ELL784 : Assignment-1 Report

Raghav Mittal (2014MT10600)

September 19, 2016

1 Basic Details

In this assignment I have designed a softmax classifier to train a neural network. There are 26 classes for the output among which only one is correct. I have used softmax function as the activation function on the output layer and ReLu function as activation on the hidden layer.

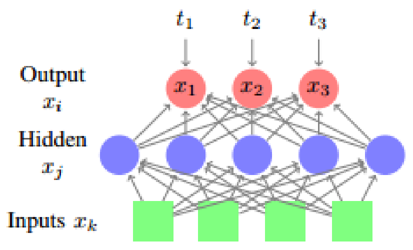


Figure 1: Sample Neural Network

The input layer contains 17 neurons, one for each of the 16 features provided and the 17th is for the bias for the frst layer. The hidden layer can contain any number of neurons. I have trained the model using 80 hidden neurons and one bias value. In the last layer there are 26 neurons, one for each class.

2 Mathematics Used Behind the Code

- The Cross Entropy cost function used for calculating error is

$$J = -\sum_{i=1}^{K} t_i log(x_i)$$

Here 'K' denotes the no. of classes

- The ReLu Activation function used in the hidden layer is

$$\text{F(x)=} \begin{cases} 0 & x \leq 0 \\ x & 0 \leq x \end{cases}$$

- The Softmax Function which is also the activation function for output layer is

$$\sigma((z)_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Here z_j is the probabilty of class 'j' for it to be the output of the neural network.Now let us derive the back propagation used for this training model

$$
\begin{aligned}
\frac{\partial E}{\partial x_i} &= -\frac{t_i}{x_i} \\
\frac{\partial x_i}{\partial s_k} &= \begin{cases} \frac{e^{s_i}}{\sum_c^{nclass} e^{s_c}} - \left(\frac{e^{s_i}}{\sum_c^{nclass} e^{s_c}}\right)^2 & i = k \\ -\frac{e^{s_i} e^{s_k}}{(\sum_c^{nclass} e^{s_c})^2} & i \neq k \end{cases} \\
&= \begin{cases} x_i(1 - x_i) & i = k \\ -x_i x_k & i \neq k \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial s_i} &= \sum_k^{nclass} \frac{\partial E}{\partial x_k} \frac{\partial x_k}{\partial s_i} \\
&= \frac{\partial E}{\partial x_i} \frac{\partial x_i}{\partial s_i} - \sum_{k \neq i} \frac{\partial E}{\partial x_k} \frac{\partial x_k}{\partial s_i} \\
&= -t_i(1 - x_i) + \sum_{k \neq i} t_k x_k \\
&= -t_i + x_i \sum_k t_k \\
&= x_i - t_i
\end{aligned}
$$

the gradient for weights in the top layer is thus

$$
\begin{aligned}
\frac{\partial E}{\partial w_{ji}} &= \sum_i \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial w_{ji}} \\
&= (x_i - t_i) x_j
\end{aligned}
$$

and for units in the second lowest layer indexed by j,

$$
\begin{aligned}
\frac{\partial E}{\partial s_j} &= \sum_i^{nclass} \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial x_j} \frac{\partial x_j}{\partial s_j} \\
&= \sum_i^{nclass} (x_i - t_i)(w_{ji})(x_j(1 - x_j))
\end{aligned}
$$

3 Training Method Used

I have used the method of cross-validation for assesing the results obtained after every epoch. The training set has been divided into 2 parts , around

88% of data has been used as the training set and remaining dataset is used as a cross validation set. The model is trained on the training set extracted and the cost is calculated using cross entropy function. After every epoch the error and also validation test and training accuracy is calculated using the same cost function. A parameter maxtestAccuracy is kept which maintains the maximum accuracy obtained till the epoch currently running.

It is verified that if the test accuracy is more than max validation test accuracy till now then only the next iteration for epoch starts else it stops there and then the result for test dataset is calculated.The parameter for maximum validation test accuracy is updated after every epoch.

