Application of LSTM Model Optimized Based on Adaptive Genetic Algorithm in Stock Forecasting

Yong He, Huan Li, and Wenhong Wei(✉)

School of Computer, Dongguan University of Technology, Dongguan 523808, China
weiwh@dgut.edu.cn

Abstract. The need to predict stock price arises in the quantitative financial transaction field is a challenging problem. Long-short term memory (LSTM) neural network has shown a good effect on this problem. There are two main issues when implementing this method. One, it always suffers from huge attempts of constructing the neural network and adjustments of the hyper-parameter. Two, it often fails to find an excellent solution. We propose an AGA-LSTM algorithm, which uses an adaptive genetic algorithm to automatically optimize the network structure and hyper-parameters of the LSTM neural network. The simulation results show that the accuracy of the rise and fall of the stock outperforms LSTM as well as other previously machine learning models. Moreover, attempts are significantly less than other tuning methods.

Keywords: Stock price prediction · Long short-term memory · Adaptive genetic algorithm · Machine learning

1 Introduction

The prediction of stock prices is an important issue in the financial field. Through the prediction of price fluctuation trends, financial risks can be minimized. Obtaining the characteristics of price fluctuations and their trends have significant impacts, on individuals to choose investment, government to control the financial market, and the securities market to promote the sustainable and healthy development. The current predictive analysis technology can be roughly divided into two categories: forecasting and clustering [1]. Specifically, we centered on forecasting, as forecasting methods are divided into two categories: the traditional forecasting model based on statistical theory, and the commonly used method of exponential smoothing. With the increase in the amount of data and the continuous development of the stock market, traditional non-intelligent methods have gradually decreased [2]. The second type of intelligent method, forecasting models based on gray theory and neural networks, are more and more common in stock price forecasting due to its powerful learning ability and self-adaptive ability [3].

Among prediction methods of machine learning, artificial neural network (ANN) is a commonly used method, since it is suitable for dealing with complex relationship problems. But the testing and training speed is relatively slow, and it is easy to fall into local extreme values and over-fitting. In order to avoid the black box problem of neural networks, Li-Ping Ni, Zhi-Wei Ni, Ya-Zhuo Gao used support vector machines (SVM) with strict mathematical proofs, but there is a problem of obtaining the optimal number of features in feature selection [4]. So this method is not easy to generalize. Yang Su, Yao-yuan Shi, and Heng Song used the genetic algorithm to optimize the kernel function of SVM, and got better results on the stock index prediction of the Chinese stock market [5]. Yakup Kara, Melek Acar Boyacioglu, Ömer Kaan Baykan compared the SVM model and BP neural network to predict the trend of the stock index, and found that the prediction effect of BP neural network is better than SVM model [6].

Christopher Krauss, Xuan Anh Do and Nicolas Huck applied Random Forest (RF) to the stock price prediction problem. Comparing with the random forest, SVM, BP neural network, and naive Bayes models, the overall performance of the random forest was the best. Although the accuracy of random forest is higher than most individual algorithms, the RF model is prone to over-fitting on sample sets with large noise such as stock price sequence [7]. Long short-term memory (LSTM) has always been one of the most popular methods in time series prediction. LSTM is an improvement of RNN, which can memorize important state information of longer time series, and will not have the problem of gradient disappearance or gradient explosion. Although it is seldom used for financial time series forecasting, the characteristics of this model are highly compatible with stock forecasting problems. Fischer and Krauss [8] used LSTM to predict the sample return based on the S&P 500 constituent stocks. Although this method performed better than random forest (RF), deep neural network (DNN) and logistic regression (LOG), its accuracy is between 51% and 54%. Reasons for the lower accuracy rate may be the feature samples are not appropriate, the neural network structure is not suitable, and the model is unable to jump out of local extremes. The hyper-parameter selection in LSTM mostly uses grid search algorithms and fine-tuning control variables, but their computational overhead is huge. This article proposes a stock price prediction method based on the AGA-LSTM recurrent neural network for the stock price time series. The adaptive genetic algorithm is used to optimize the hyper-parameters of LSTM. With the goal of maximizing test accuracy, the experiment is carried out with US stock data, The experimental results show that the AGA-LSTM model has good forecasting performance in the test data.

