Learning Algorithms and Regularization

Abstract

This report is motivated by different discussions on (Loshchilov & Hutter, 2017). In here, RMSProp and Adam are compared to Stochastics Gradient Descent (SGD) in terms of convergence and performance. Then, one way to improve learning rules is by using schedulers. This leads us to use the cosine annealing scheduler with and without warm restarts. They are compared between them and with constant learning rates for both SGD and Adam. Finally, the differences raised by (Loshchilov & Hutter, 2017) between L2 and weight decay in adaptive learning rules are explored by using constant and schedule learning rates. By the end, the best model is picked in terms of accuracy on the test set. All the experiments were carried out using the EMNIST (Extended MNIST) Balanced data set.

1. Introduction

Adam and RMSprop are algorithms that are supposed to converge faster than Stochastic Gradient Descent (SGD). In this work, this hypothesis is explored in an experimental way, and the best hyperparameters for the three algorithms are picked to carry out the other tasks. Then, different learning rates schedulers such as the cosine annealing with and without warm restarts are implemented and studied on Adam and SGD. These schedulers are supposed to help the loss function to converge faster. Finally, according to (Loshchilov & Hutter, 2017), Adam with weight decay should perform better than with L2. This hypothesis is explored in an experimental way using different learning rates schedulers. By the end, we will select the best models in terms of accuracy, and draw different conclusions.

The EMNIST (Extended MNIST) Balanced data set (Cohen et al., 2017) is used to carry out the different tasks. This data set contains centered images of dimension 28x28 of handwritten characters from the NIST Special Database 19. The EMNIST Balanced data set contains 47 classes in total (10 digits and 37 letters). For these experiments, the training set contains 100,000 images, and both validation and test set contain 15,800 images.

All the structures studied had 3 hidden layers with 100 units, an input layer of 728 units, and an output layer of 47 units (one unit per class). All use ReLu (Hahnloser et al.) as activation function, and cross-entropy softmax as loss function. All the experiments run 100 epochs with a batch size of 100.

Layers	η	Val. Accuracy	Test Accuracy
2	0.01	83.6 %	82.3%
3	0.005	83.4%	81.9%
3	0.01	84.1%	82.9%
4	0.01	83.5%	82.8%
5	0.01	83.4%	82.9%

Table 1. Classification accuracies of EMNIST validation and test set for various number of hidden layers and learning rates (η).

2. Baseline systems

The first step is to select a baseline system that provides a historical point of reference for the next steps. In this case, several DNNs have been trained using SGD as optimization method, and those with the highest accuracy on the validation set have been tested on a test set. The structure with the highest accuracy has been selected as a baseline system.

The models studied had between 2 and 5, and followed the structures defined previously. Different learning rates equispaced in the range of [0.1-0.001] have been studied, and in Table1 the best validation and test results are reported.

The baseline system selected is 3 hidden layers with a learning rate equal to 0.01, because it achieved the best performance on test set. Although the model with 5 hidden layers achieves the same accuracy, it is desirable to go with the simplest structure. This structure will be also used for the other experiments that involve SGD.

3. Learning algorithms – RMSProp and Adam

One problem of SGD is that it converges slowly because it is noisy. This problem can be fixed by using exponentially weighted moving averages of past gradients (Ng).

RMSProp (Tieleman & Hinton) is an algorithm that uses the exponentially weighted moving average of the gradients to control the large oscillations that the loss function experiments in the SGD. RMSProp uses different adaptive learning rates for each parameter so that the learning is speed up in the direction of the local minima, and slowed down in any other direction. The adaptive learning rate (2) is defined by the estimate of the second moment (the non-centered variance) of the gradient (1):

$$S_i(t) = \beta S_i(t-1) + (1-\beta) g_i^2(t) \tag{1}$$

METHOD	η	β / α	VAL./TEST ACCURACY
RMSPROP	10^{-4}	0.95 / -	83.4% / 82.4%
ADAM	10^{-4}	0.9995 / 0.95	83.7% / 82.9%

Table 2. Best classification accuracies of EMNIST validation and test set for Adam and RMSProp.