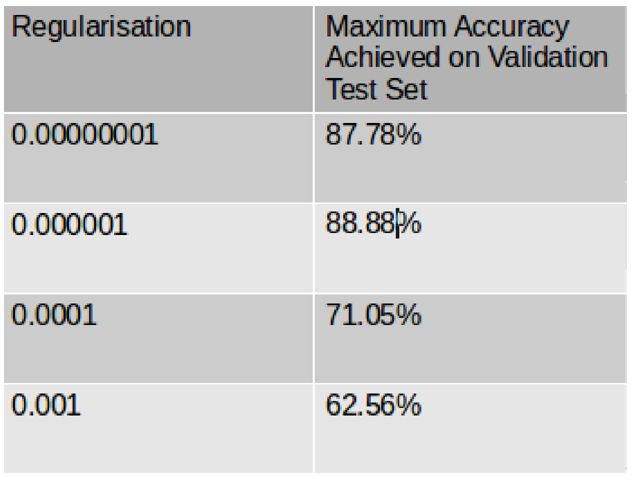
4 Optimisation Method Used

Here I have used Stochastic Gradient Descent method for training the neural network.

$$\mathrm{J}(\theta) = -\sum_{i=1}^{n} J_i(\theta)$$

Here θ denotes the weights between two layers which is the parameter of the cost function.

Stochastic Gradient Descent follows the property that the gradient of cost function is calculated for each example, weights are updated and then this process is repeated on every example in the dataset. After all the training set examples are visited once and weights are updated, new weights are used to estimate the results of the test dataset.



Regularisation	Maximum Accuracy Achieved on Validation Test Set
0.00000001	87.78%
0.000001	88.88%
0.0001	71.05%
0.001	62.56%

Learning Rate	Maximum Accuracy Achieved on Validation Test Set
0.002	82.05%
0.004	88.88%
0.001	84.22%

5 Graphical Analysis

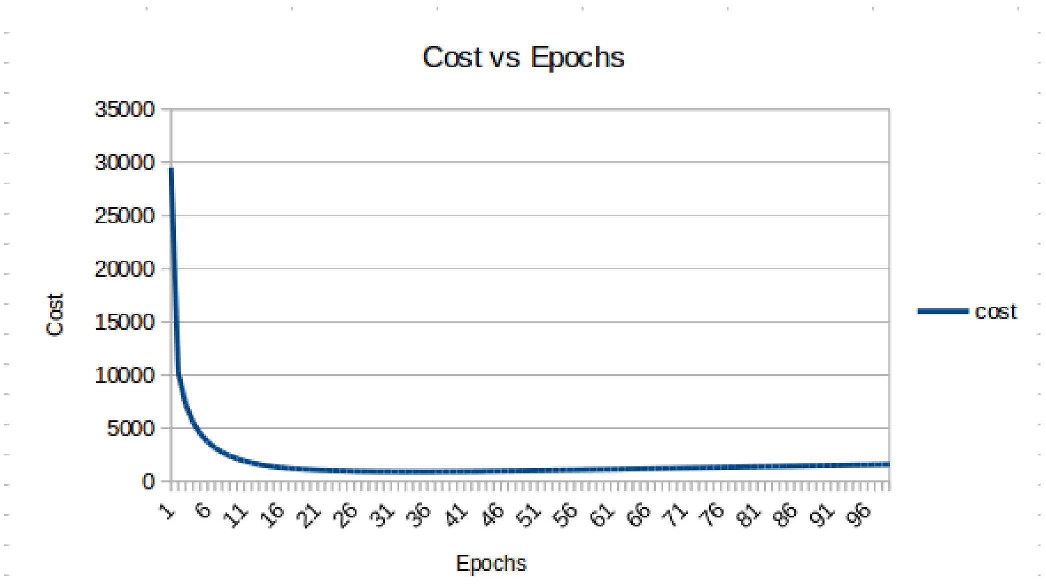
Here we will observe that how different parameters can change the value of cost and accuracy with increasing number of epochs. Parameters that we will see here are learning rate and the regularisation parameter lambda.

