Report

Machine Learning Practical

Learning rate Schedules and Rules

Author: Georgios Diakidis
University Number: s1673624

THE UNIVERSITY of EDINBURGH

Contents

General Information

In this report I present the experiments I did with various learning algorithms for solving the problem of hand written digit classification. In all my experiments I use a batch size of 50 and train for a total of 100 epochs for all reported runs.

.

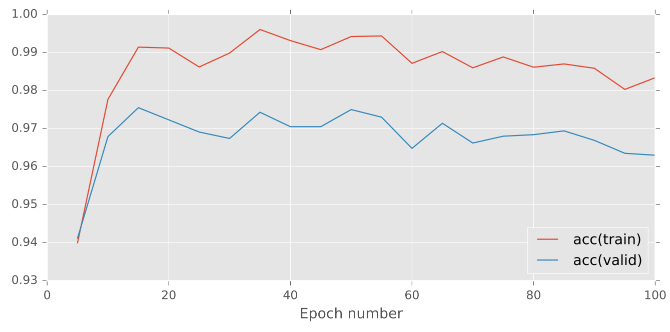
1 Part 1: Learning rate schedules

1.1 Constant learning rate vs Time-dependent learning rate

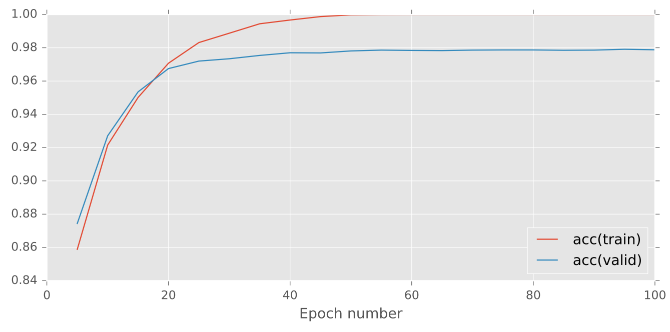
In this experiment we investigated how the accuracy and the error behave according how the learning rate changes in time. We implemented a time-dependent learning rate schedule based on the resiprocal (1) function.

$$\eta(t) = \eta_0(1 + t/r)^-1 \tag{1}$$

The difference between the constant learning rate schedule and the time-dependent learning rate schedule is that while the learning rate on the first keeps increasing as time goes by, on the second one the rate keeps decreasing on a smoother manner. It can also be observed from our plots of the evolution of the error and accuracy across the training epochs (Figures 1,2,3,4), while both rates saturate relatively fast with the same learning rate at the beginning, the constant rate becomes more unstable later on as it does bigger steps back and forth, while the time-dependent rate by doing smaller and smaller steps stabilizes on a certain point for the rest of the training.

