


Received January 15, 2019, accepted February 1, 2019, date of publication February 5, 2019, date of current version February 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2897639

An Adaptive Optimization Algorithm Based on Hybrid Power and Multidimensional Update Strategy

JUN HU¹ , (Member, IEEE), AND WENDONG ZHENG^{1,2}

¹College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

²Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

Corresponding author: Jun Hu (hujun_111@hnu.edu.cn)

This work was supported by the Guangxi Key Laboratory of Trusted Software under Grant kx201537.

ABSTRACT Recently, the adaptive learning rate optimization algorithm has shown excellent performances in the field of deep learning. However, the exponential moving average method can lead to convergence problem in some cases, such as it only converges to the sub-optimal minimum. Although AMSGrad algorithm provides solutions for convergence problems, the actual performance is close to or even weaker than that of Adam. In this paper, a new updating rule is proposed based on mixed high power historical and current squared gradients to construct a targeted first-order optimization algorithm for the adaptive learning rate. This algorithm not only overcomes the convergence problems encountered in most current optimization algorithms but also has a quick convergence. It outperforms the state-of-the-art algorithms on various real-world datasets, i.e., the forecast root-mean-square error performance is improved by about 20% on average than that of Adam and AMSGrad algorithms in time series prediction tasks.

INDEX TERMS Adaptive stochastic first order algorithm, time series prediction, temporal correlation.

I. INTRODUCTION

Since generalization ability is efficiently achieved by appropriate optimization algorithms for model training, the deep learning model has shown excellent performances in solving a wide range of problems in the scientific and industrial fields [1], [2], [32]–[34]. Although it has a good performance in most cases, there are still problems to be solved. For example, the convergence of the optimization algorithm should be ensured to avoid falling into local optimum. In addition, it is also challenging to construct optimization methods that adapt to the characteristics of learning tasks in different application areas [3]–[5].

Currently, stochastic gradient descent is a popular method for the deep learning optimization, and widely used in Adam [6], RMSProp [7], and Adadelta [8]. Through the square root of an exponential moving average of the gradient square produced by iteration, the current gradient is scaled and the weights are updated. The gradient update rule only uses the gradient information of a short time period. In some

learning tasks, this method is not ideal, because the algorithm can only converge to the sub-optimal minimum point [9]. Additionally, the temporal correlation between data cannot be represented by the stochastic gradient descent method, so that the performance of the regression task of time series should be enhanced. In this study, a new algorithm is developed with the consideration of the data temporal correlation, referring to the concept of correlation coefficient in statistics.

In statistics, the correlation coefficient of random variables is used to calculate the correlation strength between random variables, including the temporal correlation of data. Designing a first-order optimization algorithm is useful for capturing the temporal correlation of gradient information. To construct a novel first order optimization algorithm for deep learning model, a high power multidimensional, updating strategy is proposed, in which the current and historical squared gradient information are mixed and weighted. When calculating the gradient of small batch loss function about corresponding parameters in the optimization algorithm, a strong correlation coefficient for time-series tasks with strong temporal correlation is reflected by the gradient. Test results show that the consideration of the temporal correlation of data can significantly

The associate editor coordinating the review of this manuscript and approving it for publication was Li He.

improve the algorithm performance in time series tasks, even in the classification tasks with weak correlation.

The major contribution of this work includes two parts.

First, the AdaHMG (adaptive hybrid high power multi-dimensional gradient) algorithm is proposed with specific rules for updating the squared gradient according to the characteristics of temporal correlation between data. It outperforms the state-of-the-art algorithms in time series prediction tasks. Compared with the Adam algorithm, there is a slight improvement for image recognition tasks. Second, the AdaHMG algorithm performs better in convergence than the current optimization algorithms. The detailed proof of its convergence is obtained. Experimental results show that the convergence speed of this algorithm is significantly faster than that of Adam and AMSGrad algorithm.

In the following sections, the AdaHMG algorithm and the motivate update rules are described. Besides, advantages of AdaHMG over Adam and AMSGrad algorithms are discussed.

II. BACKGROUND WORKS

A. STOCHASTIC GRADIENT DESCENT METHOD

The stochastic gradient descent method and its variants are widely-used optimization algorithms in deep learning. The idea of stochastic gradient descent is extensively studied in convex optimization and non-convex optimization [10]–[12]. In the field of convex optimization, a convex optimization stochastic coordinate descent template is proposed in literature [13]. For non-convex optimization, a class of variant algorithms is further proposed to effectively solve the problem of smooth non-convex optimization in literature [14], and non-convex targets in gradient ascent are solved in literature [15]. Another branch of stochastic gradient descent method, namely the adaptive learning rate optimization algorithm, is also a hot research topic.

B. ADAPTIVE LEARNING RATE ALGORITHM

AdaGrad is the first algorithm that can independently adapt to the learning rate of all hyperparameters [16]. Although the AdaGrad is theoretically satisfied, premature or even excessive reduction in the learning rate can be caused by the accumulated squared gradients at the beginning of the deep model training in empirical studies. The subsequent RMSProp algorithm improves the AdaGrad algorithm and performs better under the non-convex optimization. RMSProp uses exponential moving average method to discard the squared values of gradients which are far away from the current time step. The algorithm converges quickly after finding the convex structure. Although RMSProp is an effective optimization algorithm, the deviation correction of the second order moment estimation of the gradient is not considered. The learning rate of the AdaGrad and RMSProp algorithms changes slowly in some data sets of smaller and more consistent gradients. The NAG algorithm proposes momentum technology by analogy with Newton's law of motion, i.e., introducing

a “momentum factor” to speed up learning. After a big jump, some correction is made to slow down the learning rate before it meets the ramp again [17]. The Adam algorithm adds the deviation correction term and incorporates the momentum technique into the first order moment estimation of the gradient, showing strong robustness in the practical application [6]. The NADAM algorithm combines the advantages of the Adam and NAG algorithms and uses the momentum to update the parameters before calculating the gradient, guaranteeing a more accurate determination of the direction of gradient change [18]. The AMSGrad algorithm overcomes the problem that Adam cannot converge to the optimal solution in some cases [19] and proposes a new parameter update method, in which maximum values of the current and historical squared gradients are used instead of the exponential moving average. However, the current study is not empirically superior to the Adam algorithm.

There are certain limitations in the AMSGrad algorithm. First, the max function is only a preliminary implementation of the long-term memory of gradient information, the value selected from the historical and the current gradient information cannot effectively capture the long-term memory of gradient information. What is more, removing the deviation correction term of the exponential moving average model can result in the accumulated error, thus empirical performance of AMSGrad algorithm is poor in most areas.

C. THE TEMPORAL CORRELATION OF THE DATA

Recently, the generalization performance of the model is improved by the dynamical capture of the spatiotemporal correlation characteristic reflected by the data. A new learning method of spatiotemporal correlation characteristic is proposed to effectively train the model by encoding the spatio-temporal correlation information of the sequence data [20], [21]. Then clustering algorithms are used to extract time patterns in time series data [22], and through empirical analysis of preschool education, the accuracy of the prediction is improved by time patterns. However, only the time pattern is mined in this research, the general method is not used for capturing the correlation characteristic of time data in the optimization algorithm.

AdaHMG algorithm in this paper not only considers the long-term memory problem in AMSGrad study, but also updates the historical and the current time step of the square gradient of the common impact. The key to AdaHMG algorithm is to design a new hybrid high power squared gradient updating strategy to capture the characteristics of the temporal correlation of the gradient information. Based on the temporal correlation characteristics of the data in the application, a new adaptive learning rate first-order optimization algorithm is constructed. To the best of our knowledge, the method of capturing data temporal correlation by the updating rules of the internal gradient information of the algorithm has not been applied by the first-order optimization algorithm in deep learning.

III. ADAHMG ALGORITHM

To overcome the convergence of the current adaptive learning rate optimization algorithms, a novel hybrid high power, multidimensional adaptive optimization algorithm (AdaHMG) is constructed to improve the performance on the datasets with strong temporal correlation. With the advantages of Adam and AMSGrad, AdaHMG algorithm owns an improved performance. A new updating rule of mixing historical and current squared gradients are designed based on the exponential moving average model, and the fast convergence and performance improvement of the algorithm are obtained. AdaHMG algorithm includes the main steps as follows: in the iterative process with time t , the gradient is calculated at first, then the first-order and second-order moment estimate of the gradient are obtained, the squared gradient update formula is added, then the respective deviation correction terms are calculated to achieve the unbiased estimate, and finally the x_t is updated until the convergence of the algorithm.

Algorithm 1 shows the pseudocode of the proposed AdaHMG algorithm. Unlike Adam and AMSGrad algorithms, AdaHMG algorithm considers the correction term of exponential moving average and involves a formula of historical and current squared gradients.

