Muhammad Nasir Khurshid, 04071913019

Bank Term Deposit Classifier

Problem

Financial institutions do marketing campaigns based on phone calls, in order to assess if the product (bank term deposit) would be subscribed by the client or not. We want to find ways to look for future strategies in order to improve future marketing campaigns for the bank and predict whether a client has subscribed a term deposit or not.

Data Set

Features

1. Age
2. Default - has credit in default? (Categorical: 'no', 'yes', 'unknown')
3. Balance
4. Housing - has housing loan? (Categorical: 'no', 'yes', 'unknown')
5. Loan - has personal loan? (Categorical: 'no', 'yes', 'unknown')
6. Day - last contact day of the week (Categorical: 'mon', 'tue', 'wed', 'thu', 'fri')
7. Duration - last contact duration, in seconds (numeric)
8. Campaign - number of contacts performed during this campaign and for this client
9. Pdays - number of days that passed by after the client was last contacted
10. Previous - number of contacts performed before this campaign and for this client

Ground Truth (Label)

- Deposit - has the client subscribed a term deposit? (binary: 'yes', 'no')

Approach

- Logistic Regression & Sigmoid Function
- Neural Network With 1 Hidden Layer & 2 Units
- K-Nearest Neighbors Using L1-Norm, L2-Norm & L∞-Norm

Data Processing

Data normalization is performed on features using following formula.

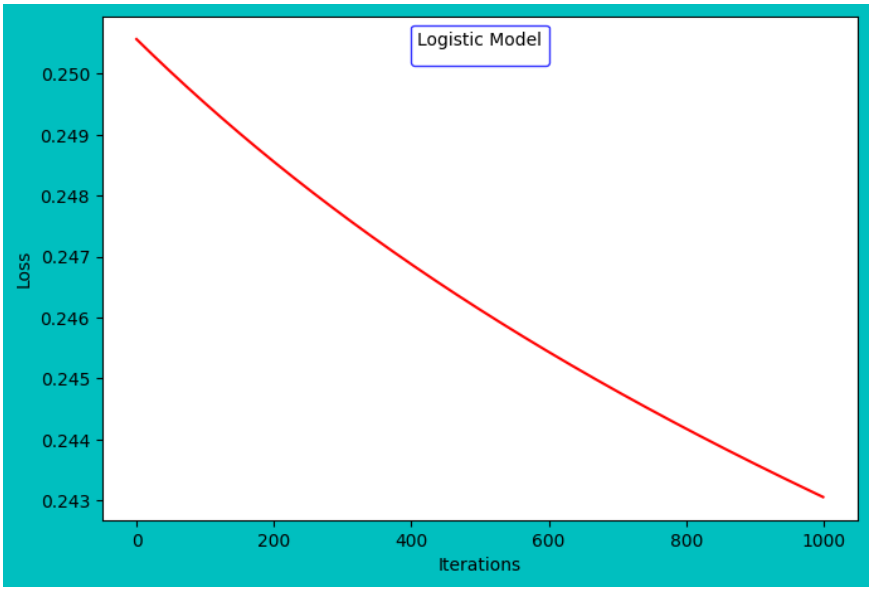
$$data = \frac{data - data.min()}{data.max() - data.min()}$$

This formula normalizes all the values between 0 and 1 inclusive.