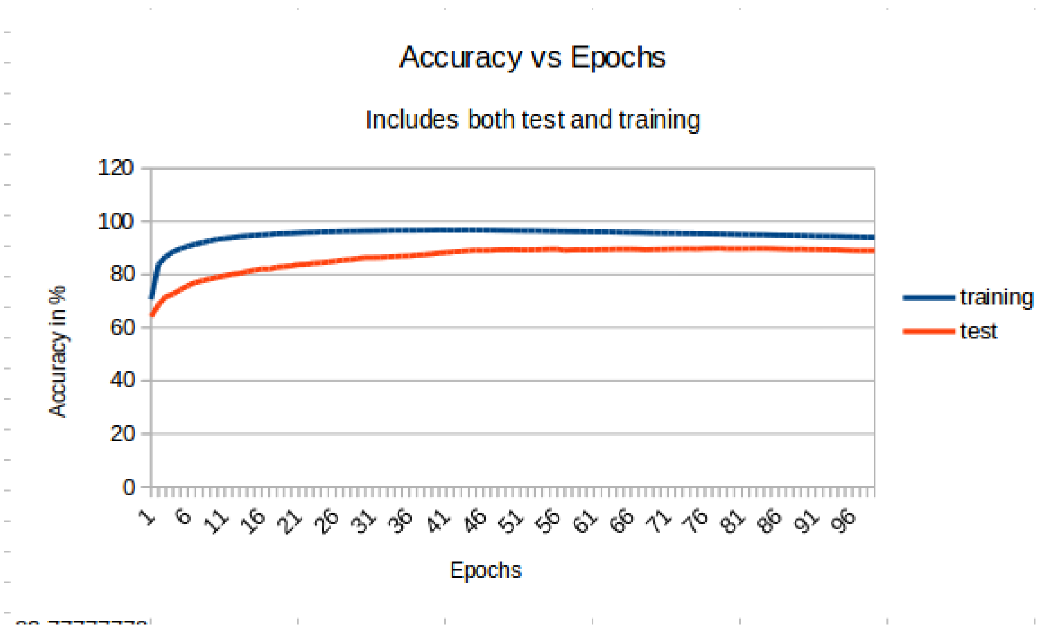
First we will see different plots of cost vs epochs, accuracy of validation set and training set vs epochs etc for different learning rates.The training model on which my code is implemented has a learning rate of 0.004.Large learning rates can lead to miss of local minima as it takes larger steps and hence can diverge easily where as small learning rates will definitely lead to local minima but it will take a lot of time to reach there.

So we keep our learning rate to be in between so that the small increased error than keeping small learning rate will be compensated by the less time taken.

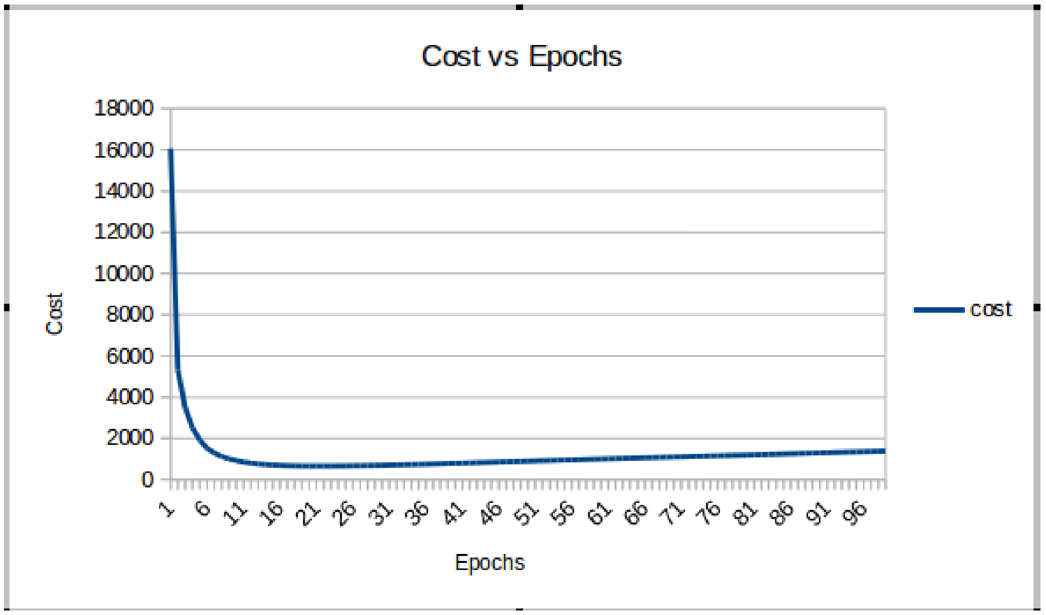
Note: Test accuracy in graphs means the accuracy of validation test set.