For the simplicity of the loop, the same approach is used as that of Adam algorithm, i.e., the pseudo-code is combined for correcting deviations on lines 4 of the while loop with the learning rate (α) over time:

$$a_t = \alpha \cdot \frac{m_t}{\sqrt{\frac{v_t}{1-\beta_2^t}}} = \alpha \cdot \sqrt{1-\beta_2^t} / (1-\beta_1^t) \cdot \frac{m_t}{\sqrt{v_t}} \quad (1)$$

First, Adam's deviation correction is kept in Eq. 1 because there is a deviation in the iterative algorithm for calculating first and second order moment estimates. If the deviation correction was excluded, there are some errors in AMSGrad algorithm when updating parameters iteratively. The combination of AdaHMG algorithm can successfully avoid the deviation. Second, it should be noted that the last step in pseudocode is to update the original v_t (in Eq.1) with the calculated \hat{v}_t . The advantages of convergence rate and the value of the hybrid update strategy for current and historical squared gradient information are discussed in sections 4 and 6.

The item $\hat{v}_t = k_1 \hat{v}_{t-1}^2 + k_2 v_t$ calculated in the while loop is directly used in the evaluation in the last line of pseudocode. As described in the pseudocode, k_1 and k_2 are introduced for adjusting the weight of historical and current squared gradient.

To ensure the dual consideration of historical and current squared gradient information, two hyperparameters of k_1 and k_2 have a range of values of $[0, 1)$, and k_1 is greater than k_2 . In addition, the squared gradient might explode when $k_1 \hat{v}_{t-1}^2$ with larger squared gradients, resulting in the NAN of loss function value. To avoid this case, a hyperparameter k_1 which is less than 1 is added to balance the equation and increase

Algorithm 1 AdaHMG, the Proposed Algorithm for Stochastic Optimization.

Require: α : Learning Rate, α_t is the learning rate that varies over time by error correction

Require: $\beta_1, \beta_2 \in [0, 1)$ and $\varepsilon = 1 \times 10^{-8}$: Exponential decay rates for the moment estimates

Require: $k_1, k_2 \in [0, 1)$: and $k_1 > k_2$ (initialize $k_1 = 0.5$, $k_2 = 0.3$)

Require: $f(x)$: Stochastic objective function with hyperparameters x

Require: x_0 : Initial vector, $m_0 = 0$, $v_0 = 0$, $t = 0$

1: While x_t not converged do:

2: $t = t + 1$

3: $g_t = \nabla_t f_t(x_t)$

4: $\alpha_t = \alpha \sqrt{1-\beta_2^t} / (1-\beta_1^t)$

5: $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$

6: $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

7: $\hat{v}_t = k_1 \hat{v}_{t-1}^2 + k_2 v_t$

8: $x_t = x_{t-1} - \alpha_t \cdot m_t / \sqrt{\hat{v}_t} + \varepsilon$

the algorithm converge. The pseudocode of the algorithm is expressed as follows:

IV. CONVERGENCE ANALYSIS

The online learning framework proposed by previous researchers is used to analyze the convergence of the AdaHMG algorithm [23]. Regret is used to evaluate the convergence performance of AdaHMG algorithm on a randomly selected sequence of convex cost functions: $f_1(x), f_2(x), \dots, f_T(x)$, defined as in:

$$R(T) = \sum_{t=1}^T (f_t(x_t) - f_t(x^*)) \quad (2)$$

where $x^* = \arg \min_{x \in X} \sum_{t=1}^T f_t(x)$. The goal is to calculate the summation of the difference between the value of the current time step $f_t(x_t)$ and the optimal solution in T time. The regret of the AdaHMG algorithm is less than $O(\sqrt{dT})$, the proof is provided in the appendix.

Before proving the algorithm convergence, the updated formula of mixed high power multi-dimensional should be adjusted by scaling method, and the following equation is derived:

$$\begin{aligned} \hat{v}_t &= k_1 \hat{v}_{t-1}^2 + k_2 v_t \\ &\geq k_1 v_{t-1}^2 + k_2 v_t \\ &\geq k_1 v_{t-1}^2 + k_2 v_{t-1} \end{aligned} \quad (3)$$

The first premise of Equation 3 is $\hat{v}_t \geq v_t$, and the second premise is the derivation as follows:

$$\begin{aligned} v_t &= (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_j^2 + (1-\beta_2) \beta_2^0 g_T^2 \\ &= v_{t-1} + (1-\beta_2) \beta_2^0 g_T^2 (g_T^2 > 0) \end{aligned} \quad (4)$$

Since $v_t > v_{t-1}$ is known, the time step is unified with easy simplification. The scaled $k_1 v_{t-1}^2 + k_2 v_{t-1}$ is brought into the algorithm pseudocode $x_t = x_{t-1} - \alpha_t \cdot m_t / \sqrt{\hat{v}_t} + \varepsilon$ to represent the \hat{v}_t contraction. Our updated rule guarantees $\|\hat{v}_t\|_2 > \|\hat{v}_{t-1}\|_2$ by analyzing the parameter x_t at each moment.

Theorem 4.1: Assuming that this objective function has a bounded gradient. First, we generalize from the two-norm of the gradient: $\|\nabla f_t(x)\|_2 < G$ to the infinite norm: $\|\nabla f_t(x)\|_\infty < G_\infty$. Suppose χ has bounded diameter D_∞ , then for all $x \in \chi$, there is $\beta_1, \beta_2 \in [0, 1)$ and $\gamma = \beta_1/\beta_2^{\frac{3}{4}} < 1$. AdaHMG algorithm supports for first-order moment estimation by adding momentum technology $\beta_{1t} = \beta_1 \lambda^{t-1}$ and $\alpha_t = \frac{\alpha}{\sqrt{t}}, t \in [1 \dots T]$. In particular, update rules for AdaHMG algorithm can lead to the decrease of the learning rate. Finally, the following regret ($R(T)$) is obtained as:

$$R(T) \leq \frac{D_\infty^2 \sqrt{T}}{\alpha(1-\beta_1)} \sum_{i=1}^d \hat{v}_{T,i}^{\frac{1}{2}} + \frac{\beta_1 D_\infty^2 G_\infty}{2(1-\beta_1)(1-\lambda)^2} + \frac{\alpha \sqrt{1+\log T}}{(1-\beta_1)^2 \sqrt{(1-\beta_2)^2(1-\gamma)} * \sqrt{\frac{4k_1 k_2}{1-\beta_2}}} * \sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}} \quad (5)$$

In Theorem 4.1, when the condition of bounded gradients is satisfied, compared with previous studies: $\sum_{i=1}^d \hat{v}_{T,i}^{\frac{1}{2}} < \sqrt{d}$ and $\sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}} < \sum_{i=1}^d \|g_{1:T,i}\|_2 \ll d\sqrt{T}G_\infty$. The result of right side of inequality is less than the boundary of the previous study, indicating the convergence of the algorithm. In addition, the use of momentum decay in Theorem 4.1 (β_{1t}) still guarantees that the regret of the algorithm satisfies $\tilde{O}(\sqrt{T})$. For all $T \geq Y_1 : \frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}} + \frac{\sqrt{1+\log T}}{T}\right)$. This result can be obtained by using Theorem 4.1 and $\sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}} < \sum_{i=1}^d \|g_{1:T,i}\|_2 \ll \sqrt{dT}$. Thus, $\frac{R(T)}{T} = 0$.

First, compared with Adam, the time and space complexity of AMSGrad are not increased additionally. The computational complexity of both algorithms is $O(\sqrt{T} + \sqrt{1+\log T} \times \sum_{i=1}^d \|g_{1:T,i}\|_2)$. Second, in Eq.5, this term ($\sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}}$) of regret values is significantly smaller than that in previous studies ($\sum_{i=1}^d \|g_{1:T,i}\|_2$). Finally, the computational complexity of the proposed algorithm is $O(\sqrt{T} + \sqrt{1+\log T} \times \sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}})$. Therefore, AdaHMG algorithm has the lower computational complexity than Adam and AMSGrad.

V. EXPERIMENT

The CPU of the experimental device is i7-7700HQ, and the GPU is NVIDIA's GTX 1050 with a memory size of 8GB. The operating system is Windows 10.

The experiment runs under the Anaconda3 integrated development environment with python 3.5. The deep learning framework is TensorFlow1.4.1-gpu [24], and the high-level framework is keras-2.1.5 [25]. Datasets of regression and classification tasks are employed, including common datasets from UCI, datasets from Kaggle Contest, and commercial datasets. The performances of Adam, AMSGrad, and AdaHMG algorithms on the same dataset are mainly compared. RMSE (Root Mean Square Error) is used to evaluate the regression result, and the accuracy of classification is emphasized. The learning rate and hyperparameters β_1, β_2 are same in three algorithms, i.e., the recommended values of previous studies are used (learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$).

A. REGRESSION TASK EXPERIMENTS

Prediction of time series tasks is usually conducted with denser and more contextual data. The datasets include a one-year-long sales dataset of mineral water from a supermarket chain in a province of China, a five-year-long PM2.5 dataset from UCI's U.S. Embassy in Beijing [26], a flight booking dataset from Kaggle [27], and a New York stock dataset [28].

