Fuzzy Systems and Data Mining IX
A.J. Tallón-Ballesteros and R. Beltrán-Barba (Eds.)

doi:10.3233/FAIA231067

Hyperparameter Adaptive Neural Network Model for E-Commerce Sales Prediction

GuangLu Zhao[a], Ran Tian[a,1] and JingXia Wang[a]
[a] *College of Computer Science & Engineering, Northwest Normal University, China*

Abstract. Sales forecasting plays a vital role in the daily operations of e-commerce companies, impacting market assessment, operational planning, and supply chain management. As the market is constantly changing, accurately predicting sales is a critical challenge that e-commerce companies need to urgently address. However, traditional statistical forecasting methods have disadvantages such as long run times, low accuracy, weak generalization, and strong data periodicity, which lead to unnecessary losses for companies. We propose the QLBiGRU model that utilizes the reinforcement learning Q-Learning algorithm combined with BiGRU to improve forecasting accuracy. Automatic parameter optimization technology is also used to reduce training time and demand for hardware resources, thereby enabling enterprices to accurately analyze the market and make informed decisions.

Keywords. Sales Forecast, Automatic Parameter Optimization, Reinforcement Learning, BiGRU

1. Introduction

With the rapid development of the e-commerce industry, the e-commerce sales system generates a large amount of data. Sales data, as typical time series data, directly reflects the characteristics of commodity circulation and can be collected and utilized to predict sales at a certain point in the future or within a certain period. This is very important for e-commerce companies to formulate reasonable business plans and sales plans based on past experience. However, due to the influence of various factors such as seasons and promotional activities, it is difficult to accurately predict product sales. Inaccurate forecasts can lead to excess inventory and increased costs or lost sales. Therefore, accurate and efficient merchandise sales forecasting is key to reducing uncertainty, minimizing inventory buildup, and reducing opportunity costs.

Traditional statistical methods, such as the Markov model[1], Prophet model[2], autoregressive moving average (ARIMA) [3], etc., have limitations due to the inability to solve problems such as data periodicity and perform poorly. The rise of deep learning has brought better prediction results, but the hyperparameters of the model, such as learning rate, number of iterations, and batch size, can be difficult to determine and require a lot of effort and cost to set. Therefore, optimizing hyperparameters is crucial for predictive models.

[1] Corresponding Author, Ran Tian, College of Computer Science & Engineering, Northwest Normal University, China; E-mail: tianran@nwnu.edu.cn

(1) We propose a prediction model QLBiGRU based on Q-Learning and BiGRU, and optimize the hyperparameters of the BiGRU model, such as the number of neurons, the number of iterations, and the learning rate, to improve the prediction accuracy.

(2) We conducted experiments with different prediction lengths and performance optimization experiments and used multiple data sets to verify the prediction performance and hyperparameter optimization capabilities of the QLBiGRU model. The experimental results validate the effectiveness and superiority of the model.

The rest of the paper is organized as follows, and in section 2, we report a summary of research related to e-commerce sales prediction and hyperparameter optimization. Section 3 presents a detailed description of QLBiGRU. In section 4, we conduct sufficient experiments to verify the validity of the proposed model in section 3. The full paper is summarized in section 5.

2. Background

2.1. E-commerce Sales Prediction

E-commerce sales forecasting is a subfield of time series forecasting that utilizes models to predict future trends, directions, and development processes within a time series. While commonly used classical statistical forecasting methods, including linear regression (LR) and autoregressive moving average model (ARIMA), have limitations such as underfitting, the Prophet-SARIMA model has been proposed by Zhao et al. [4] to solve these problems. The SARIMA model helps improve the limitations of the ARIMA model by accounting for data periodicity, and the Prophet model applies to a wide range of forecasting problems with potential unique features. Combining these models allows for better identification of change points, seasonality, holidays, and emergencies in time series data. However, since training and prediction must be conducted separately, any updates to the model require retraining. Additionally, the Prophet model is not suitable for identifying complex patterns. Markov models are also utilized in time series forecasting; however, these models have limitations in forecasting problems due to the lack of relevance between current and historical data. With the development of deep learning, the use of the recurrent neural network (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU) have shown to improve prediction results. Liu et al. [5] proposed an RNN-based model for vehicle mobility prediction and a rolling subway passenger flow prediction model, while Wang et al. [6] used the LSTM model to solve traffic forecasting problems. While the traditional statistical method has limitations with periodic data, deep learning methods have more accurate predictions but face complexity in training and parameter adjustment.