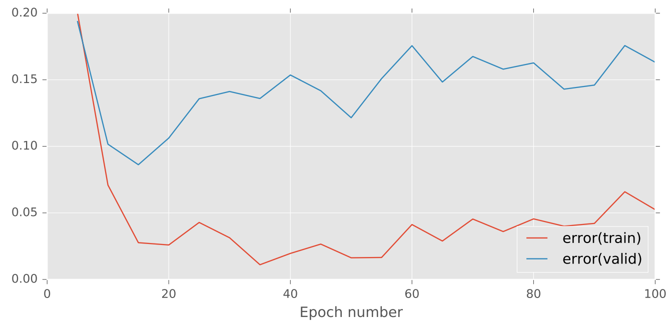


(a) Learning rate 0.1

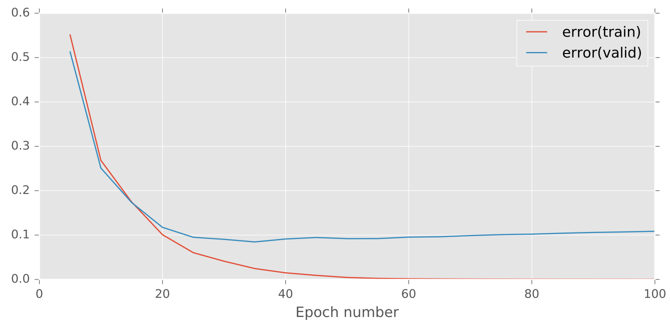


(b) Learning rate 0.01

Figure 1: Accuracy with the constant learning schedule

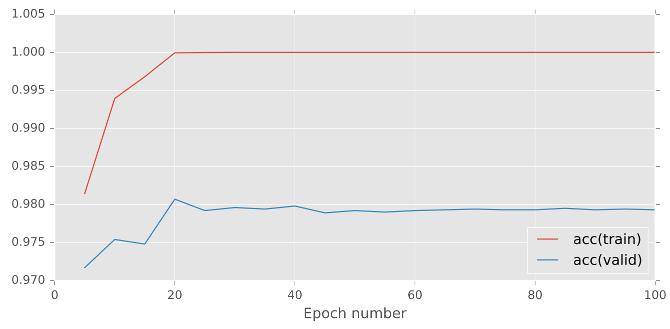


(a) Learning rate 0.1

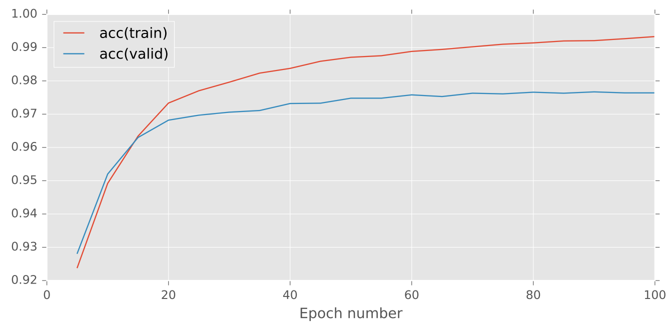


(b) Learning rate 0.01

Figure 2: Errors with the constant learning schedule

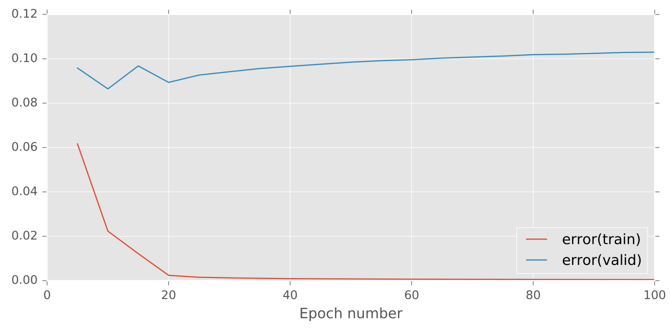


(a) Learning rate = 0.8

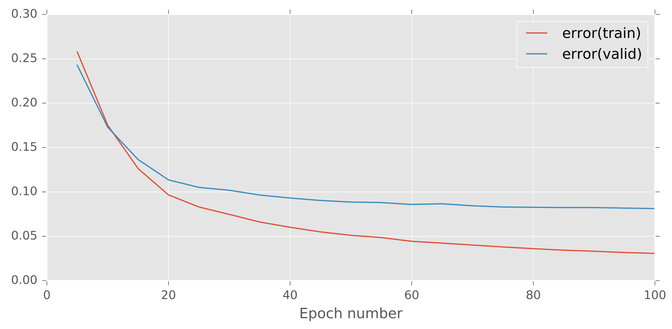


(b) Learning rate = 0.1

Figure 3: Accuracy with the Time-dependent learning schedule and decay constant = 20

