EE6132: Deep Learning for Image Processing

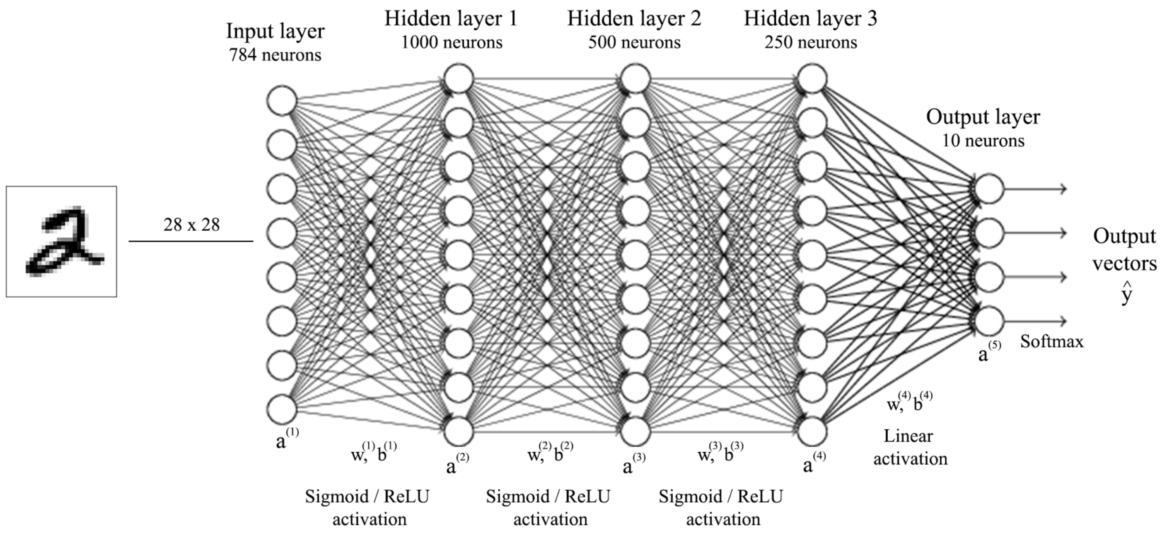
Assignment 1: MNIST Classification using Multilayer Perceptron

Adarsh B (MM14B001)

Contents

1 Overview

Following is a pictorial description of the Multilayer Perceptron model used for training and classifying on MNIST data:



Minibatch size	64
Regularization	L2
Regularization parameter (λ)	0.005
No. of training iterations	8000
Update algorithm	SGD with Momentum acceleration

2 Submissions

2.1 Backpropagation equations

Feedforwarding in the neural network takes place as follows (f can be ReLU or sigmoid):

$$\begin{aligned}
a_i^1 &= x_i \\
z_i^2 &= (w_{ij}^1 a_j^1 + b_i^1) \\
a_i^2 &= f(z_i^2) \\
z_i^3 &= (w_{ij}^2 a_j^2 + b_i^2) \\
a_i^3 &= f(z_i^3) \\
z_i^4 &= (w_{ij}^3 a_j^3 + b_i^3) \\
a_i^4 &= f(z_i^4) \\
z_i^5 &= w_{ij}^4 a_j^4 + b_i^4 \\
a_i^5 &= linear(z_i^2) \\
\hat{y}_i &= softmax(a_i^5)
\end{aligned}$$

For taking momentum into account, we initialize v^i and u^i as the velocity terms for w^i and b^i respectively. δ^i is the error derivative with respect to a^i. α is the learning rate, λ is the regularization parameter, and μ is the momentum parameter. N is the number of samples in the minibatch.

Final layer:
Since we are using cross-entropy as the loss function, the derivative for cross entropy loss can be evaluated as:

$$\begin{aligned}
\frac{\partial E}{\partial z_i^5} &= \frac{\partial E}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial z_i^5} \\
&= -\frac{(y_i - \hat{y}_i)}{\hat{y}_i(1-\hat{y}_i)}\hat{y}_i(1-\hat{y}_i) \\
&= \boldsymbol{(\hat{y}_i - y_i)}
\end{aligned}$$