2.2. Hyperparameter optimization

The selection of hyperparameters for neural network models is a test of researchers' experience and often requires numerous adjustments to achieve better results. As such, automatic optimization of hyperparameters has garnered attention from many researchers in recent years. Shao et al. employed an improved Particle Swarm Optimization (PSO) algorithm to optimize nickel price predictions based on the LSTM

neural network[7]. This method utilized the enhanced PSO algorithm to optimize the number of neurons in the LSTM network. However, the PSO algorithm is prone to falling into local optimal solutions when optimizing neural network parameters. To address this issue, Han et al. proposed an improved Adaptive Particle Swarm Optimization (APSO) algorithm for optimizing rainy road traffic speed prediction model parameters of the GRU network[8]. They used an adaptive nonlinear inertia weight method to strike a balance between the PSO algorithm's local and global search capabilities. Yang et al. suggested an enterprise network marketing prediction model based on a Genetic Algorithm (GA)-optimized BP neural network[9]. However, due to the algorithm's randomness, the local search capability of the GA is poor. Grid search is another common optimization technique; B.D. et al. utilized this method to optimize the hyperparameters of a machine learning model, leading to improved predictions[10]. Traditional grid search algorithms, though, suffer from an excessively large invalid search range and sensitivity to search step size. To overcome these drawbacks, Sun et al. proposed an improved grid search algorithm to optimize the relevant parameter values of the Support Vector Regression (SVR) model, resulting in more accurate predictions[11].

Most traditional optimization algorithms tend to operate slowly and fall into local optimal solutions. In contrast, reinforcement learning has recently gained popularity for its efficient optimization capabilities. Wu et al. introduced an efficient model-based hyperparameter optimization method that models the optimization process as a reinforcement learning model, allowing an agent to sequentially tune hyperparameters and dynamically adjust the usage model[12]. Chen et al. applied deep reinforcement learning with model acceleration to address hyperparameter optimization problems[13]. This method incorporated a predictive reward value module and a guide pool to steer the agent's exploration in the search space, ultimately reducing search time and enhancing search efficiency.

In conclusion, traditional optimization algorithms generally suffer from slow convergence speed, low solution accuracy, and a tendency to fall into local optima. Reinforcement learning-based optimization algorithms, on the other hand, demonstrate high search efficiency and deliver solution strategies from scratch, with superior efficiency, robustness, and generalization.

3. Model Definition

3.1. QLBiGRU

We establish the three parameters to be optimized as a state, generating a set of actions in each state. The agent continuously interacts with the environment to find the optimal strategy. Initially, predictions are made based on the environment's initial state, resulting in a set of actions $A = [a_1, a_2, a_3]$, which serves as the initial strategy. This strategy is then executed within the environment, and the resulting RMSE value is provided as a reward signal. The Q table is updated accordingly. Next, predictions are made based on the updated strategy using the ε-greedy strategy with $\varepsilon = 0.9$ to sample. The output is a new strategy $A' = [a_1', a_2', a_3']$, and the reward value under this updated strategy is calculated. Here, A represents the initial strategy, while A' represents the updated strategy. During the BiGRU prediction phase, standardized data is sent to the three-layer BiGRU network for computation, and the prediction result is generated (see Fig. 1).

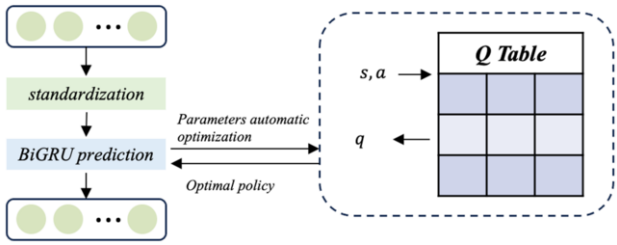

Fig. 1. QLBiGRU framework

First, we define the state. The state in this article is the three hyperparameters that need to be optimized for the BiGRU model, which are the number of neurons, the number of iterations, and the learning rate, as shown below:

$$S = [s1, s2, s3] \tag{1}$$

Among them, s_1 represents the state of the number of neurons, s_2 represents the state of the number of iterations, and s_3 represents the state of the learning rate.