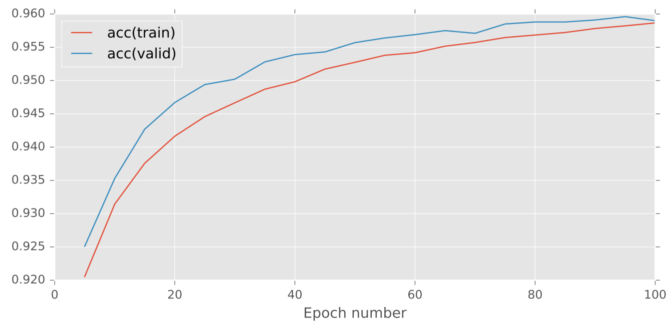


(a) Learning rate = 0.8

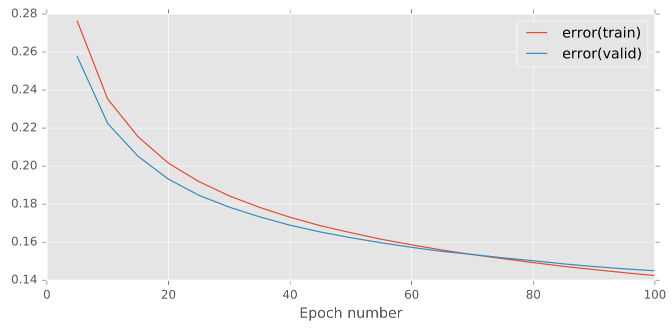


(b) Learning rate = 0.1

Figure 4: Error with the Time-dependent learning schedule and decay constant = 20



(a) Accuracy



(b) Error

Figure 5: Error and Accuracy of the Time learning schedule with learning rate=0.1 and decay constant = 2

1.2 How do the initial learning rate and the decay rate parameters affect the evolution of training

On the time-dependent learning schedule that we implemented the initial learning rate parameter as well as the decay rate have obvious results. The bigger the initial learning rate is, the faster our accuracy will increase, reaching faster at the convergence point. As the learning rate decreases, the accuracy and error saturate on the optimal training point. We need to be careful though, because if the initial learning rate is too big, then the big steps it initially does go way beyond the convergence point and do not have the chance to stabilize in time resulting in really bad weights for our model. On the other hand the decay rate has the result of decreasing our learning rate as time goes by, meaning that while it converges really fast initially, as the steps get smaller the progress we do gets smaller also. That is good if we are already close to reaching our optimal point (minimum error and maximum accuracy) but is bad if we are still far from it and our decreasing learning rate won't help the situation. The experimental results in Figure 5 and in Table 1 show that.

1.3 Final Error and Classification Accuracy

Learning rate	Error(train)	Error(val)	Accuracy(train)	Accuracy(val)
0.1	0.0524	0,163	0.983	0.963
0.01	0.00031	0.108	1.00	0.979

Table 1: Train and validation values for the Constant learning rate schedule

Learning rate	Decay rate	Error(train)	Error(val)	Accuracy(train)	Accuracy(val)
0.1	20	0.0307	0.0812	0.993	0.976
0.1	3	0.142	0.145	0.959	0.959
0.8	20	0.000467	0.103	1.00	0.979

Table 2: Train and validation values for the time-dependent learning rate schedule

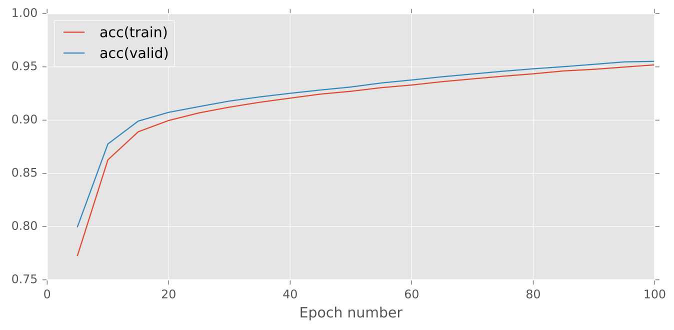
2 Part 2: Momentum learning rule

2.1 Comparison between the basic gradient descent learning rule and the momentum learning rule

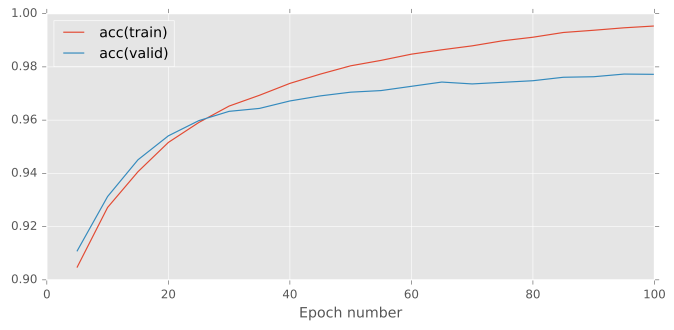
As we can see in Table 3, the momentum learning rule has better final results with higher values in the momentum coefficient than the basic gradient descent rule, both in error and accuracy. If our momentum coefficient becomes too low (0.1), we can see it then yields similar results to the basic gradient descent rule, which is logical. Furthermore, as we can observe in the diagrams below (Figures 6,7), we can see that basic gradient descent has a steeper curve, meaning faster convergence, it then loses out in gains. From that we can deduct that sometimes faster gains may become later slower gains and a smoother transition may be preferable.

