Journal of Big Data

RESEARCH Open Access

Modelling shares choice to enter in a portfolio using artificial neural networks (ANN)

Adler Haymans Manurung[1], Christina Natasha[2] and Widodo Budiharto[2*]

*Correspondence:
wbudiharto@binus.edu
[2] School of Computer Science, Bina Nusantara University, Jl. K.H. Syahdan No. 9, West Jakarta 11480, Indonesia
Full list of author information is available at the end of the article

Abstract

Shares choice to enter a portfolio is a good topic in finance and management, as it affects the portfolio performance which is managed by a Fund Manager. In this research, we aim to create an artificial neural network model to choose a share to enter a portfolio based on its financial factors and big data about the financial condition of companies. The artificial neural network model has 15 input nodes of attributes associated with a company's financial situation, 8 hidden layer nodes, and 1 output node. The accuracy of the model is 85.71%, with a learning rate of 0.05 trained over 2000 iterations.

Keywords: Shares choice, Neural network, ANN, Big data

Introduction

Shares choice to enter a portfolio a special topic in finance and management and is very interesting to be discussed by both practitioners and academics. Shares choice to enter in a portfolio is very important for a Fund Manager when he manages his client's fund. When a Fund Manager does not properly choose the share, it will affect target return and Investors will redeem their fund and looking the good performance. Share choice to enter a portfolio is a process to build a portfolio. Good shares portfolio consists of many shares that have good performance from the past and in the future. Research for shares choice becomes very important because the result will make the fund be sought by investors as a good investment. Good shares in a portfolio is a power to Fund Manager to sell it and investor to become large to buy the fund.

Markowitz [7] introduced a method of shares choice to enter a portfolio using risk and return. Elton et al. introduced an alternative to choose a share to enter a portfolio. The method is known as Excess Return to Beta which is using risk (beta) and returns. Wallingford [17] used the Single-Index, Multi-Index, and Full-Covariance Model to choose a share to enter a portfolio. Saaty et al. [15] stated that the Analytical Hierarchy Process (AHP) can be used to choose a share to enter a portfolio. AHP was developed by Saaty in 1970. Khaksari et al. [5] supported Saaty et.al's research to use AHP to choose share to

enter into a portfolio. Jones [4] introduced a network analysis to build a portfolio. Reily and Brown ([13], 272) stated that an optimal portfolio is an efficient portfolio to have the highest utility for the investor's certainty.

Previous research used risk and return to choose a share enter in a portfolio, but it never discusses characteristics of the companies whose shares were entered into a portfolio, for example, market capitalization and financial ratio company. Farrell [3] investigated stock to enter in a portfolio using homogeneous stock grouping. Based on some previous researches, Common stocks tend to group naturally, according to their price behavior [8] and also P/E [10]. Wainscott [16] found that the changing of correlation between assets will significantly affect mixed-asset portfolio from one period to another period. Peritt and Lavine [11] stated that asset allocation is very important to build a portfolio. Marmer [9] explored an efficient frontier for Canadian's Portfolio. Fama and French [2] explored another factor to affect portfolio return. Reinganum [14] did research and stated that market capitalization has an impact on managing portfolio. Manurung [6] used the Logit Method to choose a share to enter into a portfolio using financial ratios.

Shares choice to enter in a portfolio is still developed for research, especially to explore new methods. Recently, a research on Shares choice to enter in a portfolio was conducted using Artificial Neural Networks (ANN). Ashwood [1] explored ANN as a method to predict stock prices in shares portfolio, with a result of better than 50% accuracy for almost all of the stocks and portfolios tested. Rasekhschaffe and Jones [12] used Machine Learning to select stocks for building portfolio using deep neural networks (DNN). This paper explores the usage of ANN to choose a share to enter in a portfolio using financial situation data of several Indonesian companies.

Meanwhile, big data techniques, especially the volume of data, can be used in modeling shares choice to gather data to use as training and testing data for the ANN. Zhang et al. [18] stated that ANNs can extract features from raw data while big data provides tremendous training samples for the ANN, creating a mutual reinforcement. Thus, the use of big data is beneficial to improve this experiment in the future.

Proposed methods

The algorithms used for building the ANN was supervised stochastic learning with back-propagation and steepest gradient descent, with bias nodes at the hidden and output layer. The 31 data rows gathered were split into 80% training set and 20% test set. The model of the ANN was as shown in Fig. 1.

