BITS F464 - Machine Learning

Assignment 2

L Srihari	Rikil Gajarla	Raj Kashyap Mallala
2017A7PS1670H	2017A7PS0101H	2017A7PS0025H

Logistic Regression

The aim of this approach is to find whether a banknote with given features is counterfeit or real. We use logistic regression for this binary classification task by training on 60% of the given dataset and cross-validating on 20% of the dataset. The remaining 20% of the dataset is used for testing and the results are summarized.

Classification Using Logistic Regression

Hypothesis and Likelihood

The hypothesis for logistic regression is given as $h_w(x) = g(w^Tx) = 1/(1+\exp(-w^Tx))$. Here w is the coefficient or weight vector in the function h. It should also be noted that $g'(z) = g(z)(1 - g(z))$.

Hypothesis for classification is given by:

$$P(y=1 / x;w) = h_w(x)$$
$$P(y=0/ x;w) = 1 - h_{,w}(x)$$
or
$$P(y/ x;w) = (h_w(x))^y (1-h_w(x))^{1-y}$$

Likelihood function for m training examples is given by

$$L(w) = P(y/ x;w) = \prod p(y^{(i)}/ x^{(i)} ;w)$$
$$= \prod [h_w(x^i)]^y [(1 - h_w(x^i))]^{1 - y}$$

Cost function is the logarithm of likelihood function and derivative of cost function is given by

$$D = [y - g(w^Tx)]x_j$$

Weight update and Learning features

Further, the weight values in logistic regression are updated through gradient descent algorithm and is given by

$$w_j^{t+1} = w_j^t - \alpha D$$

where α is the learning rate.

Regression can also be done with L1 and L2 norm regularisation.
Weight update rule with L1 norm regularisation is given by

$$w_j^{t+1} = w_j^t - \alpha \left(\sum(y^i - g(w^T x^i)\, x_j^i\right) - (\alpha\lambda/2)\mathrm{sgn}(w_j)$$

Here sgn(x) is the signum function given by sign() method in python.

Weight update rule with L2 norm regularisation is given by

$$w_j^{t+1} = w_j^t (1 - \alpha\lambda) - \alpha \left(\sum(y^i - g(w^T x^i)\, x_j^i\right)$$

Training, Cross-Validation and Testing

The dataset is divided into 3 parts - Training set, Cross validation set and Testing set. Training set is made of 60% of the dataset. Cross Validation set is of 20% and testing set is made of remaining 20% of the dataset. This division is done after randomly permuting the dataset so that all 0s and all 1s do not occur together. This ensures proper training from the dataset.

The cross validation set is used to optimize the parameters of the model. When both Training error and cross validation errors are high, the regularisation parameter is decreased to reduce underfitting. When cross validation error is high and training error is low, regularisation parameter is increased to reduce overfitting of data. We try to minimize cross validation error to least possible value.
References: Lecture 11: Cross validation (Stanford lecture 11)Validation Error less than training error? (First answer with explanation)

Estimation of initial values of weights have been performed on the cross validation set with Gaussian,Uniform and Random distributions. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. Hence we use the test set to provide an unbiased evaluation of the model.
Reference: What is the Difference Between Test and Validation Datasets?

The test set is used to calculate the performance of the model.Accuracy and F-Score of the test set has been evaluated and reported.

Feature Scaling

The training data is first normalized so that all features are equally scaled within a -1 to 1 range. Otherwise, larger feature values tend to affect the model more than smaller feature values.

This was done by subtracting the mean of the feature vector from each feature and dividing it by the standard deviation. This step ensures that all features fall within the range of zero to with some distribution having mean as 0 and standard deviation as 1.

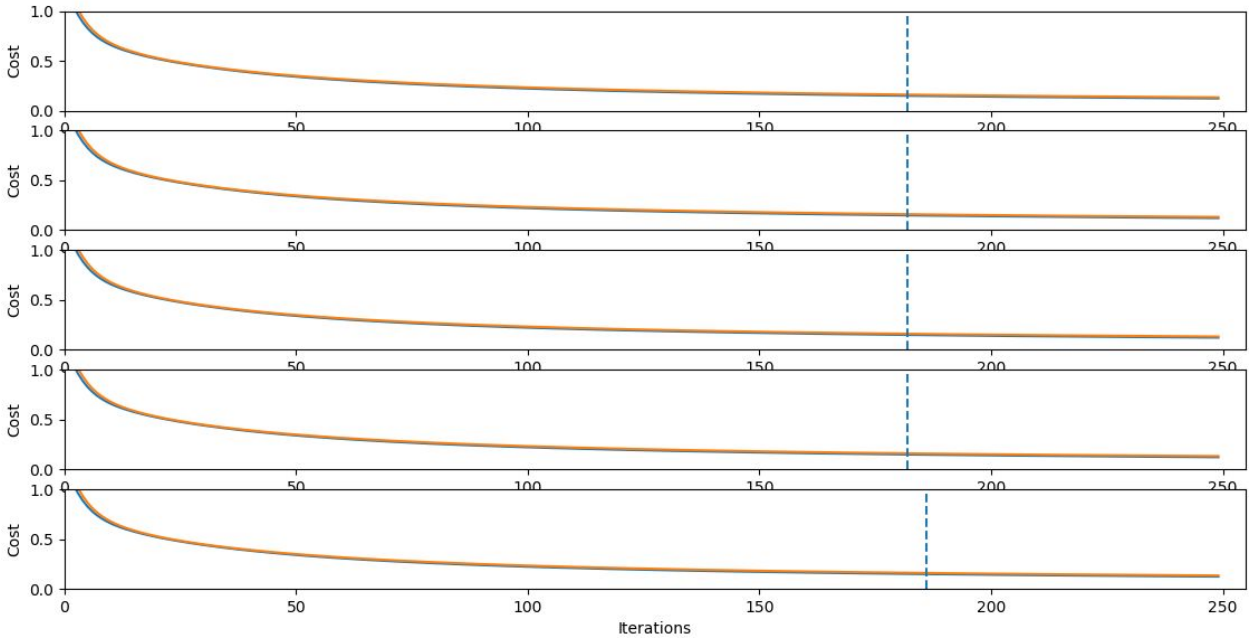
Data Analysis/Training

The cross validation data was trained across the following values.

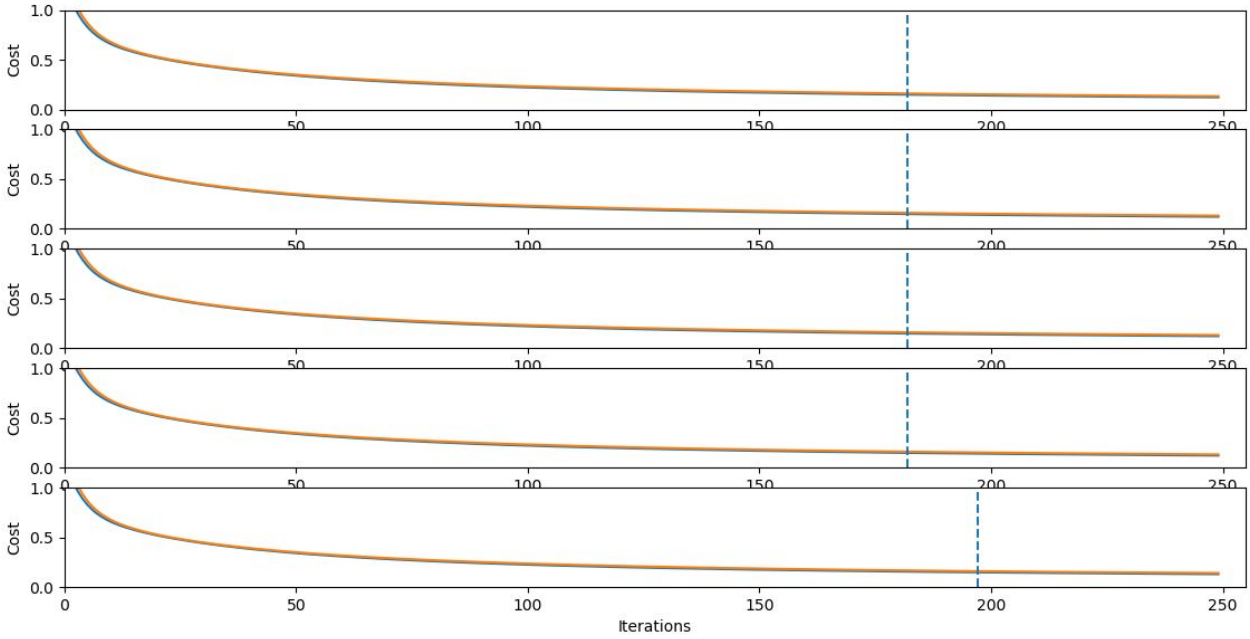
Types of regularisation: [Without, L1 norm, L2 norm]
Values for regularisation parameter (lamda): [0.0001, 0.001, 0.01, 0.1, 2]
Types of initial weight distribution : [Gaussian, Uniform, Random]
Values for learning rate (alpha) : [0.001, 0.01, 0.05, 0.1, 0.2]

Smaller values of learning rate require larger numbers of iteration to converge. All analysis was done against a fixed number of 250 iterations except for learning rate analysis. This value was chosen because large numbers of iteration spoil the representation of cost vs iterations plot and hence decrease the interpretability.
Analysis of initial weight distributions are of little use compared to regularisation. This is because even if the initial weights were poorly distributed, the gradient descent algorithm ensures larger changes towards optimal weight for higher cost, hence it is only a matter of increase in number of iterations, that too not by a very large number, before the model converges to optimal values.
But finding the right regularisation parameter is essential to not underestimate or overestimate the model.

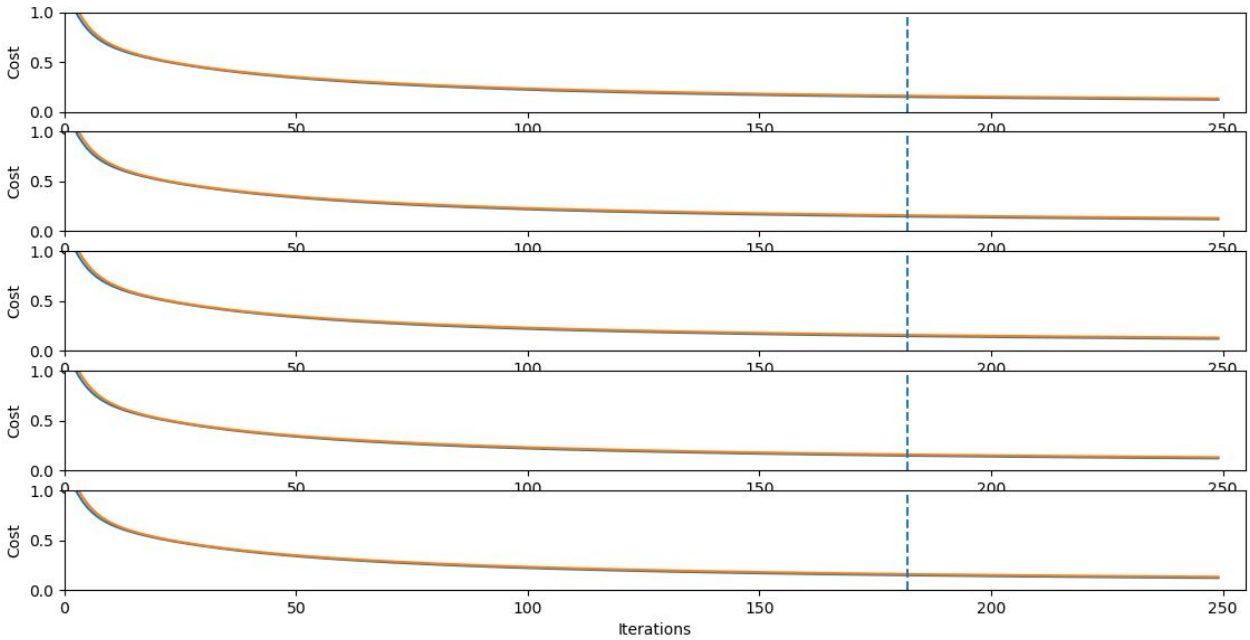
Hence, the optimal regularisation type and parameter was determined first. The plots below show their comparison when weights are fixed at [1, 0, 1, 0, 1].
Note: In these plots, the red line shows the training error against the number of iterations whereas, the blue line shows the cross-validation error. In most cases, they are very close to each other.
The dashed line in blue shows the iteration number, when the model crossed a threshold error value of 0.15.



The plots are in order of lamda = 0.0001, 0.001, 0.01, 0.1, 2 respectively for L1 norm regularisation.



The plots are in order of lamda = 0.0001, 0.001, 0.01, 0.1, 2 respectively for L2 norm regularisation.



The plots are in order of lamda = 0.0001, 0.001, 0.01, 0.1, 2 respectively for no regularisation. The varied plots are because of the random initialisation of weights.

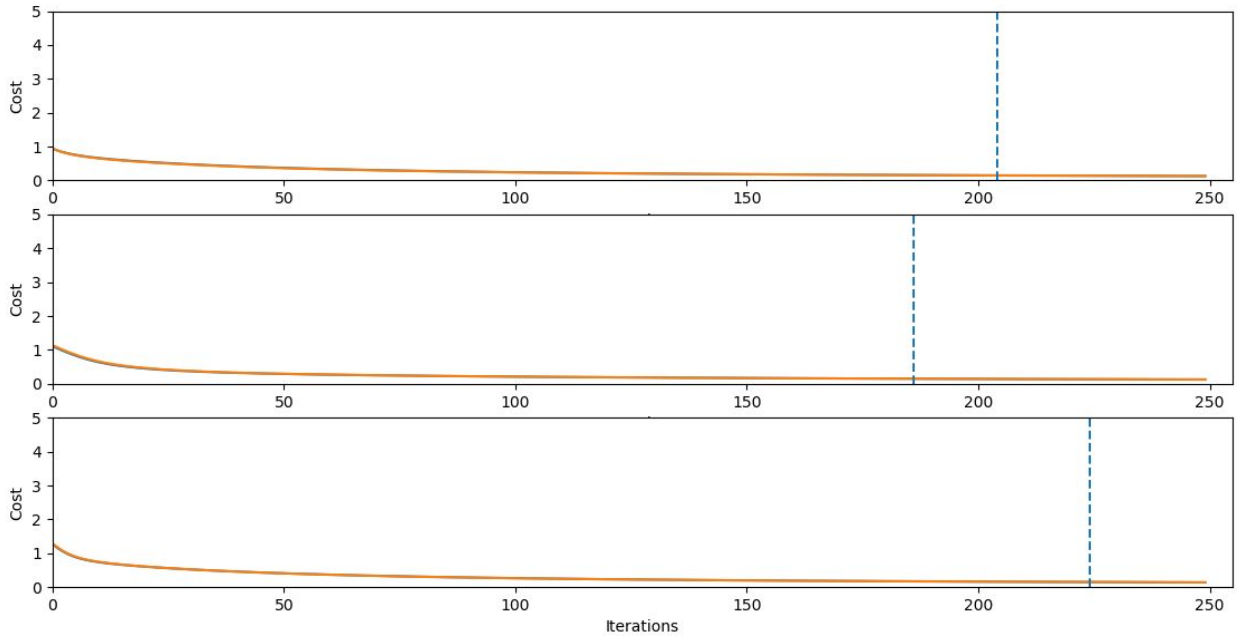
Analysis of regularisation

It can be observed that the variance in number of iterations is very low. Further, among L1, L2 and no regularisation, all the 3 have the same minimum point around 175 iterations. One thing to note here is that with a very high value of regularisation constant like 2, the model has worse performance due to underfitting. But the other values indicate that no regularisation is required. Hence we can drop regularisation from our model.

Analysis of initial weight distribution

Note: In these plots, the red line shows the training error against the number of iterations whereas, the blue line shows the cross-validation error. In most cases, they are very close to each other.

The dashed line in blue shows the iteration number, when the model crossed a threshold error value of 0.15.



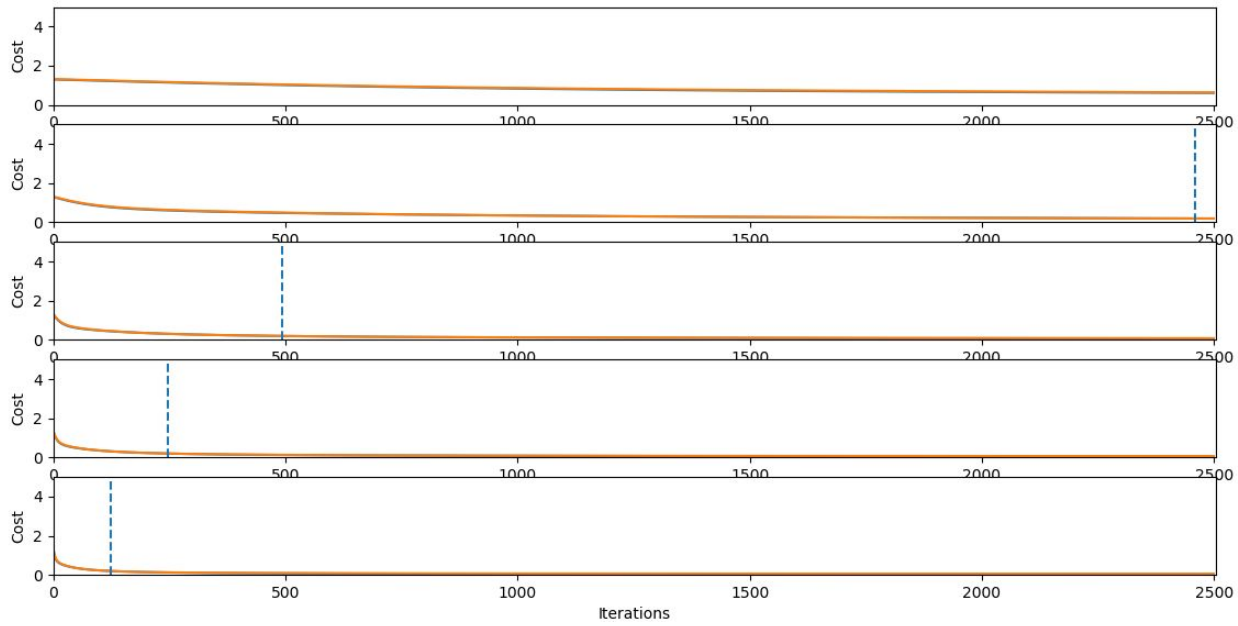
The plots are in order of Random, Gaussian and Uniform distribution respectively. The gaussian distribution has mean 0 and standard deviation 1. The Uniform distribution has minimum value -1 and maximum value 1. Random is a random sampling from uniform distribution in the range [0,1).

These values were chosen since most of the cross validation runs gave final weights in this range of values. The general trend observed in the location of error threshold is as shown in the plot. In many cases, Gaussian performed better than Uniform distribution.
One other thing that was observed over several runs was that random distribution had least initial error compared to the other two.

Analysis of learning rate

In the following plots, initial weights were fixed at w = [1,0,1,0,1]
Note: In these plots, the red line shows the training error against the number of iterations whereas, the blue line shows the cross-validation error. In most cases, they are very close to each other.
The dashed line in blue shows the iteration number, when the model crossed a threshold error value of 0.15.

A fixed number of 2500 iterations were considered for these model parameters.



The plots are in order of alpha = 0.001, 0.01, 0.05, 0.1, 0.2 respectively for no regularisation. The varied plots are because of the random initialisation of weights.

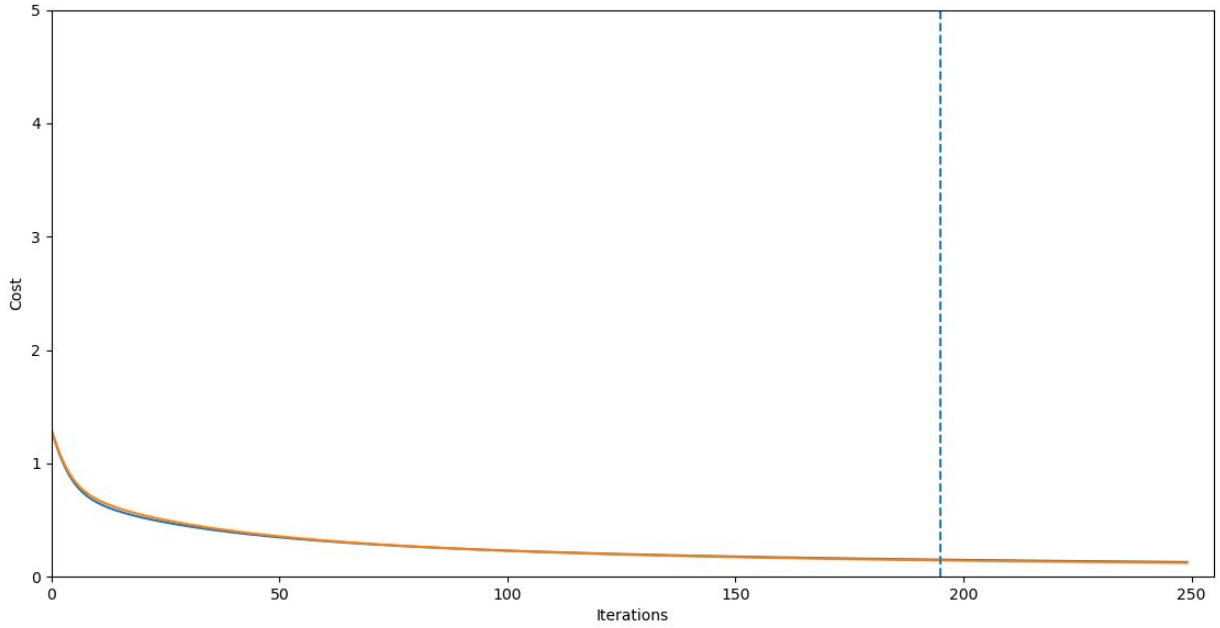
It can be clearly observed that with higher learning rate, error threshold is reached in a lesser number of iterations. But if allowed to converge to minimum possible cost without any bound on number of iterations, then it is expected that the model with lesser learning rate parameter reaches closer to the optimum weights even though number of iterations will be much higher.

Code Analysis

Python with numpy and pandas libraries were used to write code for this classifier.
Pandas library was used only to read the csv input file.
A hold out method was used to evaluate the accuracy.

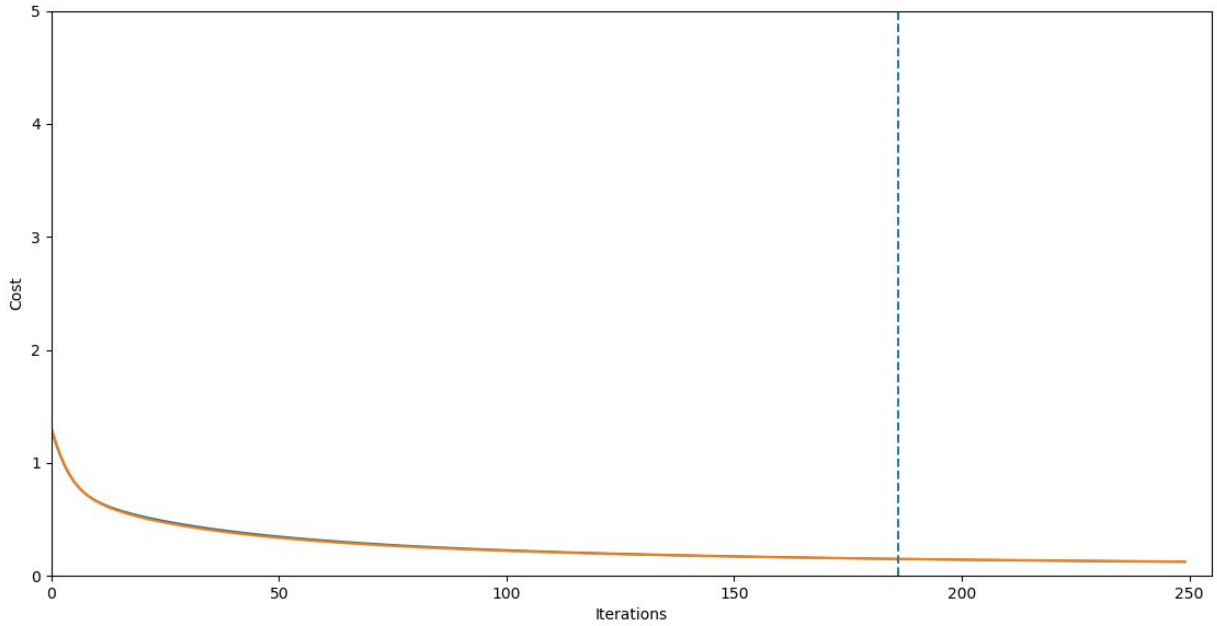
Result (On the test set)

No Regularisation



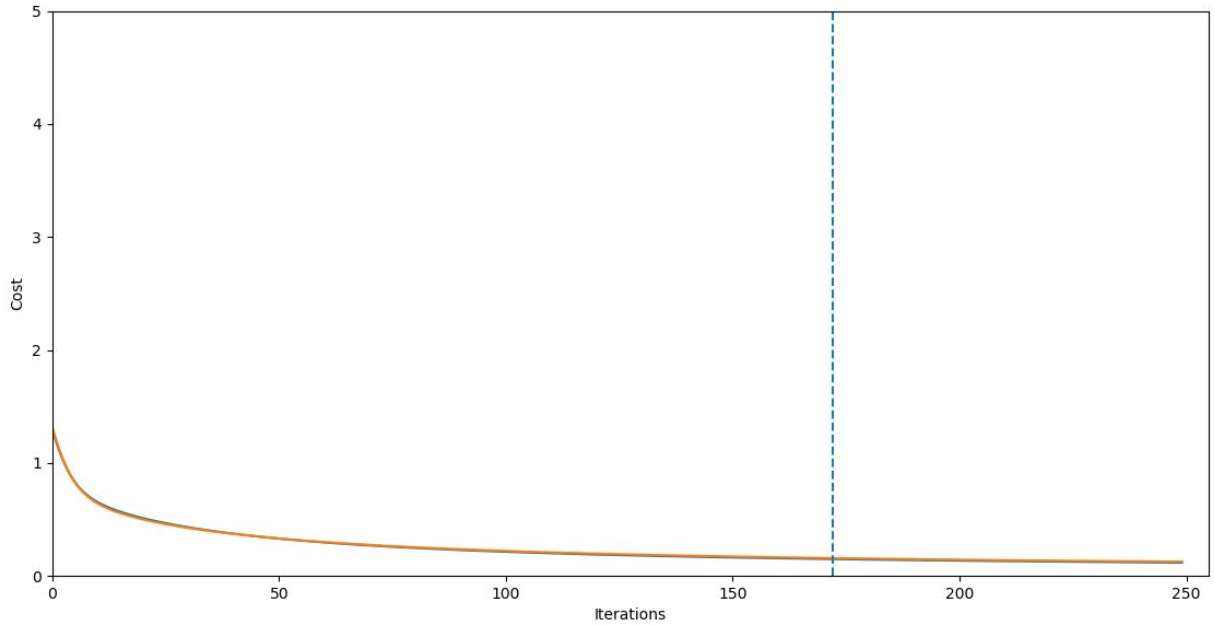

```
learning rate is :  0.2
Weight initialisation:  gaussian
Accuracy is :  98.18181818181819 %
Weight values :  [ 0.28481584 -3.59051962 -1.70474824 -1.87614632  0.32405298]
```

L1 norm Regularisation




```
regularization constant is :  0.1
regularisation type is:  L1 norm
learning rate is :  0.2
Weight initialisation:  gaussian
Accuracy is :  97.81818181818181 %
Weight values :  [ 0.23029897 -3.52004753 -1.76301594 -2.01372876  0.24646487]
```

L2 norm Regularisation



```
regularization constant is :  0.1
regularisation type is:  L2 norm
learning rate is :  0.2
Weight initialisation:  gaussian
Accuracy is :  97.81818181818181 %
Weight values :  [ 0.14936737 -3.56646641 -1.68578167 -1.92231235  0.31483831]
```

Highest accuracy obtained from using logistic regression model is 97.8 and **F-Score** calculated as (2*precision*recall) / (precision + recall) as **0.977** with **precision** (TP/(TP+FP)) as **0.963** and **recall** (TP/(TP+FN)) as **0.992.**

Feature Importance

The dataset was downloaded from banknote authentication Data Set. It specifies four features

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)

The first weight value corresponds to the bias and the remaining weights correspond to the features respectively. Importance of features can be determined by comparing the values based on magnitude and sign of respective weights. As suggested by Model-based feature importance greater positive values of weights signify higher importance of the particular feature in the prediction of positive class. Similarly, greater negative values signify higher importance in the prediction of negative class. But, it is to be noted that this is valid only when all features in training data points are positive. Since, our data has some negative training features, nothing can be said based on sign.

This can be observed from the expression for loss function in logistic regression. The gradient descent algorithm reduces loss by setting large positive weights for features more important in predicting a data point to belong to the positive class and similarly for negative class.

By this idea, we consistently observe that the bias is always positive. The first feature weight has much higher magnitude than the second, third and fourth features. This shows that the first feature (Variance of Wavelet transform) has much higher influence on finding the class. The third feature and second feature have nearly the same weights with the third feature having higher magnitude. The fourth feature has the least magnitude of weight, which shows that it is least important in classification.

Neural Network

Neural Networks is one of the most popular and also the most powerful machine learning models which can be designed to predict any non-linear kind of data. Neural networks consist of many layers of neurons which are responsible to predict data. The passing of data from input layer to output layer is called forward propagation. Each neuron has inputs connected from all neurons from the previous layer which are multiplied by weights and added bias which determines the output of neurons.

Classification using Neural Networks

Architecture

For basic understanding of neural networks, let us consider a model with 2 layers i.e., input layer - hidden layer - output layer
Usually, the output of neurons is assumed to be small so that we do not have a problem of overflowing numbers. For this purpose, we usually attach them with sigmoid (or similar) functions
Output of hidden layer:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

And for each a_j, we have output of activation function h() as $z_j = h(a_j)$
So, now we will have the output of the second layer/output layer as:

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

Passing the above output though a sigmoid layer, we can get the output in the range (0,1). So, to use a network for classification, we can set a threshold as 0.5 for binary classification

Generating output from inputs

We can use the following to get output in a 2-layer Neural Network:

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=0}^{M} w_{kj}^{(2)} h \left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right)$$

Where σ is the final sigmoid function

Training

From the above equations, we can see that there are a lot of parameters which must be taken care of (weights and biases) to get the perfect output. But directly adjusting them by hand based on the input data is too tedious and can take a lot of time. Hence, researchers devised an algorithm to compute and adjust the weights automatically. This algorithm is discussed below:

Back-Propagation

This algorithm, also known as the backprop algorithm, back propagates the error in prediction through the layers so that each corresponding weights are adjusted accordingly.
Let $E_n()$ be the error of each training point. Then the total error will be:

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w})$$

Where:

$$E_n = \frac{1}{2}\sum_k (y_{nk} - t_{nk})^2 \qquad \text{and} \qquad y_k = \sum_i w_{ki} x_i$$

The back propagation algorithm uses a technique similar to the gradient descent by adjusting the weights based on travelling a small distance in the negative direction of their gradient.
By using product of derivatives rule:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}$$

For the sake of simplicity, we can assume:

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \quad \text{where} \quad \delta_j \equiv \frac{\partial E_n}{\partial a_j} \quad \text{and} \quad \frac{\partial a_j}{\partial w_{ji}} = z_i.$$

And then,

$$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k$$

By using the above equations, we can find out the gradient and apply back-prop algorithm efficiently on each neuron to adjust weights accordingly

Example Dataset

To give predictions using the developed neural networks, we have used house price prediction dataset where we classify each house to be above or below the median price.