2.2 How does momentum coefficient influences training

As we can see in the diagrams below (more clearly in Figures 8,9) that show the accuracy and error curves with respect to epoch numbers, the bigger the momentum coefficient, the smoother the training curve will be. Meaning if our momentum coefficient is low, the convergence rate is higher at least in the beginning untill it starts to saturate.

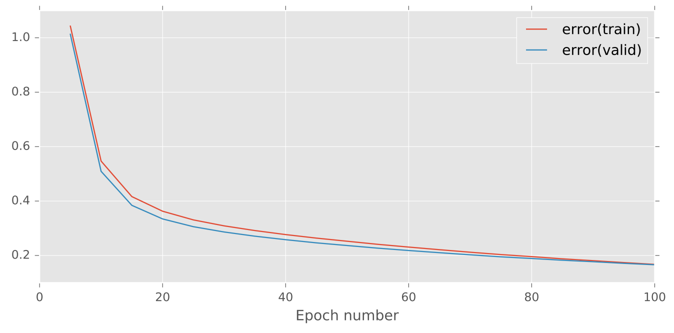


(a) Basic gradient descent rule

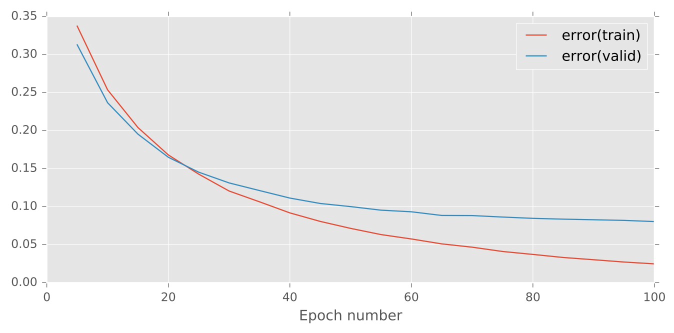


(b) Momentum learning rule

Figure 6: Accuracy of the basic gradient descent and the Momentum learning rules with momentum coefficient = 0.8 and learning rate=0.01

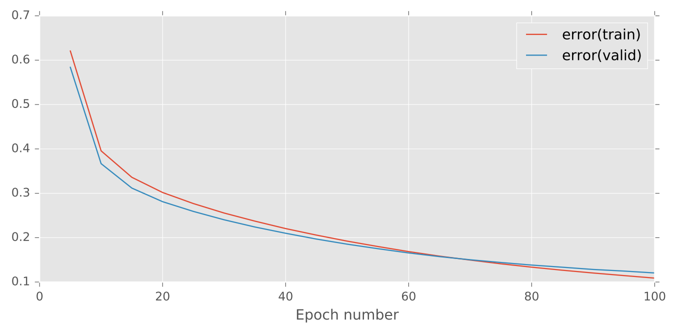


(a) Basic gradient descent learning rule

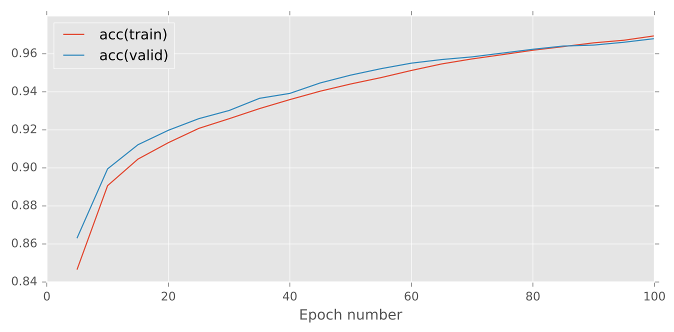


(b) Momentum learning rule

Figure 7: Error of the basic gradient descent and the Momentum learning rules with momentum coefficient = 0.8 and learning rate=0.01