We send the state S into the BiGRU model for prediction and update the model after execution. When updating the Q table (Formula 2), we expect to get the minimum Q value, so the difference between the Q value of the previous state and the subsequent state is calculated to obtain a better Q value and then a better strategy:

$$Q(s,a) = Q(s,a) - \alpha Q(s',a') - Q(s,a) \tag{2}$$

Among them, $Q(s,a)$ represents the Q value of the current state, $Q(s',a')$ represents the Q value of the next state, $Q(s',a')$is calculated as follows:

$$Q(s',a') = \begin{cases} R(s,a) - \gamma maxQ(s',a'), \text{NT} \\ R(s,a), T \end{cases} \tag{3}$$

Among them, NT is "not terminate" and T is "Terminated", α is the learning rate, $\gamma \in [0,1]$ is the discount factor, $R(s,a)$ means to take an action a in state s and transfer to s' The reward function value obtained in the prime state, a' is the next action a chosen by the greedy strategy. The calculation of the reward function in this chapter is defined as follows:

$$R(s,a) = RMSE(s,a) \tag{4}$$

Among them, the value of $R(s,a)$ is the RMSE value obtained by executing action a in state s.

3.2. BiGRU

During the experiment, we constructed a three-layer BiGRU network. Each layer of the BiGRU consists of two GRUs connected in series but in opposite directions. The output of each layer is determined collaboratively by these two GRUs, as illustrated in Fig.2.

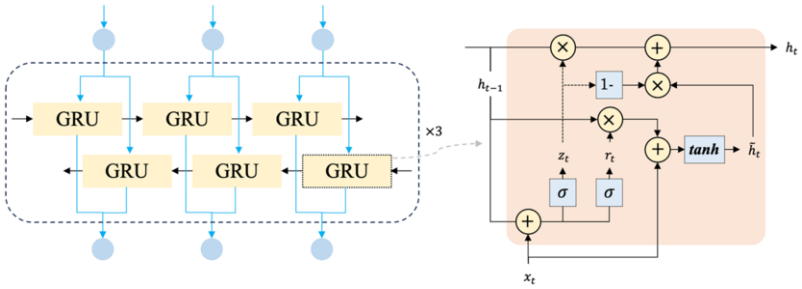

Fig. 2. BiGRU model structure

After the sales data is passed into BiGRU, the forward propagation starts. We regard all the GRUs in the two parts as a BiGRU layer, and the calculation of the first layer is as follows (Formula 5):

$$h_t^l = f\left(W_{\vec{h}_t}^l \vec{h}_t^l + W_{\overleftarrow{h}_t}^l \overleftarrow{h}_t^l + b_t^l\right) \tag{5}$$

Among them, h_t^l represents the hidden layer state at the moment of the *l*-layer BiGRU network, b_t^l is the bias of the hidden layer state at the *l*-layer moment, $W_{\vec{h}_t}^l$, $W_{h_t}^l$ represent the *l*-layer The BiGRU forward hidden state and backward hidden state weight coefficients, $\vec{h}_t^l$ and h_t^l are the states of the forward and backward hidden layers at the *l*-layer respectively, and their calculations are as follows:

$$\vec{h}_t^l = GRU^l\left(x_t^l, \vec{h}_{t-1}^l\right) \tag{6}$$

$$\overleftarrow{h}_t^l = GRU^l\left(x_t^l, \overleftarrow{h}_{t-1}^l\right) \tag{7}$$

Among them, $\vec{h}_t^l$ and htl represent the state of the forward and backward hidden layers of layer l at time $t-1$. x_t^l is the input at the moment of layer l, and the update gate calculation of layer *l* of the GRU network is as follows:

$$z_t^l = \sigma(W_z^l x_t^l + U_z^l h_{t-1}^l + b_z^l) \tag{8}$$

Among them, z_t^l is the update gate of the first layer, then W_z^l and U_z^l are the weights of the update gate of the *l* layer, b_z^l is the bias, and σ is the Sigmoid activation function. The calculation of the layer-*l* reset gate is as follows:

$$r_t^l = \sigma(W_t^l x_t^l + U_t^l h_{t-1}^l + b_t^l) \tag{9}$$

Among them, represents the l layer reset gate, $W_t^l$and U_t^l are the weights, and b_t^l is the bias. The hidden layer state of layer l is as follows:

$$h_t^l = z_t^l h_{t-1}^l + (1 - z_t^l)\tilde{h}_t^l \tag{10}$$

Among them, h_{t-1}^l and h_t^l respectively represent the hidden layer state at time and time; $\tilde{h}_t^l$ represents the candidate hidden layer state, its calculation is as follows:

$$\tilde{h}_t^l = tanh\left(W_h^l x_t^l + U_h^l (r_t^l h_{t-1}^l) + b_h^l\right) \tag{11}$$