The ANN takes 15 input data, X, of attributes associated with a company's financial situation:

- Cash Ratio (X1)
- Current Ratio (X2)
- Cash T. Assets (X3)
- Cash Debt US$ (X4)
- Free CF Debt US$ (X5)
- Debt US$ Equity (X6)
- MV E Debt US$ (X7)

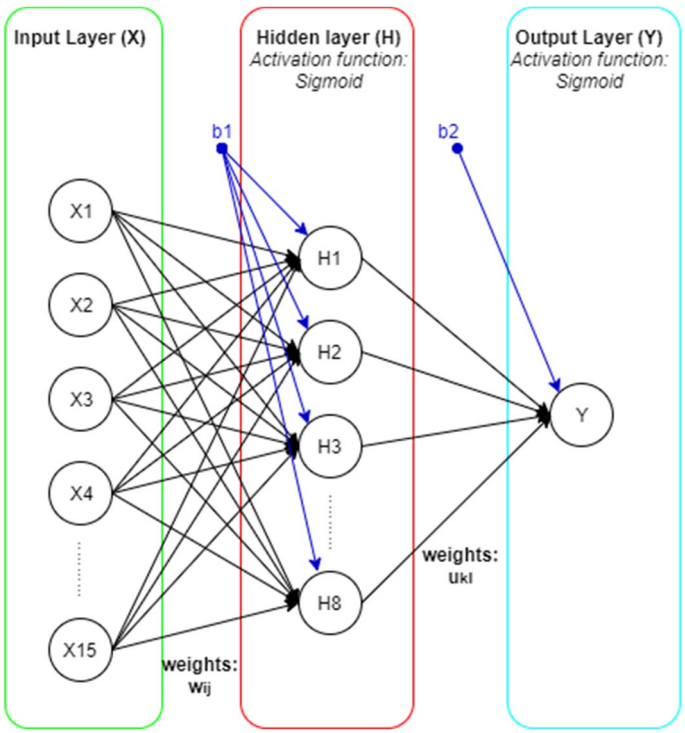

Fig. 1 The proposed model of ANN used in this research

- Sales T. Assets (X8)
- Net Inc. T. Assets (X9)
- Net Inc. Equity (X10)
- Operational Profit (X11)
- R E T. Assets (X12)
- EBIT T. Assets (X13)
- Debts T. Assets (X14)
- Free CF T. Assets (X15).

The hidden layer, H, which consists of 8 nodes, then extracts features from these data and sends a signal to the output layer, Y. The result is the probability of the company being bankrupt or not. To calculate the accuracy of the model, the test set is fed into the trained network, and if the probability is above 70%, then the company is bankrupt (Y = 1), otherwise, it is not bankrupt (Y = 0). This output is then compared with the actual labels of the test set.

The equations used in this research are as follows:

Linear combinations:

$$o_j = \sum_{i=1}^{a} w_{ij} x_i + b_1 \tag{1}$$

$$p_l = \sum_{k=1}^{b} u_{kl} h_m + b_2 \tag{2}$$

with: o_j, p_n as linear combination results for the hidden layer nodes and output layer nodes respectively; w_{ij} as weights from the input layer to the hidden layer; u_{kl} as weights from the hidden layer to the output layer; x_i as input nodes; h_m as hidden layer nodes; b_1, b_2 as biases of the hidden layer and the output layer respectively; i, k as the index of nodes in the input layer and the hidden layer respectively; j, l as the index of nodes in the hidden layer and the output layer respectively; a, b as the number of nodes in the input layer and the hidden layer respectively.

Sigmoid activation function:

$$h_j = \frac{1}{1 + e^{(-o_j)}} \tag{3}$$

$$y = \frac{1}{1 + e^{(-p_l)}} \tag{4}$$

with: h_j as the hidden layer nodes' output; y as the output layer node's output; o_j, p_l as linear combination results for the hidden layer nodes and output layer nodes respectively.

Mean squared error:

$$E = \frac{1}{2}(t - y)^2 \tag{5}$$

with: E as error (or loss); t as the target output; y as the output layer node's output

Steepest gradient descent:

$$w_{new} = w_{old} + \Delta w \tag{6}$$

$$\Delta w = -\alpha g_n(w) \tag{7}$$

$$b_{new} = b_{old} + \alpha(y - t)(1 - y)(y) \tag{8}$$

with: w_{new} as the new weight value; w_{old} as the current weight value; Δw as the adjustment made to the current weight value; α as learning rate$g_n(w)$ as a function of weight; b_{new} as the new bias value; b_{old} as the current bias value; y as a vector of the output layer nodes' output; t as a vector of the target output.

