MLP Coursework 1: Learning Algorithms and Regularization

s1882930

Abstract

Nowadays, many studies have given the evidence that generalization ability of the neural network relay on a balance between the information we can get from the training set and the complexity of the model we use. Apparently, poor generalization ability usually happens when the information got from training set do not match the complexity of the model. Under this circumstance, over-fitting will occur along with some negative results.In order to restrict the rate of the weights, some kinds of methods are used in different fields of machine learning. Here we will explore the performance of the SGD, RMSProp(Tieleman & Hinton (2012)) and Adam(Kingma & Ba (2014)) algorithms.

1. Introduction

The aim of this experiment is to explore the RMSProp and Adam learning algorithms in the context of L2 regularization. The motivation for the coursework comes from the recent paper by Loshchilov & Hutter (2017) on using the Adam optimizer with L2 regularization and weight decay. The coursework uses an extended version of the MNIST database, the EMNIST Balanced data set. Each data contains 28*28 pixel value. The whole data set is separated to training set(100000), validation set(15800), and test set(15800). Section two illustrates the process to find the hyper-parameters using SGD algorithm without regularization. Section three compares the behaviour of RMSProp and Adam with SGD, and apply the best model structure to the test set. In Section four, cosine annealing scheduler algorithm is added into the SGD and Adam to change the learning rate. Comparing the different algorithms with different methods to set the learning rate and different start. The section five trains data with Adam and compares the L2 regularization with weight decay, constant learning rate with cosine annealing schedule, and no restarts in the scheduler with using a warm restart.

2. Baseline systems

Different models should use different hyper-parameters. Firstly, some learning algorithms can be seen as a combination of a training criterion and a model. One should distinguish hyper-parameters for the model. So, in this work, we initially explore the baseline system on EMNIST using SGD without any regularization. Learning rate and the number of hidden layer of the network are the hyper-parameters in the first section. We separate the whole data set into three parts and use the validation set to choose a better structure.

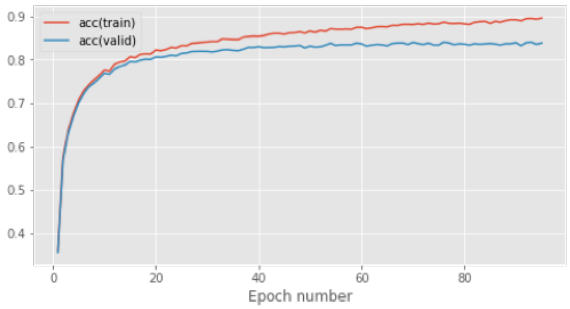


Figure 1. Accuracy Curve of SGD

LAYER	0.01	0.001	0.0001
2	83.2%	80.6%	63.6%
3	84.1%	81.8%	63.4%
4	83.9%	83.1%	63.6%
5	83.8%	82.6%	63.9%

Table 1. Classification accuracies for different baseline on validation sets.

We set learning rate at 0.01, 0.001, 0.0001 and use them in the baseline with 100 ReLu hidden units per layer. In order to compare the influence of the number of hidden layer on the network and choose the best baseline, 300 epochs are used initially along with early-stop, which is a efficient way to avoid over-fitting added in the train process to finish the task with less energy. For each computation, the baseline with different learning rate and various layers to do find the best model. It implemented 12 experiments using the cross-entropy-soft-max-error as the cost function to calculate the gradient and evaluate the model with validation set. The experiment results are listed in table 1.

The table shows the highest accuracy outcome on validation set with different model structures and learning rates. It is indicated that the network with 0.01 learning rate setting and 3 hidden layers performs best, reaching the accuracy value at 82.89%. Therefore, it is expected to use this model as the baseline in the following experiments. figure 1gives the accuracy curve about best SGD model. The early-stop also appears.

3. Learning algorithms – RMSProp and Adam

The biggest disadvantage of the random gradient drop is that each update may not be in the correct direction, so it can bring optimization fluctuations. Due to fluctuations, it will increase the number of iterations, that is to say, the convergence speed is slower.