Among them, W_h^l and U_h^l represent the weight of the hidden layer state of layer l, and b_h^l represents the bias of the hidden layer. The specific calculation process is shown in Algorithm 1.

Firstly, the model takes standardized sales time series data as input and predicts the output using the BiGRU model. During the parameter optimization process, the first step involves defining the range for the three optimization parameters. Next, the Q-Learning algorithm's parameters are initialized, and the optimization parameters are established as states. The Q table is also initialized. The process starts from state s_1, where the agent selects and executes action a_1. The agent interacts with the BiGRU environment, calculates the RMSE value, observes the reward function value and s', updates the Q table, and transitions to the next state, s_2. This cycle continues until the termination state, s_3, is reached. At this point, the algorithm updates the state and begins the cycle again from s_1. The algorithm terminates when the maximum number of iterations is reached.

Algorithm1. QLBiGRU
Input: Normalized state data
Output: The optimized parameters enter the BiGRU model, and the prediction results are obtained

1: Set ranges for the number of neurons, number of iterations, and learning rate
2: Initialize the parameters of the Q-Learning algorithm
3: Build state relationships between optimization parameters
4: Initialize $Q(s,a)$
5: **repeat:**
6: initial state s
7: **repeat:**
8: Choose action a from state s in $Q(s,a)$
9: Execute action a, enter the BiGRU model to get the predicted value,
10: To calculate the RMSE value, observe R, s'
11: Calculate the value of $Q(s,a)$by formula 2
12: $s = s'$
13: Calculate the value of $Q(s,a)$by formula 2
13: **until** reaches the terminal state s_3
14: **until** reached the maximum number of iterations
15: **return** Prediction results of BiGRU prediction under the optimal strategy (optimal hyperparameter combination)

4. Result and comparison analysis

4.1. Model parameter setting

During the experiment, we initially defined the parameter ranges for optimization. For instance, the range for the number of neurons was set to $[1, 200]$, the number of iterations was set to $[1, 100]$, and the learning rate was set to $[0.001, 0.005]$. Subsequently, during runtime, we employed the Adam optimizer for gradient descent, and the model was configured with a batch size of 128. Details of other hyperparameters can be found in Table 1:

Table 1. QLBiGRU model parameter settings.

parameters	value
Batch size	128
Loss function	RMSE
Optimizer	Adam

Dropout value	0.5
S_1 (The value range of neuron numbers)	[1, 200]
S_2 (Epoch value range)	[1, 100]
S_3 (Learning rate range)	[1e-3, 5e-3]
α init learning rate	0.01
γ discount factor	0.9
episode iterations	100

4.2. Datasets

Dataset 1: The Cainiao Demand Forecasting and Warehouse Planning Competition Dataset (ETD) from the 2016 Tianchi Competition. This dataset comprises real sales data from Alibaba's e-commerce transactions, encompassing 5,778 products between October 2014 and December 2015.

Dataset 2: The 2021 Alibaba Cloud Infrastructure Supply Chain Competition (SCD) features daily product inventory and demand data spanning June 2018 to March 2021. The product inventory data includes 7 columns: unit, date, product demand, geographical information, geographical aggregation dimension, product information, and product aggregation dimension.

Dataset 3: The Society for Supply Chain and Operations Management-Shanshu Technology Practice-Driven Research Competition Dataset (SSD) contains historical sales data of 72 SKU products across 18 DC warehouses. The dataset includes daily sales information for each DCSKU combination from January 1, 2018, to July 30, 2020. In total, there are 1,080 unique DCSKU combinations and over 830,000 records. The historical sales data encompasses fields such as sales date, warehouse ID, commodity ID, DC*SKU combination ID, and sales volume (box).

