# MLP Coursework 1: Learning Algorithms and Regularization

s1836694

## Abstract

The report focuses on the experiments about adaptive learning rate algorithm and regularization problem using 2-5 hidden layer neural networks. The report compares RMSProp and Adam with baseline systems using SGD. The two adaptive algorithms does improve the speed of converge. The investigation on Cosine annealing is also displayed in the report. The final experiment is about the difference of L2 regularization and Wight decay in Adam. In every experiment, we trained our models on EMNIST training data with monitoring on the validation set before comparing the performance on the test set. In the end, it is proved that adaptive learning rate algorithms speed up the converge and Adam with weight decay works the best on EMNIST data set.

## 1. Introduction

The primary focus of the report is to investigate the advantages of adaptive learning rate algorithms and how to apply regularization to these algorithms (Adam in particular). This report shows the experiments on the impact of different hypermeter settings and the improvement of using the adaptive learning rate algorithms such as RMSProp and Adam. Another aim of the experiment is to implement cosine annealing learning rate scheduler and to understand how it works as well as what kind of improvement can be made. Thirdly, we also conduct the research on the problem of regularization of an adaptive learning rate algorithm - Adam. Finally Adam with Weight Decay (AdamW) algorithm is proved as the best one in our experiments.

All the experiments are conducted on the EMNIST dataset (Cohen et al., 2017). There are 47 classes containing digits as well as both upper and lower-case letters. The whole dataset is split into three parts: training set (100,000 instances), validation set (15,800 instances) and test set (15,800 instances).

The report has 4 main parts. In first section, baseline systems (multi-layer neural networks using Stochastic Gradient Descent) are built to compare with advanced algorithms. Then two adaptive learning rate algorithms are implemented, followed by the experiment of Cosine annealing scheduler in Section 3. The last section shows the investigation on Adam with Weight Decay algorithm.

## 2. Baseline systems

In this section, baseline systems are established using stochastic gradient descent (SGD). Comparisons between these systems are made to investigate the influence of the number of layers and hyperparameters. These systems are consist of 2-5 hidden layers with 100 ReLU units per layer.

| Batch size | LR = .001 | LR = .01 | LR = .1 |
|---|---|---|---|
| 50 | 79.29% (120) | 72.28% (119) | 84.53% (21) |
| 100 | 75.02% (120) | 84.24% (117) | 84.54% (21) |
| 500 | 60.74% (120) | 79.26% (119) | 84.09% (21) |

*Table 1.* Classification accuracies of **SGD** on the whole **validation set** and optimal epoch numbers for **2-hidden-layer** model. LR represents learning rate. The numbers in the brackets are the optimal epoch numbers corresponding to specific hyperparameters.

| Systems | LR = .01 | LR = .05 | LR = .1 |
|---|---|---|---|
| 3 hidden layers | 84.11% (103) | 84.15% (34) | 84.13% (19) |
| 4 hidden layers | 84.35% (97) | 84.23% (27) | 84.12% (21) |
| 5 hidden layers | 82.65% (119) | 84.07% (29) | 84.13% (59) |

*Table 2.* Classification accuracies of **SGD** on the whole **validation** set and optimal epoch numbers for the models with different number of hidden layers. LR represents learning rate. The numbers in the brackets are the optimal epoch numbers corresponding to specific hyperparameters.

Typically, learning rate of a neural network ranges from $10^{-6}$ to 1 (Bengio, 2012). The learning rates for the 2-hidden layer model were set to .001, .01 and .1. The mini-batch size is usually greater than or equal to 1 and less than a few hundred (Bengio, 2012). In the experiments, they were set to 50, 100 and 500 respectively. Due to the limitation of

| Systems | Accuracy |
|---|---|
| 2 hidden layers | 83.2% |
| 3 hidden layers | 83.0% |
| 4 hidden layers | 83.1% |
| 5 hidden layers | 83.0% |

*Table 3.* Classification accuracies of the **SGD** models with different number of hidden layers on **test** set.