Spring Water Sales Forecasting. In the first experiment, a one-year-long sales dataset of spring water from a supermarket chain in a province of China is used, containing 148,377 sales records. Firstly, the sales volume field from a timestamp count is changed to an hourly count, and the character attributes are individually encoded. Secondly, all data are normalized, and time labels are put on these results. Finally, attribute columns that do not need to be predicted are picked out from the data. The data from the first season are used as a training set and the rest as a testing set (Figure 1 shows only the 24-hour prediction effect) since seasonal factors affect sales. LSTM model provided by Keras (the number of memory cells is 128) and three different optimization algorithms are adapted for training. The loss functions are MAE (mean absolute error), the number of epochs is 50 times, and the batch_size for each epoch is 150. The hyperparameters of the optimization algorithm are recommended values from Adam. L2 regularization is used in the full connection layer to prevent the overfitting.

RMSE value of AdaHMG algorithm is improved by 19.51% compared with that of Adam. Figure 1 (a) shows three algorithms.

As shown in Figure 1 (b) and (c), the loss function value of AdaHMG algorithm is always lower than that of the previous optimization algorithm. After several repeats, the test results show that AdaHMG algorithm has a performance improvement of about 20% over Adam in this task, and the loss function value (training set and validation set) decreases faster during training than that of Adam and AMSGrad.

PM2.5 Air Quality Forecasting. This dataset is the related air data which is obtained from UCI and collected near the US Embassy in China during 2010 to 2014. The preprocessing of data is similar with the previous experiment. The data from the first 1.5 years serve as the training set and the

Adam Test RMSE: 6.842

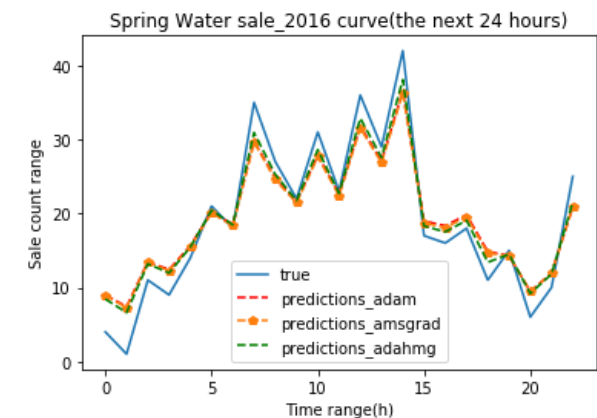
Amsgrad Test RMSE: 6.906

AdaHMG Test RMSE: 5.507

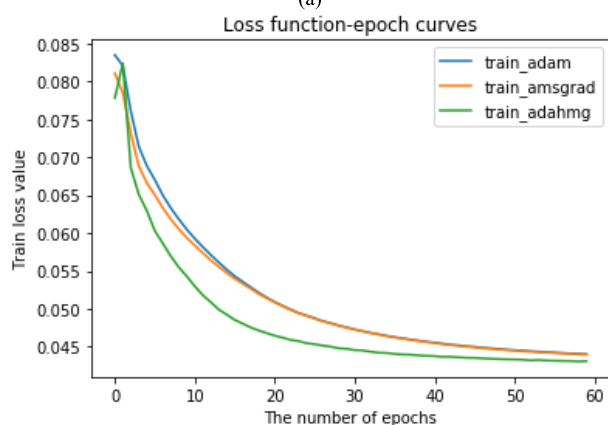
Adam Test RMSE: 7.181

Amsgrad Test RMSE: 7.336

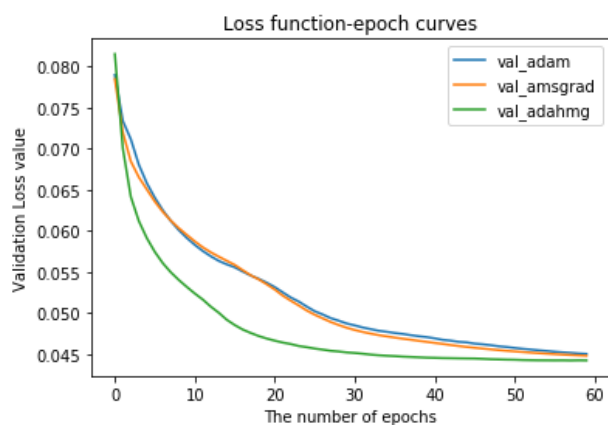
AdaHMG Test RMSE: 5.492



(a)



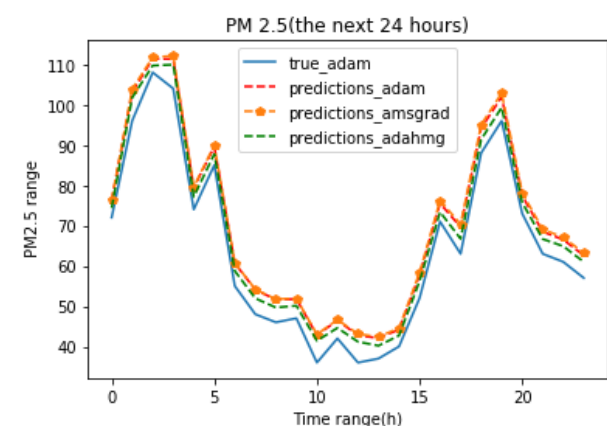
(b)



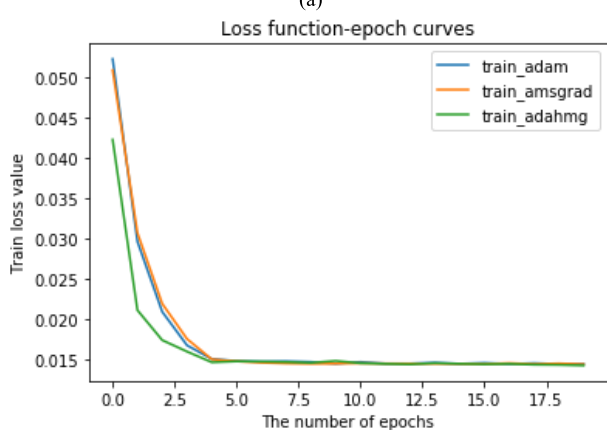
(c)

FIGURE 1. (a) Three algorithms to predict results in 24 hours. (b) Training loss function effect. (c) Validation loss function effect.

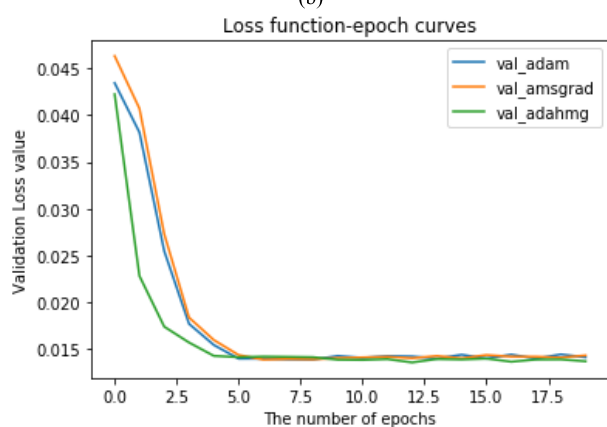
rest as a testing set (or validation set). LSTM model with 128 memory cells is employed. Fully connected layers are used as hidden layers. MAE is used as the loss function in the training model. The epochs number is 20, and the number of batch_size is 120. After training, the RMSE values of the



(a)



(b)



(c)

FIGURE 2. (a) Predicted t results of three algorithms in 24 hours. (b) Training loss function effect. (c) Validation loss function effect.

three algorithms on the testing set are calculated respectively, as shown in Figure 2(a).

As illustrated in Figure 2(b) and (c), loss function values of AdaHMG algorithm on both training set and testing set are significantly lower than that of Adam and AMSGrad

algorithms at the beginning of epochs. After several repeats, the performance of AdaHMG algorithm is improved from 20% to 25% compared with Adam, and the decrease of loss function values (training set-validation set) are slightly faster than Adam and AMSGrad.

New York Stock Market Forecasting. This dataset is taken from a prediction task about the New York Stock Exchange of the United States on Kaggle. The dataset covers daily opening and closing prices and price adjustment information from 2010 - 2016. First, data are preprocessed in the same way as that in the previous experiment. The proportion of data used as training set and testing set is 9 to 1. Then the training set is divided into training data and validation data as 3:1. A two-layer LSTM network with 256 memory cells is used in the model. The unit number of hidden layer is 128. There is a single output unit in the output layer. L2 regularization is used to handle the problem of overfitting. The loss function of the model training is MSE. Adam, AMSGrad and AdaHMG optimization algorithms are compared. Batch_size is 512 and the number of epochs is 30. Finally, the model corresponding to each optimization algorithm is scored on the testing sets, as shown in Figure 3 (a)-(c).

RMSE of AdaHMG algorithm is improved by 44.56% compared with the Adam and AMSGrad optimization algorithms. The effect curve of 175-day prediction is also closer to the true value. The loss function curve on the training set and validation set is also significantly lower than that of the Adam and AMSGrad algorithms. Note that, there are barely fluctuations on the loss function curve through AdaHMG algorithm on the training set, whereas there are some fluctuations on the loss function curves through Adam and AMSGrad algorithms.

