**Assignment 2**

BY

**Ravisanker E**

**2017AAPS0433H**

**S Ankit**

**2017A7PS0297H**

Table of Contents

[**2.** **Neural Networks** 3](#_Toc38561438)

[**1.1** **Best Results** 3](#_Toc38561439)

[**1.2** **Loss Function Plot** 4](#_Toc38561440)

[**1.3** **Experiments with Hidden Layer Nodes** 5](#_Toc38561441)

[**1.4** **Experiments with Hidden Layers** 6](#_Toc38561442)

[**1.5** **Experiments with Activation Functions** 7](#_Toc38561443)

[**1.6** **Experiments with Weight Initializations** 8](#_Toc38561444)

[**1.7** **Experiments with Learning Rates** 9](#_Toc38561445)

[**1.8** **Dropout** 10](#_Toc38561446)

# **Neural Networks**

## **Best Results**

Let,

tp = True Positives (Positive points classified as positive)

fp = False Positives (Negative points classified as positive)

tn = True Negatives (Negative points classified as negative)

fn = False Negatives (Positive points classified as negative)

Precision = tp/(tp+fp)

Accuracy = (tp+tn)/(tp+tn+fp+fn)

Recall = tp/(tp+fn)

F-score = 2\*Precision\*Recall/(Precision + Recall)

**Model Parameters:**

Number of layers = 3 (including the input and output layers)

Number of nodes = [10, 5, 1] (1 output layer node for binary classification)

Data scaler used = 'standardization'

Activation Functions used = ['relu','sigmoid']

Number of iterations = 50000

Size of batch = 1168 (Gradient Descent)

Learning rate = 0.0001

Loss Function = Cross Entropy Error Function

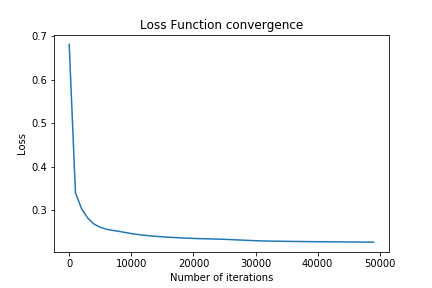
**Training Set:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Precision | Accuracy | Recall | F-Score |
| 523 | 61 | 526 | 58 | 0.895548 | 0.898116 | 0.900172 | 0.897854 |

**Test Set:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | Precision | Accuracy | Recall | F-Score |
| 140 | 14 | 131 | 7 | 0.909091 | 0.928082 | 0.952381 | 0.930233 |

## **Loss Function Plot**



***Inference***

We observed that the Loss Function reduces with iterations and converges at close to 50000 iterations. This is how we decided on the number of epochs for training of ANN model.

## **Experiments with Hidden Layer Nodes**

**Model Parameters:**

Number of layers = 3 (including the input and output layers)

Number of nodes = [10, varied, 1]

Data scaler used = 'standardization'

Activation Functions used = ['relu','sigmoid']

Number of iterations = 50000

Size of batch = 1168 (Gradient Descent)

Learning rate = 0.0001

Loss Function = Cross Entropy Error Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Hidden Nodes** | **Training Accuracy** | **Testing Accuracy** | **Training F Score** | **Testing F score** |
| 1 | 0.88955 | 0.921233 | 0.88964 | 0.923588 |
| 5 | 0.892979 | **0.928082** | 0.89307 | **0.930233** |
| 10 | 0.89126 | 0.924658 | 0.89154 | 0.92667 |
| 15 | **0.898116** | 0.9212 | **0.897854** | 0.9235 |
| 20 | 0.890411 | 0.917808 | 0.890223 | 0.92 |

***Inference***

We observed that the accuracy and f-score was maximum when the number of hidden nodes were 5. We observed a trend of training accuracy approximately increasing with increasing hidden nodes (highest for 15 hidden nodes) while the testing accuracy is maximum at 5 hidden nodes suggesting overfitting.

## **Experiments with Hidden Layers**

**Model Parameters:**

Number of layers = varied (including the input and output layers)

Data scaler used = 'standardization'

Activation Functions used = sigmoid-output layer and relu-other layers

Number of iterations = 50000

Size of batch = 1168 (Gradient Descent)

Learning rate = 0.0001

Loss Function = Cross Entropy Error Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Hidden Layers** | **Training Accuracy** | **Testing Accuracy** | **Training F Score** | **Testing F score** |
| 1[10, 5, 1] | 0.892979 | **0.928082** | 0.89307 | **0.930233** |
| 2[10, 5, 3, 1] | 0.877568 | 0.910959 | 0.879325 | 0.913907 |
| 3[10, 5, 3, 2, 1] | 0.875648 | 0.921233 | 0.874138 | 0.923077 |
| 4[10,5,5,8,2,1] | **0.8964** | 0.924658 | **0.8956** | 0.926174 |

***Inference***

We observed that the accuracy and f-score was maximum for 1 hidden layer. We observed that the training accuracy and f-score was maximum for 4 hidden layers but the testing accuracy and f-score was maximum for 1 hidden layer.