The contribution of this work includes: (1) The use of the AGA-LSTM algorithm, slightly improves performance compared with other models. (2) Real number coding and adaptive methods are implemented for computationally efficient. (3) The automatically building optimal network model is much faster than manual trial, error method and grid search method.

2 Algorithm Background

Long short-term memory neural network (LSTM) is a neural network structure that connects the LSTM unit as a directed graph [9]. The characteristic of the LSTM neural network is to store the input of the neuron at the current time and the output at the previous time. The LSTM is an improvement of Recurrent Neural Network (RNN). RNN neural network can store the relationship between the current input of the neuron and the previous output [10]. By introducing the LSTM module in the chain structure, the LSTM has a complex internal structure, which can learn long-term calculation and memory information from input data (Fig. 1). It has been widely used in the prediction of nonlinear time series data in recent years [11].

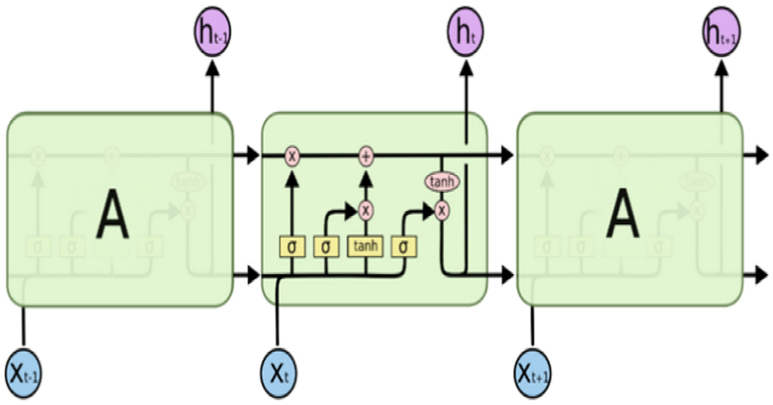

Fig. 1. The internal structure of the LSTM calculation unit.

First, the A gate is an optional way to let information through. It consists of a Sigmoid layer and a point multiplication operation. LSTM has three gates to protect and control the state of a single computing unit. The forget gate determines the information that needs to discard, which can be expressed as:

$$f_t = \sigma\left(W_f * [h_{t-1}, x_t] + b_f\right) \tag{1}$$

W_f represents the connection weight of the previous output, h_{t-1} as the previous output, x_t as current input, b_f as bias vector, and σ as activation function. The input gate determines the information needs to be updated, which is obtained by multiplying the two vectors created by the input gate layer and the tanh layer. It can be described as:

$$i_t = \sigma\left(W_f * [h_{t-1}, x_t] + b_i\right) \tag{2}$$

$$\tilde{C}_t = \tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{3}$$

The output gate provides information of the input gate and the forget gate. It can be defined as:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

The calculation unit outputs the result after the above operation. It contains two processes. First, the Sigmoid layer determines the output. The cell state passes through the Tanh layer to normalize the value to between –1 and 1, and then performs dot multiplication. The output of this process determines the part of the output, which can be expressed as:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * \tanh(C_t) \tag{6}$$

Connecting each computing unit into a chain structure can get a neural network (Fig. 2).

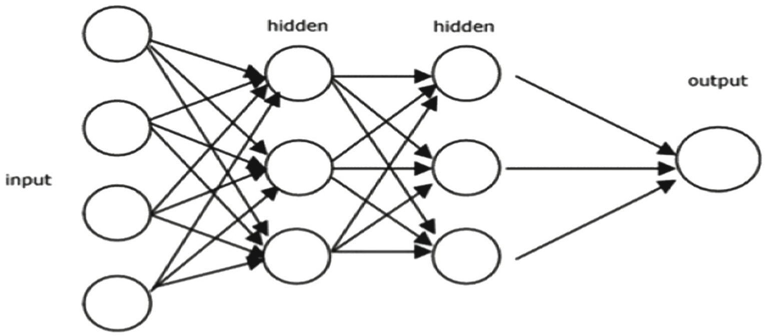

Fig. 2. A typical network structure.