Passenger Airfare Reservation Prediction. The fourth experiment is a prediction task of international airline passengers from Kaggle. The data is preprocessed at first. Then 33% of the data are used as training sets and the rest of data as testing sets (or validation set). The model includes a simple LSTM model with 100 memory cells, fully connected hidden layers, and L2 regularization added to the output layer. The loss function for model training is MSE, the number of batch_size is 2 and the number of epochs is 30. The three compared algorithms are Adam, AMSGrad and AdaHMG. The model corresponding to each optimization algorithm is scored on the testing set, as shown in Figure 4 (a)-(b).

RMSE of AdaHMG algorithm (97.261) is increased by 45.20% compared with that of Adam optimization algorithm (RMSE = 180.472). The loss function curve on the training set and validation set is also significantly lower than that of the Adam and AMSGrad algorithms.

B. CLASSIFICATION TASK EXPERIMENTS

Handwritten digit recognition task. Convolution neural networks are used to process classical MNIST datasets [29]. A total of 60,000 records in the training set are transformed into 60,000 images of $28 \times 28 \times 1$ pixels (length, width and monochrome), respectively. Thus, a 4D matrix

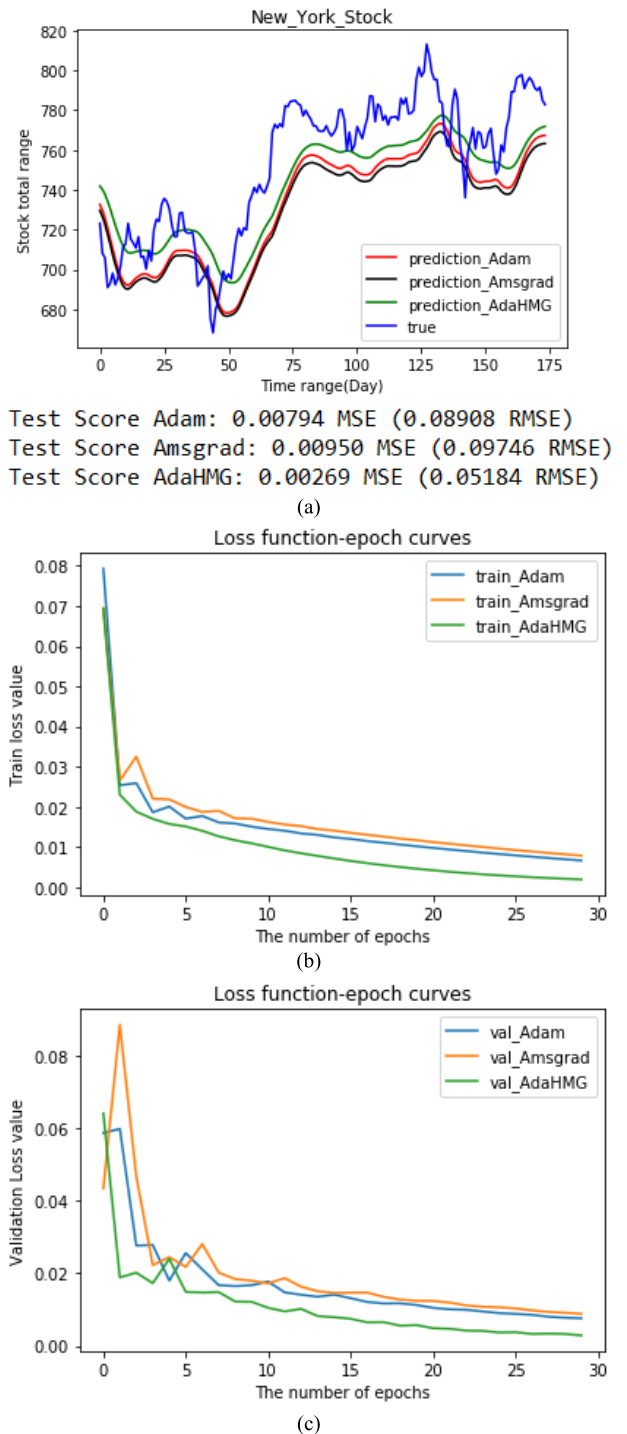


FIGURE 3. (a) Test Set Three Algorithms MSE and RMSE Values. (b) Training loss function effect. (c) Validation loss function effect.

of $60,000 \times 28 \times 28 \times 1$ is generated and then the pixel values are standardized. The dataset is divided into a training set and a validation set as 4:1, and the testing set is an additional 10,000 pieces of data. Finally, the labels of the training and test dataset are one hot encoded. The model includes two layers of convolution and pooling. The hidden layer is fully connected with 128 neurons, and the

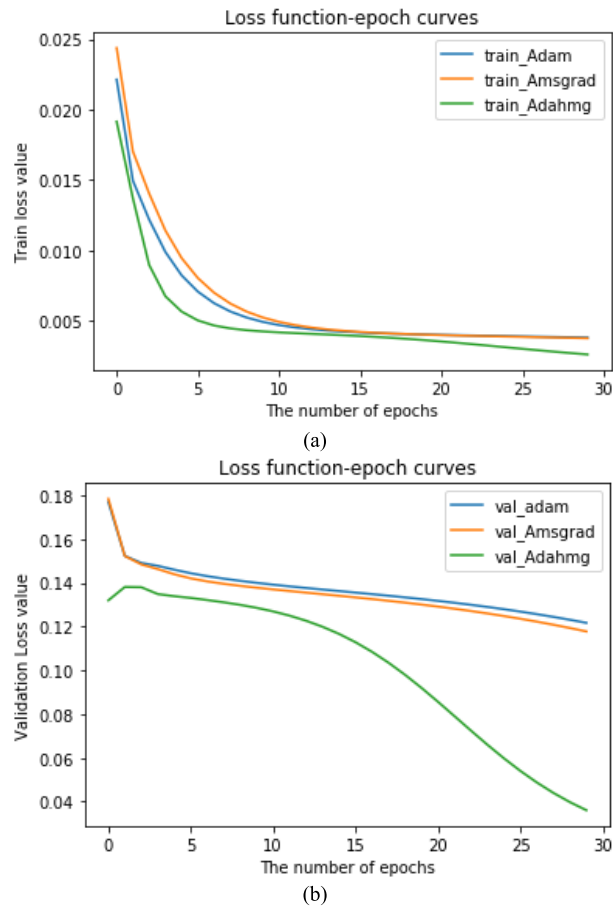


FIGURE 4. (a) Training loss function effect. (b) Validation loss function effect.

dropout function is added to avoid the overfitting. There are 10 neurons in the output layer. As an activation function, Softmax can convert the output value into the prediction probability of each number. The result is shown in Figure 5 (a)-(d).

The results show that the initial iteration of the proposed method is more accurate than the Adam and AMSGrad algorithms on the training set and validation set.

The overall accuracy is slightly higher than that of Adam and AMSGrad algorithms, namely a higher accuracy value is obtained. The proposed AdaHMG algorithm has an average accuracy of 0.9921 for ten repetitions.). The loss function is slightly lower than that of algorithms on the training-validation set.

The test results show that AdaHMG algorithm is better than the Adam and AMSGrad algorithms for time series prediction task, and the performance on classification task is improved slightly. The loss function curve shows that the convergence can be achieved with less epochs than that of Adam and AMSGrad algorithms. The experimental results show that AdaHMG algorithm accelerates convergence and improves the performance of classification and regression tasks. It is worth to explore performance in other classification tasks. Experiments are conducted for other

