**BITS F464**

**Machine Learning - Assignment 2**

**LOGISTIC REGRESSION**

|  |  |
| --- | --- |
| **Name** | **ID** |
| **Praneetha Vaddamanu** | **2017AAPS0319H** |

**Problem Statement and Methodology**

The given task is to implement logistic regression for the provided dataset, so as to detect forged banknotes. Thus, it is a binary classification task. Logistic regression finds a linear discriminant boundary between the positive and negative classes, in feature space, and is implemented by gradient descent. The model learns a set of weights, which are used to predict the class for a new test sample. To optimize the learning of these weights, experimentation with different weight initializations, regularisation, and learning rates is performed. Further, the results are recorded and analysed for test samples.

**Dataset Description and Preprocessing**

It is required to test the implementation on the dataset provided in datasets/data\_banknote\_authentication.txt.

The dataset consists of 1372 examples, and has 4 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)

We take the dataset with examples stacked as rows, and after shuffling the entire dataset, perform a train : validation : test split of 0.75 : 0.15 : 0.15. Thus, the training set has 960 examples, the validation set has 206 examples and the test set, 206 examples.

We experiment with two types of feature scaling, namely *min-max normalisation* and *standardisation*, which are defined as follows:

**Min-Max Normalisation**

Min-max normalisation constrains the dataset to lie between -1 and 1. However, it does not bring all parameters to the same distribution. The cost function may be scaled to fit between -1 and 1, however, it is not made symmetric across all dimensions.

**Standardisation**

where **,** are the mean and standard deviation of **X**, respectively.

When performed for each feature, standardization makes sure that all parameters are from a distribution with mean 0 and variance 1. Thus, it makes the cost function symmetric across all dimensions, and enables gradient descent to proceed stably and smoothly.

For these reasons, we choose to employ standardisation for feature scaling.

In addition, the parameters **,** are calculated after splitting the dataset, and the train, validation and test sets are separately scaled using these parameters. This is done so that all preprocessing uses information from the training data set, as we require the test set to be untouched in order to give an unbiased estimate.

**Metrics for Performance Evaluation**

**Accuracy**

The accuracy calculated as,

captures how many data points (as a fraction of the total number of data points fed to the algorithm) were classified correctly. It ranges between 0 and 100 %. A higher accuracy is indicative of better performance by the model.

Relying only on accuracy to judge model performance however, may, in some cases, be misleading. Hence, we also calculate the F-score in order to better evaluate the model’s performance.

**F1-score**

The F1-score is calculated as,

where

,

and

.

Recall captures how many data points were correctly classified as positive points, as a fraction of the total number of actual positive data points in the dataset.

Precision captures the number of data points correctly classified as positive points, as a fraction of the total number of points classified as positive by the model.

Ideally, we would want a high recall, as well as high precision, but often there is a tradeoff between the two. In order to capture correctly the effect of both recall and precision on the evaluation of performance of the model, we use the *F1-score* which is the harmonic mean of recall and precision.

The F1-score ranges between 0 and 1. A higher F1-score is indicative of better performance by the model.

**Loss function**

As logistic regression is a binary classification model, we use the binary cross entropy loss function for gradient descent :

where , are the actual and predicted values of the target variable respectively, *n* is the number of training examples.

**Loss function with L1 regularisation**

where are the regularisation parameter and the number of features respectively.

