Learning Algorithms and Regularization

s1886437

Abstract

This paper documents experiments performed on the classification of the EMNIST dataset to compare the learning algorithms SGD, RMSProp and Adam. As well as the effects of using a Cosine annealing learning rate scheduler on training on a deep network and experimenting with warm restarts. The paper ends with experiments on AdamW and L2 regularization and a conclusion of the finding of the experiments.

1. Introduction

In this report we will outline experiments and finding on; Adam and RMSProp learning algorithms; Cosine annealing learning rate scheduler and regularization and weight decay with Adam. The aim is examine the effects of these techniques on classification when applied to the EMNIST dataset and to examine how they effect the learning of the network.

The EMNIST dataset (Cohen, 2017) is a set of handwritten characters and digits designed to be harder to classify than the MNIST dataset. The set represents the characters as 28x28 gray scale images and has 47 different classes, because of this are networks with have 784 input nodes and 47 output nodes. During training and testing of the networks we will use a set if 100,000 data points for training and two sets of 15,800 data points for testing and validation.

2. Baseline systems

When creating classifiers for a dataset its is very useful to have a baseline classifier. This is a simple classifier that is very unlikely to encounter technical difficult during implementation. The baseline gives us an idea of the sort of classification rate we should expect from our more complex algorithms in the data set. If the baseline is out performing our more complex techniques it is very like that we have made mistakes in the implementation of these techniques.

For the baseline we are using stochastic gradient decent (SGD) on a network of 100 ReLU hidden units per hidden layer. Using a learning rate of 0.1 and batch size of 100 we then tested the network with varying numbers of layers. We found the best average classification rate on test data to be just over 0.8 (figure 1).

Despite this we will be using a standard network architecture of 3 hidden layers of 100 ReLU units for the rest of this report.

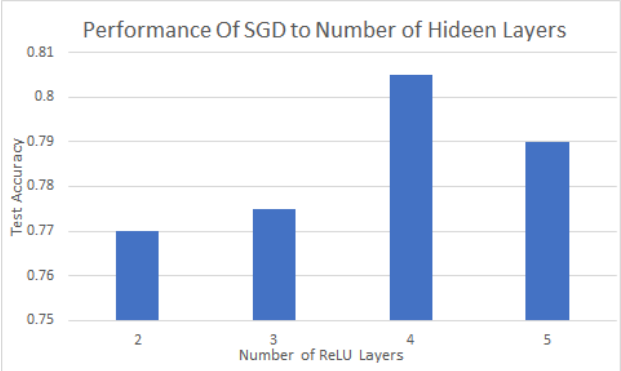

Figure 1. Average performance of SGD on EMNIST data set using ReLU hidden layers of size 100 units.

3. Learning algorithms – RMSProp and Adam

We have implemented the adaptive learning algorithms RMSProp and Adam learning into our network so we can compare their performance.

3.1. RMSProp

RMSProp updateds weight in a similar style to SGD but it adapts the learning rate for each weight by normalizing the learning rate with a moving average of the squared error gradient at the weight.

The moving average for weight i at time step t is given by:

$$S_i(t) = \beta S\, i(t-1) + (1-\beta) d_i(t)^2 \tag{1}$$

Where d_i is the gradient of error with respect to the weight and β is the decay rate (~ 0.9). And the weight update is given by:

$$w_i(t) = w_i(t-1) \frac{-\eta}{\sqrt{S_i(t) + \epsilon}} d_i \tag{2}$$

Where ϵ is a very small constant (~1e-8) to prevent dividing by 0 errors.

We optimized the hyper parameters and found that a learning rate of 0.005 and decay rate of 0.85 averaged an accuracy of 0.841 on the validation set.

3.2. Adam

Adam (Kingma & Ba, 2014) can be seen as RMSProp with momentum. It introduces a momentum term to the error

gradient. Given by:

The moving average for weight i at time step t is given by:

$$M_i(t) = \alpha Mi(t-1) + (1-\alpha)d_i(t) \tag{3}$$

Where α is the momentum (~ 0.9). The weight update is given by:

$$w_i(t) = w_i(t-1)\frac{-\eta}{\sqrt{S_i(t)+\epsilon}}M_i(t) \tag{4}$$

In Adam the decay rate term for $S_i(t)$ is ~0.999

We optimized the hyper parameters and found that a learning rate of 0.0006, a decay rate of 0.999 and a momentum of 0.85 averaged an accuracy of 0.843 on the validation set.