4.3. Analysis

To verify the effectiveness of our proposed QLBiGRU model and evaluate its performance in sales forecasting, we conducted experiments of various scales on three datasets: ETD, SCD, and SSD. The prediction lengths considered were 3, 5, 7, and 14. The results of these experiments are presented in Tables 2, 3, and 4.

In addition, we compared our QLBiGRU model with six other baseline methods, namely RNN, LSTM, BiGRU, PSO-LSTM, ACO-LSTM, and IACO-BiGRU. We performed experiments using these methods and analyzed the results. Overall, the QLBiGRU model demonstrated superior performance compared to the other methods. RNN performed the worst due to the issue of gradient disappearance or explosion, which resulted in poor prediction results. LSTM, on the other hand, generally outperformed RNN, but its larger parameter size led to longer running times. GRU, with fewer parameters and slightly faster execution, offered a viable alternative to LSTM. Moreover, BiGRU exhibited the ability to capture contextual information effectively, making it a more reasonable choice for an e-commerce sales forecasting model.

From the experimental data, it is evident that the prediction results obtained after optimizing the model parameters were superior to manually set ones. This highlights the limitations of relying solely on experiential parameter settings, which can affect the

predictive performance of the model. Automatic optimization proves to be more accurate and therefore yields better results. Specifically, in the SSD dataset experiment (Table 4), considering product 4 as an example, when the prediction length is 7, the evaluation indicators of the predicted values obtained by PSO-LSTM were 47.4%, 3.3%, and 27.5% higher than those of LSTM, respectively. Similarly, when the prediction length is 14, the evaluation indicators of IACO-BiGRU improved by 37.5%, 14.4%, and 20.9% compared to BiGRU. These findings affirm the necessity of optimizing the model parameters.

Table 2. Experimental results of the ETD dataset

(Best result in bold, * indicates sub-optimal result)

		Step=3			Step=5		
Commodity	**Method**	**MSE**	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**
	RNN	0.4440	0.3025	0.6663	0.7726	0.4991	0.8790
	LSTM	0.5061	0.2948	0.7114	0.4265	0.2758	0.6531
	BiGRU	0.2617	0.1986	0.5115	0.2858	0.2181	0.5346
1	PSO-LSTM	0.0739	0.1881	0.2718	0.0744	0.1790	0.2728
	ACO-LSTM	0.0661	0.1723	0.2571	0.0639	**0.1516**	0.2528
	IACO-BiGRU	0.0552*	**0.1502**	0.2350*	0.0601*	0.1520*	0.2453*
	QLBiGRU	**0.0354**	0.1539*	**0.1883**	**0.0391**	0.1611	**0.1978**
	RNN	0.6464	0.4040	0.8040	0.7279	0.3358	0.8531
	LSTM	0.4042	0.2893	0.6358	0.4949	0.2984	0.7034
	BiGRU	0.2836	0.2425	0.2836	0.2797	0.2280	0.2797
2	PSO-LSTM	0.0746	0.2019	0.2732	0.0819	0.2050	0.2862
	ACO-LSTM	0.0694	0.1679	0.2635	0.0638	0.1572*	0.2526
	IACO-BiGRU	0.0541*	0.1377*	0.2326*	0.0532*	0.1626	0.2308*
	QLBiGRU	**0.0269**	**0.1330**	**0.1642**	**0.0273**	**0.1368**	**0.1654**

Table 3. Experimental results of the SCD dataset

(Best result in bold, * indicates sub-optimal result)

		Step=3			Step=5		
Commodity	**Method**	**MSE**	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**
	RNN	0.8336	0.3293	0.9130	0.8783	0.2976	0.9372
	LSTM	0.8091	0.3092	0.8995	0.7676	0.3102	0.8761
	BiGRU	0.7247	0.3024	0.8512	0.7511	0.3141	0.8666
1	PSO-LSTM	0.1937	0.3123	0.4401	0.2361	0.2780	0.4859
	ACO-LSTM	0.0419	0.1314*	0.2048	0.0455	0.1404*	0.2133
	IACO-BiGRU	0.0313*	0.1361	0.1771*	0.0378*	0.1606	0.1944*
	QLBiGRU	**0.0239**	**0.1295**	**0.1546**	**0.0245**	**0.1278**	**0.1565**
	RNN	0.8046	0.3215	0.8970	0.8942	0.3606	0.9456
	LSTM	0.7438	0.3267	0.8624	0.7306	0.3021	0.8547
	BiGRU	0.6229	0.3482	0.7892	0.6865	0.2959	0.8286
2	PSO-LSTM	0.1861	0.1849	0.4314	0.2195	0.2045	0.4685
	ACO-LSTM	0.0437	0.1433	0.2091	0.0457	0.1302*	0.2139
	IACO-BiGRU	0.0313*	0.1250*	0.1771*	0.0349*	0.1384	0.1869*
	QLBiGRU	**0.0217**	**0.1149**	**0.1474**	**0.0242**	**0.1246**	**0.1558**