**Loss function with L2 regularisation**

**Results**

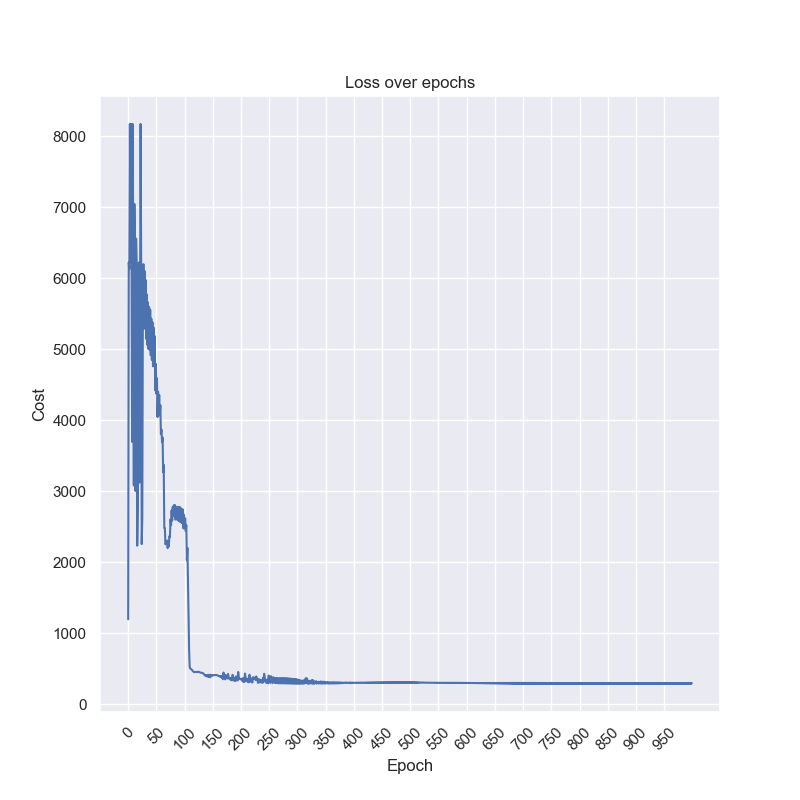
**i) Without regularisation**

1. **Choosing learning rate**

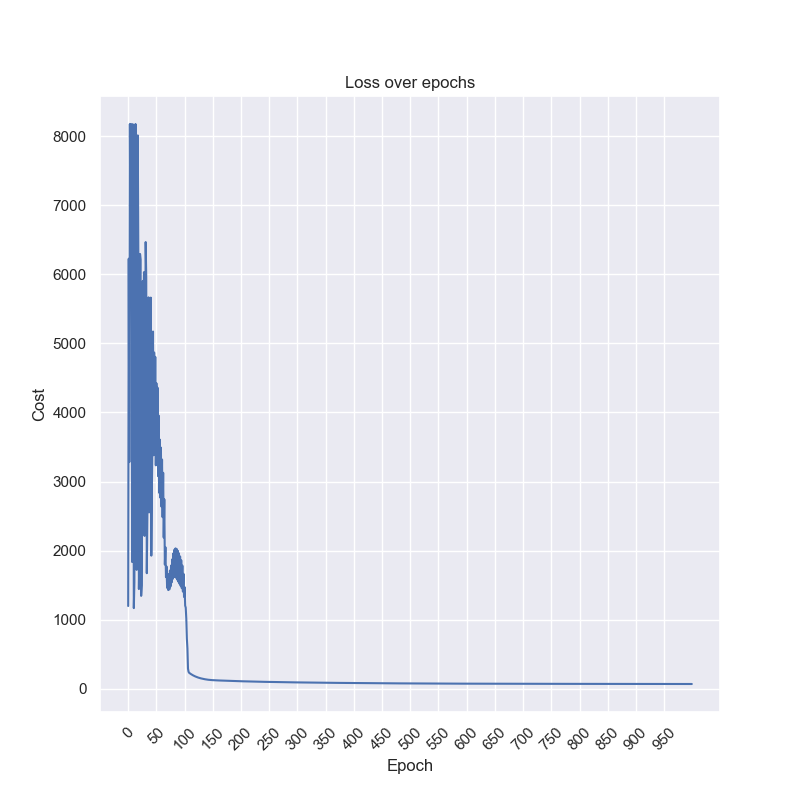
**Number of epochs = 1000**

To determine the learning rate which is better for faster convergence of the algorithm, we look to see the highest learning rate that continuously decreases the loss. This is done by plotting the loss function. If the loss function spikes, we look for a smaller learning rate.