(a) Error

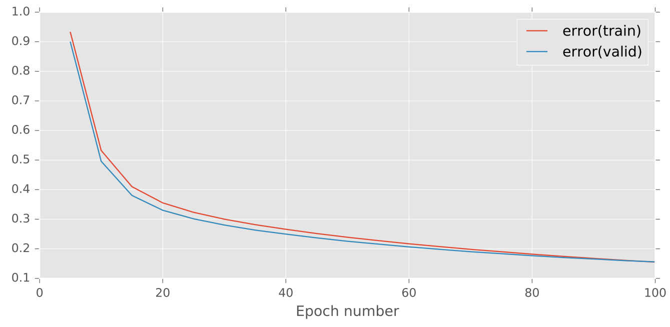


(b) Accuracy

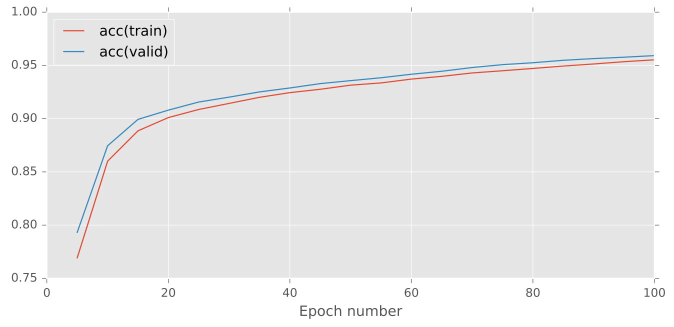
Figure 8: Error and Accuracy of the Momentum learning rules with momentum coefficient = 0.4 and learning rate=0.01

2.3 Gradually increasing momentum coefficient to improve the model

Starting with a low value of the momentum coefficient and gradually increase it as the epochs go by does not seem like an impactfull idea for our purpose. We would like the "speed" created by the momentum



(a) Error



(b) Accuracy

Figure 9: Error and Accuracy of the Momentum learning rules with momentum coefficient = 0.1 and learning rate=0.01

Learning rule	**Momentum Coeff**	**Error(train)**	**Error(val)**	**Accuracy(train)**	**Accuracy(val)**
Gradient descent	-	0.167	0.166	0.952	0.955
Momentum	0.8	0.0248	0.0804	0.995	0.977
Momentum	0.4	0.109	0.12	0.969	0.968
Momentum	0.1	0.155	0.156	0.952	0.955

Table 3: Comparison between basic gradient descent learning rule and Momentum learning rule with different momentum and learning rate = 0.01

coefficient to overcome the initial shallow local minimal and as we go closer at the saturation point to stabilize our position. As a result, by using equation 2 to gradually augment our momentum coefficient does not yield any noteworthy changes for our purpose, at least to my observations. Concerning the parameters τ, γ, they have different affects, which in our situation may not help very much. To explain further, if gamma and tau are close in value, the momentum coefficient rises gradually with time, untill it reaches a_∞, which is the maximum value of our momentum coefficient. If $\tau >> \gamma$, then the momentum coefficient increases a lot faster and yields almost the same results as the regular momentum learning rule (with constant a).

