Deep Learning
CS7015
Programming Assignment 1
Backpropagation

Shubhangi Ghosh - EE15B129
Monisha J - CS15B053

February 2018

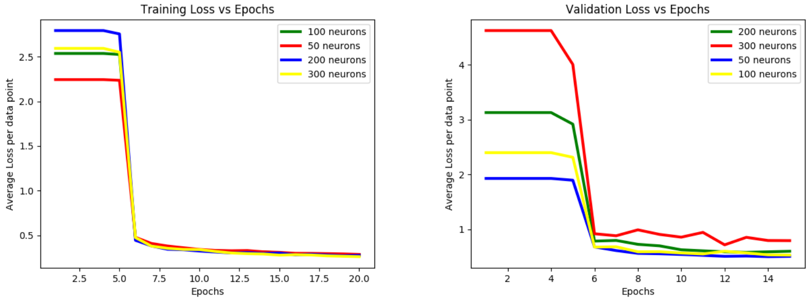
1 Model

The implementation of neural network was done using python and numpy.
The important functions include :
$forward\_propagation()$
$back\_propagation()$
$gradient\_descent()$

2 Supported Hyperparameters

The activation functions supported are *sigmoid*, *tanh* and *relu*.
The final layer activation is fixed to be softmax.
The loss functions supported are mean squared error and cross-entropy.

The implementation supports the following forms of gradient descent :
- Mini-batch gradient descent : Computes the gradient of m instances before making a parameter update
- Stochastic : Updates parameters with gradient for each instance. m has to be set to 1.
- Batch : Computes the gradient of the whole batch and then update gradient. m has to be set to the training set size.



(a) Training Loss

(b) Validation Loss

Figure 1: Single Layer Networks

The optimization algorithms supported are :
- Gradient Descent
- Momentum based gradient descent
- Nesterov based gradient descent
- Adam optimization

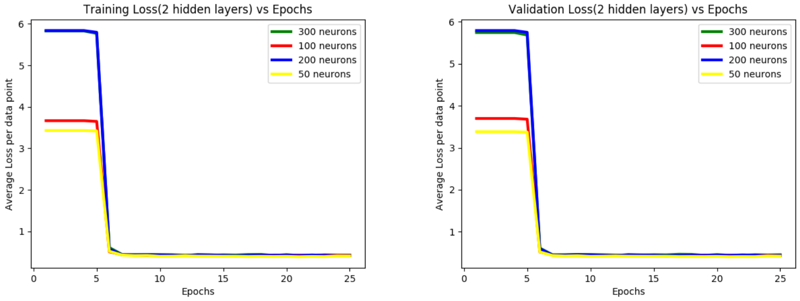
Annealing : When annealing of learning rate is enabled, the learning rate is reduced by half if the validation loss increases after the epoch.
In addition, we also implemented L2 regularization and tried dataset augmentation.

The dataset is first standardized, then trained on using an appropriate setting of the above hyperparameters, the learning rate is tuned accordingly, and the accuracy and F-score are scored on the validation set.

3 Sample Plots

3.1 Single Layer networks

Loss function : Cross-Entropy
Activation : Sigmoid
Optimization algorithm : Adam
Initial learning rate : 0.001
Batch size : 20
Figure 1 shows training and validation loss plots for 50, 100, 200 and 300 neurons.



(a) Training Loss

(b) Validation Loss

Figure 2: Two Layer Networks

3.2 Two Layer Networks

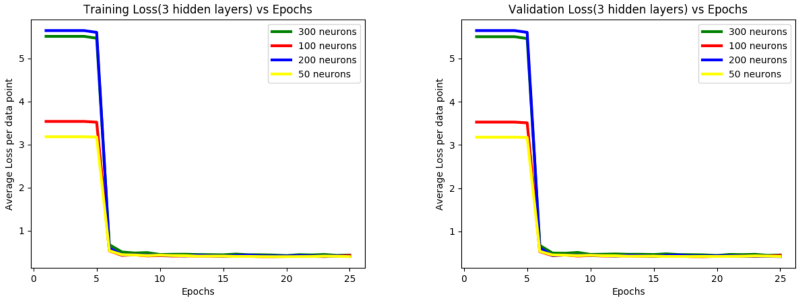
Loss function : Cross-Entropy
Activation : Sigmoid
Optimization algorithm : Adam
Initial learning rate : 0.001
Batch size : 20
Figure 2 shows training and validation loss plots for 50, 100, 200 and 300 neurons.

3.3 Three Layer Networks

Loss function : Cross-Entropy
Activation : Sigmoid
Optimization algorithm : Adam
Initial learning rate :0.001
Batch size : 20
Figure 3 shows training and validation loss plots for 50, 100, 200 and 300 neurons.