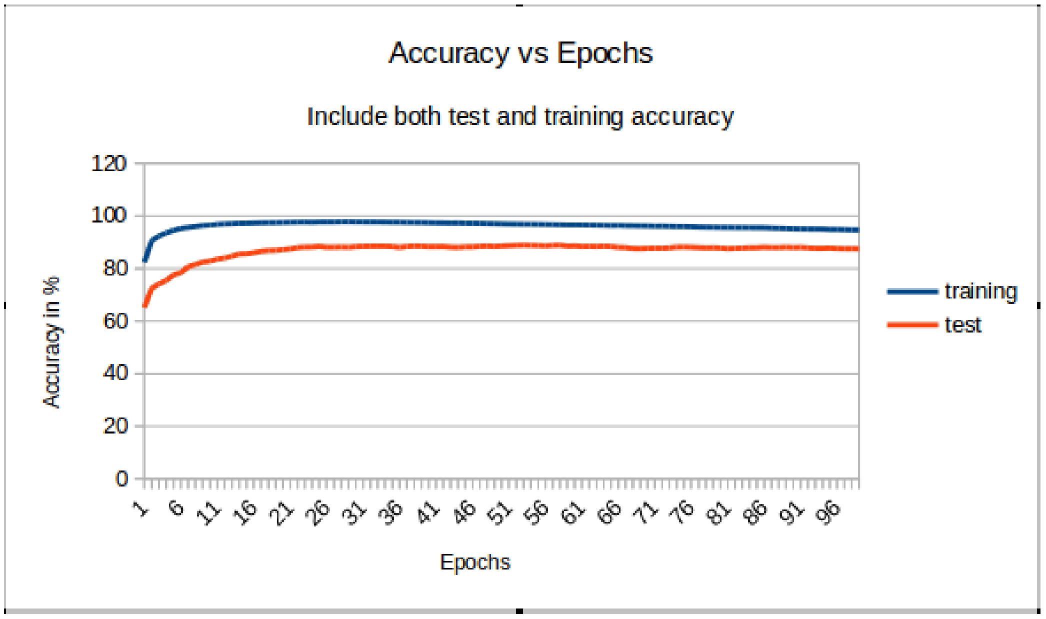
- **Learning Rate = 0.002**



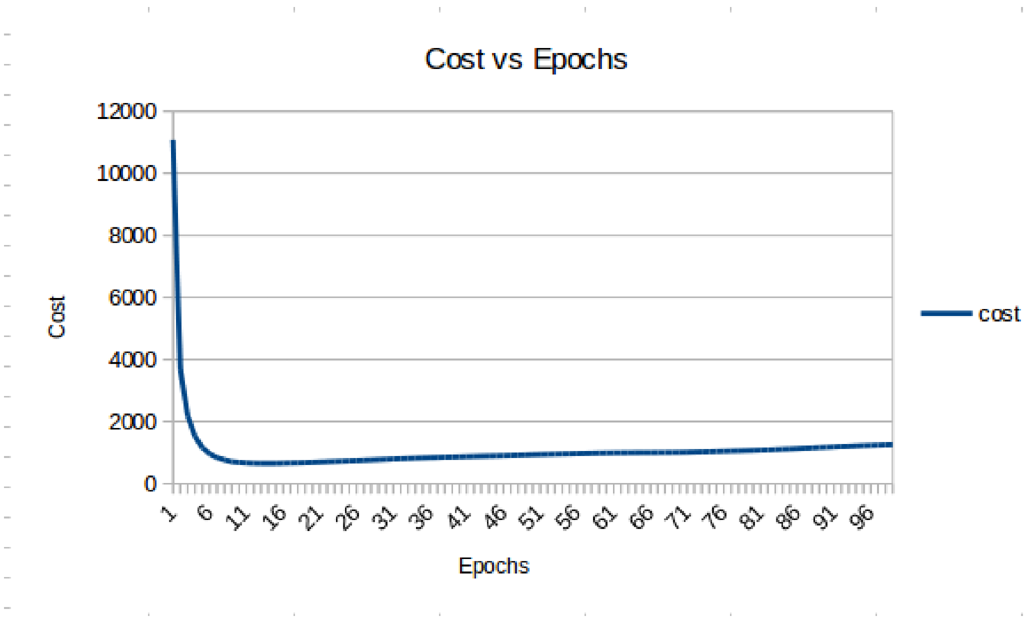


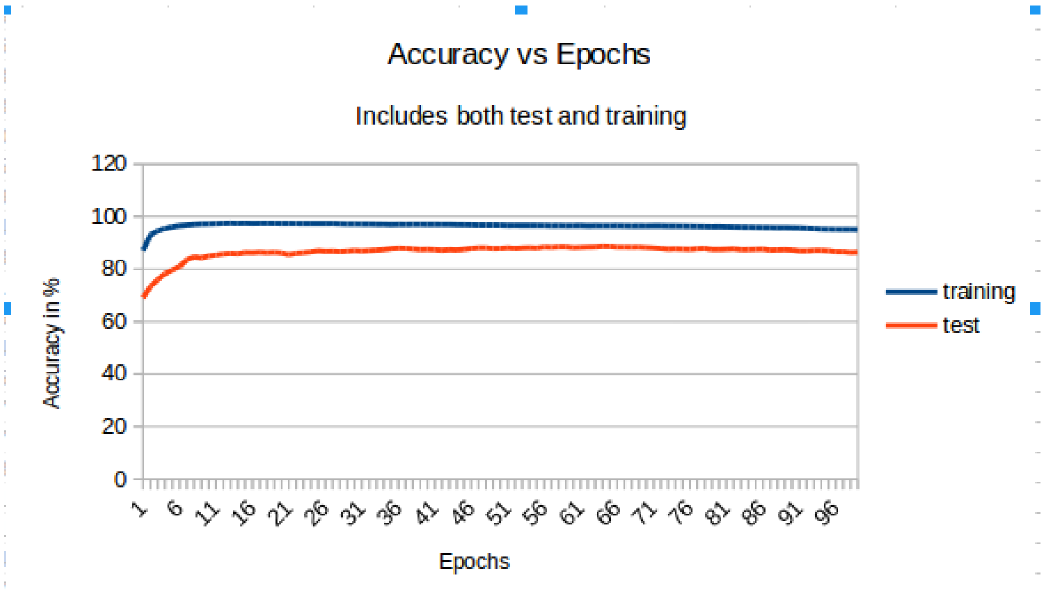
- **Learning Rate = 0.004**





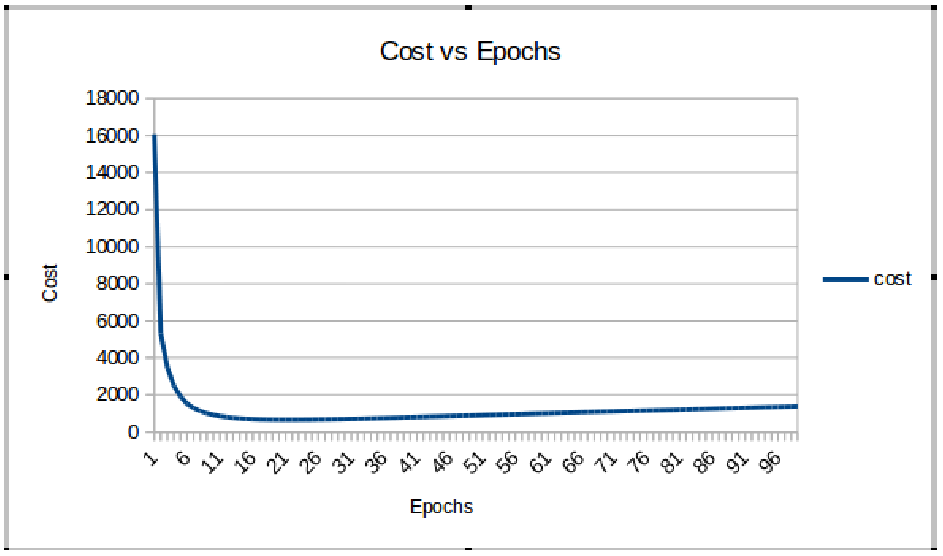