$$\begin{aligned}
\delta_i^5 &= \frac{\partial E}{\partial z_i^5} \\
&= (\hat{y}_i - y_i) \\
\frac{\partial E}{\partial w_{ij}^4} &= \delta_i^5 \frac{\partial z_i^5}{\partial w_{ij}^4} \\
&= \boldsymbol{(\hat{y}_i - y_i)a_j^4 + \lambda w_{ij}^4} \\
v_{ij}^4 &= \mu v_{ij}^4 - \alpha.\frac{1}{N}\sum^N \frac{\partial E}{\partial w_{ij}^4} \\
w_{ij}^4 &= \boldsymbol{w_{ij}^4 + v_{ij}^4} \\
\frac{\partial E}{\partial b_i^4} &= \frac{\partial E}{\partial z_i^5}\frac{\partial z_i^5}{\partial b_i^4} \\
&= \boldsymbol{(\hat{y}_i - y_i)} \\
u_i^4 &= \mu u_{ij}^4 - \alpha.\frac{1}{N}\sum^N \frac{\partial E}{\partial b_i^4} \\
b_i^4 &= \boldsymbol{b_i^4 + u_i^4}
\end{aligned}$$

Hidden layer 3:

$$\begin{aligned}
\delta_i^4 &= \frac{\partial E}{\partial a_i^4} \\
&= (\delta_j^5 w_{ji}^4) f'(z_i^4) \\
\frac{\partial E}{\partial w_{ij}^3} &= \delta_i^4 \frac{\partial a_i^4}{\partial w_{ij}^3} \\
&= \boldsymbol{((\delta_j^5 w_{ji}^4) f'(z_i^4))a_j^3 + \lambda w_{ij}^3} \\
v_{ij}^3 &= \mu v_{ij}^3 - \alpha.\frac{1}{N}\sum^N \frac{\partial E}{\partial w_{ij}^3} \\
w_{ij}^3 &= \boldsymbol{w_{ij}^3 + v_{ij}^3} \\
\frac{\partial E}{\partial b_i^4} &= \frac{\partial E}{\partial a_i^5}\frac{\partial a_i^5}{\partial b_i^4} \\
&= \boldsymbol{((\delta_j^5 w_{ji}^4) f'(z_i^4))} \\
u_i^3 &= \mu u_{ij}^3 - \alpha.\frac{1}{N}\sum^N \frac{\partial E}{\partial b_i^3} \\
b_i^3 &= \boldsymbol{b_i^3 + u_i^3}
\end{aligned}$$

Hidden layer 2:

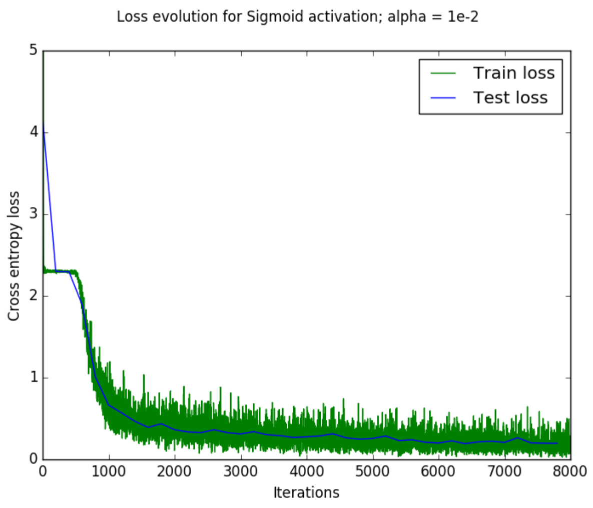
$$\begin{aligned}
\delta_i^3 &= \frac{\partial E}{\partial a_i^3} \\
&= (\delta_j^4 w_{ji}^3) f'(z_i^3) \\
\frac{\partial E}{\partial w_{ij}^2} &= \delta_i^3 \frac{\partial a_i^3}{\partial w_{ij}^2} \\
&= \boldsymbol{((\delta_j^4 w_{ji}^3) f'(z_i^3)) a_j^2 + \lambda w_{ij}^2} \\
v_{ij}^2 &= \mu v_{ij}^2 - \alpha . \frac{1}{N} \sum^N \frac{\partial E}{\partial w_{ij}^2} \\
w_{ij}^2 &= \boldsymbol{w_{ij}^2 + v_{ij}^2} \\
\frac{\partial E}{\partial b_i^3} &= \frac{\partial E}{\partial a_i^4} \frac{\partial a_i^4}{\partial b_i^3} \\
&= \boldsymbol{((\delta_j^4 w_{ji}^3) f'(z_i^3))} \\
u_i^2 &= \mu u_{ij}^2 - \alpha . \frac{1}{N} \sum^N \frac{\partial E}{\partial b_i^2} \\
b_i^2 &= \boldsymbol{b_i^2 + u_i^2}
\end{aligned}$$