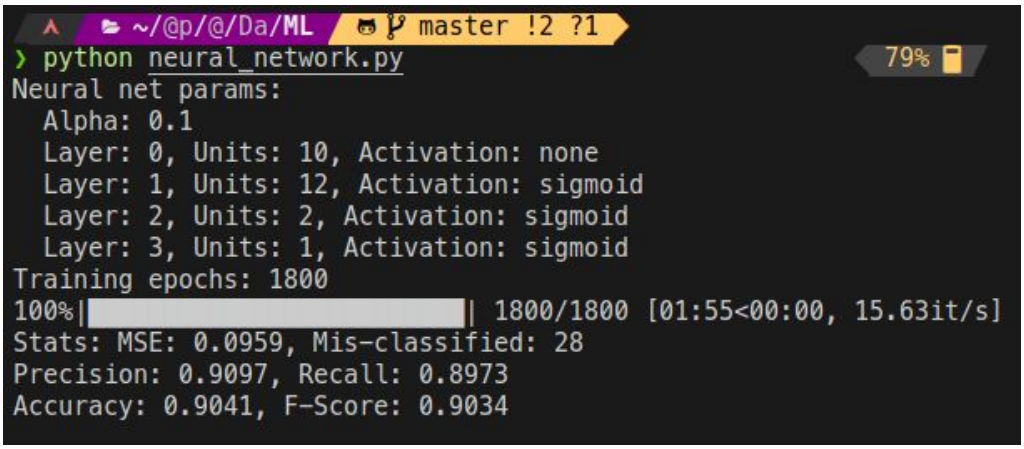
Model Analysis

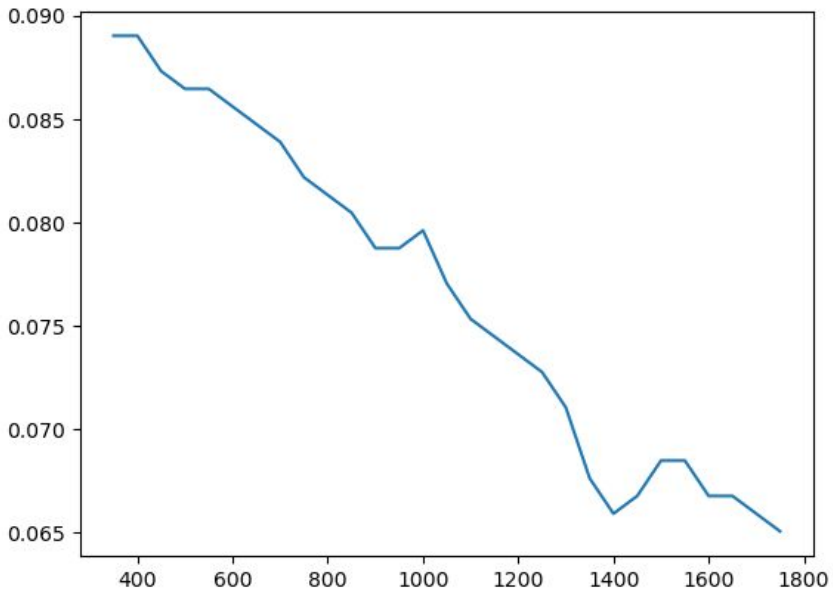
Tuning various parameters like learning rate, no of epochs, no of layers and no of neurons per layer, the following results were obtained:

Layers / Epochs / Learning rate - No. of misclassifications - accuracy - (layer activations)
10 12 1 / 5000 / 0.1 - 33 - 88.69% - (tanh, tanh, tanh)
10 12 1 / 5000 / 0.1 - 32 - 89.04% - (relu, relu, relu)
10 12 2 1 / 1800 / 0.001 - 31 - 89.38% (sigmoid, sigmoid, tanh)
10 12 2 1 / 1800 / 0.001 - 31 - 89.38% (relu, relu, tanh)
10 12 1 / 5000 / 0.1 - 30 - 89.72 - (sigmoid, sigmoid, sigmoid)
10 8 4 2 1/ 500 / 0.03 - 30 - 89.72 - (sigmoid, tanh, relu, sigmoid)
10 12 12 1 / 5000 / 0.1 - 29 - 90.2%- (sigmoid, sigmoid, sigmoid, sigmoid)
10 12 2 1 / 1800 / 0.1 - 28 -90.41% - (sigmoid, sigmoid, sigmoid, sigmoid)
10 12 2 1 / 4500 / 0.03 - 28 - 90.41% - (sigmoid, sigmoid, sigmoid, sigmoid)

Results

Final result:
The following parameters resulted in the highest possible testing accuracy:
Learning rate: 0.1
No of layers: 3
Units: Layer1: 12, Layer2: 2, Layer3: 1
Layer Activations: All sigmoid give better results
Training epochs: 1800
Misclassified points: 28, Accuracy: 90.41%
Precision: 0.90, Recall: 0.89, F-Score: 0.90




0.090
0.085
0.080
0.075
0.070
0.065
400
600
800
1000
1200
1400
1600
1800