Machine Learning Assignment-2

Documentation

1. Logistic Regression

Logistic Regression is a machine learning algorithm primarily used for binary classification algorithms. Here the model is required to predict whether the given point belongs to class A or to class B, where the classes are mutually exclusive.

Implementation

The dataset was trained and tested on 10 random splits of 70:30 ratio.

For gradient descent, a descent function was used which was called <number of iteration> times.
This function used the derivative of the loss function and subtracted the dot product of the derivatives and entire Training Data from the weights.

$$C = -\frac{1}{m}(y\ log(y^T log(h) + (1-y)^T log(1-h)$$

$$h = s(W.x^T)$$

$$dW = \frac{x.(h-y)}{m}$$

$$db = \frac{h-y}{m}$$

All the equations above are vectorized.

For stochastic gradient descent, the descent function was called <number of iteration> times, updating weights for each data-point one at a time and instead of taking dot product of the entire training set, only the current sample is multiplied with the derivative of loss for updating the weights.

Difference between SGD and GD:

In stochastic gradient descent the weights are updated very fast i.e. for each sample the weights are adjusted. As a result the optimum weights are arrived upon in a lesser number of iterations. This is less computationally intensive but might take longer as the weight update for each input can't be vectorized like in the case of GD wherein the entire dataset is passed in one go.
In GD since the entire dataset is passed in one go, the weights are updated very less frequently and since it is vectorized it is more computationally intensive. However if vectorization is possible then GD is faster even if it takes a larger number of iterations, because computations are evaluated parallely.
So the choice between these two depends on the kind of hardware available and the kind of dataset (for smaller datasets, GD should generally be much faster). If you one has a larger memory and faster CPU then GD/MBGD is expected to perform better, however if resources are limited one is better off using SGD even though it might take longer to evaluate.

Results

Final test and train metrics:

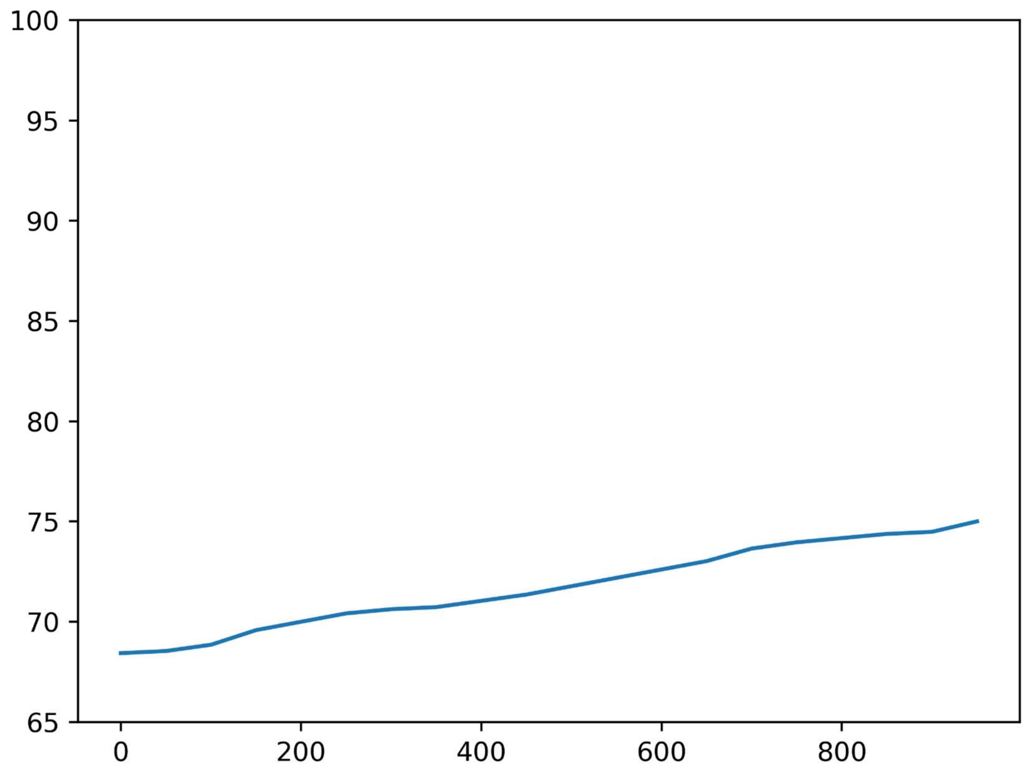
GD:
Training Accuracy (10 random splits) : 98.85416666666666
Training Precision (10 random splits) : 0.9835231691737334
Training Recall (10 random splits) : 0.990896422829312
Training F-Score (10 random splits) : 0.9871960286975288
Training loss (10 random splits) : 0.03704528357332083

Testing Accuracy (10 random splits) : 98.95631067961165
Testing Precision (10 random splits) : 0.9841458085479381
Testing Recall (10 random splits) : 0.9924925479416025
Testing F-Score (10 random splits) : 0.9883015553806088