Hidden layer 1:

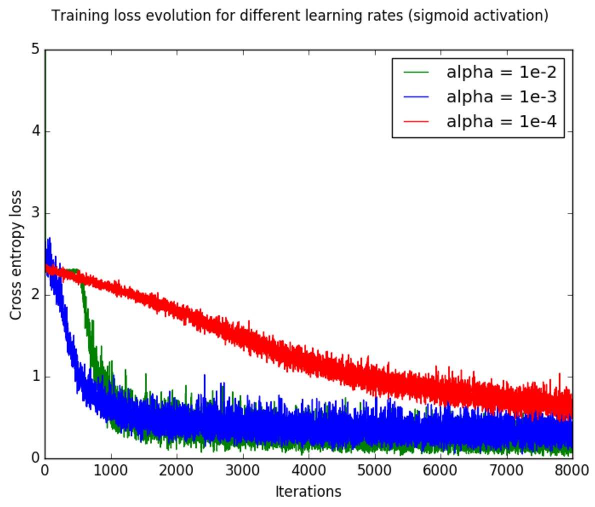
$$\begin{aligned}
\delta_i^2 &= \frac{\partial E}{\partial a_i^2} \\
&= (\delta_j^3 w_{ji}^2) f'(z_i^2) \\
\frac{\partial E}{\partial w_{ij}^1} &= \delta_i^2 \frac{\partial a_i^2}{\partial w_{ij}^1} \\
&= \boldsymbol{((\delta_j^3 w_{ji}^2) f'(z_i^2)) a_j^1 + \lambda w_{ij}^1} \\
v_{ij}^1 &= \mu v_{ij}^1 - \alpha . \frac{1}{N} \sum^N \frac{\partial E}{\partial w_{ij}^1} \\
w_{ij}^1 &= \boldsymbol{w_{ij}^1 + v_{ij}^1} \\
\frac{\partial E}{\partial b_i^2} &= \frac{\partial E}{\partial a_i^3} \frac{\partial a_i^3}{\partial b_i^2} \\
&= \boldsymbol{((\delta_j^3 w_{ji}^2) f'(z_i^2))} \\
u_i^1 &= \mu u_{ij}^1 - \alpha . \frac{1}{N} \sum^N \frac{\partial E}{\partial b_i^1} \\
b_i^1 &= \boldsymbol{b_i^1 + u_i^1}
\end{aligned}$$

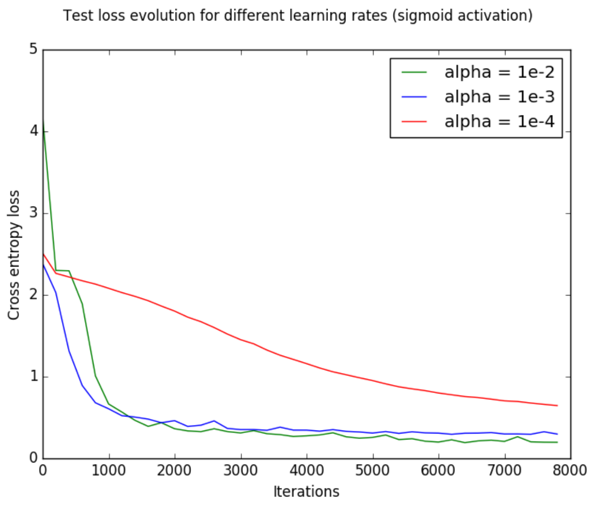
2.2 Learning curve plots

Following is the plot of train and test loss vs iterations for a network with Sigmoid activations:



Following plots show the comparison of loss evolution between different learning rates:



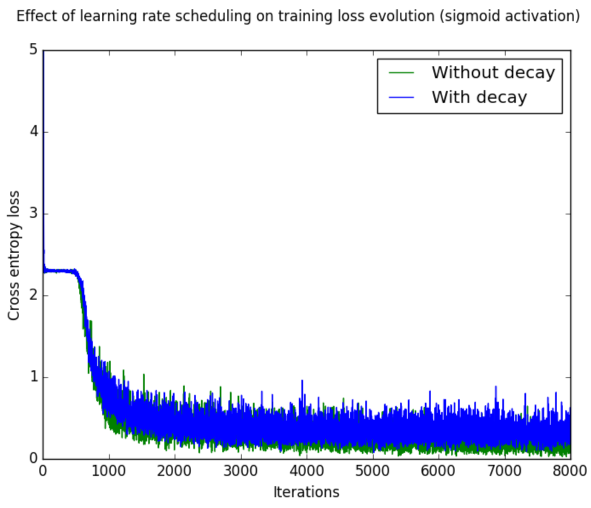


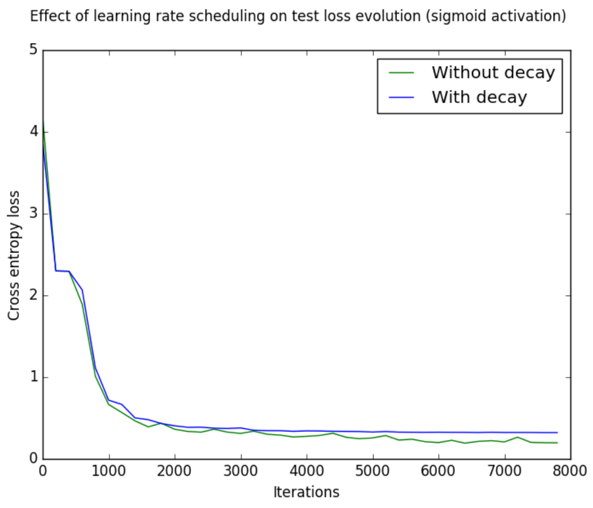
The test accuracies for various cases is tabulated as shown. Clearly, it can be observed that convergence is faster in case of higher learning rates.

alpha	**Accuracy**
1e-2	94.89%
1e-3	90.49%
1e-4	84.36%

2.3 Learning rate scheduling

Learning rate has been decayed by a factor of 0.85 for every 250 iterations. The comparison between scheduled and unscheduled learning rates is turned in below:





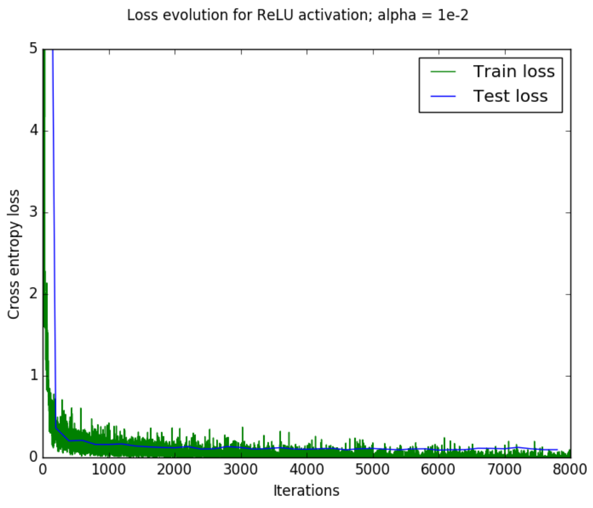
The accuracies obtained in both the cases are tabulated bwlow:

Experiment	Accuracy
Constant learning rate	94.89%
Scheduled learning rate	90.60%

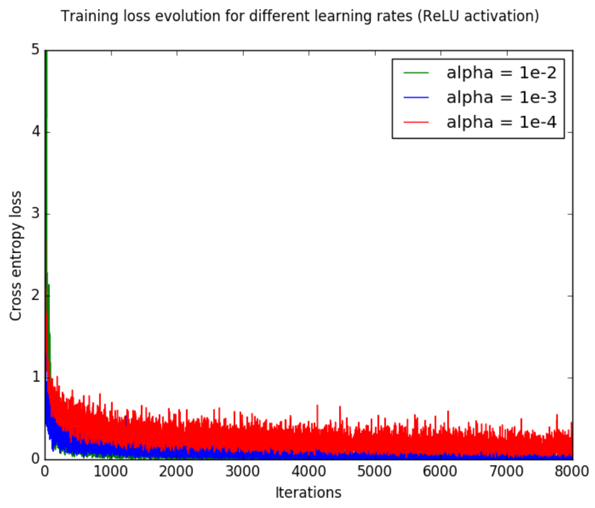
Despite having a lower accuracy in case of scheduled learning rate, the convergence is smooth. With a high fixed learning rate, the system can be thought of having too much **kinetic energy** and the parameters oscillate around rapidly, unable to settle down into deeper, but narrower parts of the loss function. This can be seen in the unsteady green line from the test loss plot. Sometimes, the loss function sometimes can even get stagnated at a local minima. But with a decaying learning rate, **deeper** along with **smoother convergence** over long iterations helps the network to learn better.

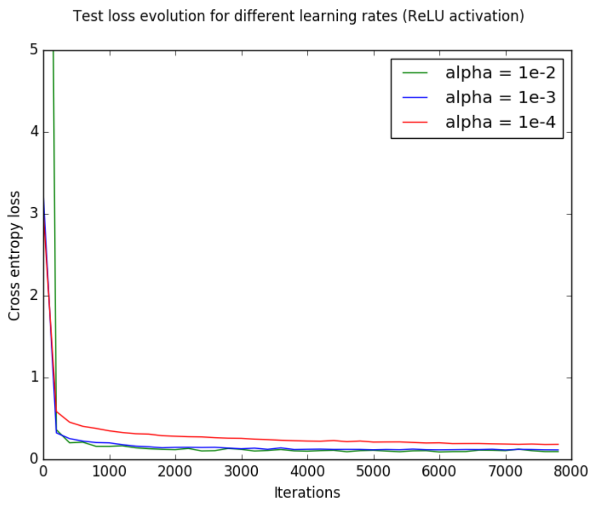
2.4 Experimenting with ReLU activation function

Following is the plot of train and test loss vs iterations for a network with ReLU activations:



Following plots show the comparison of loss evolution between different learning rates:



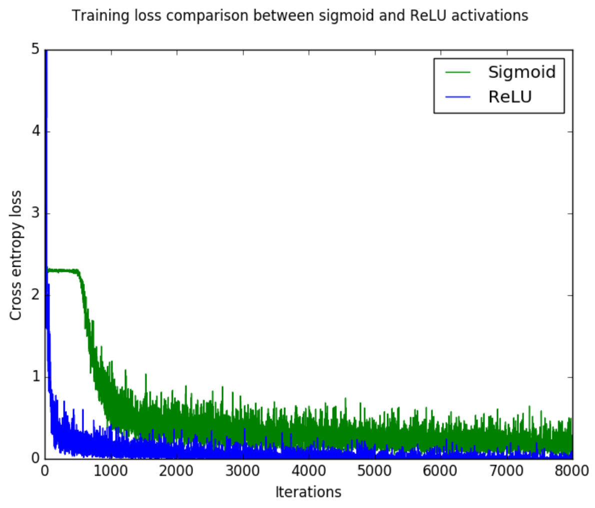


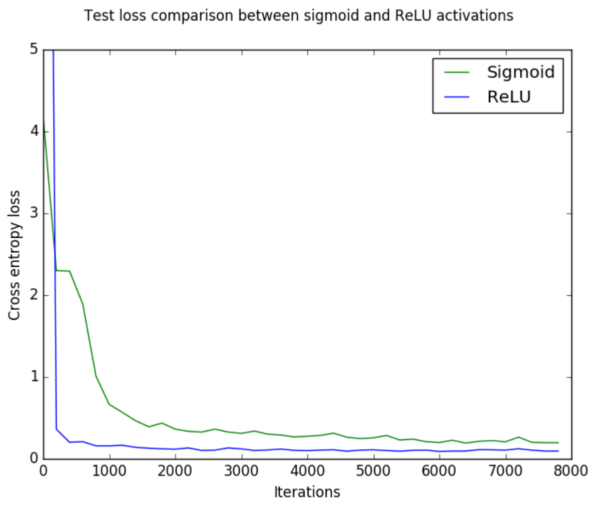
The test accuracies for various cases is tabulated as shown. Again, it can be observed that convergence is faster in case of higher learning rates.