time and the computational ability, the epoch number was set to 120 for every training procedure. Although larger epoch number is able to give more information on when the model converges and overfits, the small constant epoch number makes sense in these experiments. The reason is that we can compare the convergence speed between different settings and also get the optimal epoch number by early stopping. Table 1 depicts the best classification accuracies (on the whole validation set) of 2-hidden-layer model and corresponding training epoch number under specific batch size and learning rate. The larger batch size and lower learning rate makes parameter updating slower (see the first column and any row of Table 1). In fact, when learning rate is .001, the error of models cannot converge within 120 epoch in this experiment. Therefore, for next experiments we would use .01, .05 and .1 instead.

As the models using 100 batch size performs better than others (see Table 1), next experiments on 3-5 hidden layers used 100 batch size only. Table 2 gives the classification performance of 3-5 hidden layers setting batch size to 100 and learning rate to .01, .05 and .1. In Table 1 and Table 2, we chose the best hyperparameter settings for each number of hidden layer model: 2 hidden layer (learning rate = .1, batch size = 100, epoch 21), 3 hidden layer (learning rate = .05, epoch 34), 4 hidden layer (learning rate = .01, epoch 97), 5 hidden layer (learning rate = .05, epoch 29). Then, these trained models (including weights and biases) are applied on test data set. The results are shown in Table 3. Theoretically, deeper networks have better performance. However, 2-hidden-layer system performs better than 5-hidden-layer system. One potential reason is that 5-hidden-layer model is too complex to fit the dataset.

## 3. Learning algorithms – RMSProp and Adam

In this section RMSProp and Adam were implemented and compared with stochastic gradient descent. As shown in Eq. (1), RMSProp algorithm normalizes learning rate $\eta$ by a moving average of the squared gradient (Tieleman & Hinton, 2012).

$$Si(t) = \beta Si(t-1) + (1-\beta)di(t)^2$$
$$\triangle wi(t) = \frac{-\eta}{\sqrt{Si(t)} + \epsilon} di(t) \quad (1)$$

Kingma and Ba 2014 added momentum and proposed Adam algorithms. Equation (2) illustrates the only difference - *momentum* term which is used to keep previous direction of updating.

$$Mi(t) = \alpha Mi(t-1) + (1-\alpha)di(t)$$
$$\triangle wi(t) = \frac{-\eta}{\sqrt{Si(t)} + \epsilon} Mi(t) \quad (2)$$

Table 4 depicts the experiments on different hyperparameters of RMSProp applied on validation set. The best performance is $\beta = .90$, closely followed by the performance of $\beta = .85$ , which means that too low weighted coefficient (i.e. less contribution) of current squared gradient makes the model perform worse. The similar experiments were conducted on Adam - 3 learning rates (.0005, .001 and .005), 3 values of $\alpha$ (.85, .90 and .95) as well as 3 values of $\beta$ (.9, .999 and .9999). Table 5 lists the best $\beta$ and $\alpha$ for each learning rate. Compared with 3-hidden-layer SGD (84.15%), RMSProp and Adam achieved lower accuracy (82.4% and 82.3% respectively) in the experiments. However, RMSProp and Adam converges roughly twice faster than SGD (see Figure 1). This is because the normalization of gradient reduces the extent of zigzag parameter update. Furthermore, Adam also utilises the *momentum* to keep previous update direction, which makes the learning curve more smooth.

| $\beta$ | LR = .0005 | LR = .001 | LR = .005 |
|---|---|---|---|
| .85 | 83.57% (14) | 83.91% (12) | 75.28% (4) |
| .90 | 83.67% (19) | **83.92%** (12) | 76.40% (2) |
| .95 | 83.85% (19) | 83.80% (14) | 77.44% (4) |

*Table 4.* Classification accuracies on the whole validation set and optimal epoch numbers for **RMSProp** model. LR represents learning rate. The numbers in the brackets are the optimal epoch numbers corresponding to specific hyperparameters. The **bolded** item is the best performance

| Learning rate | Best hyperparameter settings | Accuracy |
|---|---|---|
| .0005 | $\alpha = .95, \beta = .999$, epoch = 21 | **83.95%** |
| .001 | $\alpha = .95, \beta = .9$, epoch = 10 | 83.82% |
| .005 | $\alpha = .95, \beta = .999$, epoch = 23 | 80.38% |

*Table 5.* Best hyperparameters and classification accuracies on validation set for **Adam** model. The **bolded** item is the best performance

## 4. Cosine annealing learning rate scheduler

This section describes the investigation on the benefits that Cosine annealing scheduler (Loshchilov & Hutter, 2017) brings. The learning rate is multiplied with the multiplier, which enables the learning rate to be adaptive. The multiplier is actually a half period of cosine function during which the value monotonically decreases (see Eq. (3)).

$$\eta_t = \eta_{min}^{(i)} + 0.5(\eta_{max}^{(i)} - \eta_{min}^{(i)})(1 + cos(\frac{\pi T_{cur}}{T_i})) \quad (3)$$

where $\eta_{max}^{(i)}$ and $\eta_{min}^{(i)}$ are the upper and lower boundary of learning rate multiplier $\eta_t$ and $T_{cur}$ represents the epoch number during the new period $T_i$. In practical an expansion factor of each period and an discount factor for $\eta_{max}$, which makes the learning rate become lower and each learning rate keep longer time (Bengio, 2012). This is because the parameters of network get closer to the optimal values and small updating steps are desired. In addition, periodic restart (i.e. dramatic increase in learning rate suddenly)

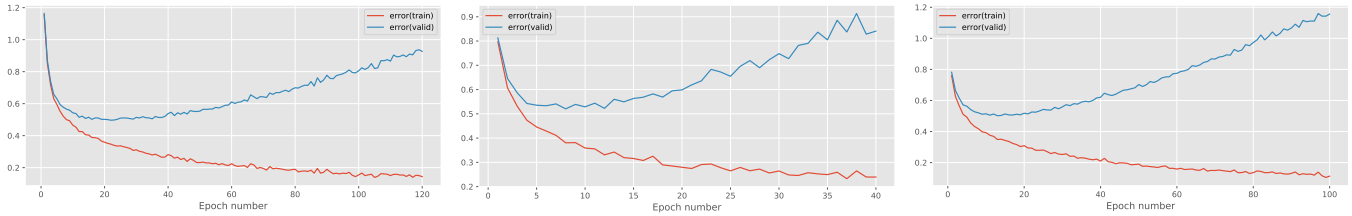


*Figure 1.* Learning curves of SGD (1st), RMSProp (2nd) and Adam (3rd).

| MODEL | BEST HYPERPARAMETER SETTINGS | ACCURACY ON TEST SET |
|---|---|---|
| SGD (CONSTANT LEARNING RATE) | LR = .05 | 83.0% |
| SGD (COSINE ANNEALING WITHOUT RESTART) | LR = .05 | 83.4% |
| SGD (COSINE ANNEALING WITH RESTART) | LR = .05 | 83.8% |
| ADAM (CONSTANT LEARNING RATE) | LR = .0005 $\alpha$ = .95, $\beta$ = .999 | 83.0% |
| ADAM (COSINE ANNEALING WITHOUT RESTART) | LR = .0005, $\alpha$ = .95, $\beta$ = .999 | 82.9% |
| ADAM (COSINE ANNEALING WITH RESTART) | LR = .0005, $\alpha$ = .85, $\beta$ = .999 | 83.2% |

*Table 6.* Classification accuracies (on **test** set) of 3-hidden-layer models using **SGD** and **Adam** with best hyperparameter settings. The table shows the influence of **cosine annealing** learning rate scheduler.

allows the model to be likely to skip out the local "valley bottom". Figure 2 shows the shape of cosine annealing learning rate scheduler.