- **Learning Rate = 0.006**

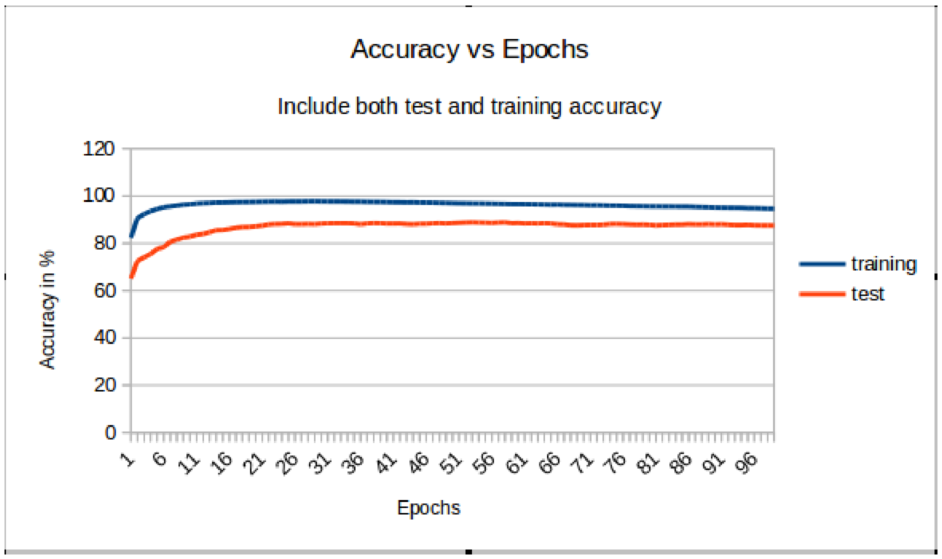




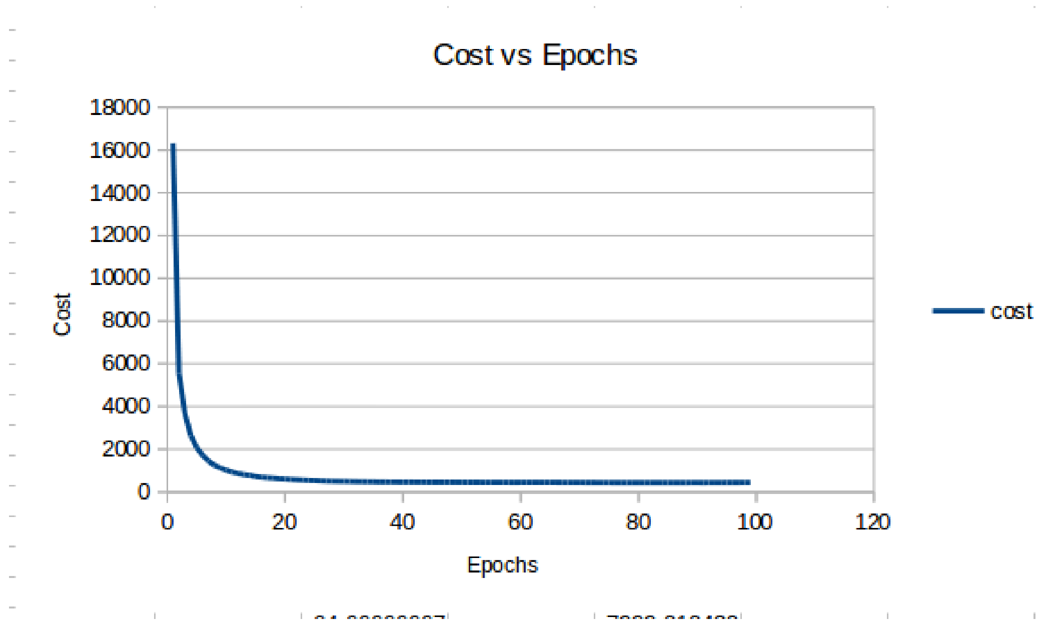
Lambda is a parameter which makes our predictions very horrible either if it is kept quite large or small. It is to be given a value that is neither to large nor to small depending upon our model. The graphs plotted below proves the statement. Lets observe the cost and accuracy of training and validation test sets by varying the regularisation parameter lambda.