$$a(t) = a_\infty \left(1 - \frac{\gamma}{t+\tau}\right) \tag{2}$$

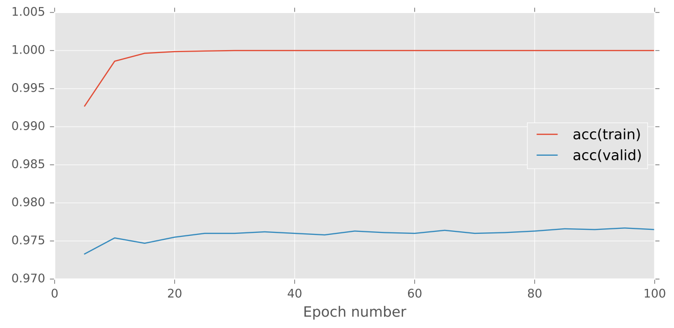
where $a_\infty \in [0, 1]$

3 Part 3: Adaptive learning rules

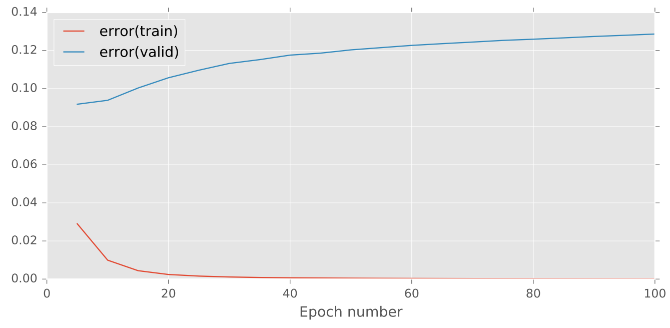
3.1 AdaGrad : The Adaptive Grading learning rule

The AdaGrad learning algorithm updates each weight separately by normalizing with the sum squared gradient. That means that by normalizing, the weights that have high gradients will have their learning rate reduced while others with lower gradient values will have their learning rate increased. By comparing the results of Adagrad with the momentum training rule we observe that with the same learning rate, AdaGrad has a little smoother convergence curve than the momentum training rule, but eventually yields similar results. While basic gradient descent learning rule has a steeper error and accuracy curves, faster convergence initially, eventually AdaGrad enjoys better results, both in accuracy and in error.

In Figure 10 we can see that with learning rate equal to 0.1 AdaGrad converges almost instantly and reaches the saturation point. Moreover, in this if we compare AdaGrad with learning rate 0.1 and 0.01

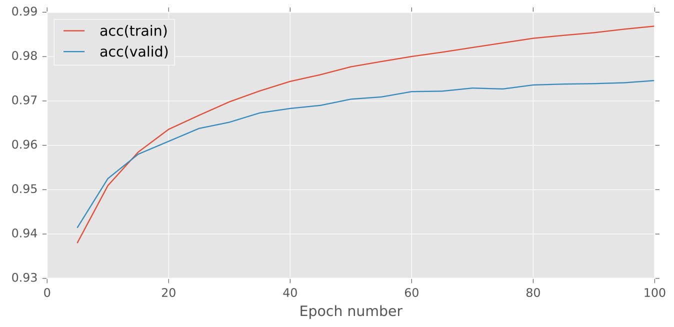


(a) Accuracy

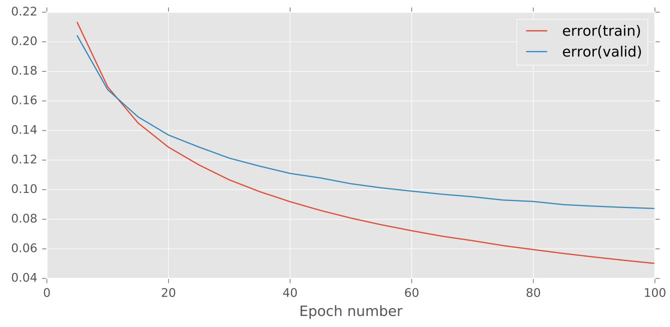


(b) Error

Figure 10: Accuracy and Error of the AdaGrad learning rule with learning rate=0.1



(a) Learning rate = 0.01



(b) Learning rate = 0.01 and Mom coefficient=0.8

Figure 11: Accuracy and Error of the AdaGrad learning rule with learning rate=0.1

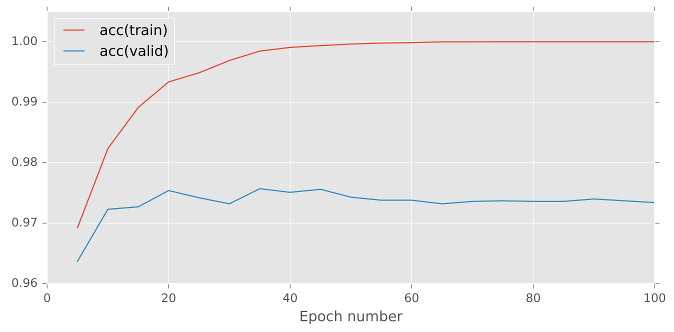
Learning rate	**Error(train)**	**Error(val)**	**Accuracy(train)**	**Accuracy(val)**
0.01	0.0501	0.0873	0.987	0.975
0.1	0.000018	0.129	1.00	0.976