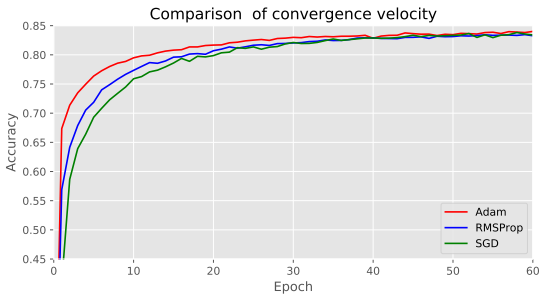

Figure 1. Comparison of convergence velocity of SGD ($\eta = 0.01$), RMSProp ($\eta = 0.0001$ and $\beta = 0.95$), and Adam ($\eta = 0.0001$, $\beta = 0.9995$, and $\alpha = 0.95$) using the validation set. Hyperparameters were selected based on the best performance on test set.

$$\eta_i(t) = \frac{\eta}{\sqrt{S_i(t)} + \epsilon} \tag{2}$$

The β in (1) is usually set around 0.9, and controls abrupt oscillations of the gradient, ϵ in (2) is used to avoid division over zero in the first update, and usually set equal to 10^{-8}, and η in (2) is the learning rate. $S_i(t = 0)$ is zero. Finally, each weight is update as:

$$\theta_i(t) = \theta_{i-1}(t) - \eta_i(t) g_i(t) \tag{3}$$

RMSProp is similar to AdaGrad (Duchi et al., 2010), but RMSProp is more powerful in non-convex problems because β controls abrupt oscillations of the gradient.

Adam (Kingma & Ba (2014)) is another algorithms that uses exponentially weighted moving average of gradients. The difference between RMSProp and Adam is that Adam uses a running average of first order of the gradient (the mean) (4) in place of the gradient(5):

$$M_i(t) = \alpha M_i(t-1) + (1-\alpha) g_i(t) \tag{4}$$

$$\theta_i(t) = \theta_{i-1}(t) - \eta_i(t) M_i(t) \tag{5}$$

The α in (4) is usually set to 0.9, and defines the exponential decay rate for the first moment estimates of the gradients. The effective learning rate ($\eta_i(t)$) is calculated as in (2),and β is usually set to 0.999. $M_i(t = 0)$ is zero.

Different hyperparameters (η, β and α) have been used for both Adam and RMSProp to train the DNNs. The $\eta's$ tried ranged from 0.001 (proposed by Kingma & Ba (2014)) to 0.0001, and both β and α were tested around their default values defined by Kingma & Ba (2014). Those models with the best performance on the validation set where selected to be tested on the test set. The best hyperparameters for a 3 hidden layers DNNs are reported in Table 2. There is no significance difference on accuracy among the three learning rules, as reported in Table 2. However, the difference among the three models is in the learning process. Figure 1. shows a comparison of the convergence velocity for the three different algorithms on the validation set. Indeed, Adam and RMSProp converge faster than SGD. Also, Adam converges faster than RMSProp. This agrees with (Kingma & Ba (2014)), because it says that Adam usually converges faster than other methods.

The best hyperparameters (Table 2) are the ones that are going to be used in the future tasks.This is because they present good results, and to safe time tuning hyperparameters.

4. Cosine annealing learning rate scheduler

One way to safe time training DNNs is to use a learning rate scheduler. A learning rate scheduler uses high learning rates at the beginning so that it can learn fast, and then reduces the learning rate to achieve high accuracy. One way to do this is by decreasing the learning rate with a cosine function, Figure3. However, having a high learning rate at the beginning and slowing it down after some epochs is not enough in high-dimensional non-convex optimization .

(Loshchilov & Hutter (2017)) proposes the cosine annealing scheduler with warm restarts.A learning rate with warm restarts that decreases in a non-linear way helps the loss function to avoid non-optimal places in the weight space. The value of the learning rate, for the epoch number T_{cur} since the last restart, is:

$$\eta(t) = \eta_{min}^{(i)} + 0.5(\eta_{max}^{(i)} - \eta_{min}^{(i)})(1 + cos(\pi \frac{T_{cur}}{T_i})) \tag{6}$$

where T_i defines the periodicity, and η_{min} and η_{max} define the range of the learning rate. Performance can be improved if T_i is multiplied by an expansion factor T_{mult} after each restart. Sometimes a discount factor is applied to the learning rate after each warm restart. Both discount and expansion factors are used so that the DNNs can learn slower when reaching the minima.

So, the goal is to compare a cosine annealing scheduler with warm restarts (CAWR), with cosine annealing without warm restarts (CAN) and with a baseline system (constant learning rate). For these experiment, T_i is set to 25 epochs, T_{mult} to 3 and discount factor equal to 0.9.