Table 4. Experimental results of the SSD dataset

(Best result in bold, * indicates sub-optimal result)

Commodity	**Method**	Step=7			Step=14		
		MSE	**MAE**	**RMSE**	**MSE**	**MAE**	**RMSE**
	RNN	0.2017	0.3863	0.4491	0.2391	0.4045	0.4890
	LSTM	0.1487	0.2040	0.3856	0.1646	0.3008	0.4057
	BiGRU	0.1304	0.2895	0.3611	0.1466	0.1978	0.3829
1	PSO-LSTM	0.1062	0.2249	0.3259	0.1140	0.2267	0.3377
	ACO-LSTM	0.0956	0.2082	0.3092	0.0976	0.1925*	0.3124
	IACO-BiGRU	0.0874*	**0.1682**	0.2956*	0.0965*	0.1932	0.3106*
	QLBiGRU	**0.0653**	0.1885*	**0.2556**	**0.0615**	**0.1728**	**0.2480**
2	RNN	0.2193	0.2166	0.4683	0.2356	0.2238	0.4854

	LSTM	0.1314	0.1786	0.3626	0.1407	0.1854	0.3751
	BiGRU	0.1131	0.1774*	0.3363	0.1367	0.1828*	0.3697
	PSO-LSTM	0.1081	0.2590	0.3288	0.1083	0.2613	0.3292
	ACO-LSTM	0.1035	0.2218	0.3217	0.1174	0.2115	0.3427
	IACO-BiGRU	0.0930*	0.2010	0.3049*	0.0923*	0.1937	0.3039*
	QLBiGRU	**0.0566**	**0.1758**	**0.2380**	**0.0608**	**0.1758**	**0.2465**
3	RNN	0.2086	0.2080	0.4567	0.2454	0.2139	0.4954
	LSTM	0.1794	0.2221	0.4235	0.1887	0.2665	0.4344
	BiGRU	0.1322	0.1855	0.3635	0.1388	0.2283	0.3726
	PSO-LSTM	0.1117	0.2535	0.3343	0.0990	0.2228	0.3146
	ACO-BiGRU	0.0929	0.1805*	0.3048	0.0936	0.2025*	0.3060
	IACO-BiGRU	0.0877*	0.2129	0.2962*	0.0891*	0.2061	0.2985*
	QLBiGRU	**0.0500**	**0.1648**	**0.2237**	**0.0541**	**0.1653**	**0.2327**
4	RNN	0.2081	0.2064	0.4562	0.2144	0.1984	0.4630
	LSTM	0.1904	0.1934	0.4363	0.1919	0.2147	0.4380
	BiGRU	0.1186	0.1739	0.3444	0.1397	0.1899	0.3738
	PSO-LSTM	0.1000	0.1869	0.3163	0.1140	0.2267	0.3377
	ACO-BiGRU	0.0912	0.1806*	0.3021	0.0970	0.1922	0.3114
	IACO-BiGRU	0.0822*	**0.1806**	0.2867*	0.0873*	**0.1625**	0.2954*
	QLBiGRU	**0.0693**	0.1983	**0.2633**	**0.0686**	0.1803*	**0.2620**

Additionally, to further validate the performance of the QLBiGRU algorithm, we compared the super-parameter optimization time among different methods. We conducted five experiments for each method and calculated the average value. The results are presented in Figure 3. As observed, the QLBiGRU model exhibited the shortest running time and high search efficiency.