The following experiments were carried out to get insight of the impact of Cosine annealing scheduler without restart and with warm restart. Before this, the best hyperparameter settings were chosen from 3 learning rates, 3 values of $\alpha$ and 3 values of $\beta$ on validation set. Table 6 gives the information on the classification accuracies of SGD and Adam with best hyperparameter settings on test set. As shown in Table 6, Cosine annealing scheduler improves the accuracy of SGD method on test set. For adaptive learning rate algorithm - Adam, cosine annealing seems not to improve the performance on both accuracy and converge speed in the particular experiment. However, it should also work on Adam but just not have as great impact as that on SGD (Loshchilov & Hutter, 2017). In addition, cosine annealing learning rate scheduler with warm restart works better than that without restart (see Table 6). This is because warm restart is more likely to help the error function to reach the global minima by the large step which happens when the learning rate restarts.

## 5. Regularization and weight decay with Adam

In this section, we will discuss about the problem of using regularization on Adam and the solution - using weight decay instead, followed by the experiments on the improvement of using weight decay and cosine annealing scheduler on Adam algorithm.

Regularization works well on constant learning rate algorithm like SGD because the form of the gradient of regularization exactly equals to the form of weight decay

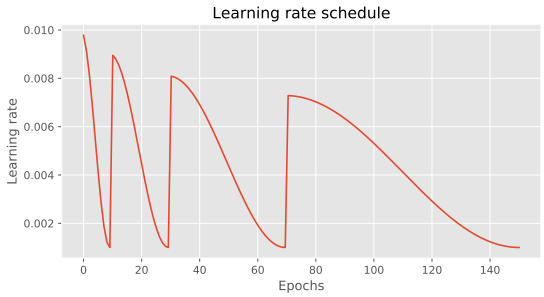


*Figure 2.* The shape of cosine annealing learning rate scheduler ($\eta_{min}$=0.001, $\eta_{max}$=0.01, $T_i$=10., discount=0.9, expansion=2.0).

Loshchilov & Hutter (2017). However, in Adam algorithm, the momentum (i.e. previous gradient) and squared gradient contains the weight decay part when L2 regularization is applied. This means that the update of weights does not exactly subtract a particular proportion of the values of last weights. The solution is rather straightforward. Instead of regularization, original version of weight decay is applied (i.e. subtracting at the final weight update step instead of the step of calculating the gradients).

The first experiment in the section is to find the best hyperparameter settings of the neural network using Adam with Weight Decay (AdamW) according to the accuracy on validation set. As we have investigate that $\alpha$= .95 and $\beta$ = .999 are the best (see Section 2, Table 4 and 5), they will be kept in the following experiment. Table 7 compares Adam with weight decay with two different learning rates and 3 coefficients of weight decay. Table 8 compares Adam with L2 regularization with three different coefficients of regularization. Then we compared the two methods on test

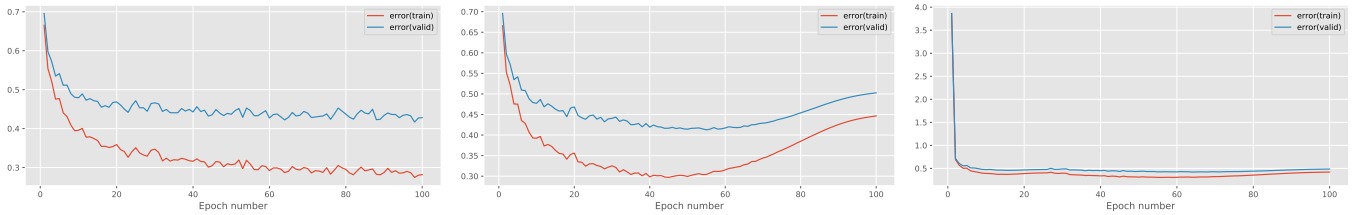


*Figure 3.* Learning curves of Adam with weight decay (1st), AdamW with cosine annealing but without restart (2nd) and AdamW with cosine annealing with warm restart (3rd).

