolio. Results on 21 companies selected to be our testing targets from Taiwan 50 Index Constituents demonstrate the feasibility of optimal investment weights and superiority of ROI based on buy-and-hold trading strategy. Through the experimental results, the complex portfolio asset allocation management would be solved effectively by our model. From our experiments, it appears that using allocated learning based neural network model will converge to two dimensions (the highest expected return and the lower RMSE), simultaneously. Using MLALNN model produces better return of investment than TSE in each sliding window. Our MLALNN model recommends increasing/decreasing investment weight of optimistic/pessimistic prospects of assets effectively.

5. References

[1] Craig Ellis and Patrick J. Wilson, "Can a neural network property portfolio selection process outperform the property market?" Journal of Real Estate Portfolio Management, Vol.11, pp.105-121, 2005.

[2] Darius Plikynas, Leonidas Salalauskas and Alina Poliakova, "Analysis of foreign investment impact on the dynamics of national capitalization structure:a computational intelligence approach," Research in International Business and Finance, Vol. 19, pp.304-332, 2005.

[3] Kei-Keung Hung, Yiu-Ming Cheung and Lei Xu, "An Extended ASLD Trading System to Enhance Portfolio Management," IEEE Transaction on Neural Networks, Vol.14, pp.413-425.

[4] Nicolas Chapados and Yoshua Bengio, "Cost functions and model combination for VaR-based asset allocation using neural network," IEEE Transaction on Neural Networks, 2001, Vol. 12, pp. 890.

[5] Shin-Yan Hung, Ting-Peng Liang and Victor Wei-Chi Liu, "Integrating arbitrage pricing theory and artificial neural networks to support portfolio management," Decision Support Systems, Vol. 18, pp. 301-316, 1996.

[6] Stanley G. Eakins and Stanley R. Stansell, "Can value-based stock selection criteria yield superior risk-adjusted returns: an application of neural networks," International Review of Financial Analysis, Vol. 12, pp.83-97, 2003.