MLP Assignment 1

Christopher Sipola (s1667278)

Part 1: Learning rate schedules

Before running any experiments, I created a baseline by running a constant learning rate schedule. I used a learning rate $\eta = 0.05$ (after manually running some pilot tests and observing where the validation error seems to be lowest). As prescribed in the assignment, I set the batch size to 50 and the number of epochs to 100.

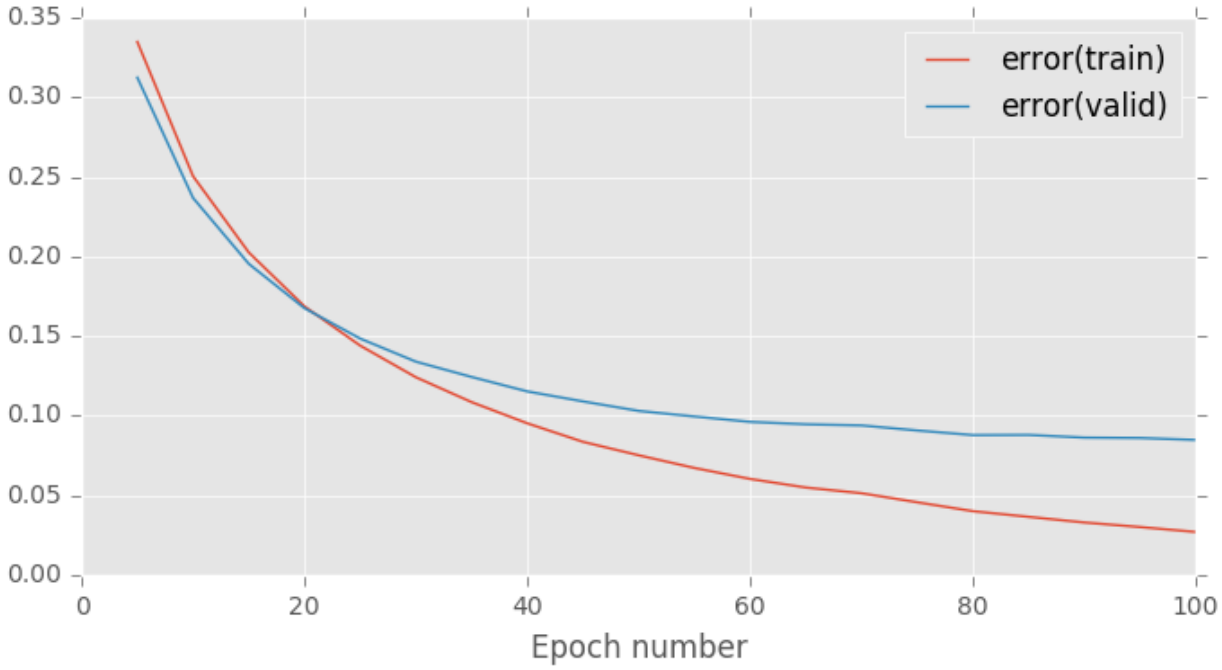
The results of the baseline are as follows:

- Validation error: 0.0848
- Validation accuracy: 0.9753
- Run time: 142

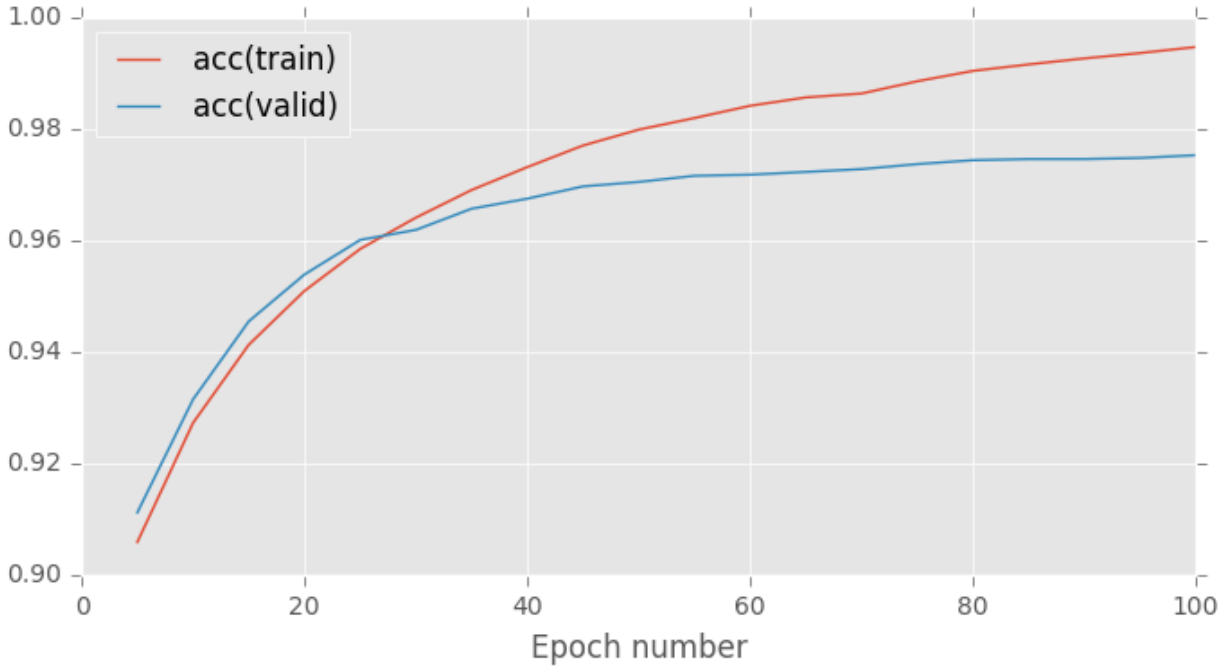
Figure 1 shows the evolution of the error and accuracy across epochs for the baseline.