RMSProp and other methods manage to solve this by automatically adjusting the step size. In this circumstance, the step is on the same scale as the gradients, that is to say, as the average gradient gets smaller, the coefficient in the SGD update gets bigger to compensate. Combining the idea of only using the the gradient with the idea of the step size separately for each weight, RMSProp was proposed. The idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. The specific algorithm 1 is shown the process. Adam is

Algorithm 1 RMS

Input: step count$t$,initial learning α,decay rate$beta$, sum square gradientsum_sq_grad

repeat

$g_t = \nabla_\theta f_t(\theta_{t-1})$

$sum\_sq\_grad_t = \beta_1 \cdot sum\_sq\_grad_{t-1} + (1-\beta) \cdot g_t^2$

$\theta_t = \theta_{t-1} - \alpha \cdot g_t / \sqrt{(sum\_sq\_grad + \epsilon)}$

until θ_t is converged

upgraded from the RMSProp optimizer. Averages of both the first moments of gradients and the second moments of the gradients are implemented in this algorithm. Instead of using the parameter learning rates based on the mean of first moment as in RMSProp, it also takes the average of the second moments of the gradients, which calculates an exponential moving average of the gradient and the squared gradient. Hyper-parameters beta1 and beta2 control the decay rates of these moving.algorithmic 2 describes how it works. Based on the theories above, we carry out ex-

Algorithm 2 adam

Input: step count$t$,initial learning α,momentum parameter$beta_1$,decay rate$beta_2$

repeat

$t = t + 1$

$g_t = \nabla_\theta f_t(\theta_{t-1})$

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

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

$\hat{m}_t = m_t / (1-\beta_1^t)$

$\hat{v}_t = v_t / (1-\beta_2^t)$

$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$

until θ_t is converged

periments to compare these algorithms on EMNIST using the standard 3 hidden layer architecture. For the RMSProp algorithm, learning rate is separately set at 0.01, 0.001, 0.0001, and the decay rate is at 0.9, 0.99, 0.999, (gradually close to 1.0). In order to avoid over-fitting and figure out the best model structure, early-stopping is also used in the process.

LEARNING RATE / DECAY RATE	0.9	0.99	0.999
0.01	61.9%	63.3%	21.8%
0.001	83.4%	84.2%	82.3%
0.0001	83.4%	83.5%	83.5%

Table 2. Classification accuracies for different baseline on validation sets with RMSProp.

	SGD	RMSPROP	ADAM
VALIDATION	83.9%	84.2%	84.3%
TEST	82.8%	82.6%	83.9%

Table 3. Classification accuracies for different baseline on test sets.

table 2 shows the experiment results that the using early-stop to get the weights from validation set in the condition that the value of cost function do not raise for several iterations. When the learning rate setting is large, the early-stop will occur due to the fast change of weights and the potential over-fitting on training set. In this circumstance, the accuracy values of these models are far less than the performance of other models with smaller learning rate. From those experiment, we get ideal RMSProp model with accuracy 84.2% with learning rate at 0.001 and decay weight at 0.99 on validation set.

For the Adam algorithm, learning rate is separately set at 0.1,0.01, 0.001, 0.0001, and the beta1 is at 0, 0.3, 0.6, 0.9, and beta2 is at 0.99, 0.999, 0.9999. 64 sets of experiments were completed using the computer's loop calculation, and the hyper-parameter combination setting with highest accuracy value was obtained, at 84.3% with learning rate at 0.001, beta1 at 0.9, and beta2 at 0.999.

For the test set, outcomes are listed in table 3. The classification accuracy of three different models are shown clearly as we expect. It shows the outcomes of the application of best weights chosen from validation set to the test set. With basic three hidden layers, three algorithms get different accuracy values on validation set and test set.

In general, Adam algorithm gets the best performance on validation set, followed by RMSProp and SGD algorithms. Meanwhile, on the test set, the outcome is the same. Adam still get best classification result,but SGD as baseline outcomes the RMSProp.

figure 2 shows the accuracy curves during the training process. It is illustrated that Adam has the fastest learning speed and can get the convergence better than RMSProp.

(a) RMS (b) ADA

Figure 2. Accuracy Curve of RMS and Adam

LEARNING RATE SETTING	0.0001:0.001	0.0001:0.005
WARM-START	84.70%	82.6%
NO-RESTART	83.76%	80.6%

Table 4. Classification accuracy values in Adam on validation sets with different scheduler.

LEARNING RATE	0.1	0.01	0.001
WARM-START	84.4%	83.8%	70.0%
NO-RESTART	84.0%	84.1%	70.6%

Table 5. Classification accuracy values in SGD on validation sets with different scheduler.