3.4 Four Layer Network

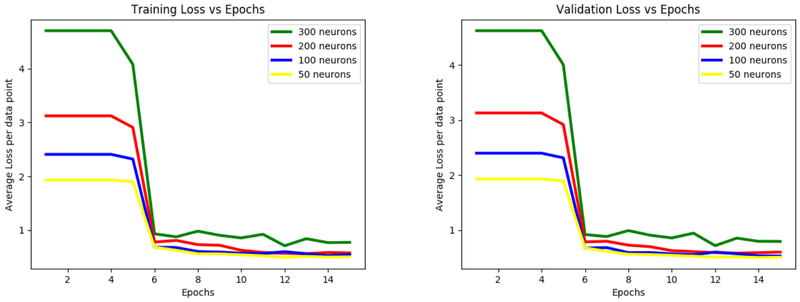
Loss function : Cross-Entropy
Activation : Sigmoid
Optimization algorithm : Adam
Initial learning rate : 0.001
Batch size : 20



(a) Training Loss

(b) Validation Loss

Figure 3: Three Layer Networks



(a) Training Loss

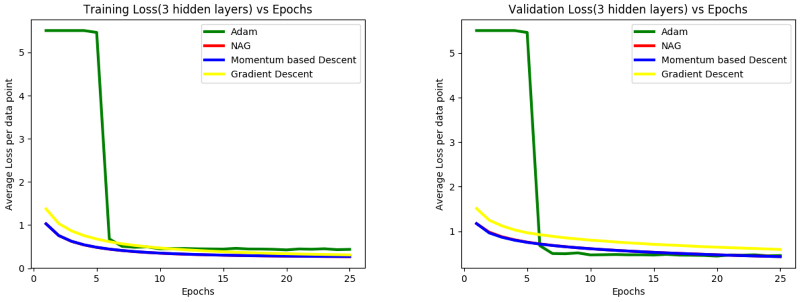
(b) Validation Loss

Figure 4: Four Layer Networks

Figure 4 shows training and validation loss plots for 50, 100, 200 and 300 neurons.

3.5 Adam, NAG, Momentum, GD

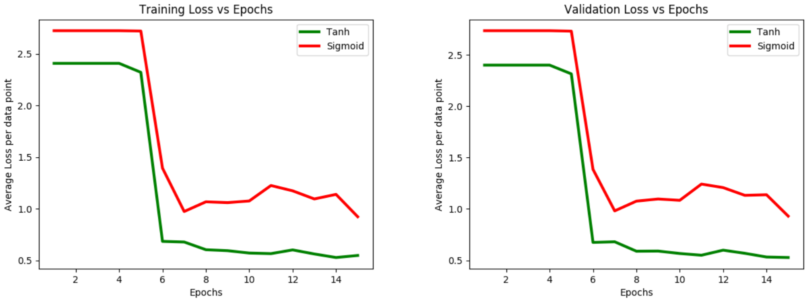
Network structure : 300, 300, 300
Loss function : Cross-Entropy
Activation : Sigmoid
Initial learning rate :
Batch size : 20
Figure 5 shows training and validation loss plots.



(a) Training Loss

(b) Validation Loss

Figure 5: Adam, NAG, Momentum, GD



(a) Training Loss

(b) Validation Loss

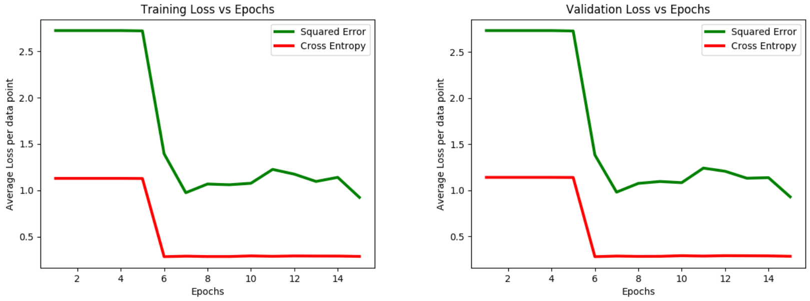
Figure 6: Sigmoid v/s Tanh activation

3.6 Sigmoid v/s Tanh activation

Network structure : 100, 100
Loss function : Cross-Entropy
Optimization algorithm : Adam
Initial learning rate : 0.001
Batch size : 20
Figure 6 shows training and validation loss plots.

3.7 Cross entropy loss v/s Squared error loss

Network structure : 100, 100
Activation : Sigmoid
Optimization algorithm : Adam



(a) Training Loss

(b) Validation Loss

Figure 7: Cross entropy loss v/s Squared error loss

Initial learning rate : 0.001
Batch size : 20
Figure 7 shows training and validation loss plots.

4 Regularisation Methods

We tried the following Regularisation Methods :

1. **Bagging**
   Classifiers with best performance (F1- score) were bagged to give better performance. Models with best performance were trained and majority vote was given as class output. This helped achieve some improvement in performance.

2. **Dataset Augmentation**

   - Data augmentation code for images was used from: https://github.com/xkumiyu/numpy-data-augmentation
   - Vertical flip, Horizontal flip and random rotation were tried.
   - Didn't result in a significant increase in performance.

....

5 Results

We got the best results (on the validation as well as test data) with the following parameters b ensembling five different neural networks.
The five networks were :
- 100,100,100,100,100
- 100, 100, 50, 50, 50
- 200, 200, 200, 100, 100, 50
- 200,200,200,100,100,100
- 300,300,200,100,50

The parameters for each of these networks were :
Epochs : 20
Activation : Tanh
Optimization algorithm : NAG
Initial learning rate : 0.1
Batch size : 20
Loss function : Cross Entropy
The majority vote of these 5 networks was taken as the final prediction.