Chain rule for the hidden layer → output weights (u):

$$\frac{\partial E}{\partial u} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial q}\frac{\partial q}{\partial u}$$

$$\frac{\partial E}{\partial u} = \big(y - t\big)\big(1 - y\big)\big(y\big)(h_2) \tag{9}$$

with: $g_n(u)$ as a function of the hidden layer → output layer weights; $\frac{\partial E}{\partial u}$ as the partial derivative of Error against the hidden layer → output layer weights; $\frac{\partial E}{\partial y}$ as the partial derivative of Error against the output layer; $\frac{\partial y}{\partial q}$ as the partial derivative of the output layer against the output layer's linear combination; $\frac{\partial q}{\partial u}$ as the partial derivative of the output layer's linear combination against the hidden layer → output layer weights.

Chain rule for the input layer → hidden layer weights (w):

$$g_n(w) = \frac{\partial E}{\partial w}$$

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial q}\frac{\partial q}{\partial h}\frac{\partial h}{\partial p}\frac{\partial p}{\partial w}$$

$$\frac{\partial E}{\partial w} = \big(y - t\big)\big(1 - y\big)\big(y\big)(u)(1 - h)(h)(x) \tag{10}$$

with: $g_n(w)$ as a function of the input layer → hidden layer weights; $\frac{\partial E}{\partial w}$ as the partial derivative of Error against the input layer → hidden layer weights; $\frac{\partial q}{\partial h}$ as the partial derivative of the output layer's linear combination against the hidden layer; $\frac{\partial h}{\partial p}$ as the partial derivative of the hidden layer against the input layer's linear combination; $\frac{\partial p}{\partial w}$ as the partial derivative of the hidden layer's linear combination against the input layer → hidden layer weights.

Meanwhile, the pseudocode of the algorithm used in this experiment is:

```
begin
import libraries
import dataset as X (features) and y (label)
normalize X
split dataset into training and test set
define ANN architecture
define learning rate as lr
define number of iterations as number_of_iter
initialize output_errors array
function train(X_train, y_train, lr, number_of_iter):
  initialize weight and biases randomly using numpy.random.randn()
  for iter in range(number_of_iter):
    store shuffled list of len(X_train) as r
    for i in r:
      reshape biases into vectors
      do forward pass
      reshape vector variables into matrices
      gradient descent backpropagation algorithm
      update weights and biases
    end for
  end for
  calculate mean of errors
  if iter is 0.1 * number_of_iter:
    report mean of errors
  end if
  return weights and biases
end function
function predict(X_test, weights and biases):
  initialize array y_pred
  for i in range(len(X_test)):
    do forward pass
  end for
  return y_pred
end function
initialize array new_ypred
for yp in y_pred:
  if yp > 0.7:
    yp = 1
  else:
    yp = 0
  end if
  new_ypred.append(yp)
end for
y_pred = new_ypred
initialize int trueCount = 0
for i in range(len(y_pred)):
  if y_pred[i] equals y_test[i]:
    add 1 to trueCount
  end if
end for
print trueCount/len(y_pred) * 100 as accuracy
print error vs. iteration graph
end
```

Experimental results

We used a total of 31 lines of data for training and testing. The data was split into 80% training set and 20% test set prior to executing the training algorithm using the architecture defined in Fig. 1. It was trained with various learning rates: 0.5, 0.1 and 0.05. It was also trained with various amounts of iterations.