Logistic Regression with Sigmoid Function

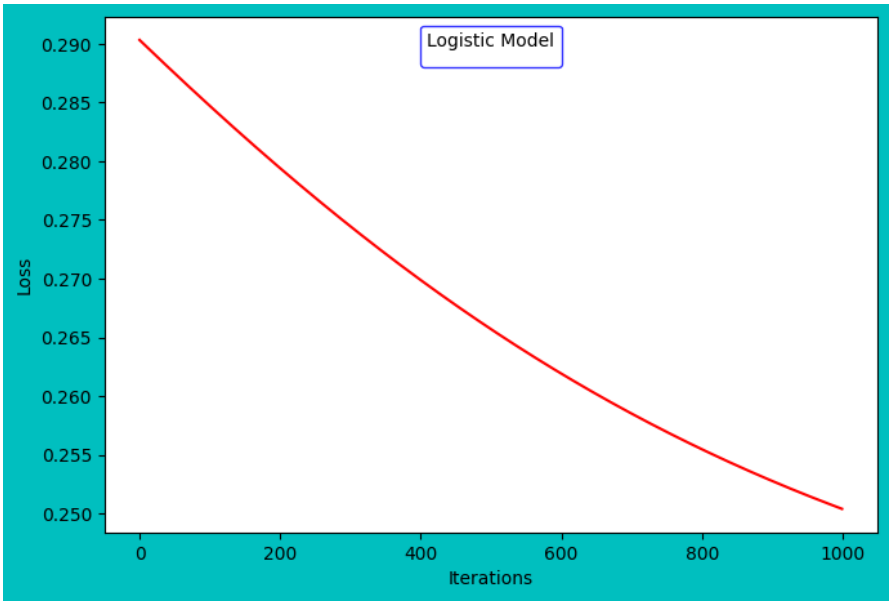
Initialization of W with Zeros

- Learning rate: 0.01
- Iterations: 1000
- Loss: 0.24305545356230504

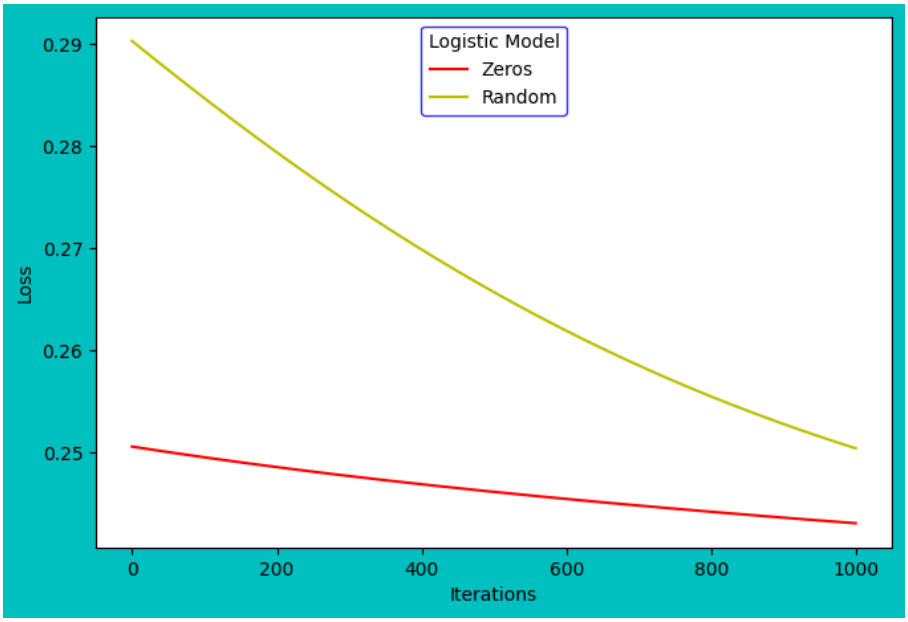


Initialization of W with Random Values

- Learning rate: 0.01
- Iterations: 1000
- Loss: 0.25039892110335304



Comparison of both initializations

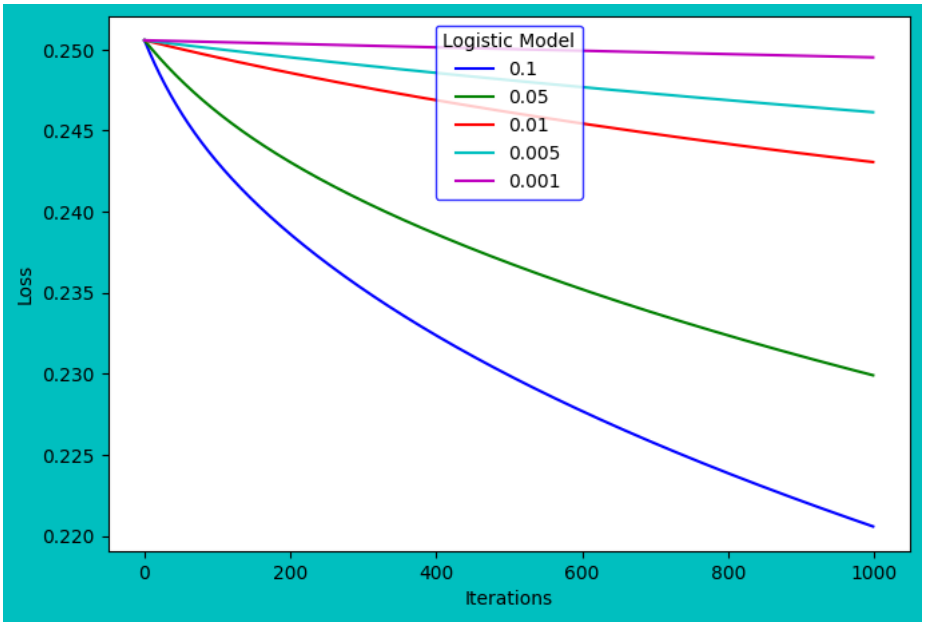


It can be clearly seen from the comparison that initialization of W with zeros performs better for the current problem because the loss for Zeros is minimized earlier than the loss in case of random values.