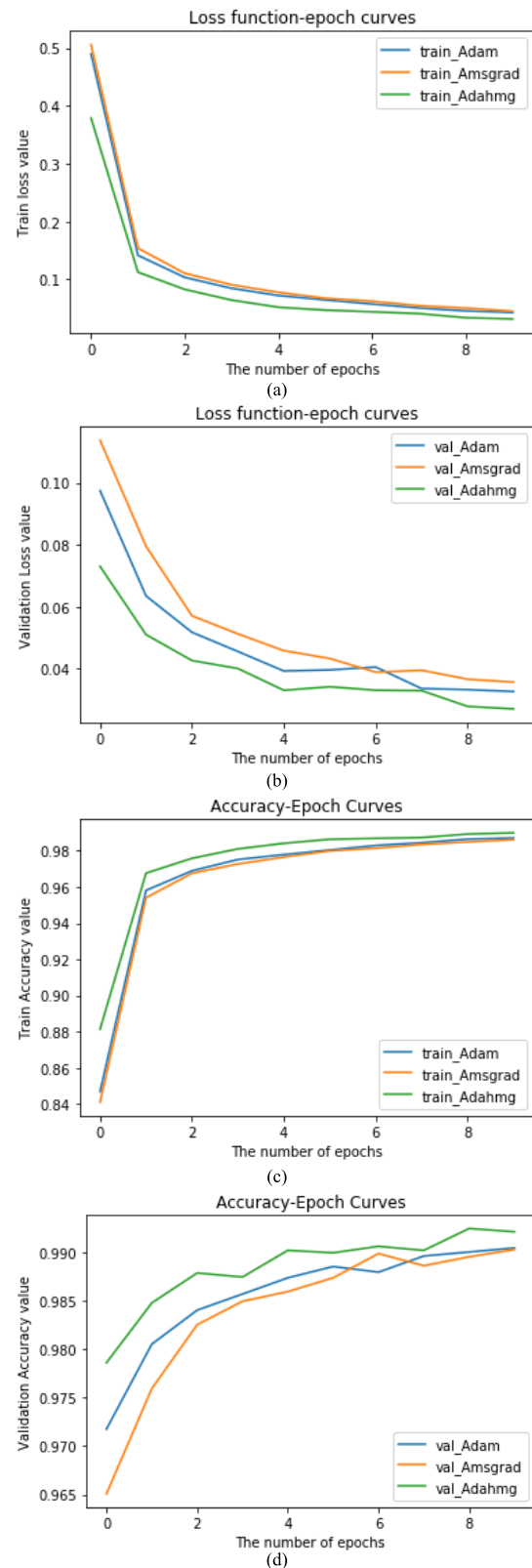


FIGURE 5. (a) Training loss function effect. (b) Validation loss function effect. (c) Training accuracy effect. (d) Validation accuracy effect.

classification tasks. However, the performance improvements are not as prominent as time-series tasks. For example, the accurate value of AdaHMG algorithm is only 0.01 higher

TABLE 1. Performance comparison of three algorithms on different datasets.

dataset	algorithm	RMSE
Spring Water Sales	Adam	6.842
	AMSGrad	6.906
	AdaHMG	5.507
PM2.5 Air Quality	Adam	7.181
	AMSGrad	7.336
	AdaHMG	5.492
New York Stock Market	Adam	0.08908
	AMSGrad	0.09746
	AdaHMG	0.05184
Passenger Airfare Reservation	Adam	180.472
	AMSGrad	177.495
	AdaHMG	97.261
Handwritten digit recognition	Adam	0.9912
	AMSGrad	0.9914
	AdaHMG	0.9921

TABLE 2. Performance ranking table for algorithm comparison.

Dataset	AdaHMG	Adam	AMSGrad
Spring Water Sales	1	2	3
PM2.5 Air Quality	1	2	3
New York Stock Market	1	2	3
Passenger Airfare Reservation	1	3	2
Handwritten digit recognition	1	3	2
Average ranking	1	2.4	2.6

than Adam's in IMDB Review's Affective classification task. This is probably caused by the weak correlation between the data involved in these classification tasks.

The performances of three algorithms on five different datasets are listed in Table 1. The bold numbers in the table represent the best performing results of three algorithms.

C. STATISTICAL TESTING

Typically, the performance of multiple algorithms over multiple datasets need to be compared. A Friedman test of statistical testing is used based on the algorithm sorting. Specific algorithms are sorted in Table 2.

Then, the Friedman test is used to determine whether all three algorithms have the same performance. If the same, their average ranking value should be the same. From the average ranking in Table 2, three algorithms obviously have different performances. If k algorithms are compared on N datasets, r_i represents the average ranking value of the

algorithm i . Friedman test variables are obtained as follows:

$$\begin{aligned}\tau_{\chi^2} &= \frac{k-1}{k} \cdot \frac{12N}{k^2-1} \sum_{i=1}^k \left(r_i - \frac{k+1}{2} \right)^2 \\ &= \frac{12N}{k(k+1)} \left(\sum_{i=1}^k r_i - \frac{k(k+1)^2}{4} \right)\end{aligned}\quad (6)$$

The data is substituted into Eq. 6, and Friedman test variables is obtained as 7.6.

However, the original Friedman test is too conservative and now variables are used:

$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1) - \tau_{\chi^2}} \quad (7)$$

The value of bringing our data into Eq. 7 is 12.67. The threshold of F test is 4.459 when $\alpha = 0.05$, $N = 5$ and $k = 3$. τ_F is larger than a = 0.05 Friedman test threshold of 4.459, so the assumption that all algorithms perform the same is rejected. Then post-hoc tests should be carried out to further distinguish the performance of each algorithm. Nemenyi post-hoc test is the commonly used method. The threshold range for calculating the difference of average ranking value is:

$$CD = q_{\alpha} \sqrt{\frac{k(k+1)}{6N}} \quad (8)$$

where q_{α} is the threshold of the Tukey distribution. When $\alpha = 0.1$ and $k = 3$, q_{α} is 2.052. These parameters are taken into Eq. 8 and a $CD_{\alpha=0.1}$ value of 1.296 is obtained. When $\alpha = 0.05$ and $k = 3$, $q_{\alpha} = 2.344$. These parameters are taken into Eq. 8 and a $CD_{\alpha=0.05}$ value of 1.481 is obtained. If the difference of the average ranking of the two algorithms exceeds the value of CD, the assumption that the performance of the two algorithms is the same is rejected with the corresponding confidence. It is obvious that the difference between r_{AdaHMG} and r_{Adam} is 1.6, r_{AdaHMG} and $r_{AMSGrad}$ is 1.4. Their differences are greater than those of CD. Finally, the performance of AdaHMG is significantly different from that of Adam with the confidence of $1 - \alpha$ equals 0.9. Since the difference between the average ranking of the AdaHMG and AMSGrad algorithm is larger than that of $CD_{\alpha=0.05}$, the performance of the two algorithms is significantly different with the confidence of 0.95. At this time, from the statistical test, it is proved that the proposed algorithm is superior to the other two algorithms in five datasets.

VI. DISCUSSION

The core idea of Adam's algorithm is the exponential moving averages method and correcting the bias introduced by v_t . Adam has the excellent optimization performance in a wide range of machine learning fields. However, Adam's algorithm converges to suboptimal solutions rather than optimal solutions in some conditions, e.g. machine translation [30] and object recognition [9] tasks.

Researchers [19] found the reason why Adam's algorithm converges to suboptimal solutions in small batch learning tasks. Adam's exponential moving average method may reduce the impact of the vast gradient information provided by the small batch samples in these tasks. To overcome this convergence problem, the researchers proposed a new solution, AMSGrad. The major improvement is that Max function is used as the update rule for \hat{v}_t . Such update rule can get a non-incremental step size, and maintain "long-term memory" of the squared gradient information, thus the convergence of the algorithm is guaranteed. However, AMSGrad has the following shortcomings. First, the jointly effect of the current (v_t) and the historical moment (\hat{v}_{t-1}) on updating \hat{v}_t is not considered. The Max function simply selects the largest from \hat{v}_{t-1} and v_t , which only satisfy the convergence requirement theoretically. Second, the AMSGrad algorithm without the error correction of moment estimation can produce the accumulated error when learning small batches of samples. This problem may lead to the lack of significant improvement in its empirical performance in a wide range of fields. Previous experiments found that AMSGrad has the similar or even weaker performance than Adam [31]. The experimental results in the present study also showed that the performance of AMSGrad algorithm is unsatisfactory.

AdaHMG algorithm is greatly improved where the Max function of AMSGrad algorithm is replaced by a new hybrid high-power and multi-dimensional updating rule. The major advantages of AdaHMG algorithm are the representation of the temporal correlation of data and the good convergence performance. First, the correlation between the data is calculated by the correlation coefficient (e.g., $C = \frac{\text{cov}(X,Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}}$, C is the correlation coefficient, Cov is the covariance, Var is the variance) in AdaHMG algorithm. The update rule of AdaHMG algorithm represents the characteristics of the strong temporal correlation of the data by mixing the current and historical squared gradient information, so that the long-term memory of the gradient information is ensured. Second, the convergence analysis in Section 4 proves that AdaHMG algorithm has a better convergence feature than AMSGrad.

The Max function in AMSGrad (Refer to (9)) is compared with the proposed equation (Refer to (10)).

$$\hat{v}_t = \begin{cases} \hat{v}_{t-1} & \text{if } \hat{v}_{t-1} > v_t \\ v_t & \text{if } \hat{v}_{t-1} < v_t \end{cases} \quad (9)$$

$$\hat{v}_t = k_1 \hat{v}_{t-1}^2 + k_2 v_t \quad (10)$$

The Max function has a simple change rate, and the proposed updating formula has a quadratic change rate. Therefore, AdaHMG algorithm has better convergence performance than AMSGrad. Moreover, the quadratic change rate is the best choice from the following theoretical derivations.