Table 4: Adagrad learning rule's final error and accuracy results with different learning rates

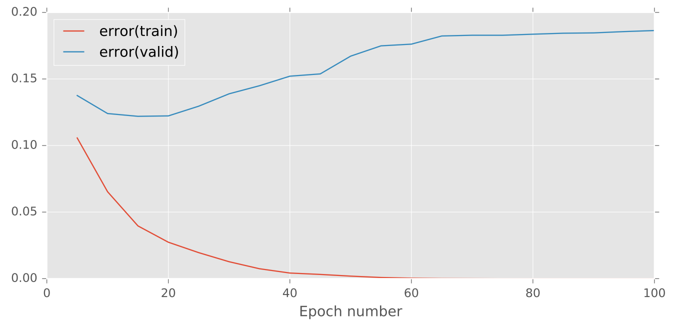
(Table 4), we can see that the first even though converges very faste. it has the same accuracy, but a little higher error, which could play some role in other situations.

3.2 RMSProp

RMSProp, uses a moving average of squared gradients as an attempt to reduce the monotonically decreasing learning rate of adagrad. At first glance RMSProp seems to have an extremely fast convergence rate compared to the previous algorithms. By making the learning rate extremely small (RMSProp's learning rate set to 0.0001) the algorithm seems to converge at least as fast as the previous methods that had a lot larger initial learning rates. By looking at the final error and accuracy results (Table 5) we observe that there weren't many differences. But in the diagrams in Figure 11 we can clearly see the aggressive convergence rate and almost instantaneous saturation and on the other hand the much smoother learning curve in Figure 12. (obviously because of the much smaller learning rate).

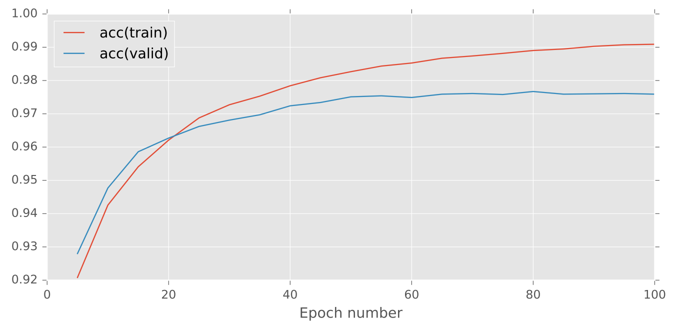


(a) Accuracy

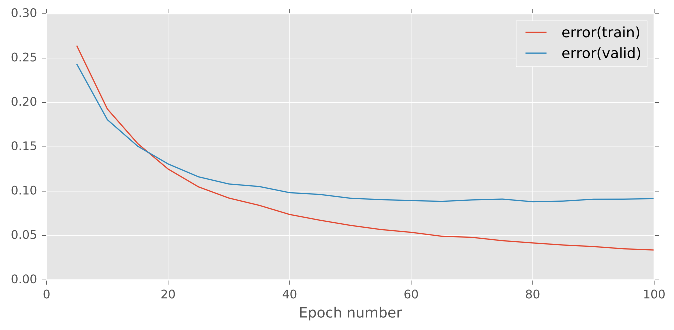


(b) Error

Figure 12: Accuracy and Error of the RMSProp learning rule with learning rate=0.001



(a) Accuracy



(b) Error

Figure 13: Accuracy and Error of the RMSProp learning rule with learning rate=0.0001

Learning rate	**Error(train)**	**Error(val)**	**Accuracy(train)**	**Accuracy(val)**
0.001	9.51e-06	0.186	1.00	0.974
0.0001	0.0337	0.0917	0.991	0.976

Table 5: RMSProp learning rule's final error and accuracy results with different learning rates

By comparing the results of RMSprop with the momentum and the basic gradient descent learning rules we observe that RMSprop has a way more aggressive convergence rate than the other two and as a result reaches the saturation point faster and achieving better results than the basic gradient descent rule, in terms of final accuracy and error, and almost the same results as the momentum learning rule. It is worth to mention that RMSProp has 100% accuracy on the training set, as well as a really really low error value (9.51e-06), something that the previous algorithms may have come close but didn't achieve with the specific learning rates we tried.