4. Cosine annealing learning rate scheduler

In this part, we use a learning rate scheduler which is presented in Loshchilov & Hutter (2017), for stochastic gradient descent(SGD)and Adam. With a budget of 100 epochs, the learning rate is controlled by two processes: a function which decreases it through time, and periodic warm restarts in which the learning rate is increased. The learning rate is updated following the method:

$$\eta_t = \eta_{min}^{(i)} + 0.5(\eta_{max}^{(i)} - \eta_{min}^{(i)})(1 + \cos(\pi T_{cur}/T_i))$$

In the experimental stage, we initially explore the SGD algorithm using the learning rate scheduler with warm start and no-restart. Initially, we still set learning rate as a hyper-parameter, using loop function to find the best model. Learning rate is separately set at 0.1, 0.01 ,0.001 in both warm-restart process and no-restart process. The outcomes are listed in table 5. It is shown that the model using SGD algorithm with warm-restart scheduler has the best performance, reaching at 85.1%on the validation set.

Next, we explore the Adam performance with different learning rate schedulers. We set Ti at 25 and 100, and T_cur at 3. As for the hyper-parameters in Adam algorithm with different learning rate scheduler, we set the maximum learning rate and minimal learning rate for different values. The motivation for the range of learning rate setting are from the previous experiment that using 0.001 as the learning rate for Adam algorithm generates the best model. In this circumstance, we use two groups to train the model. One setting is 0.001 for the maximal learning rate and 0.0001 for the minimal learning rate, and the other one is 0.005 for the maximal learning rate and 0.0001 for the minimal learning rate. The outcomes are shown in the table 4.The learning curves are shown in figure 3 .

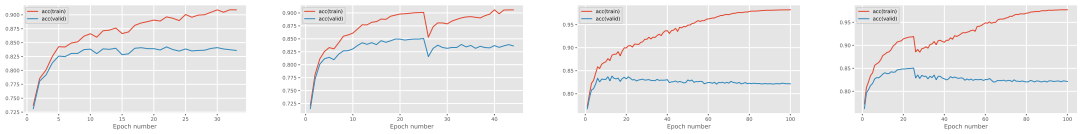
Once the ideal models are chosen given the accuracy values on validation set, we implement them on the test set. we can see the results in table 6.

In general, the learning rate scheduler with warm-restart has a direct impact on the accuracy values of the models in both algorithms. We can see from the table that the test accuracy values with warm-start are higher than that with no-restart. However, the results for Adam with warm-start scheduler did not benefit form this scheduler: its best result was obtained from baseline, the model with constant learning at 0.001. Similarly to the SGD, the usage of scheduler did not influence on the test performance. The accuracy with warm-start in Adam has the best classification result, reaching at 83.9%. This is may because the more complicated the model is, the worse outcome it gets. The simplicity of the baseline may solve the classification task more efficiently.

ADAM		
BASELINE	WARM-START	NO-RESTART
83.9%	81.5%	81.3%
SGD		
BASELINE	WARM-START	NO-RESTART
82.8%	82.9%	82.2%

Table 6. Classification accuracy values in SGD on validation sets with different scheduler.

From the learning curves, we can see that models with the scheduler can fast change learning rate. For Adam model with warm-restart function, it converged at the 24th epoch, while the model with no-restart function converged at the 16th epoch. The convergence situation is similar to that of SGD but much slower. The results about the accuracy of the models is slightly different from what we expect. It may due to the fact that the batch size is small and the influence of hyper-parameters. We use 0.01 as learning rate at SGD and 0.001 at Adam. The initial learning rate setting may influence the learning process and it should be explored in the future work.



(a) SGD-NO (b) SGD-WS (c) ADA-NO (d) ADA-WS

Figure 3. a

5. Regularization and weight decay with Adam

When training neural networks, Adam is widely used because of its fast convergence. But the best results on many data sets are fine-tuned with SGD . It can be seen that Adam's generalization is not as good as SGD with Momentum. An important reason for the poor performance of Adam's generalization in that article is that the L2 regularization term in Adam is not as effective as in SGD, and this problem has been fixed by the original definition of Weight Decay.

In the paper of Loshchilov and Hutter, the problem that the poor generalization of the most popular adaptive gradient method, Adam , is resulted from that L2 regularization is not nearly very effective as we expect. So they proposed a weight decay regularization by the separate of the Weight Decay from the gradient first, followed by normalizing the weight decay taken other parameters into consideration , and finally adding cosine annealing learning schedule and warm restarts to fasten the whole process.

Based on the thoughts above, we implement the following experiment like the paper. First, we use a constant decay rate in Adam learning rule and updated it with the weight decay calculated by the formula associated with the batch size, the total number of training points per epoch and total number of epochs .Finally, the Cosine Annealing and Warm Restarts are implemented in the whole process to train the model.