In the convergence analysis, weights and inequalities are used to scale the derivation of formulas. The algorithm converges only when the inequality for weights summation is true. The power of the numerator is required to be greater than the power of denominator, i.e., $n > m$ is satisfied in the

following formula: $\frac{(\sum_{i=1}^n a_i)^n}{(\sum_{i=1}^n b_i)^m} \leq \sum_{i=1}^n \frac{a_i^n}{b_i^m}$. If the number of high power terms is too high, the inequality for the weight summation cannot be true, thus the convergence cannot be proved. For example, when the high power terms are cubic, the intermediate result is obtained as follows.

$$\begin{aligned} \sum_{t=1}^T \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 &\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 \\ &+ \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \\ &\times \sum_{i=1}^d \sum_{j=1}^{T-1} \frac{\beta_1^{T-1-j} g_{j,i}^2}{\left(\beta_2^{T-1-j} g_{j,i}^2 \right)^1} \\ &+ \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \\ &\times \sum_{i=1}^d \frac{g_{T,i}^2}{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^1} \end{aligned} \quad (11)$$

The powers of $g_{j,i}^2$ in the numerator and denominator are equal, thus the inequality for the weight summation cannot be true. Similarly, when the higher order term is uneven power, the squared gradient information represented by $g_{j,i}^2$ will be counteracted. Thus, more gradient information will be lost and the algorithm will not be able to capture enough historical and current gradient information, which will badly limit the performance of the algorithm. Therefore, the quadratic term in the proposed update rule is the best choice. The detailed analysis is given in the appendix.

In this experiments, the shape of loss function curve on the training set show that AdaHMG algorithm has better convergence than AMSGrad and Adam. As shown in Figure 1 (b), the loss function curve of AdaHMG algorithm, decreases faster than that of Adam and AMSGrad algorithms in the first 20 epochs and it basically converges when around epoch 40. The final loss function value of AdaHMG is the lowest of the three. These results show that AdaHMG algorithm converges faster, indicating that the update rules for \hat{v}_t in AdaHMG algorithm are the key to significantly improve the performance of the algorithm.

In the practical application, it is critical to predicting the future 24 hours PM2.5 index accurately by using the weather index data and PM2.5 index in the past 48 hours. For haze weather without wind and rain (PM2.5 is higher), to predict the future highest PM2.5 index in the next 24 hours, AdaHMG algorithm trains a more accurate prediction model by mixing square gradient information. It helps environmentalists to prepare for dust reduction in advance. Instead, the real value of PM2.5 is 46 at some point in the future, and the predicted value of AdaHMG algorithm is 48, while predicted values of other algorithms are above 50. Based on the analysis of good weather PM2.5 below 50, AdaHMG algorithm is more accurate for the future. The city cleaning

department will not need to spend additional financial resources to reduce PM2.5 concentration. Figure. 2 (a) shows the predictive effect of AdaHMG algorithm.

Stock trend forecast is more intuitive. The AdaHMG algorithm gets an average of 5-15 points higher than the other two algorithms, closer to the true value in Figure. 3 (a). AdaHMG algorithm helps investors make more accurate decisions. When the stock index goes up after the next 50 days, the proposed approach forecasts about 20 points higher than the others. If investors buy at this moment and use AdaHMG algorithm to observe more significant gains in the next few days, it is very beneficial to increase investment.

In conclusion, the core innovation of AdaHMG algorithm design, i.e., the updating rules of hybrid high power multidimensional with mixed historical and current squared gradient information, can effectively capture the temporal correlation of data. Thus, it has better performance in time series prediction tasks. Experimental results of the present study strongly support this theoretical idea. Both the loss function curve and the evaluation index show that AdaHMG algorithm has better performance in accuracy, RMSE, and convergence.

VII. CONCLUSION

In the present study, AdaHMG, an optimization algorithm is proposed based on Adam, including the updating rules of mixed high power multidimensional squared gradients. The convergence analysis is conducted with the proof of the basic theory. The performance of AdaHMG is greatly improved in time series prediction tasks compared with the most advanced algorithms such as Adam and AMSGrad. The curve of loss function show that AdaHMG algorithm converges faster and performs better in the early training period. Moreover, compared with the previous work, the updating rules of mixed high power multidimensional squared gradients may further reveal some of the essence of the adaptive optimization algorithm.

APPENDIX

A. PROOF OF THEOREM 4.1

The proof of Theorem 4.1 presented below is along the lines of the [19, Th. 4.1]. We provide a proof of convergence for AdaHMG.

Proof: Some of the conclusions we know come from previous studies:

$$\begin{aligned} x_{t+1} &= x_t - \alpha_t \hat{v}_t^{-\frac{1}{2}} m_t \\ &= \min_{x \in \mathcal{X}} \left\| \hat{v}_t^{\frac{1}{4}} \left(x - \left(x_t - \alpha_t \hat{v}_t^{-\frac{1}{2}} m_t \right) \right) \right\| \end{aligned} \quad (12)$$

$$\begin{aligned} &\left\| \hat{v}_t^{\frac{1}{4}} (x_{t+1} - x^*) \right\|^2 \\ &\leq \left\| \hat{v}_t^{\frac{1}{4}} (x_t - x^*) \right\|^2 + \alpha_t^2 \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 \\ &\quad - 2\alpha_t \langle m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, x_t - x^* \rangle \end{aligned} \quad (13)$$

$$\begin{aligned} \sum_{t=1}^T f_t(x_t) - f_t(x^*) &\leq \sum_{t=1}^T \langle g_t, x_t - x^* \rangle \\ &\leq \sum_{t=1}^T \left[\frac{1}{2\alpha_t(1 - \beta_{1t})} \left[\left\| \hat{v}_t^{\frac{1}{4}} (x_t - x^*) \right\|^2 \right. \right. \\ &\quad \left. \left. - \left\| \hat{v}_t^{\frac{1}{4}} (x_{t+1} - x^*) \right\|^2 \right] + \frac{\alpha_t}{2(1 - \beta_{1t})} \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 \right. \\ &\quad \left. + \frac{\beta_{1t}}{2(1 - \beta_{1t})} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_{t-1} \right\|^2 \right. \\ &\quad \left. + \frac{\beta_{1t}}{2\alpha_t(1 - \beta_{1t})} \left\| \hat{v}_t^{\frac{1}{4}} (x_t - x^*) \right\|^2 \right] \end{aligned} \quad (14)$$

For further bounding this inequality, we need the following intermediate result.

Lemma 1: According to the parameter definition and hypothetical conditions of Theorem 4.1, we need to proof the following intermediate conclusion:

$$\begin{aligned} \sum_{t=1}^T \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 \\ \leq \frac{\alpha \sqrt{1 + \log T}}{\sqrt{\frac{4k_1 k_2}{1 - \beta_2}} (1 - \beta_1) \sqrt{(1 - \beta_2)^2 (1 - \gamma)}} \sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}} \end{aligned} \quad (15)$$

Proof: We start with the following (16), as shown at the top of the next page. The first inequality comes from the definition of our update strategy. The first part of inequality $\hat{v}_{T,i}$ is deduced from the Convergence Analysis of the paper. The second inequality comes from the update strategy for m_t and v_t in Algorithm 1. We need to further scale out the inequalities above (17)–(23), as shown at the top of the next page.

The first inequality comes from Cauchy-Schwarz inequality. The second inequality comes from the fact that $\beta_{1k} \leq \beta_1$ for all $k \in [T]$. The third inequality comes from the inequality $\sum_{j=1}^T \beta_1^{T-j} \leq \frac{1}{1 - \beta_1}$. The fifth inequality comes from the deformation of the mean inequality $\sqrt{a + b} \geq \sqrt{2\sqrt{ab}}$. The sixth inequality is the inequality for the weight summation (In fact, it equal to Holder inequality): $\frac{(\sum_{i=1}^n a_i)^n}{(\sum_{i=1}^n b_i)^m} \leq \sum_{i=1}^n \frac{a_i^n}{b_i^m}$ and $n > m$.

$$\begin{aligned} &\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 \\ &\quad + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1 - \beta_2}} (1 - \beta_1) \sqrt{T(1 - \beta_2)^2}} \\ &\quad \times \sum_{i=1}^d \sum_{j=1}^{T-1} \gamma^{T-1-j} (g_{j,i})^{\frac{1}{2}} \\ &\quad + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1 - \beta_2}} (1 - \beta_1) \sqrt{T(1 - \beta_2)^2}} \sum_{i=1}^d \\ &\quad \times \frac{g_{T,i}^2}{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^{\frac{3}{4}}} \end{aligned} \quad (24)$$