## **Experiments with Activation Functions**

**Model Parameters:**

Number of layers = 3 (including the input and output layers)

Number of nodes = [10, 5, 1]

Data scaler used = 'standardization'

Activation Functions used = [varied,’sigmoid’]

Number of iterations = 50000

Size of batch = 1168 (Gradient Descent)

Learning rate = 0.0001

Loss Function = Cross Entropy Error Function

|  |  |  |
| --- | --- | --- |
| **Activation Function** | **Testing Accuracy** | **Testing F score** |
| relu | **0.928082** | **0.930233** |
| linear | 0.921233 | 0.923588 |
| sigmoid | 0.917808 | 0.92053 |
| tanh | 0.917808 | 0.92 |

***Inference***

We observed that the accuracy and f-score was maximum with relu as the activation function. So we picked relu as our activation function for or best model.

## **Experiments with Weight Initializations**

**Model Parameters:**

Number of layers = 3 (including the input and output layers)

Number of nodes = [10, 5, 1]

Data scaler used = 'standardization'

Activation Functions used = [‘relu’,’sigmoid’]

Number of iterations = 50000

Size of batch = 1168 (Gradient Descent)

Learning rate = 0.0001

Loss Function = Cross Entropy Error Function

|  |  |  |
| --- | --- | --- |
| **Weight Initializations** | **Testing Accuracy** | **Testing F score** |
| Gaussian | **0.928082** | **0.930233** |
| Uniform | 0.921233 | 0.923588 |

***Inference***

We observed that the accuracy and f-score was maximum with Gaussian Weight Initializationas.

## **Experiments with Learning Rates**

**Model Parameters:**

Number of layers = 3 (including the input and output layers)

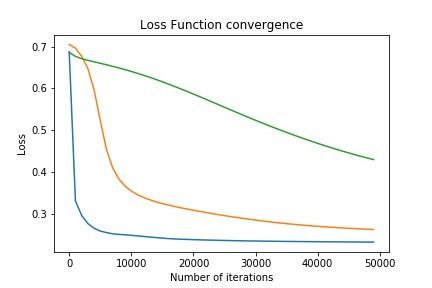
Number of nodes = [10, 5, 1]

Activation Functions used = [‘relu’,’sigmoid’]

Number of iterations = 50000

Learning rate = varied

Loss Function = Cross Entropy Error Function



**Blue Line – learning rate = 0.0001**

**Red Line – learning rate = 0.00001**

**Green Line – learning rate = 0.0000001**

***Inference***

We observed that with learning rate at 0.0001, the loss function converged the quickest. Thus we picked 0.0001 as the learning rate for our model.

## **Dropout**

**Model Parameters:**

Number of layers = 3 (including the input and output layers)

Number of nodes = [10, varied, 1]

Activation Functions used = [‘relu’,’sigmoid’]

Number of iterations = 50000

Learning rate = 0.000005

Loss Function = Cross Entropy Error Function

**5 Hidden Nodes**

|  |  |  |
| --- | --- | --- |
| **Rate** | **Accuracy** | **F\_score** |
| 0 | [0.68835616] | [0.56872038] |
| 0.1 | [0.68150685] | [0.55924171] |
| 0.2 | [0.61986301] | [0.40641711] |
| 0.3 | [0.76369863] | [0.7250996] |
| 0.4 | [0.66780822] | [0.52216749] |
| 0.5 | [0.67123288] | [0.52941176] |
| 0.6 | [0.78082192] | [0.77931034] |
| 0.7 | [0.51027397] | [0.67276888] |

**10 Hidden Nodes**

|  |  |  |
| --- | --- | --- |
| **Rate** | **Accuracy** | **F\_score** |
| 0 | [0.89041096] | [0.88965517] |
| 0.1 | [0.87328767] | [0.87457627] |
| 0.2 | [0.83219178] | [0.8164794] |
| 0.3 | [0.80479452] | [0.77470356] |
| 0.4 | [0.75342466] | [0.70731707] |
| 0.5 | [0.71917808] | [0.63392857] |
| 0.6 | [0.74657534] | [0.67826087] |
| 0.7 | [0.79452055] | [0.83050847] |

**15 Hidden Nodes**

|  |  |  |
| --- | --- | --- |
| **Rate** | **Accuracy** | **F\_score** |
| 0 | [0.91438356] | [0.91749175] |
| 0.1 | [0.81506849] | [0.79850746] |
| 0.2 | [0.87671233] | [0.8902439] |
| 0.3 | [0.78767123] | [0.8258427] |
| 0.4 | [0.80136986] | [0.78030303] |
| 0.5 | [0.77739726] | [0.75095785] |
| 0.6 | [0.82534247] | [0.84210526] |
| 0.7 | [0.81849315] | [0.80442804] |

**20 Hidden Nodes**

|  |  |  |
| --- | --- | --- |
| **Rate** | **Accuracy** | **F\_score** |
| 0 | [0.92465753] | [0.92666667] |
| 0.1 | [0.87328767] | [0.86545455] |
| 0.2 | [0.85273973] | [0.87087087] |
| 0.3 | [0.84589041] | [0.83146067] |
| 0.4 | [0.85958904] | [0.87461774] |
| 0.5 | [0.86986301] | [0.88271605] |
| 0.6 | [0.83219178] | [0.83044983] |
| 0.7 | [0.85616438] | [0.87195122] |