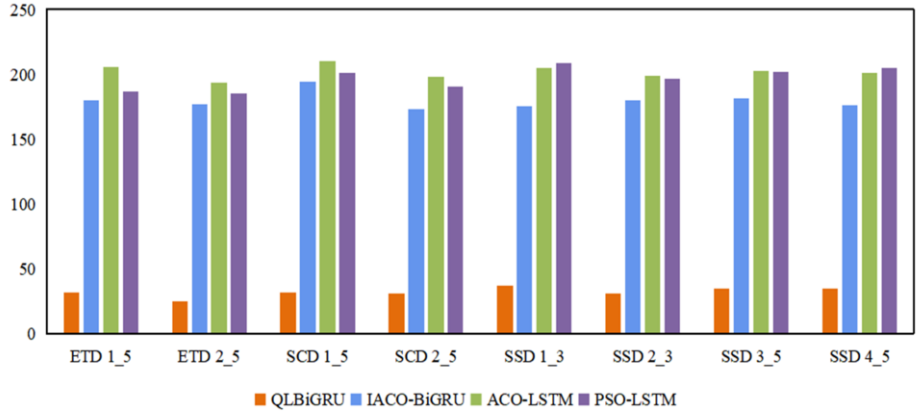

Fig. 3. Comparison of running time

We compared the experimental errors using three different reward functions: RMSE, MAE, and MSE (Fig. 4). Based on the results, it is evident that the use of RMSE as a reward function yields the smallest prediction error. Therefore, it is more reasonable to utilize RMSE as the preferred reward function.

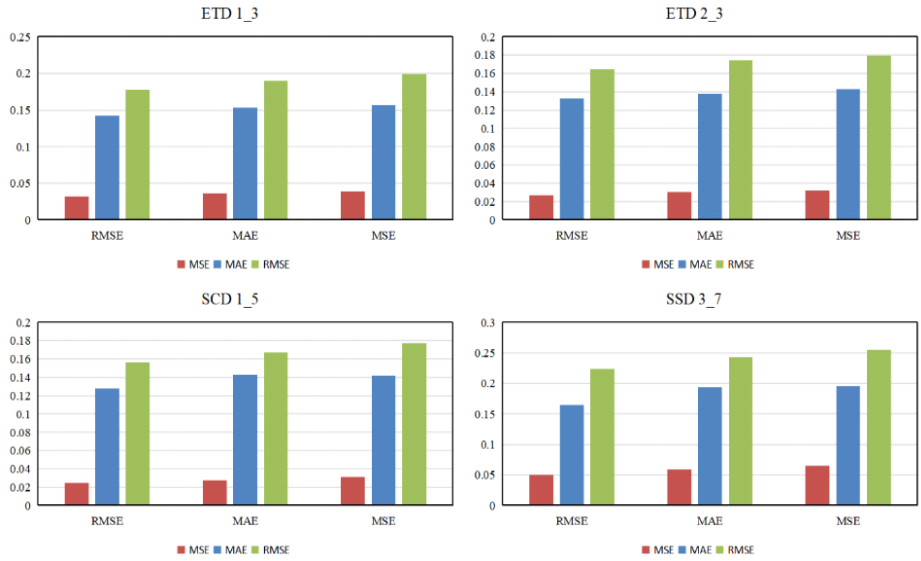

Fig. 4. Error comparison when three evaluation indicators are set as reward functions

To analyze the convergence performance of the model, we compared the convergence curves of its loss function across different datasets (Fig. 5). The results demonstrate that QLBiGRU exhibits faster convergence specifically on the SSD dataset.