Figure 2. reports the comparision for SGD. It is clear that both schedulers outperform the constant learning rate, because both coverge faster. However, both schedulers perform very similar (CAWR performs slightly better). Before the restart, the CAWR is smoother than the CAN, because CAWR decreases faster to the η_{min}. Moreover, the plot

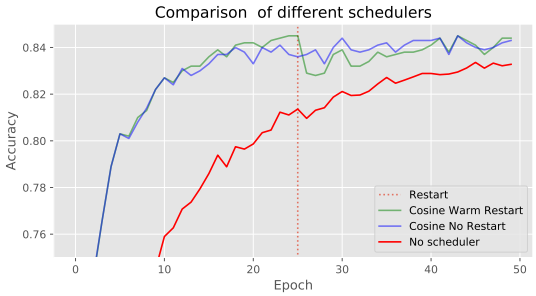

Figure 2. Comparison of different schedulers for SGD.

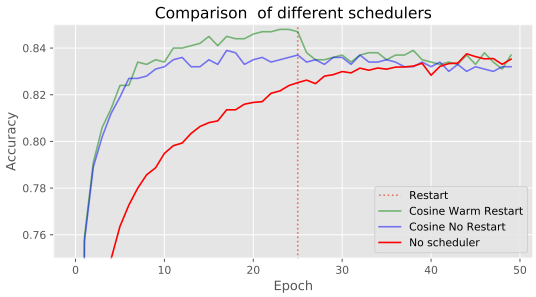

Figure 3. Comparison of different scheduler for Adam.

shows that after the restarts, both CAWR and CAN perform very similar. This is because the same η_{min} and η_{max} are used for both schedulers, and Figure 3. shows that after epoch 25 both schedulers are very similar.

The hyperparameters used for the baseline system are those defined in 2, because we were motivated by their results. Both schedulers have $\eta_{min} = 0.001$ and $\eta_{max} = 0.05$. The η_{max} was defined to be slightly bigger than that found for the baseline system, so that in the first epochs the loss function can converge faster.

And for Adam we used the model with the best test performance found previously, Table 2. Both schedulers have $\eta_{min} = 0.00001$ and $\eta_{max} = 0.0005$. The η_{max} was defined

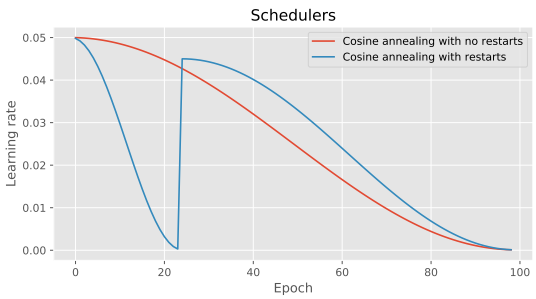

Figure 4. Comparison of CAWR and CAN. For both schedulers, $\eta_{min} = 0.001$ and $\eta_{max} = 0.05$. For CAWR, T_i is 25 epochs and $T_{mult} = 0.9$.

Method	Scheduler	Val./Test Accuracy
SGD	*NA*	84.1% / 82.9%
SGD	CAWR	84.5% / 83.2%
SGD	CAN	84.0% / 83.0%
Adam	*NA*	83.7% / 82.9%
Adam	CAWR	82.2% / 81.9%
Adam	CAN	82.7% / 81.7%

Table 3. Best classification accuracies of EMNIST validation and test set for SGD and Adam for different schedulers. *NA* in schedulers means that the learning rate was constant.

to be slightly bigger than that found in Table 2, so that in the first epochs the loss function can converge faster. As Figure 3 presents, Adam with CAWR and Adam with CAN converge faster than Adam with the a constant learning rate. Adam with CAWR converges slightly faster than Adam with CAN.

Among all the results, the one that converges the fastest is Adam with CAWR, followed by Adam with CAN. However, as we can see in Table 3, there are no subtantial differences on the accuracy.

The different parameters defined in this section for both SGD and Adam are the ones that we will use in the next task.

5. Regularization and weight decay with Adam

(Loshchilov & Hutter, 2017) discusses about the poor performance of Adam when L2 regularization is applied. They hypothesize and proof that there is no L2 in which the regularization factor (λ) and the learning rate (η) are decoupled for adaptive gradient algorithms.