W initialization with Zeros

Training model for various Learning rates and comparing results:

- Iterations: 1000



Steps:

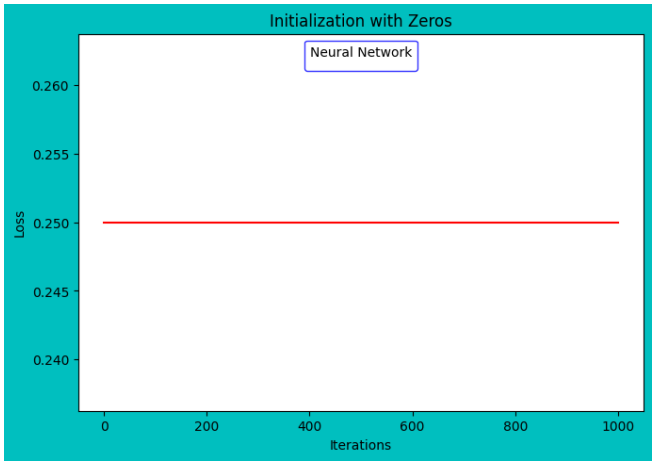
- Taking the best learning rate (0.1)
- Splitting the data into train and test in 80:20 using sklearn library's function train_test_split
- Training the model with 0.1 learning rate at 10000 iterations
- Getting the trained parameter W
- Predicting the values of test data using the W =
  ([-0.44455653, -0.23935802, 0.76122738, -1.08064025, -0.5999505 ,
  -0.46416935, 7.37075107, -1.4630327 , 2.44260597, 1.02248575])
- Comparing the obtained results with original results and computing accuracy
- Obtained accuracy is **0.7393767705382436.**

(continued…)

Single Layered Neural Network with 2 Units

Initialization of V & W with Zeros

- Learning rate: 0.01
- Iterations: 1000
- Loss: 0.25

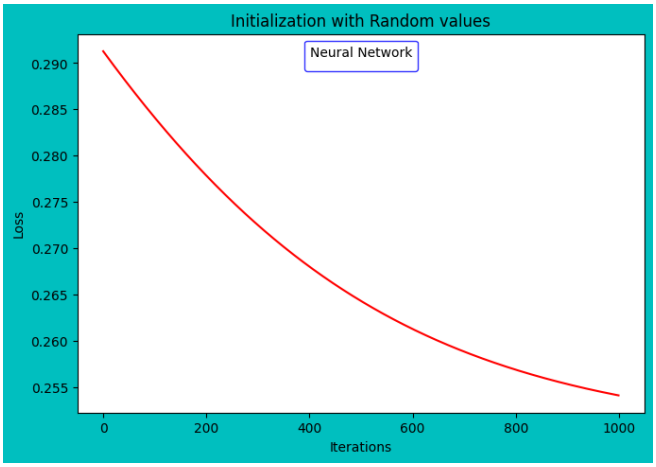


When we initialize V and W with zeros then loss remains constant. It is because in the formula of gradients for V and W, the whole expression is multiplied with W which makes the results of gradients zero. So, when we get zero gradient, we can not update the value of W as per following formula:

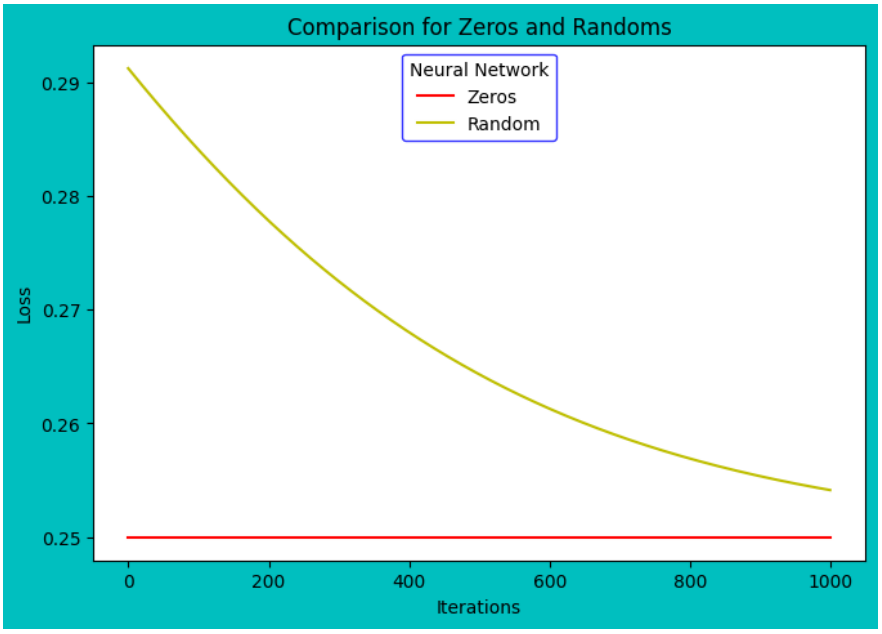
$$w = w - \propto \nabla_w TrainLoss$$

Initialization of V & W with Random Values

- Learning rate: 0.01
- Iterations: 1000
- Loss: 0.25534165796820346



Comparison of both initializations

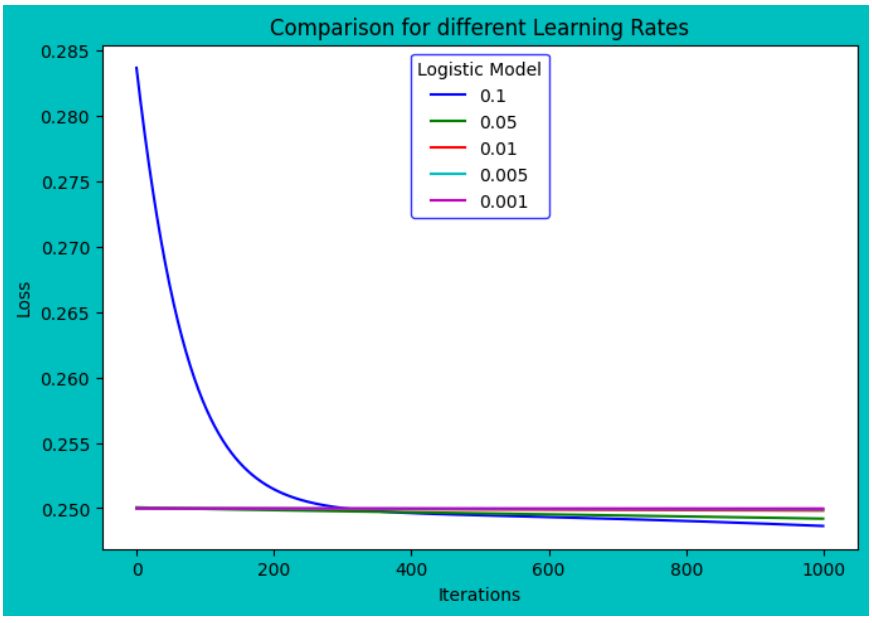


It can be clearly seen from the comparison that initialization of V and W with random values performs better for the current problem because the loss for Zeros never changes and in case of random values the loss is minimized efficiently.

V & W initialization with Random Values

Training model for various Learning rates and comparing results:

- Iterations: 1000



Steps:

- Taking the best learning rate (0.1)
- Splitting the data into train and test in 80:20 using sklearn library's function train_test_split
- Training the model with 0.1 learning rate at 10000 iterations
- Getting the trained parameter W = ([4.23808431, -4.25331626])
- Predicting the values of test data using the W
- Comparing the obtained results with original results and computing accuracy
- Obtained accuracy is **0.7393767705382436.**

(continued…)

K – Nearest Neighbors for K = 3

There are three common similarity measures used in KNN:

1. L1-Norm
2. L2-Norm
3. L∞-Norm

All the three similarity measures are used to predict the values.

Steps:

- Splitting the data into train and test in 80:20 using sklearn library's function train_test_split
- Setting K = 3 to predict the classes
- Predicting the classes using L1-Norm
- Accuracy using L1-Norm: 0.49291784702549574
- Predicting the classes using L2-Norm
- Accuracy using L2-Norm: 0.49575070821529743
- Predicting the classes using L∞-Norm
- Accuracy using L∞-Norm: 0.5042492917847026
- Setting K = 5 to predict the classes
- Predicting the classes using L1-Norm
- Accuracy using L1-Norm: 0.5042492917847026
- Predicting the classes using L2-Norm
- Accuracy using L2-Norm: 0.49008498583569404
- Predicting the classes using L∞-Norm
- Accuracy using L∞-Norm: 0.5042492917847026

From above results, it can be seen that for L1-Norm and L∞-Norm, K = 5 performs relatively better than K = 3. The small value of accuracy for all the Norms may be due to the noise or some missing features in data.