FIgure 1: Constant learning rate schedule (baseline)

a. Training and validation error



b. Training and validation accuracy



For the first experiment, I used the exponential time-dependent learning rate schedule

$$\eta(t) = \eta_0 \; exp(-t/r)$$

where η_0 is the initial learning rate, t the epoch number, $\eta(t)$ the learning rate at epoch t and r a free parameter governing how quickly the learning rate decays.[1]

I created a grid of various values of η_0 and r for which I tested the error and accuracy of the validation set. For η_0, I used the values 0.067, 0.2 and 0.6, which are (respectively) 1/3, 1 and 3 times the constant learning rate schedule used for the baseline. For r, I tested the values 16, 50 and 150, which are (respectively) 1/3, 1 and 3 times the size of the training set. (In the notes, it was suggested we set r approximately equal to the size of the training set.)

Figure 2 shows the validation error and validation accuracy for this grid of η_0 and r values. The lowest error is achieved at $\eta_0 = 0.2$ and $r = 50$, while the highest accuracy is achieved at $\eta_0 = 0.2$ and $r = 150$.

Figure 2: Exponential learning rate schedule
(best values in bold)

a. Validation error

	$r = 16$	$r = 50$	$r = 150$
$\eta_0 = 0.67$	0.1581	0.0970	0.0854
$\eta_0 = 0.2$	0.0921	**0.0839**	0.0917
$\eta_0 = 0.6$	0.0858	0.1025	0.1104

b. Validation accuracy

	$r = 16$	$r = 50$	$r = 150$
$\eta_0 = 0.67$	0.9563	0.9716	0.9733
$\eta_0 = 0.2$	0.9751	0.9782	**0.9788**
$\eta_0 = 0.6$	0.9769	0.9767	0.9775

[1] This language was taken from the assignment, restated here for clarity.

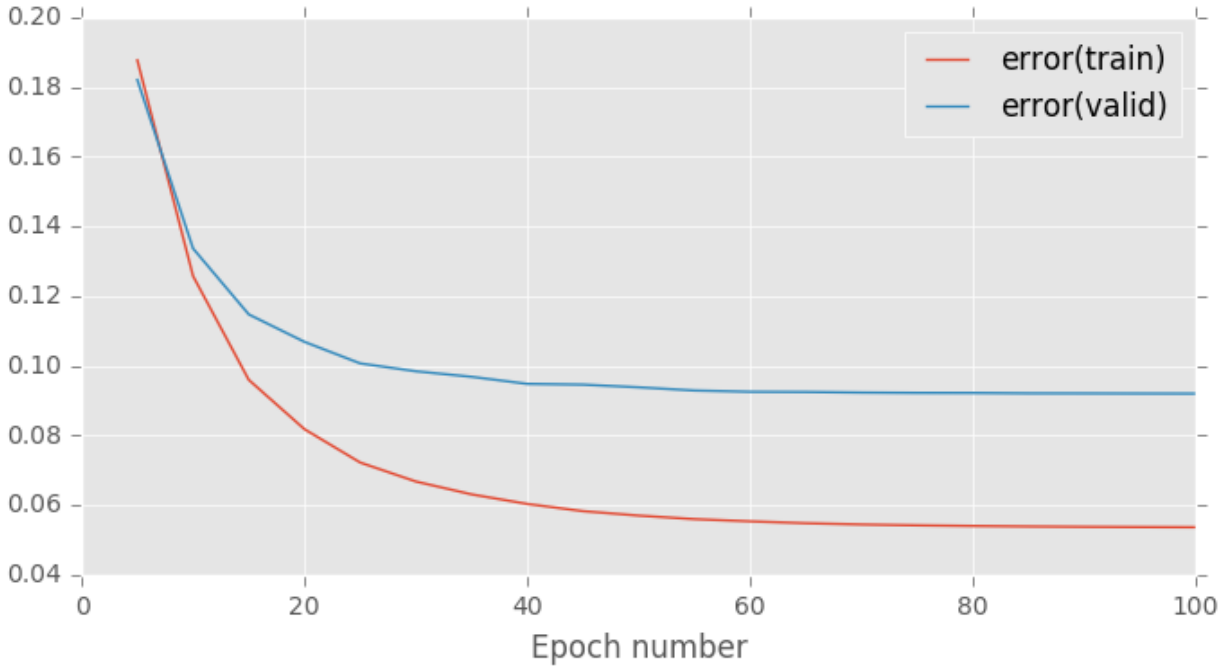
The error for the best exponential learning rate schedule model (0.0839) was lower than that of the the constant learning rate schedule baseline (0.0848), and the validation accuracy (0.9788) was a bit higher than the baseline (0.9753).

Generally, increasing the value of r increases the speed of convergence (and in our case, resulted in some overfitting). See the difference in the the divergence of the training and validation error in figure 3a ($r = 16$) and figure 3b ($r = 150$). I kept the learning rate schedule constant between the two at $\eta_0 = 0.2$.

Similarly, increasing the value of η_0 (figure 4a and figure 4b) gives faster convergence—but with a much greater effect than increasing r by a similar magnitude. In fact, with $\eta_0 = 0.6$ we began overfitting as soon as epoch 15.

Figure 3: Exponential learning rate schedule with varying r

a. Low r value: $\eta_0 = 0.20, r = 16$



b: High r value: $\eta_0 = 0.20, r = 150$

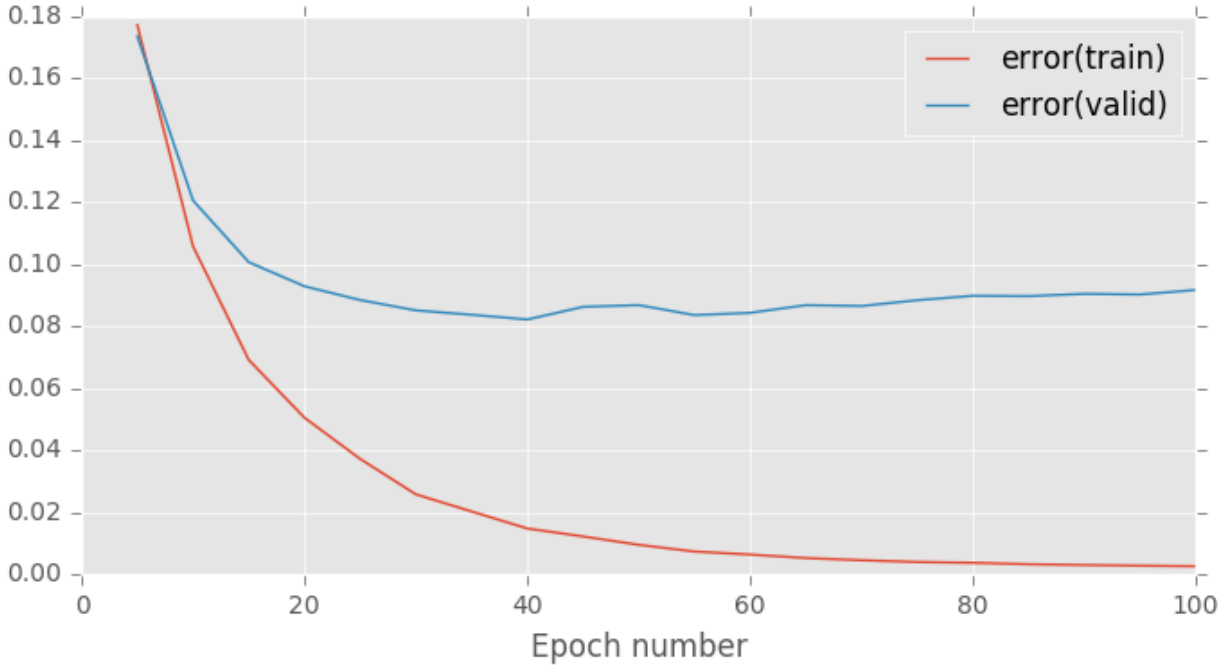
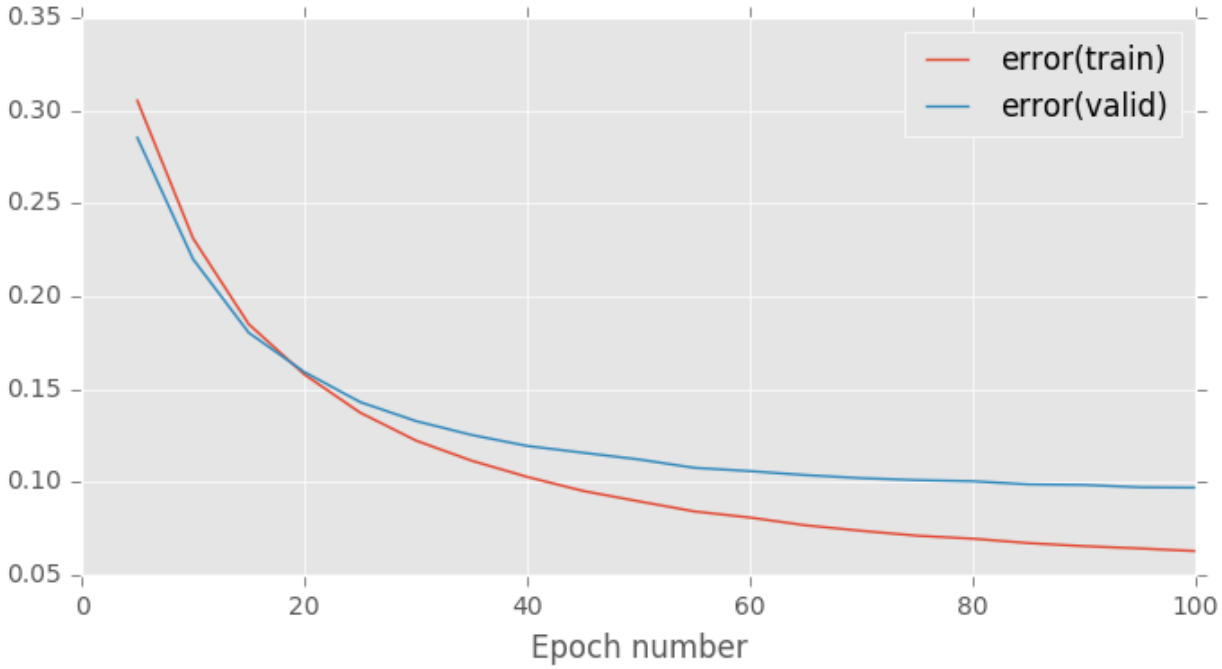
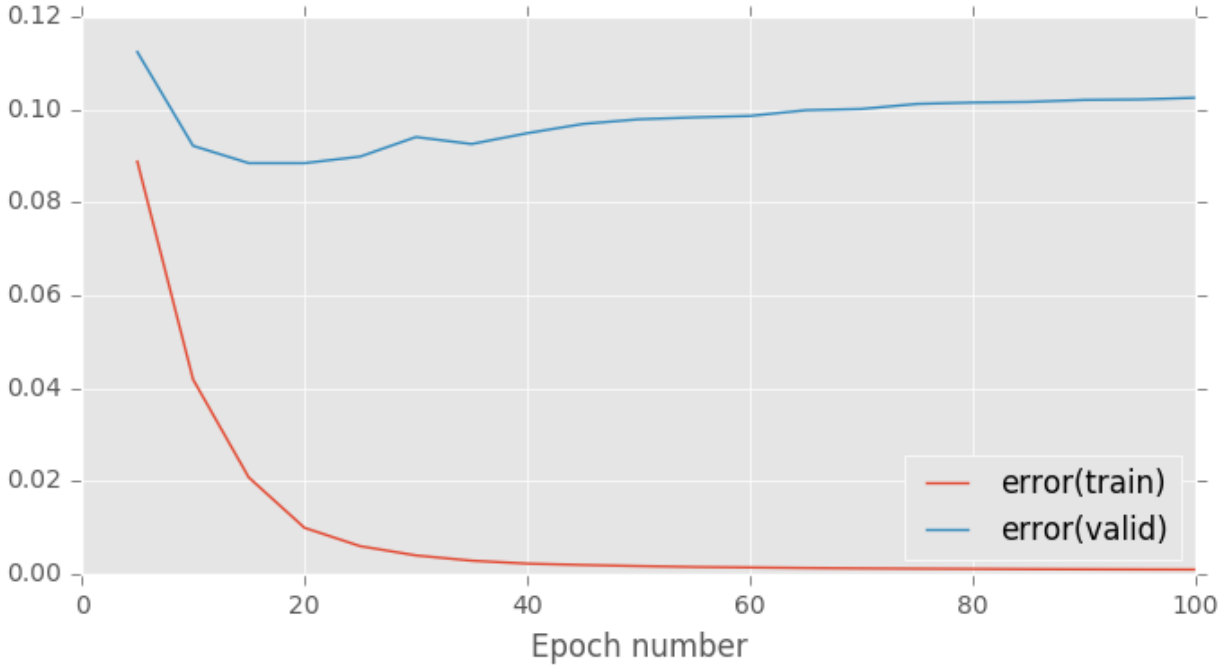


Figure 4: Exponential learning rate schedule with varying learning rate

a. Low learning rate: $\eta_0 = 0.067, r = 50$



b. High learning rate $\eta_0 = 0.6, r = 50$



Part 2: Momentum learning rule

For this part, I vary the momentum coefficient $\alpha \in [0, 1]$ in

$$\Delta w_i(t) = -\eta D_i(t) + \alpha \Delta w_i(t-1)$$

where

$$D_i(t) = \frac{\partial E}{\partial w_i(t)}$$

and w_i is the weight for input x_i and $\Delta w_i(t)$ is the gradient for w_i at time t.

I chose the following values of α: 0, 0.05, 0.75, 0.90, 0.95, 0.99. I chose more values around 0.90 because the lecture notes recommend we choose $\alpha \sim 0.9$.

Values of α around the recommended 0.9 resulted in drastic decreases in training time. To account for this, I used a learning rate of 0.005, which is much smaller than the baseline constant learning rate schedule of 0.2. Without making this adjustment, there was severe instability and overfitting even in early epochs.

Figure 5 shows the the validation error and accuracy for various values of α. The lowest validation error is seen at $\alpha = 0.90$, while the highest accuracy is at $\alpha = 0.99$.

Figure 5: Validation error and accuracy for momentum learning rule (best values for each column in bold)

α	Validation error	Validation accuracy
0.00	0.2348	0.9318
0.05	0.2295	0.9329
0.75	0.1140	0.9680
0.90	**0.0846**	0.9757
0.95	0.0865	0.9776
0.99	0.1033	**0.9781**

Figure 6 and figure 7 show how the error and accuracy (respectively) are affected by increases in α. As stated earlier, increases in α lead to much faster convergence. For $\alpha = 0.00$ (which collapses the learning rate schedule to one without momentum), both the training and validation error have yet to reach a value of 0.2 by epoch 100. In contrast, for $\alpha = 0.90$, a training and validation error of 0.2 is reached at around epoch 15. When we increase α from 0.90 to 0.99, then we get errors of around 0.1 by epoch 5 and then see overfitting beyond epoch 15. The slightly jagged curves for $\alpha = 0.99$ also hint at the beginning of instability and suggest we shouldn't go much higher than 0.99.

We can also vary the momentum according to the schedule

$$\alpha(t) = \alpha_\infty\left(1 - \frac{\gamma}{t+\tau}\right)$$

where $\alpha_\infty \in [0, 1]$ determines the asymptotic momentum coefficient and $\tau \geq 1$ and $0 \geq \gamma \geq \tau$ determine the initial momentum coefficient and how quickly the coefficient tends to α_∞.[2]

For this increasing momentum learning rule, I kept the learning rate η_0 at 0.005 and iterated over every combination of the following values:

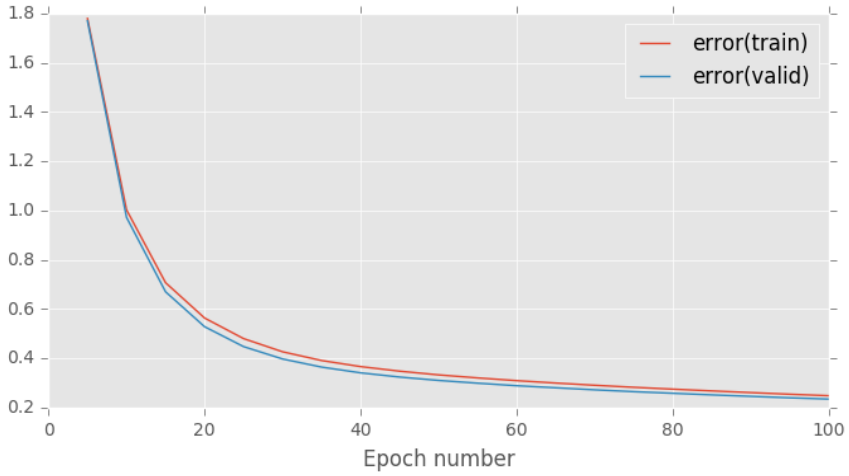
- α_∞: 0.1, 0.5 and 0.9;
- τ: 1, 3 and 9; and
- γ: τ times 0.1, 0.5 and 0.9 (for example, for $\tau = 9$, the values of γ would be $\tau * 0.1 = 0.9$, $\tau * 0.5 = 4.5$ and $\tau * 0.9 = 8.1$).

Figure 8 shows the final validation error and accuracy for these hyperparameter values. The hyperparameter combination $\alpha_\infty = 0.9, \tau = 1, \gamma = 0.1$ produced the lowest error (0.0848), which is comparable to that of the constant momentum learning rule (0.0846). That hyperparameter combination also produced the highest accuracy (0.9753), which is slightly lower than the 0.9781 for the the constant momentum learning rule.

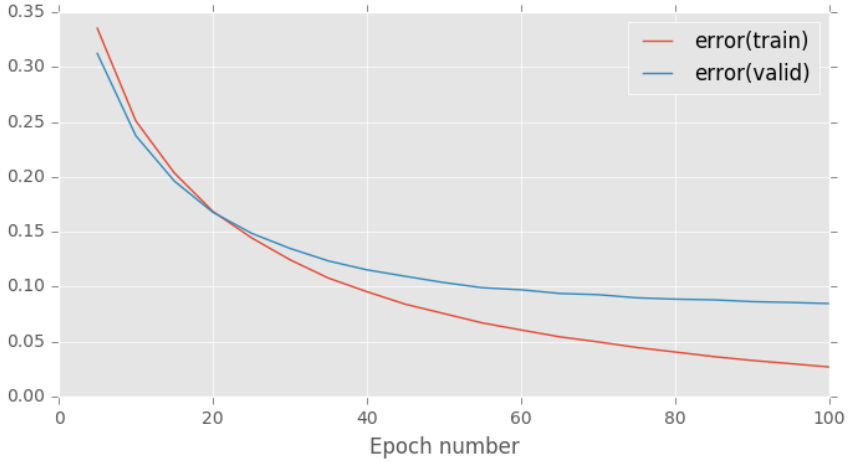
[2] This language was taken from the assignment, restated here for clarity.

Figure 6: Error for increasing values of α

$\alpha = 0.00$



$\alpha = 0.90$



$\alpha = 0.99$

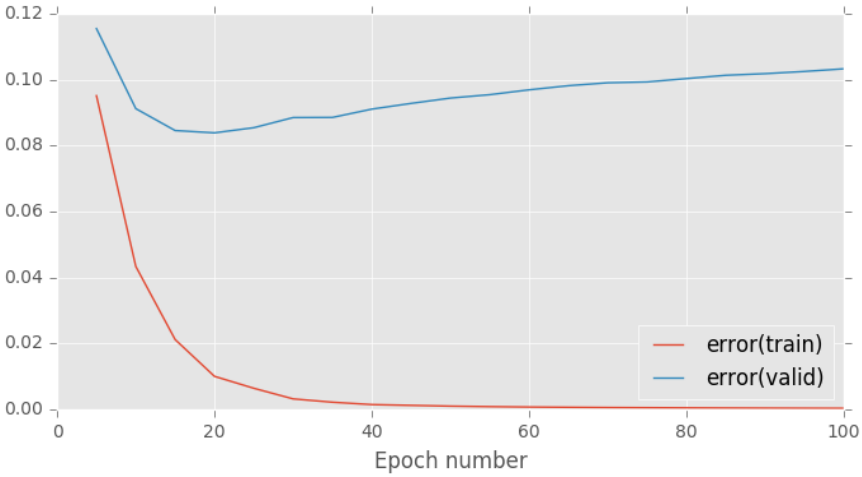
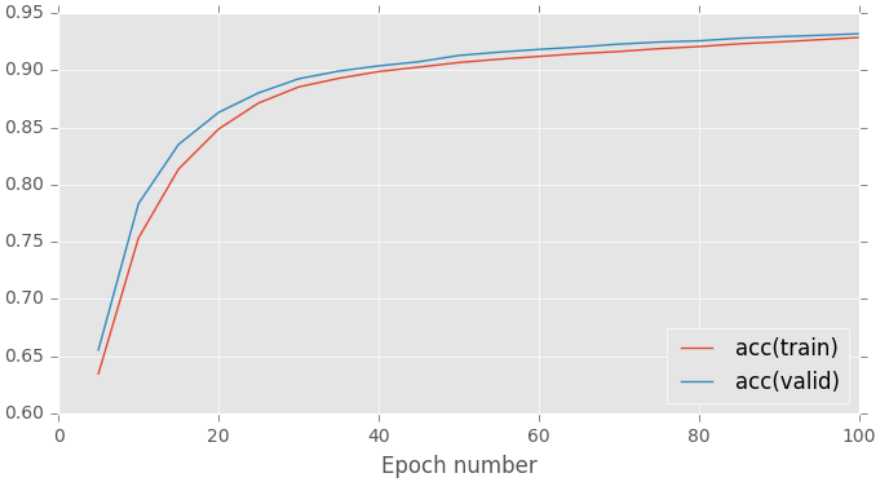
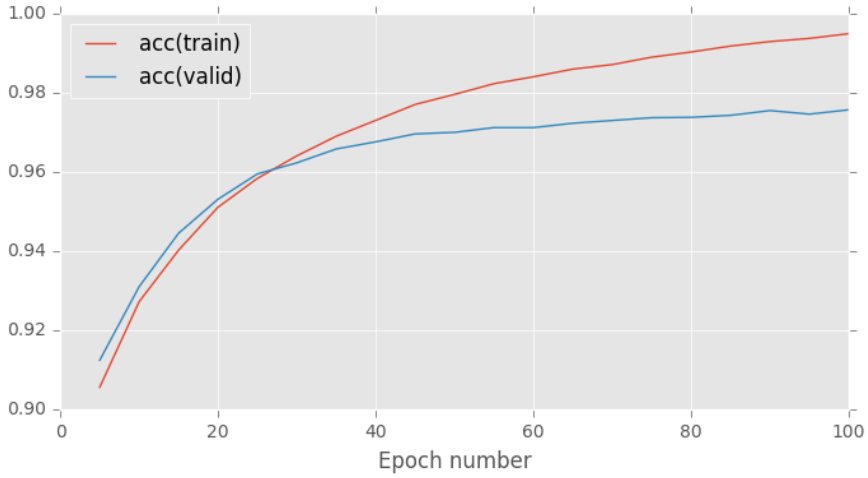


Figure 7: Accuracy for increasing values of α

$\alpha = 0.00$



$\alpha = 0.90$



$\alpha = 0.99$

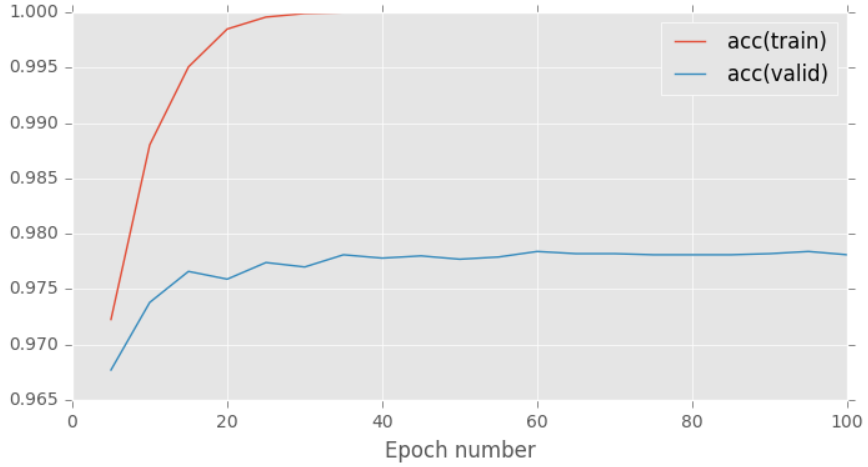


Figure 8: Increasing momentum rate validation error and accuracy (best values in bold)

Hyperparameter values	Validation error	Validation accuracy
$\alpha_\infty = 0.1, \tau = 1, \gamma = 0.1$	0.2240	0.9356
$\alpha_\infty = 0.1, \tau = 1, \gamma = 0.5$	0.2242	0.9354
$\alpha_\infty = 0.1, \tau = 1, \gamma = 0.9$	0.2244	0.9353
$\alpha_\infty = 0.1, \tau = 3, \gamma = 0.3$	0.2240	0.9356
$\alpha_\infty = 0.1, \tau = 3, \gamma = 1.5$	0.2245	0.9351
$\alpha_\infty = 0.1, \tau = 3, \gamma = 2.7$	0.2250	0.9348
$\alpha_\infty = 0.1, \tau = 9, \gamma = 0.9$	0.2242	0.9354
$\alpha_\infty = 0.1, \tau = 9, \gamma = 4.5$	0.2252	0.9348
$\alpha_\infty = 0.1, \tau = 9, \gamma = 8.1$	0.2262	0.9343
$\alpha_\infty = 0.5, \tau = 1, \gamma = 0.1$	0.1666	0.9536
$\alpha_\infty = 0.5, \tau = 1, \gamma = 0.5$	0.1682	0.9533
$\alpha_\infty = 0.5, \tau = 1, \gamma = 0.9$	0.1697	0.9528
$\alpha_\infty = 0.5, \tau = 3, \gamma = 0.3$	0.1671	0.9536
$\alpha_\infty = 0.5, \tau = 3, \gamma = 1.5$	0.1707	0.9528
$\alpha_\infty = 0.5, \tau = 3, \gamma = 2.7$	0.1739	0.9517
$\alpha_\infty = 0.5, \tau = 9, \gamma = 0.9$	0.1681	0.9533
$\alpha_\infty = 0.5, \tau = 9, \gamma = 4.5$	0.1756	0.9510
$\alpha_\infty = 0.5, \tau = 9, \gamma = 8.1$	0.1822	0.9490
$\alpha_\infty = 0.9, \tau = 1, \gamma = 0.1$	**0.0848**	**0.9753**
$\alpha_\infty = 0.9, \tau = 1, \gamma = 0.5$	0.0860	0.9748
$\alpha_\infty = 0.9, \tau = 1, \gamma = 0.9$	0.0876	0.9749
$\alpha_\infty = 0.9, \tau = 3, \gamma = 0.3$	0.0853	0.9751
$\alpha_\infty = 0.9, \tau = 3, \gamma = 1.5$	0.0896	0.9738
$\alpha_\infty = 0.9, \tau = 3, \gamma = 2.7$	0.0946	0.9725
$\alpha_\infty = 0.9, \tau = 9, \gamma = 0.9$	0.0868	0.9745
$\alpha_\infty = 0.9, \tau = 9, \gamma = 4.5$	0.0991	0.9716
$\alpha_\infty = 0.9, \tau = 9, \gamma = 8.1$	0.1120	0.9687

Part 3: Adaptive learning rules

For this part of the assignment, I compared AdaGrad and RMSProp.

For AdaGrad, I tested learning rates $\eta = 0.2 * 3^{-r}$ for r from 0 to 5 (inclusive).[3] That is, I started at the baseline learning rate for the constant learning rate schedule and divided by factors of 3.

Figure 9 shows the results that the learning rate $\eta = 0.0222$ is the clear winner, with a validation error of 0.0842 and a validation accuracy of 0.9796. Note that it could've been possible to tune even further since it seems that the optimal training error is achieved at around epoch 60 (figure 10).

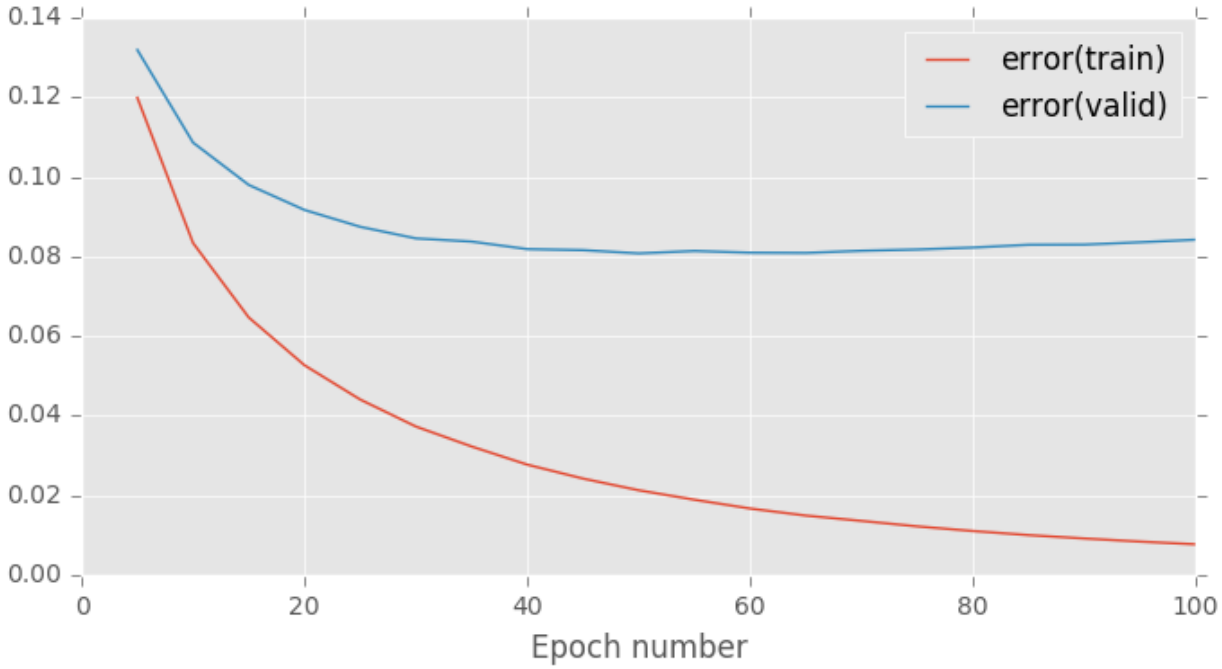
Figure 9: AdaGrad results by learning rate η (best value for each column in bold)

η	Validation error	Validation accuracy	Run time
0.2000	0.1831	0.9712	**199**
0.0667	0.1007	**0.9796**	201
0.0222	**0.0842**	0.9769	233
0.0074	0.1058	0.9687	225
0.0025	0.1843	0.9489	239
0.0008	0.3158	0.9163	230

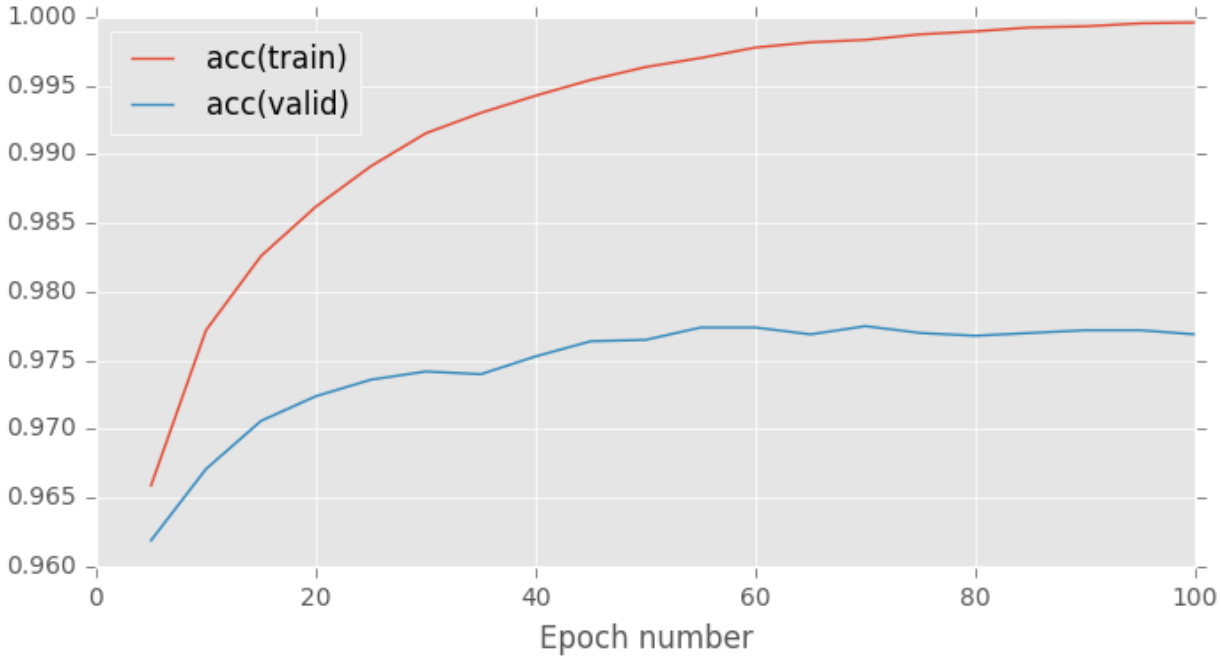
[3] I used this to be consistent with the third lab where 0.2 was used at the learning rate for three affine layers. I choose to multiply and divide by factors of 3 because this was suggested in Dr. Andrew Ng's online machine learning course; it seems to allow you to cover a good amount of the parameter space without skipping over important values.

Figure 10: AdaGrad error and accuracy across epochs for $\eta = 0.0222$

a. Training and validation error



a. Training and validation accuracy



For RMSProp, where we normalize by the moving average

$$S_i(t) = \beta S_i(t-1) + (1-\beta) D_i(t)^2$$

with the decay rate β, I tested the following hyperparameters:

- $\eta = 0.2 * 3^{-r}$ for r from 6 to 8 (inclusive)
- β: 0.85, 0.90, 0.95

Figure 10 shows the results of the experiments for RMSProp. We see the best validation error at $\eta = 0.914 * 10^{-4}$ and $\beta = 0.95$ and the best accuracy at $\eta = 2.740 * 10^{-4}$ and $\beta = 0.95$. Figure 11 shows the evolution of the error and accuracy across epochs.

FIgure 10: RMSProp results for various η and β
(best value for each grid in bold)

a. Validation error

	$\beta = 0.85$	$\beta = 0.90$	$\beta = 0.95$
$\eta = 2.740 * 10^{-4}$	0.1209	0.1293	0.1435
$\eta = 0.914 * 10^{-4}$	0.0977	0.0932	**0.0882**
$\eta = 0.304 * 10^{-4}$	0.1428	0.1391	0.1345

b. Validation accuracy

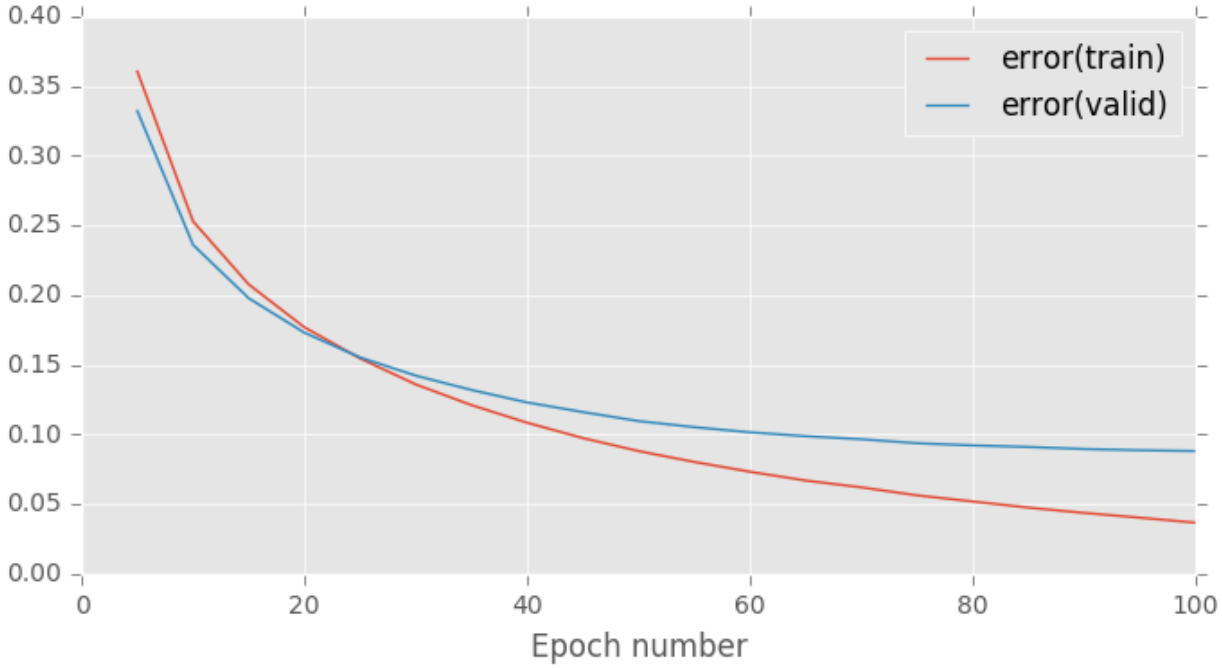
	$\beta = 0.85$	$\beta = 0.90$	$\beta = 0.95$
$\eta = 2.740 * 10^{-4}$	0.9742	0.9744	**0.9751**
$\eta = 0.914 * 10^{-4}$	0.9720	0.9727	0.9733
$\eta = 0.304 * 10^{-4}$	0.9591	0.9611	0.9617

c. Run time

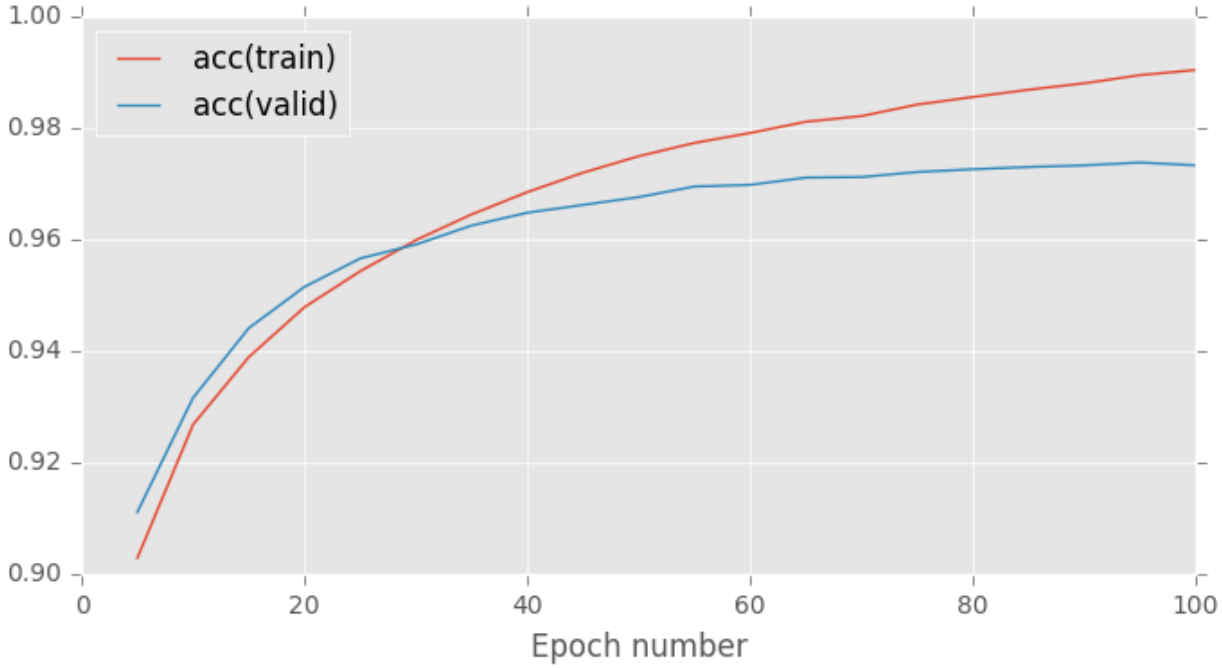
	$\beta = 0.85$	$\beta = 0.90$	$\beta = 0.95$
$\eta = 2.740 * 10^{-4}$	187	188	184
$\eta = 0.914 * 10^{-4}$	207	229	230
$\eta = 0.304 * 10^{-4}$	237	**238**	218

Figure 10: RMSProp error and accuracy across epochs for $\eta = 0.914 * 10^{-4}$ and $\beta = 0.95$

a. Training and validation error



a. Training and validation accuracy



It seems that high β increases validation error when the model is overfitting but decreases it when the learning rate is in a reasonable range (or perhaps underfitting). Although I only tested β in a small range around 0.9, the results are not nearly as sensitive as they are to the learning rate η. (In fact, I had to search quite a bit to find learning rate values that fell between extreme underfitting and overfitting.)

In terms of speed of convergence, RMSProp and AdaGrad have run times that are comparable at a little over 230. However, these two models have a higher run time than the baseline and the other learning rate schedules, which have values around 140. Given that I generally tried to tune the learning rate so that the validation error curve was at a minimum at epoch 100, I did not test whether it would be possible to reach convergence in fewer epochs with one learning rule versus another.

Figure 11 gives a summary of the results for all learning rate schedules, where each uses the hyperparameters that give the lowest validation error. Of all models, the exponential model with the hyperparameters $\eta_0 = 0.2$ and $r = 50$ returned the lowest validation error (as well as the highest accuracy)—with a run time that was not as high as the most computationally expensive learning rate schedules (AdaGrad or RMSProp). However, a denser search for the optimal hyperparameters may return different error and accuracy.

Figure 11: Results of all models

Model	Hyperparameters used	Validation error	Validation accuracy	Run time
Baseline	$\eta = 0.05$	0.0848	0.9753	142
Exponential	$\eta_0 = 0.2$, $r = 50$	0.0839	0.9782	140
Constant momentum coefficient	$\alpha = 0.90$	0.0846	0.9781	134
Increasing momentum coefficient	$\alpha_\infty = 0.9, \tau = 1, \gamma = 0.1$	0.0848	0.9753	136
AdaGrad	$\eta = 0.0222$	0.0842	0.9769	225
RMSProp	$\eta = 0.914 * 10^{-4}$, $\beta = 0.95$	0.0882	0.9751	230