3.3. Comparison

We will now compare the learning algorithms RMSProp, Adam and SGD. We trained the 3 algorithms for 25 epochs using the hyper parameters highlighted earlier and a batch size of 100.

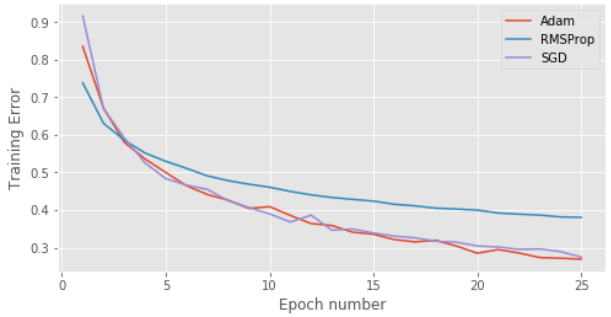

Figure 2. Training Error of Adam RMSProp and SGD on EMNIST

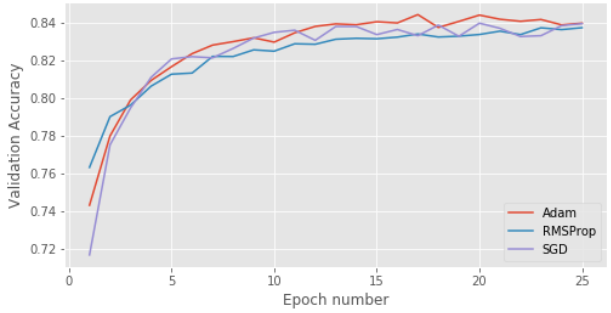

Figure 3. Validation Accuracy of Adam RMSProp and SGD on EMNIST

Looking at the training error in Figure 2 we can see that RMSProp and Adam have fit the training data far better than SGD. They are however much nosier, this could be due to the momentum of the algorithms preventing them from smoothing. Figure 3 shows us the the accuracy on the validation set of the algorithms, despite better training error performance of Adam and RMSProp all 3 algorithms perform roughly equally here.

As a final test of the 3 algorithms we ran them multiple times and recorded accuracy on the test set. Table 1 shows us that Adam performed the best of the 3 algorithms but all were comparatively similar.

	Test 1	**Test 2**	**Test 3**	**Test 4**	**Average**
SGD	0.85	0.80	0.78	0.84	**0.82**
RMSProp	0.79	0.81	0.80	0.84	**0.81**
Adam	0.84	0.83	0.86	0.82	**0.84**

Table 1. Test accuracy results of the learning algorithms, Average is to 2s.f.

4. Cosine annealing learning rate scheduler

We have implemented a Cosine annealing learning rate scheduler (Loshchilov & Hutter, 2017) to control the learning rate through the training of the network. The scheduler reduces the learning rate as the network trains by the formula:

$$\eta_t = \eta_{min} + 0.5(\eta_{max} - \eta_{min})(1 + cos(\pi T_{cur}/T_i)) \tag{5}$$

Where T_{cur} is epochs since restart and T_i epoch count at which a restart will be performed. We can see that as T_{cur} reaches T_i $(1 + cos(\pi T_{cur}/T_i))$ will reduce from an initial output of 2 down to 0 where a warm restart is preformed. This simply sets T_{cur} to 0 and multiplies T_i by a factor called T_{mult} this causes the next restart to take more epochs to reach. The learning rate will then be set to its maximum and reduce down to its minimum again.

The idea of the scheduler is to quickly approach a good solution then make very fine adjustments to the weights towards the end of the training, following the cosine annealing. And with warm restarts as proposed in (Loshchilov & Hutter, 2016) that are intended to accelerate the training of the network.

To test the scheduler we will train SGD and Adam for 100 epochs with a fixed learning rate; Cosine annealing with no warm restarts and Cosine annealing with warm restarts. This will let us examine the effects for both the Cosine annealing and the restarts.

4.1. Fixed Learning Rates

This will serve as a baseline for making comparisons with the other 2 methods we will use. We trained SGD with a learning rate of 0.01 and 0.00015 for Adam (0.85 for momentum and 0.999 for decay rate). The baselines were trained for 100 epoches.