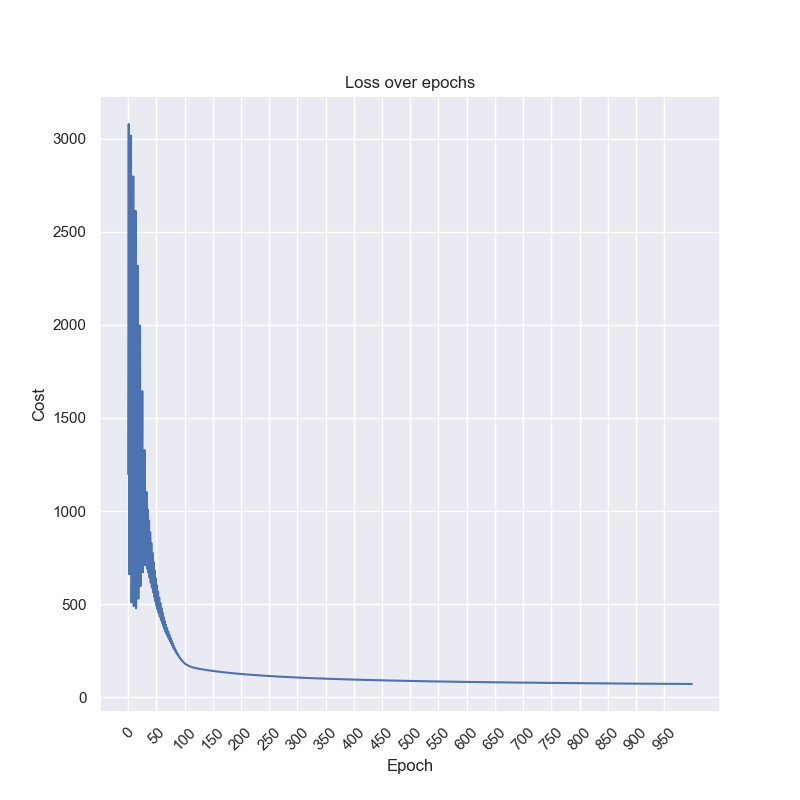
**Learning rate = 1**

****

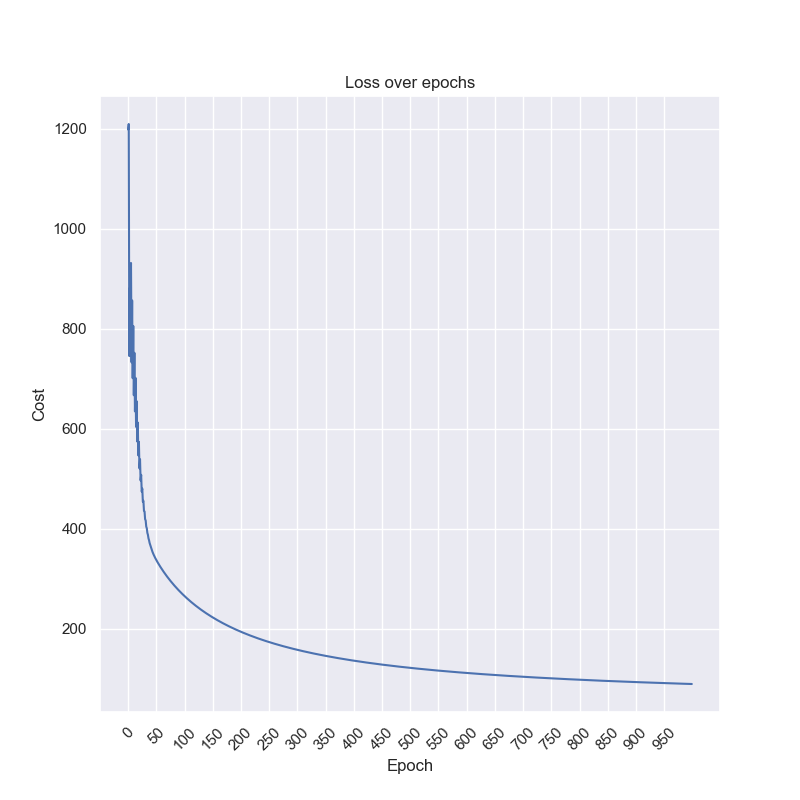
**Learning rate = 0.1**

****

**Learning rate = 0.01**

****

**Learning rate = 0.005**

****

As the learning rate 0.005 seems to provide smooth stable learning, we fix learning rate as 0.005.

1. **Choosing number of epochs**

**Learning rate = 0.005**

|  |  |  |
| --- | --- | --- |
| **Number of epochs** | **Train Accuracy** | **Validation Accuracy** |
| **100** | **91.98 %** | **90.77 %** |
| **500** | **97.60 %** | **96.12 %** |
| **1000** | **97.81 %** | **96.60 %** |
| **5000** | **98.125 %** | **96.60 %** |
| **10000** | **98.125 %** | **96.60 %** |

As the training accuracy does not significantly improve after 5000 epochs of the entire dataset, we fix the number of epochs as 5000.

1. **Choosing weight initialization**

**Learning rate = 0.005, Number of epochs = 5000**

We experiment with uniform and gaussian initialization of weights.

For uniform initialization, we constrain the weights to lie between -1 and 1. Gaussian initialization returns random float values for the weights, such that their mean is 0 and standard deviation is 1. Thus, the weights are initialized such that they are not far from zero, both on the positive and negative side.

|  |  |  |
| --- | --- | --- |
| **Initialization** | **Training accuracy** | **Validation accuracy** |
| **Uniform** | **98.125 %** | **96.60 %** |
| **Gaussian** | **98.125 %** | **96.60 %** |

As logistic regression on this dataset reached the global minima of the loss function quite soon, we don’t see much difference in uniform and gaussian initialization. Hence, we proceed with gaussian initialization.