The training method of the LSTM model is to calculate the output value of the LSTM unit according to the forward propagation method first. Second step is to propagate the error value of the LSTM unit back, and calculate the weight gradient according to the error value. Then use the optimization algorithm to perform gradient descent. Last step to continuously Update weight.

Most optimization algorithms use the stochastic gradient descent (SGD), the adaptive gradient algorithm (AdaGrad), and the adaptive moment estimation algorithm (Adam). The Adam algorithm is different from the stochastic gradient descent. The Stochastic gradient descent maintains a single learning rate, while the Adam algorithm calculates first-order moment estimation and second-order moment estimation for parameter to obtain an independent adaptive learning rate [12].

As the number of network layers increases, it often leads to long training time and over-fitting problems. The integrated learning method is used to train a different model to improve the generalization ability. By adding a dropout layer, randomly omitting a certain ratio of feature detectors in each training case to reduce the over-fitting phenomenon [13].

The Adaptive genetic algorithm is a heuristic search algorithm based on the improvement of genetic algorithm, by dynamically changing the crossover probability (P_c) and the mutation probability (P_m). When the fitness of each individual is lower than the average fitness of the group, P_c and P_m increase to jump out of the local optimum. When the fitness of each individual of the population is lower than the average fitness of the group, P_c and P_m are reduced to ensure the survival probability of excellent individuals.

The prediction accuracy of the LSTM model is closely related to the value of the hyper-parameters. The number of layers in the LSTM need to be optimized in the search space that includes the dense layer, the number of neurons, the forgetting rate of the dropout layer and the number of iterations. The grid search algorithm and the fine-tuning method are essentially traversal optimization of the search space. This paper used adaptive genetic algorithm (AGA) to adjust the LSTM hyper-parameters to obtain the optimal solution in the search space to form a composite AGA-LSTM model.

3 Problem Description

The stock price data is a non-stationary random sequence. We use the stock data of Google (from 2004-08-19 to 2018-03-27), Amazon (from 2000-11-01 to 2018-03-27) and Apple (from 2000-11-01 to 2018-03-27) provided by the quandl website. According to Krauss' sample processing method [7], for each sample, five-dimensional data (opening price, closing price, highest price, lowest price, and trading volume of a certain stock.) of 50 days before the current time are used as input, and the closing price of the current time as output. Table 1 shows data examples.

Table 1. The first 10 rows of Google data example

Date	Open	High	Low	Close	Volume
2004/08/19	100.01	104.06	95.96	100.35	44659000
2004/08/20	101.01	109.08	100.50	108.31	22834300
2004/08/23	110.76	113.48	109.05	109.40	18256100
2004/08/24	111.24	111.60	103.57	104.87	15247300
2004/08/25	104.76	108.00	103.88	106.00	9188600
2004/08/26	104.95	107.95	104.66	107.91	7094800
2004/08/27	108.10	108.62	105.69	106.15	6211700
2004/08/30	105.28	105.49	102.01	102.01	5196700
2004/08/31	102.32	103.71	102.16	102.37	4917800
2004/09/01	102.70	102.97	99.67	100.25	9138200

Table 1 shows the data examples. Here are the first 10 rows of the Google data example.

In order to eliminate the influence of different dimensions of the above characteristic, this article normalized the data. The data is mapped to (0, 1).

$$x_{norm} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{7}$$

The original data sequence is x = {x_1, x_2,…, x_n}, and the x norm is the normalized value. We obtain a new data sequence D = {d_1, d_2,…, d_n}. There are training set and test set, d_{tr} = {d_1, d_2,…, d_m} and d_{te} = {d_{m+1}, d_{m+2},…, d_n}. Seting the timestamp of the data to 50 as S, the input data is:

$$X = \{X_1, X_2, \ldots, X_n\} \tag{8}$$

$$X_i = \left\{d_i, d_{i+1}, \ldots, d_{(m-S+p-1)}\right\} \quad s.t 1 \leq i \leq S;\ S \in N \tag{9}$$

The output data is converted into the rise or fall of stock price, which is a classification label calculated based on the closing price. C_{t+1} represents the closing price on day t + 1, and the classification label on day t is calculated by the following formula:

$$y_t = \begin{cases} 1, & C_{t+1} > C_t \\ -1, & others \end{cases} \tag{10}$$

The actual and predicted values are:

$$Y = \{Y_{m+1}, Y_{m+2}, \ldots Y_n\} \tag{11}$$

$$y_i = \{y_{m+1}, y_{m+2}, \ldots y_n\} \tag{12}$$

In order to quantitatively analyze the model performance of AGA-LSTM, model accuracy is used as a measurement index and compared with models such as SVM, BP, random forest, and LSTM. Those indicators are calculated based on true-TP, true-negative-TN, false positive-FP, false negative-FN, and the prediction accuracy. It is expressed as:

$$acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{13}$$

4 Algorithm Description

4.1 Genes Code

The coding method of the genetic algorithm adopted in this paper is real number coding, and directly performs genetic operations on the phenotype of the solution. There are 8 genes in one sample. The first gene represents the number of LSTM layers (1–2, and the step size is 1). The second gene indicates the number of dense layers (1–2, with a step size of 1). The third and fourth genes represent the number of neurons in each layer of the LSTM layer (32–128, and the step size is 1). If the indicated layer does not exist, the number of neurons is coded as 0. The fifth and sixth gene represents the number of neurons in the dense layer. Similarly, if the represented layer does not exist, the number of neurons is coded as 0. The seventh and eighth genes represent the dropout layer forget rate (0.1–0.5, the step size is 0.1) and the number of network iterations (20–60, and the step size is 20).

4.2 Crossover Operator

This paper adopts the adaptive single-point crossover method, and randomly sets a crossover point in the individual string. When the crossover operator is performed, the genes are at the corresponding positions of the individual cross. The first and second digits do not cross, because they represent the number of layers. After the crossover, it will affect the number of neurons in the back. If the third to sixth digits have exchanged genes of 0, they should not be crossed. The adaptive crossover probability is expressed as:

$$p_c = \begin{cases} k_3, & f_c \le f_{avg} \\ \frac{k_1(f_{\max} - f_c)}{f_{\max} - f_{avg}}, & f_c \ge f_{avg} \end{cases} \tag{14}$$

In the formula, f_c is the larger fitness value of the two individuals to be crossed. For individuals with higher fitness, the smaller p_c is in line with the setting of "preserving good individuals". When the individuals with lower fitness are bigger, they are in line with the setting of "changing bad individuals". Set $k_1 = 1$ and $k_3 = 1$.

4.3 Mutation Operator

The mutation method used in this paper is an adaptive mutation single-point operator, which only adjusts the value of the individual after the two genes. It randomly selects the value according to the corresponding range. The third to sixth digits are randomly selected from (32, 128), and the seventh value is randomly selected from (0.1, 0.5). The last bit is randomly selected from (1, 3), where the adaptive mutation probability is expressed as:

$$p_m = \begin{cases} k_4, & f_m \le f_{avg} \\ \frac{k_2(f_{\max} - f_m)}{f_{\max} - f_{avg}}, & f_m \ge f_{avg} \end{cases} \tag{15}$$

In this formula, f_m is the fitness value of the individual to be mutated. For the individual with the highest fitness, the crossover probability is 0. If the mutation probability is also 0, it will lead to mass reproduction and produce "premature" phenomenon. Therefore, the individual with lower fitness is used to search for the global optimal solution. Set $k_2 = 0.5$, $k_4 = 0.5$.

4.4 Steps of the Algorithm

The specific steps of the algorithm are as follows: The first step is to initialize the population and determine the value of each chromosome. Neural networks are determined according to the genes of the chromosomes, the number of network layers, the number of hidden neurons and the number of iterations of the LSTM. Then, the neural network is trained to calculate the fitness value. Drawing on formula (13), the accuracy of model prediction is used as a fitness function. Further, roulette generates the next generation. Adaptive crossover and mutation operations on the solved individuals generate new individuals. Repeating the above steps until reaches the maximum number of iterations, and the best individual would emerge among them. Last is to train the LSTM neural network with the best hyper-parameter combination. Input test samples, output prediction are compared with true values (Fig. 3).

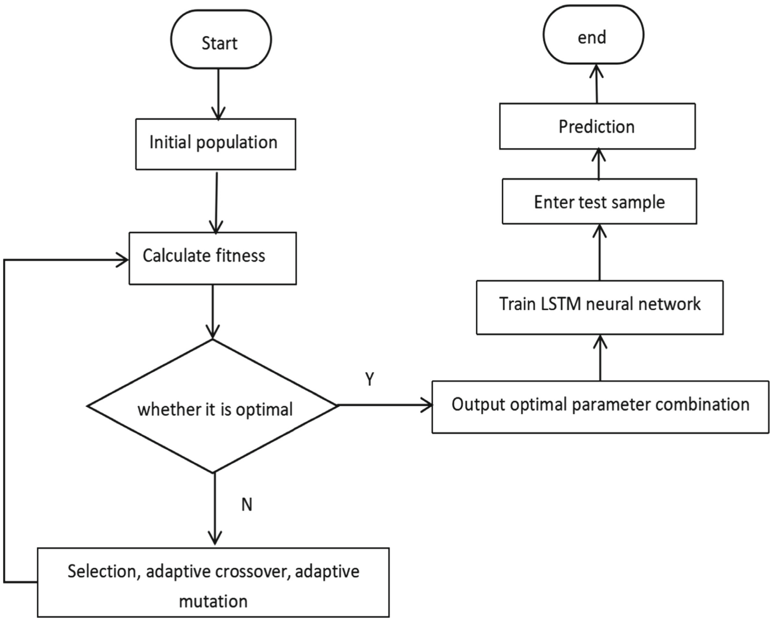
Start
Initial population
Calculate fitness
whether it is optimal
Y
N
Selection, adaptive crossover, adaptive mutation
Output optimal parameter combination
Train LSTM neural network
Enter test sample
Prediction
end

Fig. 3. The summary of the AGA-LSTM algorithm.

5 Experimental Result

The computer configuration and software environment used in the experiment are as follow: The processor is Intel Xeon E3–1230 v6, with 16.0 GB memory and NVIDIA Quadro M2000 graphics card; the system is Windows 10 (64-bit); the programming language version is Python 3.6.2; IDE is Pycharm 2020.3 in the Anaconda package. AGA-LSTM and LSTM are implemented in the Keras library with Tensorflow as the backend. The SVM, random forest, and KNN model used for the comparison of prediction are implemented in the scikit-learn library. The hyper-parameters of the genetic algorithm are as follow: the number of populations in each generation is 20, the length of individual genes is 8, and the number of iterations is 100.

After the data is trained with the AGA-LSTM model, it is possible to quickly find better solutions and approximately optimal solutions in the search space than the sub-optimal model (Fig. 4).

A neural network is constructed by setting the parameters of the optimal solution. As the number of iterations increases, the verification error converges quickly. The mean square error in the LSTM neural network is reduced to achieve the optimal solution in the search space (Fig. 5).

In order to show the experimental result, this article uses the fitting curve of the verification set to compare with the true curve of the test set. The test set data is from 82%–100% of the price series of the total data set. The horizontal axis is time, and the vertical axis is the normalized stock price. The prediced effects of the three data samples under the four models are drawn respectively. As shown in Figs. 6, 7, and 8, all five algorithms can fit the real data curve. Compared with the KNN, SVM, RF, and LSTM algorithms, AGA-LSTM reacts more sensitively to changes in stocks. The adaptability to dramatic change is stronger, so the trend changes rapidly. The “time lag” problem caused by over-fitting is over comed to some extent. With a sequence, the prediction accuracy is higher.

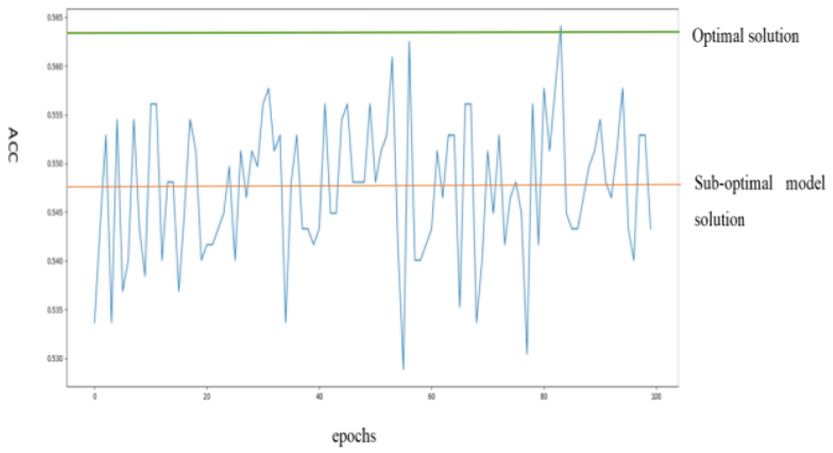

Fig. 4. The best accuracy of each generation.

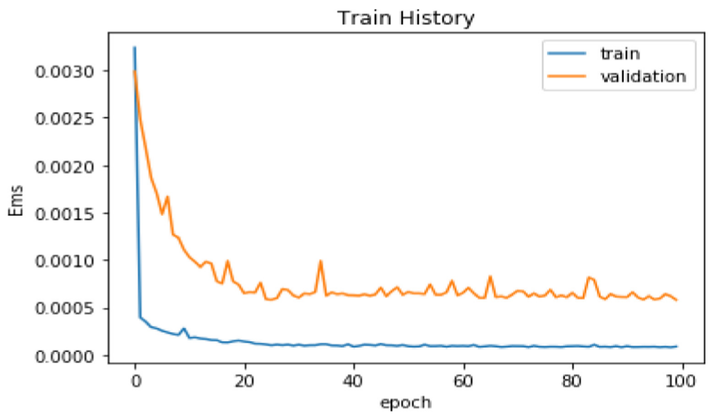

Fig. 5. The mean square error of the optimal solution is decreasing, and the optimal solution in the search space is realized.

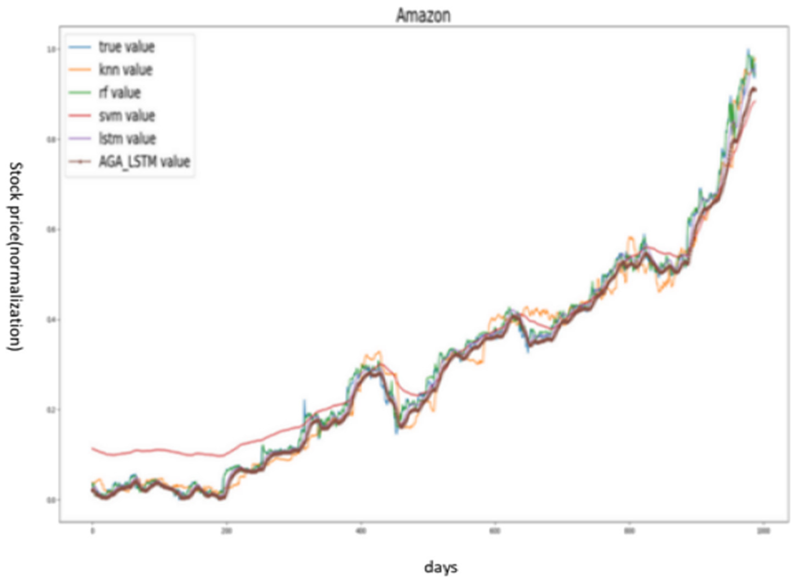

Fig. 6. Amazon stock price prediction in five models

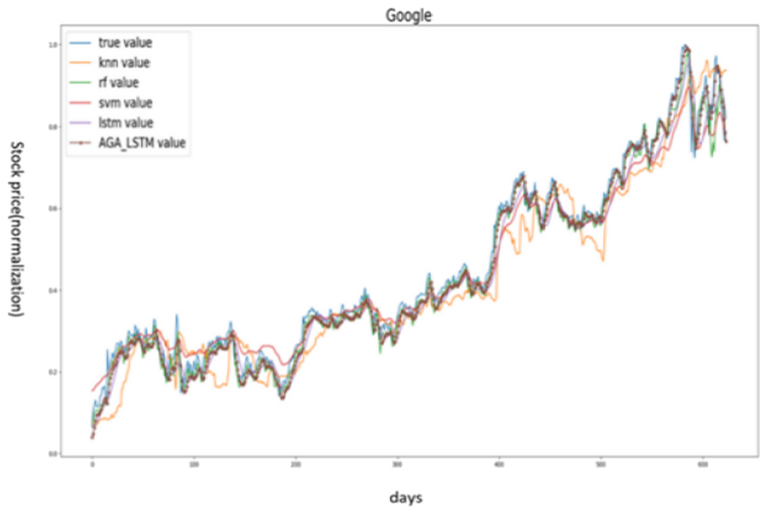

Fig. 7. Google stock price prediction in five models

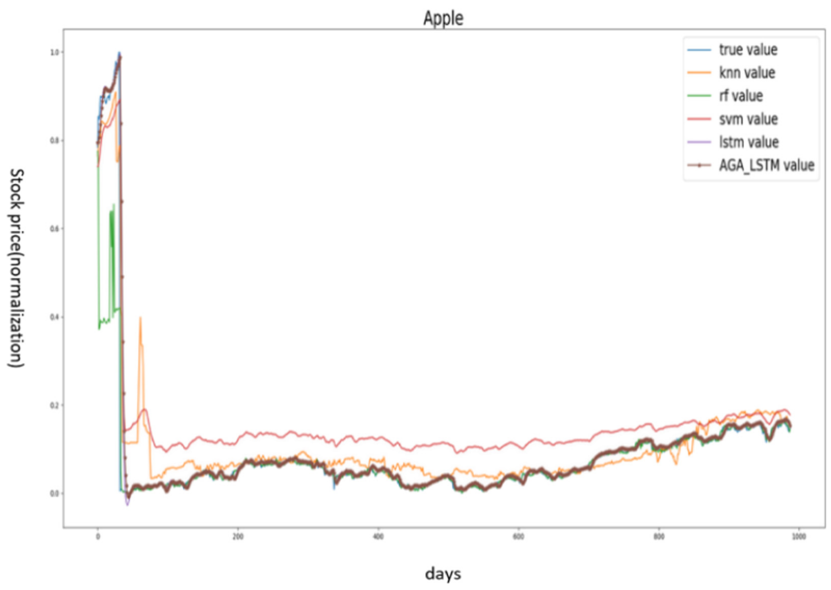

Fig. 8. Google stock price prediction in five models

After the prediction of 5 models, the prediction accuracy indicators of each stock are shown in Table 2 and Fig. 9. The accuracies of AGA-LSTM are 3.29%, 4.79%, and 1.43% higher than other sub-optimal models respectively. The average accuracy is 3.28% higher.

Those are the prediction accuracy and the average accuracy of each model in each stock.

Table 2. Prediction accuracy of each stock

Acc	KNN	RF	SVM	LSTM	AGA-LSTM
Apple	0.514	0.493	0.547	0.533	**0.565**
Amazon	0.507	0.502	0.542	0.521	**0.568**
Google	0.536	0.485	0.556	0.516	**0.564**
Average	0.519	0.493	0.548	0.523	**0.566**

In order to find the appropriate key hyper-parameters to adapt the machine learning model, a grid search (GS) or random search (RS) method is used. In this question, the time of each model training is set to t. GS can be regarded as an exhaustive search, and the model training time is (2 * 2 * 96 * 96 * 96 * 96 * 5 * 3) * t = 5096079370t. This time consuming is undoubtedly very huge. Although RS method can avoid excessive training time in a small area with poor performance, it does not use the previously well-performing parameter combination. If the RS method is used in a sufficiently large configuration space, the time performance of this method is relatively poor. The AGA-LSTM algorithm has strong early local search capabilities, adaptively adjusts the crossover, and mutation probability. However it is not easy to fall into the local extreme value in the early stage and maintain good individuals in the later stage, As shown in Fig. 4 the 50th round of iteration. To obtain a parameter combination (individual) close to the optimal solution, the model training time required is only (50 * 20) * t = 1000t, which is 1.96 * 10–7 of the time required by the grid search algorithm.