To address this inconvenient, they propose to directly apply a penalty (w) when updating the weights (7), instead of penalizing the loss function.

$$\theta_i(t) = \theta_i(t-1) - \eta(t)(\alpha \frac{\hat{M}_i(t)}{\sqrt{\hat{S}_i(t) + \epsilon}} + w\theta_i(t-1)) \quad (7)$$

where $\eta(t)$ is a scheduler multiplier (can be a scalar or dependent on time), and $\hat{M}_i(t)$ and $\hat{S}_i(t)$ are the bias correction of $M_i(t)$ and $S_i(t)$, respectively:

$$\hat{M}_i(t) = \frac{M_i(t)}{1 - \beta^t} \quad (8)$$

$$\hat{S}_i(t) = \frac{S_i(t)}{1 - \alpha^t} \quad (9)$$

The bias correction is used to fix the bias of $M_i(t)$ and $S_i(t)$ towards to zero for the first epochs.

SCHEDULER	λ	w	VAL./TEST ACCURACY
NA	10^{-4}	-	84.1% / 82.6%
NA	-	$5 * 10^{-5}$	84.1% / 83.3%
CAWR	-	$5 * 10^{-5}$	84.3% / 83.7%
CAN	-	$5 * 10^{-5}$	84.5% / 83.9%

Table 4. Best classification accuracies of EMNIST validation and test set for Adam with L2 and AdamW with different schedulers. *NA* mean that no scheduler was pplied.

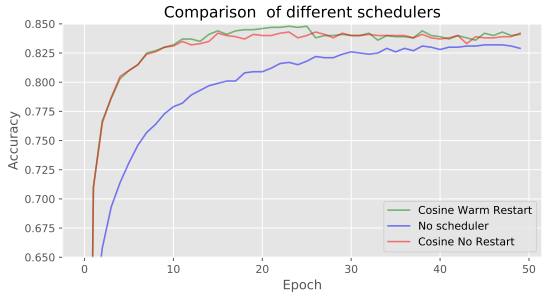

Figure 5. Comparison of different scheduler for AdamW.

In table 4, the results show that Adam with Weight decay (AdamW) performs slightly better than Adam with L2 penalty. This is because the learning rate used in both cases is the one reported in Table 2, and AdamW decouples the penalty of the learning rate, while L2 does not. To achieve better results with L2, we should try different combinations of η and λ, which can become tricky. Different λ and w inspired by (Loshchilov & Hutter, 2017) were tried, but just the best results are reported in Table 4. These results agree with those of (Loshchilov & Hutter, 2017), which show that AdamW decouples η and w, and that AdamW generalizes better than Adam with L2.

Finally, Table 4. also has the best performance results of AdamW with constant learning rate, CAWR and CAN. The best performance is for AdamW with CAN, followed by AdamW with CAWR. As Figure 5 shows, AdamW with CAN and CAWR converge faster than AdamW with constant learning rate. The parameters of both CAWR and CAN are those defined in the previous point.

6. Conclusions

Even though that cosine annealing with warm restarts performed better for any previous experiment, the best model in terms of performance accuracy on the test set is AdamW with cosine annealing without warm restarts, followed by AdamW with cosine annealing with warm restarts. This result may be because the scheduler and the weight decay are coupled in AdamW.

Indeed, we have seen that Adam and RMSProp converge faster than SGD, and that Adam is the fastest one, as (Kingma & Ba, 2014) stated. We also found the best hyperparameters for these models, and used them in the all the other experiments. Beyond this, we have clearly seen that the cosine annealing scheduler with and without warm restarts helps to converge faster for both Adam and SGC.

Finally, AdamW performs better than Adam with L2. By following (Loshchilov & Hutter, 2017), we could decouple the learning rate and the weight decay for adaptive learning algorithms. However, a further question would be if the scheduler could be decoupled from the weight decay.

Another thing that could be explored is to use a reduction factor for η_{min} in (7), so that the firsts η_{min} are close to η_{max} to speed up the learning rate, and then η_{min} goes decreasing to achieve a good accuracy.

References

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.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-24.html.

Hahnloser, Richard H. R., Sarpeshkar, Rahul, Mahowald, Misha A., Douglas, Rodney J., and Seung, H. Sebastian. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–951, 2000.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://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.

Ng, Andrew. Exponentially weighted averages. URL https://www.coursera.org/lecture/deep-neural-network/exponentially-weighted-averages-duStO.

Tieleman, T. and Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.