**Weights obtained with these specifications (learning rate = 0.005, number of epochs = 5000, gaussian initialization of weights):**

**w0 = 36.88978382**

**w1 = -27.85759607**

**w2 = -25.02620473**

**w3 = -25.77764824**

**w4 = -0.92504928**

Here, **w0** is the bias, **w1, w2, w3, and w4** correspond to the first, second, third and fourth features respectively.

**Test accuracy = 99.03 %**

**Test F1-score = 0.98969072**

**ii) With regularisation**

1. **L2 regularisation**

**Finding the best L2 regularisation parameter, for learning rate = 0.005, 5000 epochs, gaussian weight initialization**

|  |  |  |
| --- | --- | --- |
| **Lambda** | **Training accuracy** | **Validation Accuracy** |
| **1.5** | **94.17 %** | **91.747 %** |
| **1** | **96.04 %** | **93.20 %** |
| **0.1** | **97.81 %** | **96.11 %** |
| **0.01** | **97.92 %** | **96.60 %** |

As the test accuracy itself was very high without regularisation, on increasing lambda value, we don’t see any improvement in the validation accuracy, and just a decrease in training accuracy. Hence, the model is not overfitting. Thus we choose a low value for regularisation parameter lambda, and fix it at 0.01.

**Weights obtained with these specifications (learning rate = 0.005, number of epochs = 5000, gaussian initialization of weights, L2 lambda = 0.01):**

**w0 = 33.21095657**

**w1 = -25.44401713**

**w2 = -22.51962104**

**w3 = -23.17605131**

**w4 = -0.47824845**

Here, **w0** is the bias, **w1, w2, w3, and w4** correspond to the first, second, third and fourth features respectively.

As can be observed, the weights obtained are smaller than those obtained without regularisation.

**Test accuracy (with L2 regularisation parameter = 0.01) = 99.03 %**

**Test F1-Score = 0.98969072**

1. **L1 regularisation**

**Finding the best L1 regularisation parameter, for learning rate = 0.005, 5000 epochs, gaussian weight initialization**

|  |  |  |
| --- | --- | --- |
| **Lambda** | **Training accuracy** | **Validation Accuracy** |
| **10** | **84.27 %** | **86.41 %** |
| **5** | **96.98 %** | **95.63 %** |
| **3.5** | **97.19 %** | **95.63 %** |
| **2.5** | **97.71 %** | **96.11 %** |
| **2** | **97.81 %** | **96.60 %** |
| **1.5** | **97.81 %** | **96.11 %** |

As the test accuracy itself was very high without regularisation, on increasing lambda value, we don’t see much improvement in the validation accuracy. There was a slight increase in validation accuracy on increasing the regularisation parameter to 2 from 1.5. Hence we choose the L1 regularisation parameter as 2. There is very little (if any) overfitting.

**Weights obtained with these specifications (learning rate = 0.005, number of epochs = 5000, gaussian initialization of weights, L1 lambda = 2):**

**w0 = 19.5168441**

**w1 = -16.2206245**

**w2 = -12.7291103**

**w3 = -12.4134922**

**w4 = 0.005936077**

As can be observed, the weights obtained are even smaller than those obtained with L2 regularisation. The last weight corresponding to the feature 4 is almost reduced to 0, a characteristic observation in L1 regularisation (weights tend to zero).

**Test accuracy (with L1 regularisation parameter = 2) = 99.03 %**

**Test F1-Score = 0.98969072**

**Feature importance**

If we compare the magnitudes of the weights that have been obtained, we notice that in each case, the weight **w0** (bias) has the greatest value. Next highest magnitude value is that of weight **w1,** corresponding to feature 1. Weights **w2 and w3** (corresponding to features 2 and 3) are almost equal in magnitude, and the weight **w4** (corresponding to feature 4) is much smaller than the rest.

The weights **w1** through **w4** signify the importance given to the numerical value of their respective features. As all features share the same variance and mean, as guaranteed by standardizing the data, we can interpret this numerical weighting as the significance of a particular attribute in predicting the class of that test sample. The weight **w0** represents the bias term, which in context of a linear classifier such as logistic regression, simply means that the discriminant line does not pass through the origin.

Following this analysis, we can conclude that **feature 1 (variance of wavelet transformed image)** is of the highest significance for prediction, closely followed by both **feature 2 (skewness of wavelet transformed image) and feature 3 (curtosis of wavelet transformed image)**, which have nearly the same significance. **Features 1, 2 and 3** also exhibit negative correlation with the target attribute, as can be seen from the negative values of the weights 1, 2 and 3 respectively. However, **feature 4 (entropy of image)** has very little significance in prediction, and does not greatly influence model performance.