| Weight Decay | LR = .001 | LR = .0005 |
|---|---|---|
| 1e-06 | 83.81% (9) | 84.01% (18) |
| 1e-05 | 84.09% (14) | 84.35% (18) |
| 1e-04 | **85.63**% (98) | 85.6% (82) |

*Table 7.* Classification accuracies of the models using **Adam with weight decay** with two different learning rates and 3 coefficients of weight decay.

| Weight Decay | Accuracy |
|---|---|
| 1e-03 | 85.69% (70) |
| 1e-02 | 78.01% (74) |
| 1e-01 | 2.3% (0) |

*Table 8.* Classification accuracies of the models using **Adam with L2 regularization** with 3 coefficients of regularization. The learning rate is .0005, $\alpha = .95$ and $\beta = .999$.

set with the best hyperparameter settings. As expected, Adam with weight decay achieves higher accuracy (85.0%) than that with L2 regularization (84.8%).

The following experiment is to investigate the performance of the combination of cosine annealing learning rate schedule and Adam with weight decay (AdamW). Here we use the best hyperparameter settings which are decided above (learning rate = .001, weight decay = .0001). Then we ran the model on the test dataset, the accuracies of cosine annealing without restart and with restart ($T_i = 25, expansion = 3, discount = .9$) reached 85.0% - 85.2% which is almost the same as Adam with weight decay without cosine annealing scheduler; but AdamW with cosine annealing only need 50 epochs to converge, compared with 98 epochs of Adam. As shown in the third picture of Figure 3, there is an sudden change of the learning curve, which is caused by the warm restart at 25th epoch. Additionally, Adam with weight decay is better than Adam (roughly 83%) as expected. When comparing pure Adam, Adam with L2 and Adam with weight decay across Table 5, 7 and 8 (see the best epoch number roughly representing when the model start to overfit), we can prove that L2 regularization can improve the overfitting problem to some extent and Adam with weight decay work better. As illustrated in Figure 3, AdamW using cosine annealing without restart can converge and overfit much faster than pure AdamW. This means that weight decay of Adam cannot domains when consine annealing without warm-restart is utilised. However, when warm restart is applied at the meantime, the algorithm can work better with higher accuracy and nice learning curve.

## 6. Conclusions

In the investigation, baseline systems using simple SGD were built which is able to achieve a rather high accuracy of classification (83.2%) using only 2 hidden layers. More layers makes the speed of optimization slower. The large learning rate can speed up the converge procedure but it is hard for model to reach the optimal state if the learning rate keeps constant. Therefore, adaptive learning rate algorithm such as RMSProp and Adam are desired. Due to benefits of normalized gradients and momentum, RMSProp and Adam did improve the converge speed twice approximately with keeping or even improving the accuracy (Kingma & Ba, 2014; Tieleman & Hinton, 2012). Cosine annealing learning rate scheduler can significantly improve the optimization performance of constant learning rate algorithm SGD, but does not show an obvious increase in the accuracy of Adam with cosine annealing. The possible reason is that Adam already have adaptive learning rate itself. Weight decay can improve the overfitting problem of Adam algorithm better than L2 regularization does, because it decouples the weight decay with momentum and normalised squared gradients (Loshchilov & Hutter, 2017).

Adam with Weight Decay performs the best on the classification task on the balanced EMNIST data with the accuracy of 85.1%(without cosine annealing) and 85.0% (with cosine annealing with warm restart - $T_i = 25$ and $T_{multi} = 3.0$ in particular). As shown above, the benefits of cosine annealing learning rate scheduler in the experiments does not appear to be obvious in the results, so we need to do some investigation on it for future work. For example, we may change the different initialization settings of weights and biases or use a larger data set.

## References

Bengio, Yoshua. Practical recommendations for gradient-based training of deep architectures. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012. ISSN 03029743. doi: 10.1007/978-3-642-35289-8-26.

Cohen, Gregory, Afshar, Saeed, Tapson, Jonathan, and van Schaik, André. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017. URL http://arxiv.org/abs/1702.05373.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, December 2014.

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*, 2012.