We can see in figure 4 that both the algorithms begin to slightly over fit the training set causing the validation error

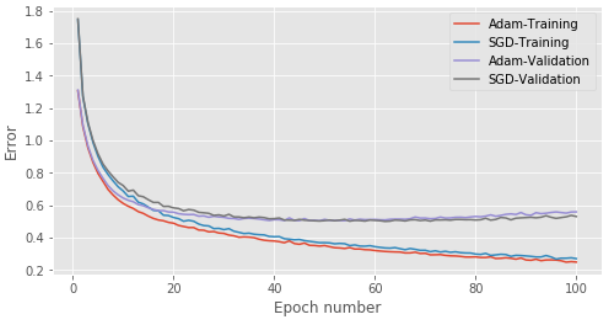

Figure 4. Error on train and validation sets of Adam and SGD on EMNIST using fixed learning rates

to slightly increase. Despite this the results on the test data were strong. Adam had a accuracy rating of 0.89 and SGD had 0.86.

4.2. Cosine Annealing With No Warm Restarts

We have assumed that no warm restarts means T_i will equal 100. This will cause the learning rate to decrease to 0 over the course of the 100 epochs. SGD has used a η_{max} of 0.02 and η_{min} of 0.002, Adam has used a η_{max} of 0.0005 and η_{min} of 0.

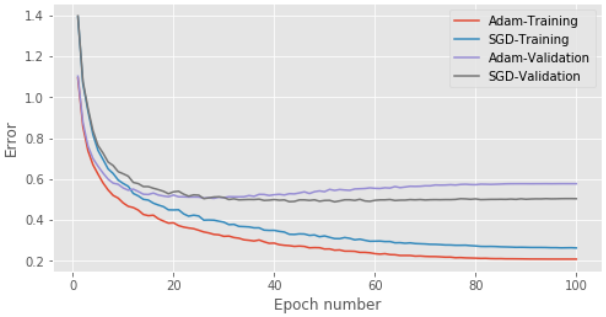

Figure 5. Error on train and validation sets of Adam and SGD on EMNIST using Cosine Annealing without restarts

Again in figure 5 we see that SGD and Adam have higher levels of validation error as in the baseline and similar errors in the training set. The Cosine Annealing has yeilded worse results than on the baseline. SGD model scored 0.79 accuracy on the test set and while Adam scored 0.85. These worse results could be caused by the model over fitting the training data too much as indicated by the higher validation error.

4.3. Cosine Annealing With Warm Restarts

For the warm restarts T_i has been set to 25 and T_{mult} to 3. This give us one warm restart at epoch 25 and then the learning rate to decrease to 0 for the final, 100^{th}, epoch. A max learning rate discount factor has also been included. This lowers the maximum learning rate after a warm restart. SGD and Adam have the same hyper parameters as in the last section and both are using a max learning rate discount factor of 0.8. SGD has used a η_{max} of 0.02 and η_{min} of 0.002, Adam has used a η_{max} of 0.0002 and η_{min} of 0.

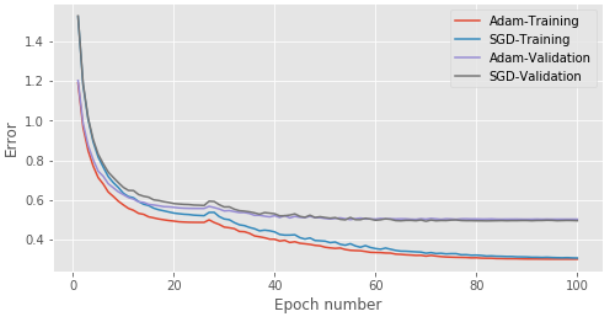

Figure 6. Error on train and validation sets of Adam and SGD on EMNIST using Cosine Annealing with warm restarts

We can in figure 6 the effects of the warm restart takes place at epoch 25. The error lines spike slightly as the learning rate is increased. Then continue to fall lower than it was plateauing before the restart. The restart also lead to greater performance in the testing set than without the restart with SGD scoring 0.90 and Adam scoring 0.89.

This result is slightly unexpected as Adam did not outperform its baseline. It is possible that more time was needed to be spent on hyper parameter optimization.

5. Regularization and weight decay with Adam

In this section we will be exploring the effects of L2 regularization and weight decay for the Adam learning algorithm. As well as testing learning rate schedules with AdamW.

L2 regularization aims to prevent over fitting in training by adding a regularization term to the error gradient. However, regularization has been shown to be less effective in adaptive gradient methods such as Adam. This is due to L2 regularization not working as intended in adaptive gradient algorithms

To over come this Adam with weight decay (AdamW) has been proposed (Loshchilov & Hutter, 2017) that decouples weight decay from the error gradient and adds it too the weight update.