alpha	**Accuracy**
1e-2	97.79%
1e-3	96.87%
1e-4	94.28%

Comparison between ReLU and Sigmoid:

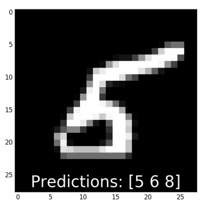
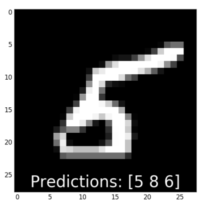
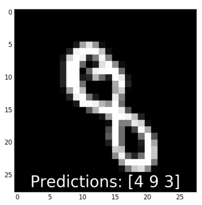
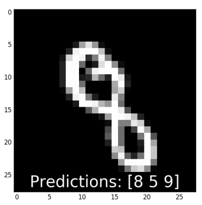
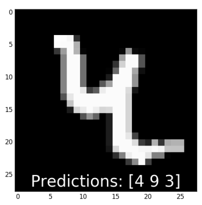
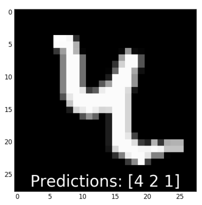
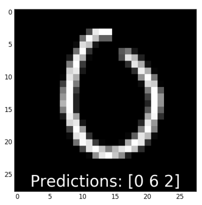
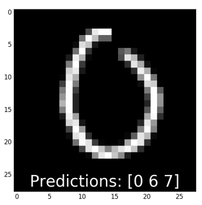
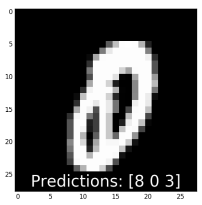
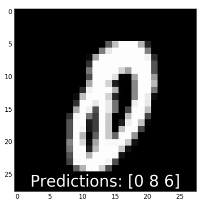
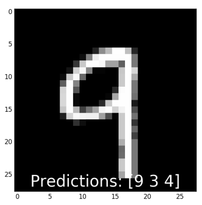
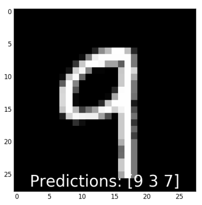
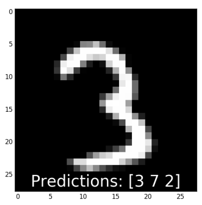
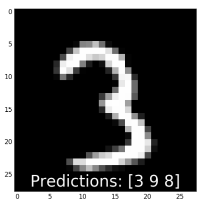
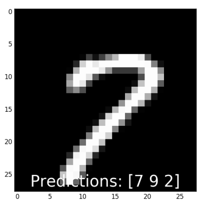
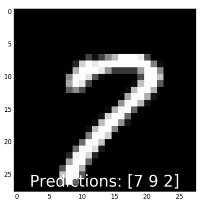
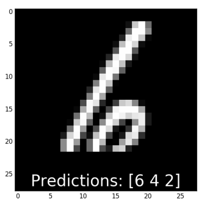
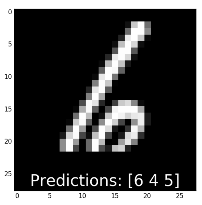
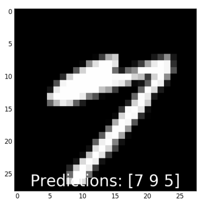
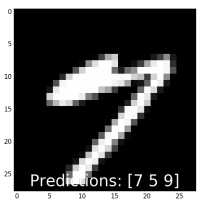
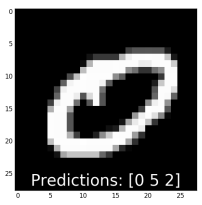
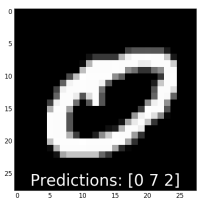
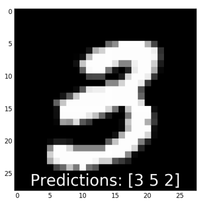
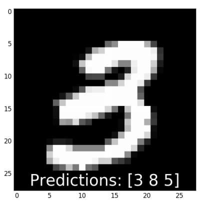
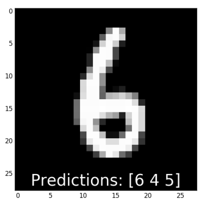
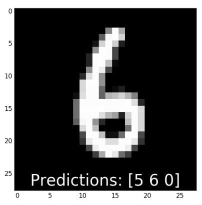
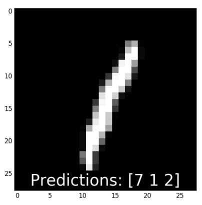
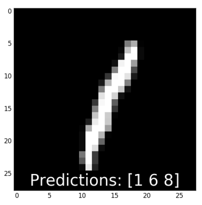
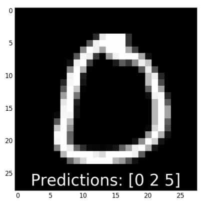
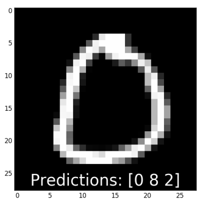
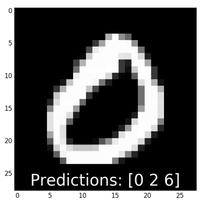
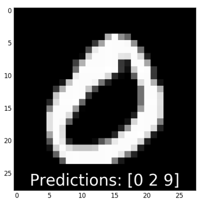
Following plots depict the convergence comparison between ReLU and sigmoid activations for alpha=1e-2.