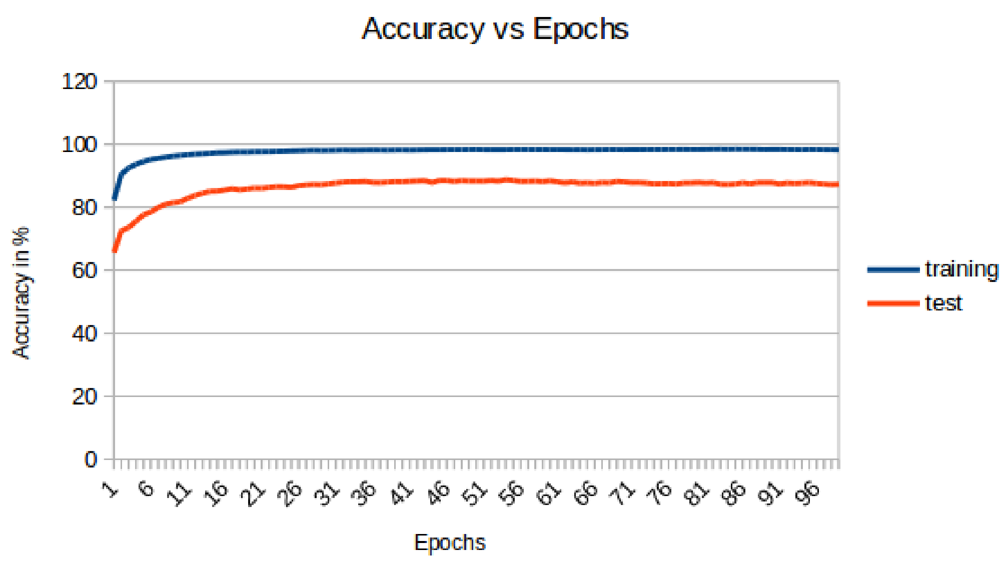
- **Lambda = 0.000001**



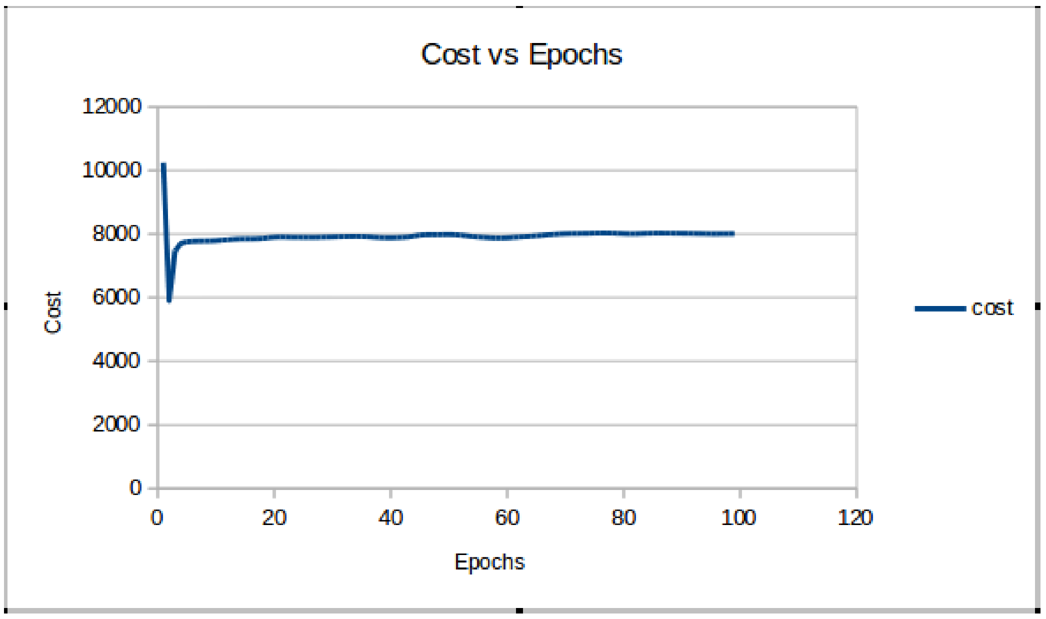


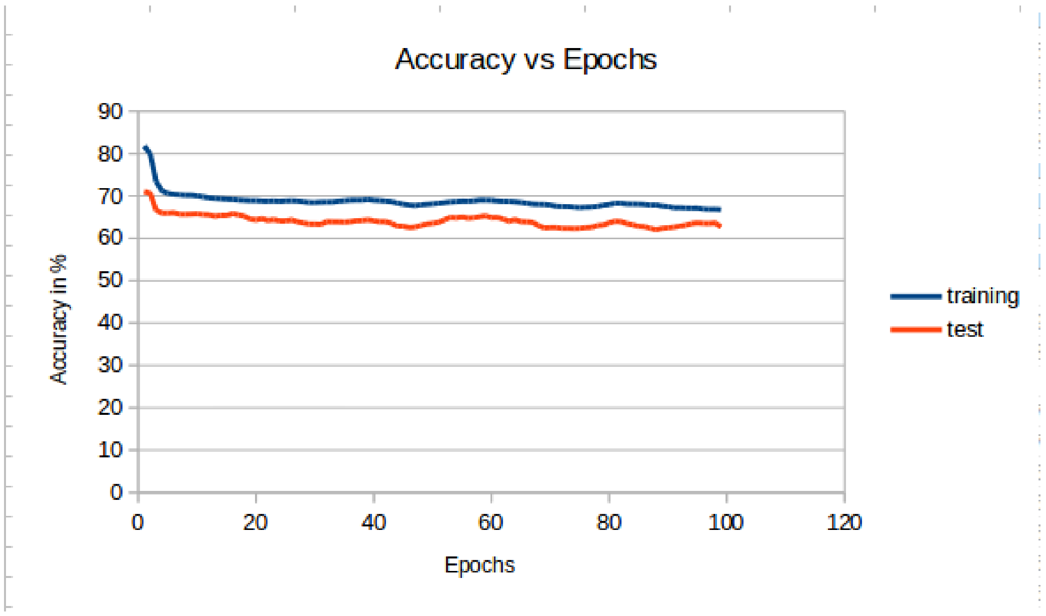
- **Lambda = 0.00000001**





- **Lambda = 0.0001**





References

[1] Research paper. https://www.ics.uci.edu/ pjsadows/notes.pdf.

[2] Wikipedia. https://en.wikipedia.org/wiki/Softmax_function.

[3] Youtube tutorial. https://www.youtube.com/watch?v=bxe2T-V8XRs&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU.