AGA-GRU: An Optimized GRU Neural Network Model Based on Adaptive Genetic Algorithm

Chenyao Bai
Department of Public Education, Shanghai Customs College, Shanghai, 201204, China
baichenyao@shcc.edu.cn

Abstract. The weight adjustment of gated recurrent unit (GRU) network depends on the gradient descent algorithm heavily, therefore this paper proposes an improved GRU neural network model based on adaptive genetic algorithm (AGA-GRU) to solve this problem. In this model, AGA is used to construct the optimization system, and the parameters of neural network model are optimized to improve the prediction performance. The results on UCI dataset show that the prediction accuracy of AGA-GRU model is significantly improved, and the generalization performance is stronger.

1. Introduction
The era of big data has promoted the development of deep learning and artificial intelligence [1]. Gated recurrent unit (GRU) [2] is a new cyclic neural network model proposed in 2014. GRU is a variant of Long shot-term memory (LSTM) [3] and has been widely used in speech recognition [4], machine translation [5, 6], computer vision [7] and time series prediction [8].

However, when the number of hidden layer neurons is large, the GRU model still has some shortcomings, such as low learning efficiency, slow convergence speed, and even the phenomenon of under fitting and low prediction performance. For this reason, Schuster et al. [9] proposed the bidirectional recurrent neural network (BRNN), which can learn from two time dimensions, and has higher learning efficiency and better prediction accuracy. Lei et al. [10] proposed a simple recurrent unit (SRU) neural network, which enhanced the generalization ability in large data volume scenarios. Liu Q et al. [11] proposed the statistical recurrent unit (TSRU) neural network, which realized associative learning, maintained the long-term dependence of information, and significantly reduced the computational cost.

Most of the improved models aim to improve the structure of neurons and ignore the weight adjustment between GRU neurons. Therefore, an optimized GRU neural network model based on AGA algorithm (AGA-GRU) is proposed in this paper. The model uses adaptive genetic algorithm to establish the optimization space and update the weights of nodes to enhance the accuracy and time series prediction ability of the model.

2. Adaptive genetic algorithm and the GRU neural network

2.1. Adaptive genetic algorithm
Adaptive genetic algorithm [12] is an improved genetic algorithm proposed by Srinivas et al. It can dynamically adjust the crossover and mutation probabilities, retain the excellent individuals in the population to the maximum extent, avoid the occurrence of premature phenomenon, and have better
optimization performance. Compared with genetic algorithm, the crossover and mutation probabilities in the algorithm are updated in real time to ensure the diversity of the population. Therefore, this paper uses AGA algorithm to construct the optimization space, and uses it to adjust the weight parameters of GRU model.

Formulas (1) and (2) are the derivation formulas of crossover probability and mutation probability in AGA algorithm. Among them, \( f_{avg} \) is the average fitness value, \( f_{max} \) is the maximum fitness value of all individuals, \( f' \) is the larger fitness value in the cross process, \( f \) represents the variant individual, \( k_1, k_2, k_3 \) and \( k_4 \) are real numbers between 0 and 1.

\[
P_c = \begin{cases} 
  k_1 \frac{f_{max} - f'}{f_{max} - f_{avg}} & f' \geq f_{avg} \\
  k_2 & f' < f_{avg} 
\end{cases}
\]

(1)

\[
P_m = \begin{cases} 
  k_3 \frac{f_{max} - f}{f_{max} - f_{avg}} & f \geq f_{avg} \\
  k_4 & f < f_{avg} 
\end{cases}
\]

(2)

The adaptive genetic algorithm can make full use of the parameters, \( k_1, k_2, k_3 \) and \( k_4 \). In the early time of iteration, the crossover probability and mutation probability can be adjusted appropriately to protect the individuals with larger fitness value; in the later time of iteration, the crossover probability and mutation probability can be increased to jump out of the local optimum and find the global optimal individual.

2.2. The GRU neural network
The GRU neural network optimizes the structure of the LSTM model, and merges the three gated units to two gated units, namely the update gate and the reset gate. The GRU model has fewer parameters and lower training overhead, its neuron structure of GRU is shown in Figure 1.

![Fig.1 The structure of the GRU neuron.](image)

If the input data is \((x_1, x_2, ..., x_t)\), the derivation formulas for the update gate and reset gate at time \( t \) are as follows:

\[
r_t = \sigma(W_r * [h_{t-1}, x_t])
\]

(3)

\[
z_t = \sigma(W_z * [h_{t-1}, x_t])
\]

(4)

\[
n_t = \tanh(W * [r_t, h_{t-1}, x_t])
\]

(5)
Among them, at time $t$, $r_t$ is the output of the reset gate; $z_t$ is the output of the update gate; $h_t$ is the hidden state of the candidate; $W_r$ represents the weight matrix between the input $x_t$ and $h_{t-1}$ in the reset gate; $W_z$ represents the weight matrix between the input $x_t$ and $h_{t-1}$ in the update gate; $W$ represents the input weight matrix between $r_t * h_{t-1}$ and $x_t$ at the candidate hidden state; where $h_{t-1}$ represents the hidden state of the layer at time $t-1$; $x_t$ represents the input at time $t$; $\sigma$ is the activation function sigmoid, tanh is also a common activation function. The GRU model continuously discards redundant timing information through the control of the gated unit, and uses the long-term dependency relationship of the candidate hidden state to maintain the information, thereby improving the prediction accuracy.

3. AGA-GRU model

3.1. Chromosome coding

In this paper, AGA is used to optimize the weights between the nodes of the GRU neural network, which could improve the generalization ability and prediction performance of the model.

![Figure 2. The structure of GRU neural network with 3 hidden layers](image)

A chromosome is a set of weight values between nodes in the GRU model. The chromosomes of all individual are encoded with real numbers. Figure 2 shows the structure of GRU model with 3 hidden layers, Figure 3 shows the coding method of a certain chromosome. Among them, $w_{11}$ represents the weight between the first neuron of the input layer and the first neuron of the hidden layer in the first layer, and so on.

![Figure 3. The coding of chromosomes](image)

3.2. Fitness value function

In AGA-GRU model, the mean square error between the output value and the actual value of the GRU model is defined as the fitness value function. The smaller the value is, the more reasonable the weight setting of the GRU neural network is, and the higher the accuracy of the model is. The mean square error is given by:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

where $N$ is the number of data points, $y_i$ is the actual value, and $\hat{y}_i$ is the predicted value.
error, usually expressed by, is the mean square error of chromosome. The calculation formula is shown in (8).

$$MSE_i = \frac{1}{T_{num}} \sum_{j=1}^{T_{num}} (d_{ij} - y_{ij})^2$$ (8)

Among them, $num$ is the number of population; $T_{num}$ is the number of test set; $d_{ij}$ is the prediction result of GRU model corresponding to chromosome $i$; $y_{ij}$ is the actual value.

3.3. The step of AGA-GRU model

The AGA-GRU model maps the weight values between the nodes of the GRU neural network through the adaptive genetic algorithm and maps each weight value to a certain dimension of the chromosome, making the chromosome a solution set of candidate weights for the GRU neural network. Afterwards, the adaptive genetic algorithm is used to establish the optimization space and iterate continuously to make the weights between the neurons more reasonable, improving the prediction performance and the accuracy of the GRU model.

![Flowchart of AGA-GRU model](image)

Figure 4. the flow of AGA-GRU model

Figure 4 is the flowchart of the AGA-GRU model optimized based on adaptive genetic algorithm. The main flow of AGA-GRU model is as follows:

1. Initialize the algorithms of the GRU model and AGA. Train the model to obtain the default weights.
2. Calculate the fitness value of chromosomes, and use AGA algorithm to build the optimization system.
3. Select the chromosomes to the crossover pool.
4. Do the adaptive crossover operation by formula (1), and do the adaptive mutation operation by formula (2).
5. Determine whether the iteration is terminated. If yes, terminate the iteration, otherwise skip to (2).
6. Get the globally optimal chromosome, which is the optimal weight distribution of GRU model.

4. Experimental results and analysis

We empirically compare the performance of LSTM network, GRU network and the proposed AGA-GRU on several public datasets. AGA-GRU is efficient to train and enable to outperform other network architectures.