$$\begin{aligned}
& \sum_{t=1}^T \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 \\
&= \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha_T \sum_{i=1}^d \frac{m_{T,i}^2}{\sqrt{\hat{v}_{T,i}}} \\
&\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha_T \sum_{i=1}^d \frac{m_{T,i}^2}{\sqrt{k_1 v_{T-1,i}^2 + k_2 v_{T-1,i}}} \\
&\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha \sum_{i=1}^d \frac{\left(\sum_{j=1}^T \prod_{k=1}^{T-j} \beta_{1(T-k+1)} g_{j,i} \right)^2}{\sqrt{T \left(k_1 \left[(1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right]^2 + k_2 (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \quad (16)
\end{aligned}$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha \sum_{i=1}^d \frac{\left(\sum_{j=1}^T \prod_{k=1}^{T-j} \beta_{1(T-k+1)} \right) \left(\sum_{j=1}^T \prod_{k=1}^{T-j} \beta_{1(T-k+1)} g_{j,i}^2 \right)}{\sqrt{T \left(k_1 (1-\beta_2)^2 \left[\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right]^2 + k_2 (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \quad (17)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha \sum_{i=1}^d \frac{\left(\sum_{j=1}^T \beta_1^{T-j} \right) \left(\sum_{j=1}^T \beta_1^{T-j} g_{j,i}^2 \right)}{\sqrt{T \left(k_1 (1-\beta_2)^2 \left[\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right]^2 + k_2 (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \quad (18)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{1-\beta_1} \sum_{i=1}^d \frac{\sum_{j=1}^T \beta_1^{T-j} g_{j,i}^2}{\sqrt{T \left(k_1 (1-\beta_2)^2 \left[\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right]^2 + k_2 (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \quad (19)$$

$$\begin{aligned}
&\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{(1-\beta_1) \sqrt{T (1-\beta_2)^2}} \sum_{i=1}^d \left(\frac{\sum_{j=1}^{T-1} \beta_1^{T-1-j} g_{j,i}^2}{\sqrt{k_1 \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^2 + \frac{k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \right. \\
&\quad \left. + \frac{\beta_1^0 g_{T,i}^2}{\sqrt{k_1 \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^2 + \frac{k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \right) \quad (20)
\end{aligned}$$

$$\begin{aligned}
&< \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{(1-\beta_1) \sqrt{T (1-\beta_2)^2}} \sum_{i=1}^d \left(\frac{\sum_{j=1}^{T-1} \beta_1^{T-1-j} g_{j,i}^2}{\sqrt{2 \sqrt{k_1 \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^2 \times \frac{k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}}} \right. \\
&\quad \left. + \frac{\beta_1^0 g_{T,i}^2}{\sqrt{2 \sqrt{k_1 \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^2 \times \frac{k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}}} \right) \\
&= \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{(1-\beta_1) \sqrt{T (1-\beta_2)^2}} \sum_{i=1}^d \left(\frac{\sum_{j=1}^{T-1} \beta_1^{T-1-j} g_{j,i}^2}{\sqrt[4]{\frac{4k_1 k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^3}} + \frac{\beta_1^0 g_{T,i}^2}{\sqrt[4]{\frac{4k_1 k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^3}} \right) \quad (21)
\end{aligned}$$

$$= \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d \left(\frac{\sum_{j=1}^{T-1} \beta_1^{T-1-j} g_{j,i}^2}{\sqrt{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^3}} + \frac{\beta_1^0 g_{T,i}^2}{\sqrt{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^3}} \right) \quad (22)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d \sum_{j=1}^{T-1} \frac{\beta_1^{T-1-j} g_{j,i}^2}{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^{\frac{3}{4}}} + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \times \sum_{i=1}^d \frac{g_{T,i}^2}{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^{\frac{3}{4}}} \quad (23)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \times \sum_{i=1}^d \sum_{j=1}^{T-1} \gamma^{T-1-j} (g_{j,i})^{\frac{1}{2}} + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d g_{T,i}^{\frac{1}{2}} \quad (25)$$

Consider that there is one more time step left. Let's scale it again. Let's get it into the all-time step. The second inequality comes from $\|g_{T,i}^2, g_{T,i}^2, \dots, g_{T,i}^2\|_2 = g_{T,i}^2 (T-1) \geq g_{T,i}^2$. By using similar upper bounds for all time steps, the quantity in (25) can further be bounded as follows:

$$\begin{aligned} & \sum_{t=1}^T \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 \\ & \leq \sum_{t=1}^T \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{t(1-\beta_2)^2}} \\ & \quad \times \sum_{i=1}^d \sum_{j=1}^t \gamma^{t-j} (g_{j,i})^{\frac{1}{2}} \\ & = \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{(1-\beta_2)^2}} \\ & \quad \times \sum_{i=1}^d \sum_{t=1}^T \frac{1}{\sqrt{t}} \sum_{j=1}^t \gamma^{t-j} (g_{j,i})^{\frac{1}{2}} \\ & = \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{(1-\beta_2)^2}} \\ & \quad \times \sum_{i=1}^d \sum_{t=1}^T (g_{j,i})^{\frac{1}{2}} \sum_{j=t}^T \frac{\gamma^{j-t}}{\sqrt{j}} \\ & \leq \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{(1-\beta_2)^2}} \\ & \quad \times \sum_{i=1}^d \sum_{t=1}^T (g_{j,i})^{\frac{1}{2}} \sum_{j=t}^T \frac{\gamma^{j-t}}{\sqrt{j}} \end{aligned}$$

$$\begin{aligned} & \leq \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{(1-\beta_2)^2}} \\ & \quad \times \sum_{i=1}^d \sum_{t=1}^T (g_{j,i})^{\frac{1}{2}} \sum_{j=t}^T \frac{1}{(1-\gamma) \sqrt{j}} \\ & \leq \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{(1-\beta_2)^2} (1-\gamma)} \\ & \quad \times \sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}} \sqrt{\sum_{t=1}^T \frac{1}{t}} \\ & \leq \frac{\alpha \sqrt{1 + \log T}}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{(1-\beta_2)^2} (1-\gamma)} \\ & \quad \times \sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}} \quad (26) \end{aligned}$$

The third inequality comes from the fact that $\sum_{j=t}^T \gamma^{j-t} \leq \frac{1}{1-\gamma}$. The fourth inequality comes from Cauchy-Schwarz inequality. The fifth inequality is due to the following bound on harmonic number to sum: $\sum_{t=1}^T \frac{1}{t} \leq 1 + \log T$. Lemma 1 completes the proof.

We can return to the proof of Theorem 4, Using some of the derivations in this article to draw conclusions [19]:

$$\begin{aligned} & \sum_{t=1}^T f_t(x_t) - f_t(x^*) \\ & \leq \frac{1}{2\alpha_1(1-\beta_1)} \sum_{i=1}^d \hat{v}_{1,i}^{\frac{1}{2}} (x_{1,i} - x_i^*)^2 + \frac{1}{2(1-\beta_1)} \sum_{t=2}^T \\ & \quad \times \sum_{i=1}^d (x_{t,i} - x_i^*)^2 \left[\frac{\hat{v}_{t,i}^{\frac{1}{2}}}{\alpha_t} - \frac{\hat{v}_{t-1,i}^{\frac{1}{2}}}{\alpha_{t-1}} \right] \\ & \quad + \frac{1}{2(1-\beta_1)} \sum_{t=1}^T \sum_{i=1}^d \frac{\beta_{1t} (x_{t,i} - x_i^*)^2 \hat{v}_{t,i}^{\frac{1}{2}}}{\alpha_t} \\ & \quad + \frac{\alpha \sqrt{1 + \log T}}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1)^2 \sqrt{(1-\beta_2)^2} (1-\gamma)} \sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}} \quad (27) \end{aligned}$$

The first three to the right of the first inequality we agree with the previous study, and the last one is the computed form of our intermediate result. Using the L_∞ bound on the feasible

$$\begin{aligned} & \sum_{t=1}^T \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 \\ &= \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha_T \sum_{i=1}^d \frac{m_{T,i}^2}{\sqrt{\hat{v}_{T,i}}} \end{aligned} \quad (29)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha_T \sum_{i=1}^d \frac{m_{T,i}^2}{\sqrt{k_1 v_{T-1,i}^3 + k_2 v_{T-1,i}}} \quad (30)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha \sum_{i=1}^d \frac{\left(\sum_{j=1}^T \prod_{k=1}^{T-j} \beta_{1(T-k+1)} g_{j,i} \right)^2}{\sqrt{T \left(k_1 \left[(1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right]^3 + k_2 (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \quad (31)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha \sum_{i=1}^d \frac{\left(\sum_{j=1}^T \prod_{k=1}^{T-j} \beta_{1(T-k+1)} \right) \left(\sum_{j=1}^T \prod_{k=1}^{T-j} \beta_{1(T-k+1)} g_{j,i}^2 \right)}{\sqrt{T \left(k_1 (1-\beta_2)^2 \left[\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right]^3 + k_2 (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \quad (32)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \alpha \sum_{i=1}^d \frac{\left(\sum_{j=1}^T \beta_1^{T-j} \right) \left(\sum_{j=1}^T \beta_1^{T-j} g_{j,i}^2 \right)}{\sqrt{T \left(k_1 (1-\beta_2)^2 \left[\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right]^3 + k_2 (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \quad (33)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{1-\beta_1} \sum_{i=1}^d \frac{\sum_{j=1}^T \beta_1^{T-j} g_{j,i}^2}{\sqrt{T \left(k_1 (1-\beta_2)^2 \left[\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right]^3 + k_2 (1-\beta_2) \sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \quad (34)$$