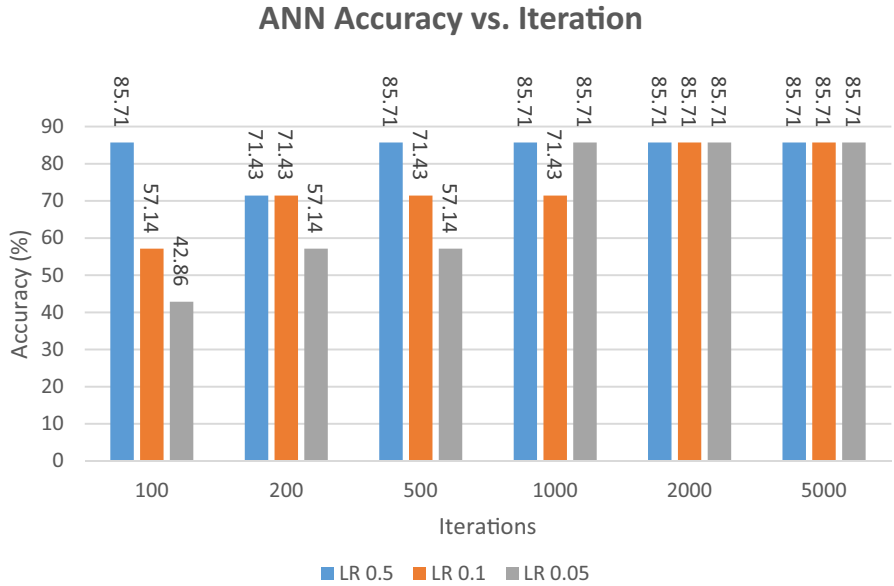

Fig. 2 Accuracy of ANN over iterations with different learning rates

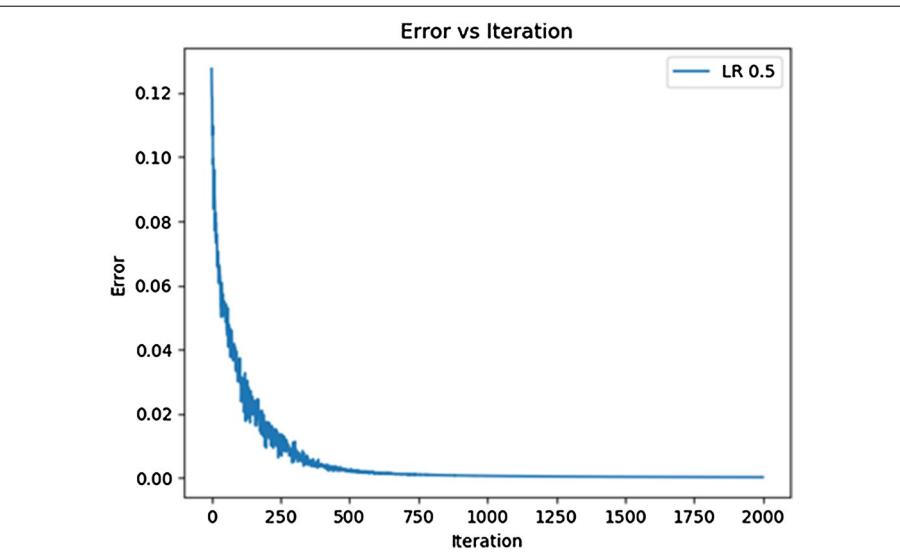

Fig. 3 Error vs. iteration graph for LR 0.5 over 2000 iterations

The test results can be seen in Fig. 2, while the error vs. iteration graph for each learning rate for 2000 iterations can be seen in Figs. 3, 4, 5.

Discussion

As can be seen in Fig. 2, the maximum accuracy of the model is 85.71%, with the results stabilizing on 2000 iterations. By comparing Figs. 3, 4, and 5, we can see that the smoothest, thus the most stable, graph is the one with 0.05 learning rate. From these results, we can say