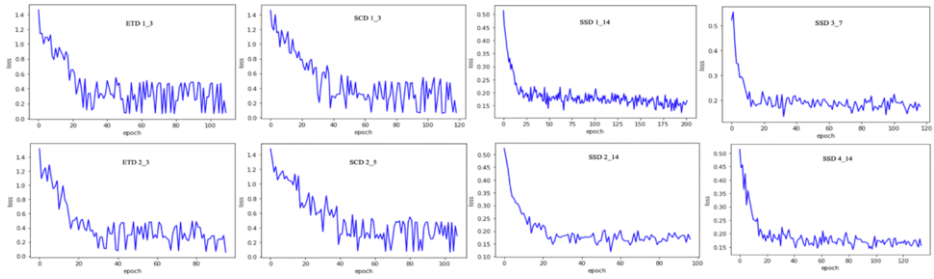

Fig. 5. Loss values of the model on three data sets

Finally, the prediction results obtained using QLBiGRU consistently outperform those of the PSO-LSTM, ACO-LSTM, and IACO-BiGRU models. Analyzing the experimental results from the ETD dataset (Table 2), we observe that for commodity 1, when the prediction length is 3, the QLBiGRU model achieves a 52.0% improvement in the three prediction evaluation indicators compared to the PSO-LSTM model, with a QLBiGRU ratio of 30.7%. Additionally, the ACO-LSTM model experiences a respective increase of 46.4%, 10.6%, and 26.7% in MSE, MAE, and RMSE. When the prediction length is 5, the QLBiGRU model demonstrates a 34.9% increase in MSE and a 19.3% increase in RMSE compared to the IACO-BiGRU model. The advantages of the Q-Learning algorithm in addressing the super-parameter optimization problem compensate for the limitations of the PSO and ACO algorithms, which are prone to local optima, thus leading to improved prediction results.

5. Conclusion

Based on the e-commerce sales forecasting, we employed the Q-Learning algorithm in combination with the BiGRU model to enhance the prediction and optimization of traditional heuristic algorithms. The traditional approaches often face challenges such as getting stuck in local optimal solutions, long execution times, and inadequate performance. The effectiveness of our proposed QLBiGRU model was evaluated across different prediction scales on three distinct datasets: ETD, SCD, and SSD. Following a comprehensive evaluation, we concluded that the QLBiGRU model exhibits superior performance in accurately predicting sales for various commodities at each scale.

Acknowledgment

Project supported by the Key Research and Development Program of Gansu Province, China (No.22YF7GA171), the University Industry Support Program of Gansu Province, China (No.2023QB-115), the Innovation Fund for Science and Technology-based Small and Medium Enterprises of Gansu Province, China (No.23CXGA0136) and the Scientific Research Project of the Lanzhou Science and Technology Program, China (No.2018-01-58).

References

[1] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology," 1967.
[2] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician,* vol. 72, no. 1, pp. 37-45, 2018.
[3] T. Van Calster, B. Baesens, and W. Lemahieu, "ProfARIMA: A profit-driven order identification algorithm for ARIMA models in sales forecasting," *Applied Soft Computing,* vol. 60, pp. 775-785, 2017.
[4] J. Zhao and C. Zhang, "Research on sales forecast based on prophet-SARIMA combination model," in *Journal of Physics: Conference Series*, 2020, vol. 1616, no. 1: IOP Publishing, p. 012069.
[5] W. Liu and Y. Shoji, "DeepVM: RNN-based vehicle mobility prediction to support intelligent vehicle applications," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 6, pp. 3997-4006, 2019.
[6] J. Q. Wang, Y. Du, and J. Wang, "LSTM based long-term energy consumption prediction with periodicity," *Energy,* vol. 197, p. 117197, 2020.
[7] B. Shao, M. Li, Y. Zhao, and G. Bian, "Nickel price forecast based on the LSTM neural network optimized by the improved PSO algorithm," *Mathematical Problems in Engineering,* vol. 2019, 2019.
[8] D. Han, X. Yang, G. Li, S. Wang, Z. Wang, and J. Zhao, "Highway traffic speed prediction in rainy environment based on APSO-GRU," *Journal of advanced transportation,* vol. 2021, pp. 1-11, 2021.
[9] R. Yang, "Enterprise network marketing prediction using the optimized GA-BP neural network," *Complexity,* vol. 2020, pp. 1-9, 2020.
[10] D. M. Belete and M. D. Huchaiah, "Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results," *International Journal of Computers and Applications,* vol. 44, no. 9, pp. 875-886, 2022.
[11] Y. Sun, S. Ding, Z. Zhang, and W. Jia, "An improved grid search algorithm to optimize SVR for prediction," *Soft Computing,* vol. 25, pp. 5633-5644, 2021.
[12] J. Wu, S. Chen, and X. Liu, "Efficient hyperparameter optimization through model-based reinforcement learning," *Neurocomputing,* vol. 409, pp. 381-393, 2020.
[13] S. Chen, J. Wu, and X. Chen, "Deep reinforcement learning with model-based acceleration for hyperparameter optimization," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019: IEEE, pp. 170-177.