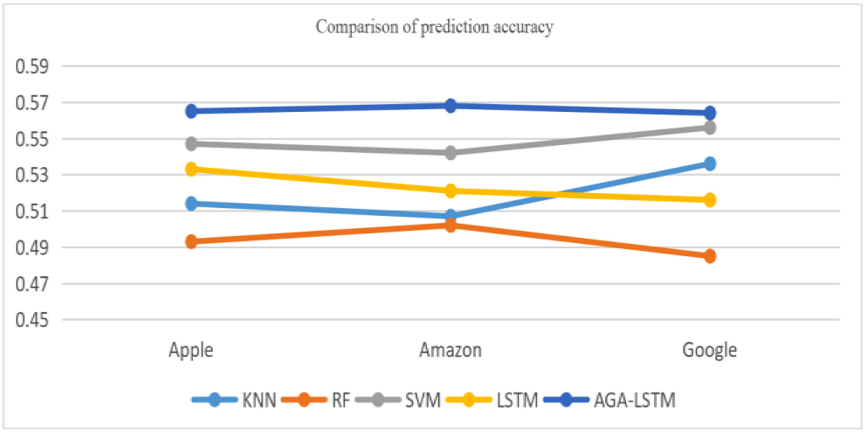

Fig. 9. Prediction accuracy of each stock and the average accuracy of each model.

6 Conclusion

The main contribution of this research is that we build a model with a high prediction accuracy. It can automatically adjust network structure and parameter combination, which significantly reduced the number of parameter adjustments. We conducted a quantitative analysis of the three stocks. The result demonstrates not only our model produces better performance than using LSTM, but also performs better than other four machine learning models. Our approach includes using actual codes to directly find the optimal solution in the solution space, adaptive crossover and mutation probability to accelerate the speed of finding the optimal solution. The found parameter combinations can achieve better prediction goals. Furthermore, comparing with grid search (GS) method and random search (RS) method, the efficiency of parameter tuning is higher. In order to get better accuracy, there are still more directions for future exploration. The use of more features of big data might be a good choice for yielding a better prediction accuracy. In this research, the disparities of fitness of each individual are not that huge, and using an adaptive algorithm based on individual order might augment the speed of finding the optimal solution.

Acknowledgement. This work was supported by the Key Project of Science and Technology Innovation 2030 funded by the Ministry of Science and Technology of China (No. 2018AAA0101301), the Key Projects of Artificial Intelligence of High School in Guangdong Province (No. 2019KZDZX1011) and The High School innovation Project (No. 2018KTSCX222).

References

1. Dattatray, P.G., Kumar, K.: Systematic analysis and review of stock market prediction techniques. Comput. Sci. Rev. **34** (2019)
2. Jin-Song, W.: An empirical study on the effectiveness of commonly used technical analysis methods in stock market. Southwestern University of Finance and Economics (2010)
3. Javad, Z., Mohammad, M.R.: Application of artificial neural network models and principal component analysis method in predicting stock prices on Tehran Stock Exchange. Physica A: Stat. Mech. Appl. **438** (2015)
4. Li-Ping, N., Zhi-Wei, N., Ya-Zhuo, G.: Stock trend prediction based on fractal feature selection and support vector machine. Expert Syst. Appl. **38**(5), 5569–5576 (2011)
5. Su, Y., Yao-Yuan, S., Heng, S.: Time series prediction algorithm of stock market based on support vector machine. Sci. Technol. Eng. **02**, 381–386 (2008)
6. Yakup, K., Melek, A.B., Ömer, K.B.: Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul Stock Exchange. Expert Syst. Appl. **38**(5), 5311–5319 (2011)
7. Christopher, K., Xuan, A.D., Nicolas, H.: Deep neural networks, gradient-boosted trees, random forests: statistical arbitrage on the S&P 500. Eur. J. Oper. Res. **259**(2), 689–702 (2016)
8. Thomas, F., Christopher, K.: Deep learning with long short-term memory networks for financial market predictions. Eur. J. Oper. Res. **270**(2), 654–669 (2018)
9. Pascanu, R., Gulcehre, C., Cho, K.: How to construct deep recurrent neural networks. In: International Conference on Learning Representations, pp. 1–13. IEEE Press, Piscataway (2014)
10. Dong, Y., Li, D.: Automatic Speech Recognition, vol. 237. Springer Verlag, London (2015)
11. Ma, X., Tao, Z., Wang, Y.: Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp. Res. Part C: Emerg. Technol. **54**, 187–197 (2015)
12. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: 2014 IEEE International Conference on Learning Representations. IEEE (2014)
13. Srivastava, N., Hinton, G., Krizhevsky, A.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)