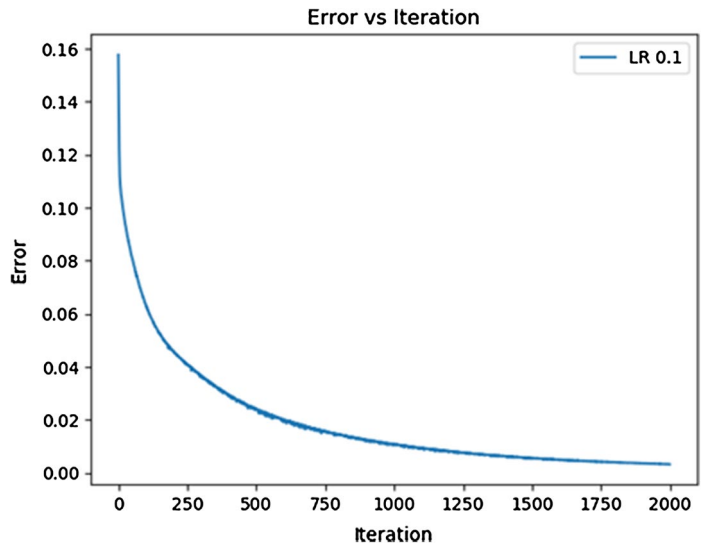

Fig. 4 Error vs. iteration graph for LR 0.1 over 2000 iterations

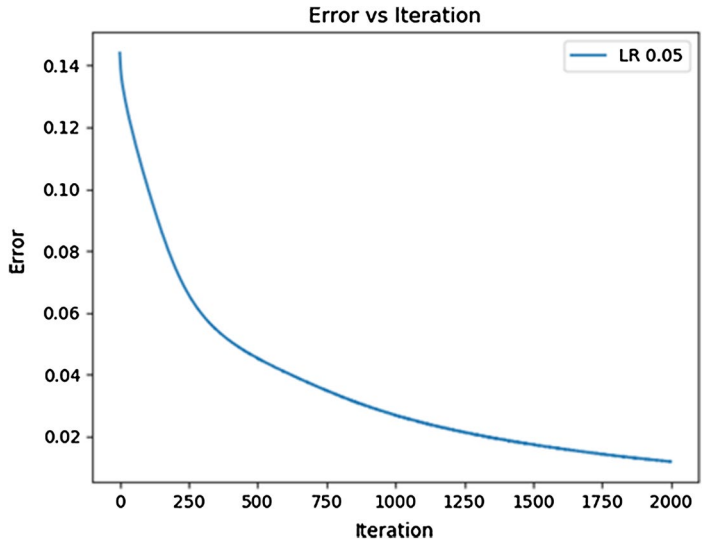

Fig. 5 Error vs. Iteration graph for LR 0.05 over 2000 iterations

that the optimal parameters for the ANN has 0.05 learning rate and trained for over 2000 iterations.

Conclusion

The data in this research are real data from some Indonesian companies. With these results, it can be said that the data used can be used to predict which shares are best for a portfolio. The algorithms used for building the ANN were supervised stochastic learning with back-propagation and steepest gradient descent, with the optimal parameters of 0.05 learning rate trained over 2000 iterations. These algorithms run with these parameters were able to predict if a company will go bankrupt with good success, that is, with 85.71% accuracy. The

accuracy may be improved by obtaining more real data to be used in training the ANN, so it can learn from more examples. Big data techniques will be very helpful in this matter.

Abbreviation
ANN: Artificial neural network.

Acknowledgements
This work is supported by BINUS University.

Authors' contributions
Both authors read and approved the final manuscript.

Funding
No funding in this research.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Management Department, BINUS Business School Doctor of Research in Management, Bina Nusantara University, Jakarta 11480, Indonesia. [2] School of Computer Science, Bina Nusantara University, Jl. K.H. Syahdan No. 9, West Jakarta 11480, Indonesia.

Received: 23 October 2019 Accepted: 21 February 2020
Published online: 09 March 2020

References

1. Ashwood AJ. Portfolio selection using artificial intelligence. DBA disertation of Queensland University of Technology 2013.
2. Fama EF, French FK. The cross-section of expected return. J Financ. 1992;47(2):611–49.
3. Farrell JL. Homogeneous stock grouping: implication for portfolio management. Financ Anal J. 1975;31(3):50–62.
4. Jones CP. Portfolio management. New York: McGraw-Hill'; 1992.
5. Khaksari S, Kamath R, Grieves R. A new approach to determining optimum portfolio mix. J Portf Manag. 1989;15(3):43–9.
6. Manurung AH. Consistency of shares choice in building optimal portfolio on Jakarta Stock Exchange by Fund Manager associated to ratio of empirical performance (Konsistensi Pemilihan Saham dalam Pembentukan Portfolio Optimal di BEJ oleh Manajer Investasi dikaitkan dengan Variabel Rasio Empirik Kinerja Perusahaan). 2002 Doctoral Disertation of University of Indonesia.
7. Markowitz HM. Portfolio selection. J Finance. 1952;7(1):77–95.
8. Markowitz HM, Perold AF. Portfolio analysis with factors and scenarios. J Financ. 1981;36(4):871–7.
9. Marmer HS. Optimal international allocation under different economic environments: a Canadian perspective. Financ Anal J. 1991;3(6):85–92.
10. Peavy JW, Goodman DA. The significance of P/Es for portfolio returns. J Portf Manag. 1983;9(2):43–7.
11. Perritt GW, Lavine A. Diversify: the investor's guide to asset allocation strategies. USA: Longman Financial Services Publishing; 1990.
12. Rasekhschaffe KC, Jones RC. Machine Learning for Stock Selection. Financ Anal J. 2019;75(3):70–88.
13. Reily FK, Brown KC. Investment analysis and portfolio management. New York: The Dryden Press; 2002. p. 272.
14. Reinganum MC. The significance of market capitalization in portfolio management over time. J Portf Manag. 1999;25(4):39–50.
15. Saaty TL, Rogers PL, Pell R. Portfolio selection through hierarchies. J Portf Manag. 1980;6(3):16–21.
16. Wainscott CB. The Stock-bond correlation and it's implication for asset allocation. Financ Anal J. 1990;2(2):55–60.
17. Wallingford BA. A survey and comparison of portfolio selection models. J Financ Quant Anal. 1967;2(2):85–106.
18. Zhang Y, Guo Q, Wang Jw. Big data analysis using neural networks. Adv Eng Sci. 2017;49:9–18.

Publisher's Note