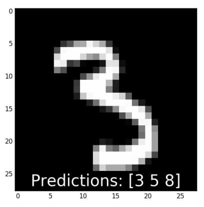
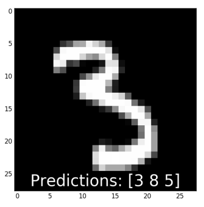
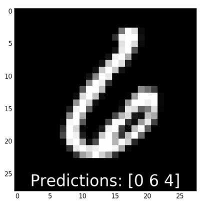
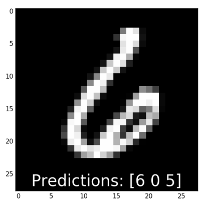
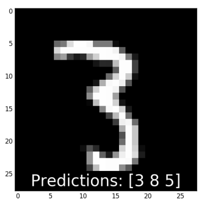
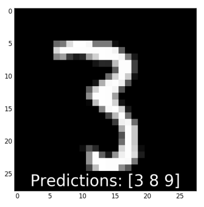
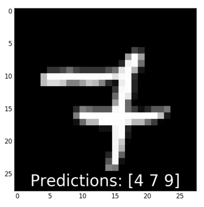
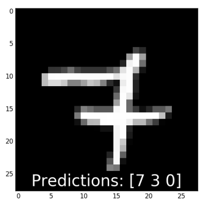




Clearly, ReLU converges faster as compared to sigmoid activation. Test accuracies are also higher for ReLU than sigmoid (**97.79%** for ReLU and **94.89%** for Sigmoid).

2.5 Sample predictions

Sigmoid	ReLU	Sigmoid	ReLU
 Predictions: [5 6 8]	 Predictions: [5 8 6]	 Predictions: [4 9 3]	 Predictions: [8 5 9]
 Predictions: [4 9 3]	 Predictions: [4 2 1]	 Predictions: [0 6 2]	 Predictions: [0 6 7]
 Predictions: [8 0 3]	 Predictions: [0 8 6]	 Predictions: [9 3 4]	 Predictions: [9 3 7]
 Predictions: [3 7 2]	 Predictions: [3 9 8]	 Predictions: [7 9 2]	 Predictions: [7 9 2]
 Predictions: [6 4 2]	 Predictions: [6 4 5]	 Predictions: [7 9 5]	 Predictions: [7 5 9]
 Predictions: [0 5 2]	 Predictions: [0 7 2]	 Predictions: [3 5 2]	 Predictions: [3 8 5]
 Predictions: [6 4 5]	 Predictions: [5 6 0]	 Predictions: [7 1 2]	 Predictions: [1 6 8]
 Predictions: [0 2 5]	 Predictions: [0 8 2]	 Predictions: [0 2 6]	 Predictions: [0 2 9]

Sigmoid	ReLU	Sigmoid	ReLU
			
			

As can be seen from the above predictions, network with ReLU activation could predict **19 out of 20 images** correctly, and network with sigmoid activation could predict **17 out of 20** predictions correctly.