First, we compare the performance in Adam with L2 regularization and weight decay. The learning rate is fixed and the only hyper-parameter is the weight decay rate. For L2 regularization, we set 0.01,0.005,0.001,0.0001 (just like the research done in Loshchilov & Hutter (2017))and the outcomes are shown in the table 7. 0.005 is the normalized weight decay. Then, we implement the different weight decay regularization on basic model with default parameters.

For the next experiment stage, the scheduler is added to constrain the learning rate as we did before: using an cosine function to change the learning rate and accelerate the convergence process with less interations. The weight decay regularization is still the hyper-parameter set separately at 1e-3, 1e-4 and 1e-5(proposed in Loshchilov & Hutter (2017)), while the maximum learning rate and minimal learning rate keep the same as the previous experiment we implemented in task three, 1e-3 at maximun learning rate and 1e-4 at minimal learning rate. Warm-start and no-restart methods are used in the experiment. After choosing the best models, they were tested on the test set. The results are shown in the table 9.

L2 REGULARIZATION	ACCURACY VALUE
0.01	75.8%
0.005	80.3%
0.001	85.5%%
0.0001	83.6%

Table 7. Classification accuracies in Adam on validation sets with different L2 regularization.

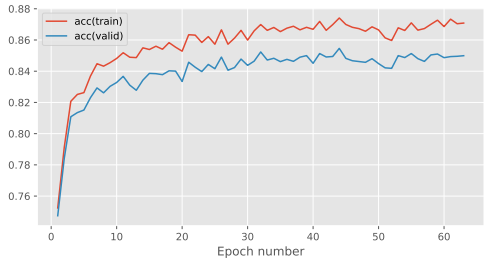
WEIGHT DECAY REGULARIZATION	ACCURACY VALUE
1E-3	77.7%
1E-4	85.8%
1E-5	84.0%

Table 8. Classification accuracies in Adam on validation sets with different weight decay regularization.

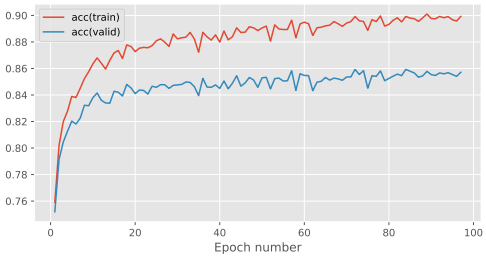
From the table 7 and table 8, we choose the best models figure 4taken the accuracy value on validation set into consideration and get the test values list in table 9. It is clear that the usage of weight decay can converge faster and optimize the model better. For L2 implementation method, the weights of the gradient value is larger, the decline is less than expected.That is to say,when the two weights are the same, the weight decay should have the same effect on them. But the gradient is relatively large. Due to Adam's normalization, the decline situation is relatively small. Above all, the weight decay can fasten the whole process and get better performance.

CONSTANT WEIGHT DECAY	L2 REGULARIZATION
84.3%	84.1%
WARM-START	NO-RESTART
84.7%	84.18%

Table 9. Classification accuracy values in Adam on set sets with different regularization.



(a) l2



(b) weight decay

Figure 4. L2 and weight decay for Adam

From the table 9 it is clear that, weight decay regularization can improve the model better than the L2 regularization.Secondly, the usage of cosine annealing learning schedule can reach a higher accuracy value than constant weight decay. It is because of the fact that with this scheduler, the learning rate can change fast and save more time to find the right direction the optimal. Thirdly, the model trained by warm start performed better than the model with no-restart. The outcome is reasonable and we can see an clearly improvement compared with baseline.

6. Conclusions

Based on the experiment, it is shown that for Adam algorithm, the implementation of weight decay regularization along with cosine annealing learning schedule and warm-start has the best performance. From the above experiment, the conclusions are as follows:

1. The SGD as baseline with 3 hidden layers is still a strong tool.

2. Warm start function can lead to a better performance.

3. L2 regularization has a different influence on the Adam.

However, there are some work needed to be discussed in the future. Although the outcome is reasonable and comply with the practical results of the paper. The best model in the whole experiment still have some hyper-parameters. For example, the cosine annealing scheduler contains the maximal , minimal learning rate and maximum learning rate discount factor. If there is a best model with this structure, the appropriate setting of these parameters may boost the accuracy. That's the point we need to work in the future.

References

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL https://arxiv.org/abs/1412.6980.

Loshchilov, Ilya and Hutter, Frank. Fixing weight decay regularization in Adam. *arXiv preprint arXiv:1711.05101*, 2017. URL https://arxiv.org/abs/1711.05101.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5 - rmsprop, coursera: Neural networks for machine learning. Technical report, University of Toronto, Toronto, ON, 2012.