5.1. L2 Regularization vs. Weight Decay

With the research shown in (Loshchilov & Hutter, 2017) it should be expected that Adam with weight decay should outperform Adam with L2.

To compare the two we have created AdamW and Adam with L2 and run them on the same three data splits three different times. Both are using a constant learning rate, the hyper parameters given in section 3 and run for 100 epochs. The weight decay rate on both is 0.00001.

	Test 1	Test 2	Test 3	Average
AdamW	0.87	0.86	0.85	**0.86**
Adam With L2	0.80	0.85	0.85	**0.83**

Table 2. Test accuracy results AdamW and Adam with L2, Average is to 2s.f.

From Table 2 we can a significant performance improvement in AdamW over Adam with L2.

5.2. Constant Learning Rate vs. Cosine Annealing Schedule

We will now compare AdamW performance using a constant Learning Rate and a learning rate controlled by Cosine Annealing. The constant learning rate was set to 0.0002 and Cosine Annealing had η_{max} of 0.0002 and η_{min} of 0.

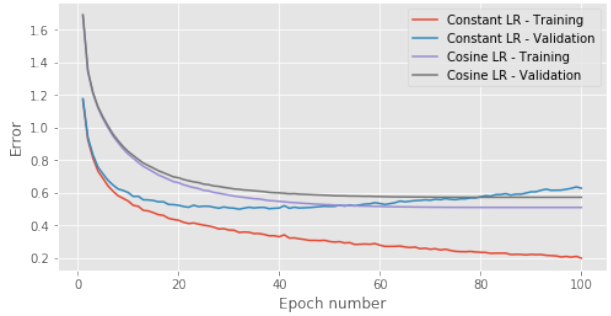

Figure 7. Error on train and validation sets of Adam on EMNIST using Cosine Annealing adjusted learning rates and constant learning rate

We can see that the Cosine Annealing adjusted learning rate kept the model from over fitting slightly better than the constant learning rate.

5.3. No Restarts vs. Warm Restart

As in 4.2 we have assumed that no restarts is the same as $T_i = 100$. η_{max} of 0.0002 and η_{min} of 0 have been used for both.

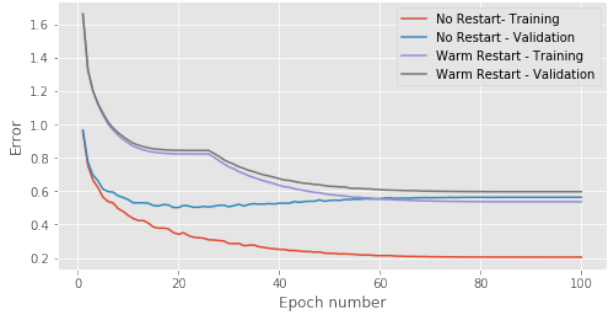

Figure 8. Error on train and validation sets of Adam on EMNIST using Cosine Annealing with and with out resets

We see the validation errors converge to roughly the same level. The two methods despite this the two scored the same result of 0.86 on the test set.

6. Conclusions

For the experiments we have conducted we can conclude that SGD can still produce strong classification results when optimized and can serve as a strong baseline. The optimum configuration we found for training of 100 epochs is training with a Cosine Annealing Schedule with warm restarts where $T_i = 25$, $\eta_{max} = 0.02$, $\eta_{min} = 0.002$ and a maximum learning rate decay of 0.8.

The best results we found using Adam for training on 100 epochs is training with a Cosine Annealing Schedule with warm restarts where $T_i = 25$, $\eta_{max} = 0.0002$, $\eta_{min} = 0.0$ and a maximum learning rate decay of 0.8. With 0.85 momentum and 0.999 decay rate.

Our experiments with Cosine Annealing With Warm Restarts have shown that in general warm restarts improve the quality of training when compared to the same method with out restarts.

We have also shown that AdamW outperforms Adam with L2 regularization.

Further work could be done to try and reduce the amount of over fitting. We would expect that drop out might be able to help in this regard. Work could as be done on finding an optimum number for hidden units in the hidden layers and the number of hidden layers.

References

Cohen, G., Afshar S. Tapson J. van Schaik A. Crafting papers on machine learning. In *EMNIST: an extension of MNIST to handwritten letters.*, 2017. URL http://arxiv.org/abs/1702.05373.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization, 2014.

Loshchilov, Ilya and Hutter, Frank. Sgdr: Stochastic gradient descent with warm restarts, 2016.

Loshchilov, Ilya and Hutter, Frank. Fixing weight decay regularization in adam, 2017.