$$\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{(1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d \left(\frac{\sum_{j=1}^{T-1} \beta_1^{T-1-j} g_{j,i}^2}{\sqrt{k_1 \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^3 + \frac{k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} + \frac{\beta_1^0 g_{T,i}^2}{\sqrt{k_1 \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^3 + \frac{k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} \right) \quad (35)$$

$$< \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{(1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d \left(\frac{\sum_{j=1}^{T-1} \beta_1^{T-1-j} g_{j,i}^2}{\sqrt{2\sqrt{k_1 \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^3 \times \frac{k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}} + \frac{\beta_1^0 g_{T,i}^2}{\sqrt{2\sqrt{k_1 \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^3 \times \frac{k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)}}} \right)$$

$$= \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{(1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d \left(\frac{\sum_{j=1}^{T-1} \beta_1^{T-1-j} g_{j,i}^2}{\sqrt[4]{\frac{4k_1 k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^4}} + \frac{\beta_1^0 g_{T,i}^2}{\sqrt[4]{\frac{4k_1 k_2}{1-\beta_2} \left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^4}} \right) \quad (36)$$

$$= \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{\sqrt[4]{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d \left(\frac{\sum_{j=1}^{T-1} \beta_1^{T-1-j} g_{j,i}^2}{\sqrt[4]{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^4}} + \frac{\beta_1^0 g_{T,i}^2}{\sqrt[4]{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^4}} \right) \quad (37)$$

$$\begin{aligned}
&\leq \sum_{t=1}^{T-1} \alpha_t \left\| \hat{v}_t^{-\frac{1}{4}} m_t \right\|^2 + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d \sum_{j=1}^{T-1} \frac{\beta_1^{T-1-j} g_{j,i}^2}{\left(\beta_2^{T-1-j} g_{j,i}^2 \right)^1} \\
&\quad + \frac{\alpha}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1) \sqrt{T(1-\beta_2)^2}} \sum_{i=1}^d \frac{g_{T,i}^2}{\left(\sum_{j=1}^{T-1} \beta_2^{T-1-j} g_{j,i}^2 \right)^1}
\end{aligned} \quad (38)$$

set and making the above property in (27), we can get this conclusion:

$$\begin{aligned}
&\sum_{t=1}^T f_t(x_t) - f_t(x^*) \\
&\leq \frac{D_\infty^2}{2\alpha_T(1-\beta_1)} \sum_{i=1}^d \hat{v}_{T,i}^{\frac{1}{2}} \\
&\quad + \frac{D_\infty^2}{2(1-\beta_1)} \sum_{i=1}^d \sum_{t=1}^T \frac{\beta_{1t} \hat{v}_{t,i}^{\frac{1}{2}}}{\alpha_t} \\
&\quad + \frac{\alpha \sqrt{1+\log T}}{\sqrt{\frac{4k_1 k_2}{1-\beta_2}} (1-\beta_1)^2 \sqrt{(1-\beta_2)^2(1-\gamma)}} \sum_{i=1}^d \|g_{1:T,i}\|_2^{\frac{1}{2}}
\end{aligned} \quad (28)$$

Equation (28) is the final result we need about regret.

A. SELECTION OF HIGH-POWER ITEMS

Our proposed algorithm has a strict limit on the number of times for high power term in the rule. Here we will show why the quadratic item is chosen instead of the other in the update rule. Why don't we choose terms that are higher than the quadratic power? Such as \hat{v}_{t-1}^3 . Similar to the proof in lemma 1, the following is the process by which we derived the update rule into $\hat{v}_t = k_1 \hat{v}_{t-1}^3 + k_2 v_t$, (29)–(38), as shown at the top of the previous page.

The last inequality in equation (37) shows that the equal number of numerators and denominators of $g_{j,i}^2$ will be simplified, resulting in conditions that do not satisfy the inequality for the weight summation and make the update rules independent of the gradient information. It is an error to capture the historical square gradient information in \hat{v}_{t-1}^3 as an update rule. It is implying that \hat{v}_{t-1}^2 in the update rule is the correct and better choice.

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
- [2] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum, "Optimizing semantic coherence in topic models," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2010, pp. 262–272.
- [3] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," in *Proc. 33rd Int. Conf. Learn. Represent.*, 2016, pp. 1–11.
- [4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1171–1179.
- [5] Y. Li, N. Du, and S. Bengio. (2017). "Time-dependent representation for neural event sequence prediction." [Online]. Available: <https://arxiv.org/abs/1708.00065>
- [6] J. Ba and D. Kingma, "Adam: A method for stochastic optimization," in *Proc. 32nd Int. Conf. Learn. Represent.*, 2015, pp. 127–142.
- [7] T. Tieleman and G. Hinton, "Lecture 6.5-RMSPROP: Divide the gradient by a running average of its recent magnitude," *COURSERA, Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.
- [8] M. D. Zeiler. (2012). "ADDELTA: An adaptive learning rate method." [Online]. Available: <https://arxiv.org/abs/1212.5701>
- [9] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
- [10] S. Ghadimi and G. H. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2341–2368, 2013.
- [11] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming," *Math. Program.*, vol. 156, no. 1, pp. 59–99, 2016.
- [12] S. J. Reddi, A. Hefny, S. Sra, B. Póczos, and A. Smola. (2016). "Stochastic variance reduction for nonconvex optimization." [Online]. Available: <https://arxiv.org/abs/1603.06160>
- [13] A. Alacaoglu, Q. T. Dinh, O. Fercoq, and V. Cevher, "Smooth primal-dual coordinate descent algorithms for nonsmooth convex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5854–5863.
- [14] L. Lei, C. Ju, J. Chen, and M. I. Jordan, "Non-convex finite-sum optimization via SCSSG methods," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2345–2355.
- [15] R. Ge and T. Ma, "On the optimization landscape of tensor decompositions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3656–3666.
- [16] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 257–269, Jul. 2011.
- [17] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.
- [18] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. Workshop ICLR*, 2016, pp. 2013–2016.
- [19] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proc. 35th Int. Conf. Learn. Represent.*, 2018, pp. 1101–1115.
- [20] J. Wissam Baddar and Y. Man, "Learning spatio-temporal features with partial expression sequences for on-the-fly prediction," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 6666–6673.
- [21] D. H. Kim, W. Baddar, J. Jang, and Y. M. Ro, "Multi-objective based spatio-temporal feature representation learning robust to expression intensity variations for facial expression recognition," *IEEE Trans. Affect. Comput.*, to be published.
- [22] J. Naito, Y. Baba, H. Kashima, T. Takaki, and T. Funo, "Predictive modeling of learning continuation in preschool education using temporal patterns of development tests," in *Proc. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 7934–7940.
- [23] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 12th Int. Conf. Mach. Learn.*, 2003, pp. 928–936.
- [24] M. Abadi. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: <https://www.tensorflow.org/>
- [25] F. Chollet. (2015). *Keras*. [Online]. Available: <https://github.com/fchollet/keras>
- [26] X. Liang et al., "Assessing Beijing's PM_{2.5} pollution: Severity, weather impact, APEC and winter heating," *Proc. Roy. Soc. A, Math. Phys. Eng. Sci.*, vol. 471, no. 2182, Feb. 2015, Art. no. 20150257.
- [27] RakanNimer. (2017). *Flight Booking Prediction*. [Online]. Available: <https://www.kaggle.com/rakanNimer/air-passengers>
- [28] Dominik Gawlik. (2017). *New York Stock Prediction*. [Online]. Available: <https://www.kaggle.com/dgawlik/nyse>

- [29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [30] M. Johnson *et al.* (2016). "Google's multilingual neural machine translation system: Enabling zero-shot translation." [Online]. Available: <https://arxiv.org/abs/1611.04558>
- [31] S. Ruder. (2016). "An overview of gradient descent optimization algorithms." [Online]. Available: <https://arxiv.org/abs/1609.04747>
- [32] N. Zhang, S. Ding, J. Zhang, and Y. Xue, "Research on point-wise gated deep networks," *Appl. Soft Comput.*, vol. 52, pp. 1210–1221, Mar. 2017.
- [33] N. Zhang and S. Ding, "Unsupervised and semi-supervised extreme learning machine with wavelet kernel for high dimensional data," *Memetic Comput.*, vol. 9, no. 2, pp. 129–139, May 2017.
- [34] N. Zhang, S. Ding, J. Zhang, and Y. Xue, "An overview on restricted boltzmann machines," *Neurocomputing*, vol. 275, pp. 1186–1199, Jan. 2018.



WENDONG ZHENG was born in Taiyuan, China, in 1994. He received the B.S. degree in software engineering from the North University of China, in 2017. He is currently pursuing the degree in computer science with Hunan University. His main research interests include multi-agent systems and deep learning.

• • •



JUN HU was born in 1971. He received the M.Sc. degree in computer application from the Kunming University of Science and Technology, Kunming, China, and the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China. In 2010, he was an Academic Visitor with the University of Southampton working on multi-agent system. He is currently an Associate Professor with Hunan University, Changsha, China. His research interests include multi-agent systems, distributed artificial intelligence, and software engineering. He is a Senior Member of China Computer Federation.