4.1. Experimental data set

We use Shampoo Sales Dataset (SSD) and Minimum Daily Temperatures Dataset (MDTD) in the DataMarket, and the Dow Jones Index Dataset (DJI) in the UCI database to evaluate the different networks. SSD dataset describes the monthly number of sales of shampoo over several years; MDTD includes the low temperature data from daily history in 1981-1990 of in Melbourne, Australia, both of which are univariate time series data sets.
4.2. Establishing prediction models

The dataset is firstly pre-processed, then we established the time series prediction models based on LSTM, GRU and AGA-GRU on the above three UCI datasets respectively. The data set is split as training set, validation set and test set, where the training set contains 60% of data, the validation data contains 20% of data and test set contains 20% of data. We use the sum of error (SSE) as the evaluation criterion of performance. The definition of SSE is shown in (9):

$$\text{SSE} = \sum_{i=1}^{n_{\text{test}}} (y_i^* - y_i)^2$$

(9)

where $n_{\text{test}}$ represents the number of the test set, $y_i^*$ represents the output result, $y_i$ represents the actual value. The smaller the SSE value, the better the performance of the model.

4.3. Analysis of empirical results

Table 1. SSE comparison of different neural networks.

| Data set | LSTM   | GRU    | AGA-GRU |
|----------|--------|--------|---------|
| SSD      | 382.3  | 338.8  | 264.4   |
| MDTD     | 312.1  | 273.1  | 225.1   |
| DJI      | 355.6  | 301.3  | 228.7   |
| Average SSE value | 351 | 304.4 | 239.4 |

Table 1 summary the SSE value of different architectures. The performance is compared on three different datasets (DJI, SSD, MDTD) and we also give the averaged performance among these datasets. Figure 5 is a histogram drawn according to Table 1.

Figure 5. SSE results of our method and two baselines
Figure 6. The comparison of averaged SSE of our method and two baselines

Form Table 1 and Figure 6, the proposed AGA-GRU has the best SSE value on each dataset. AGA-GRU gets 239.4 on average, while GRU and LSTM get 304.4 and 351, respectively. AGA-GRU has only 78% SSE compared to GRU and 68.1% SSE compared to MSE. The accuracy of long-term prediction of our method is better than LSTM and GRU, where these two methods are considered as the state-of-the-art recurrent neural network. The generalization ability of AGA-GRU is also better than other two methods because our method gets higher performance among different datasets. The genetic optimization updates the weights of neural networks to enhance the optimization ability of neural network, which is useful in learning long-term dependence and fast training. The AGA algorithm has a significant promotion effect on the weight optimization of GRU network.

5. Conclusion
We consider the weight correction and learning efficiency of the popular GRU network. The GRU network is often get stuck in long-term prediction and fall into local optimal solution. The proposed genetic optimization method solves this problem and results in a more efficient recurrent neural network for long-term prediction. The extensive experiments are conducted on popular datasets and the results show our method outperforms the LSTM and GRU. The proposed AGA-GRU model has highest prediction accuracy and better generalization performance on different datasets. The prediction error is significantly reduced compared to the original GRU network. We believe the genetic optimization method can be combined with other neural networks to improve the performance, for example, graph neural network and self-attention. We leave this for future research.

Acknowledgments
This work was supported by the scientific research project of Shanghai Education Commission on Application of Molecular Communication in Customs Clearance Virus Detection (2312409K17).

References
[1] Wu J, Long J, Liu M. Evolving RBF neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm[J]. Neurocomputing, 2015, 14(8):136-142P
[2] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.
[3] Hochreitters, SchmidhuberJ. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780
[4] Ravanelli M, Brakel P, Omologo M, et al. Light gated recurrent units for speech recognition[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2018, 2(2): 92-102.
[5] Yin W, Kann K, Yu M, et al. Comparative study of cnn and rnn for natural language processing[J]. arXiv preprint arXiv:1702.01923, 2017.
[6] Zhou G B, Wu J, Zhang C L, et al. Minimal gated unit for recurrent neural networks[J]. International Journal of Automation and Computing, 2016, 13(3): 226-234.

[7] Jung M, Lee H, Tani J. Adaptive detrending to accelerate convolutional gated recurrent unit training for contextual video recognition[J]. Neural Networks, 2018, 105: 356-370.

[8] Shen G, Tan Q, Zhang H, et al. Deep learning with gated recurrent unit networks for financial sequence predictions[J]. Procedia computer science, 2018, 131: 895-903.

[9] SCHUSTERM, PALIWALKK. Bidirectional recurrent neural networks[J]. IEEE Transactions on Signal Processing, 1997, 45(11):2673 -2681.

[10] Lei T, Zhang Y, Wang S I, et al. Simple Recurrent Units for Highly Parallelizable Recurrence[J]. 2017.

[11] Oliva J B, Poczos B, Schneider J. The Statistical Recurrent Unit[J]. 2017.

[12] Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1994, 24(4): 656-667.