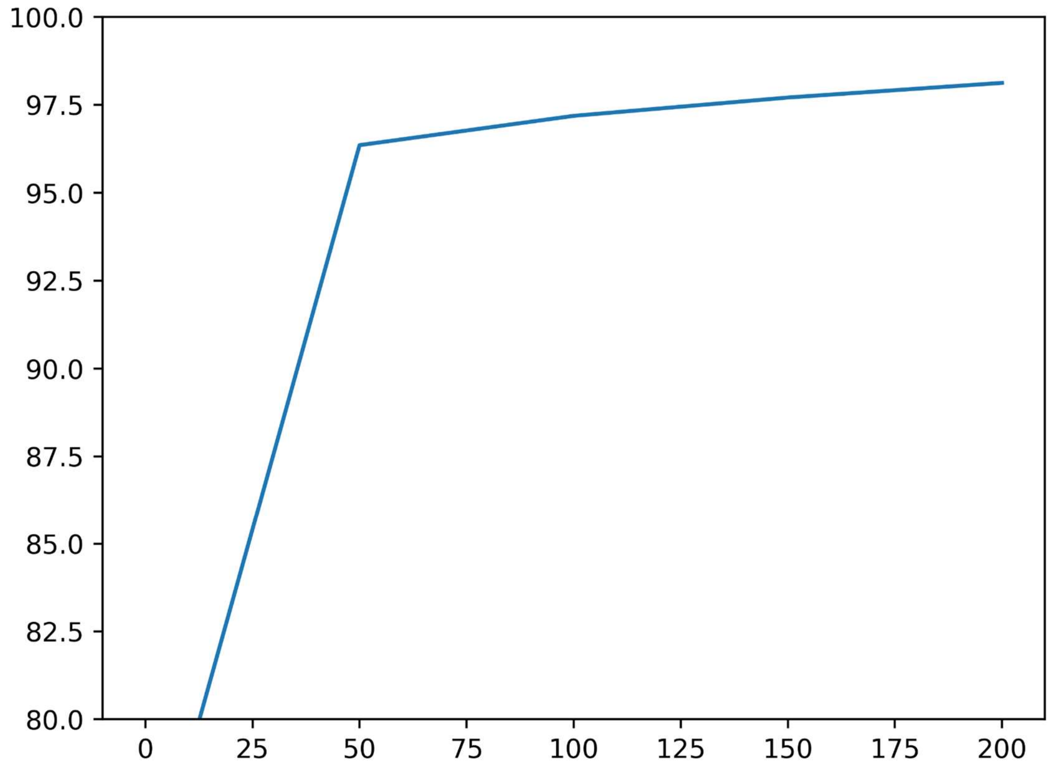
```
Testing loss (10 random splits) :  0.03676691150719037



SGD:
Training Accuracy (10 random splits) :  98.77083333333334
Training Precision (10 random splits) :  0.9846436819024855
Training Recall (10 random splits) :  0.9878123478118713
Training F-Score (10 random splits) :  0.986225469694322
Training loss (10 random splits) :  0.050865832541637124

Testing Accuracy (10 random splits) :  98.6893203883495
Testing Precision (10 random splits) :  0.985699589180606
Testing Recall (10 random splits) :  0.9847897869474439
Testing F-Score (10 random splits) :  0.9852444780298798
Testing loss (10 random splits) :  0.05514615170088646
```

Plots

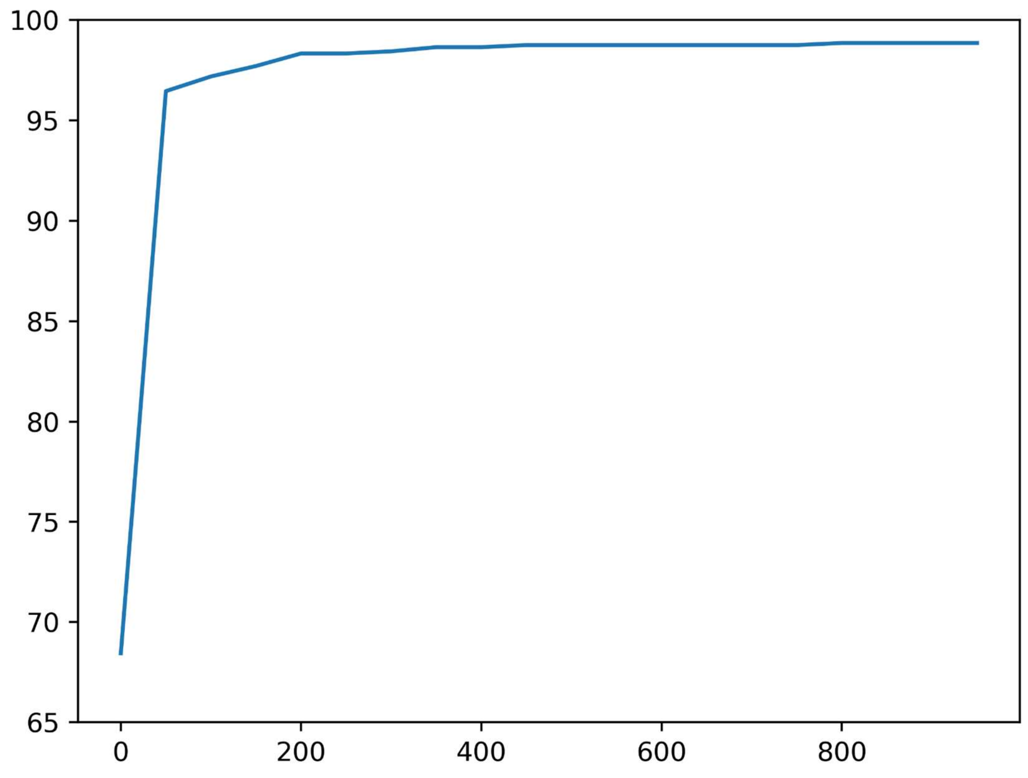
Plot of Accuracy for GD with learning rate = 0.0001



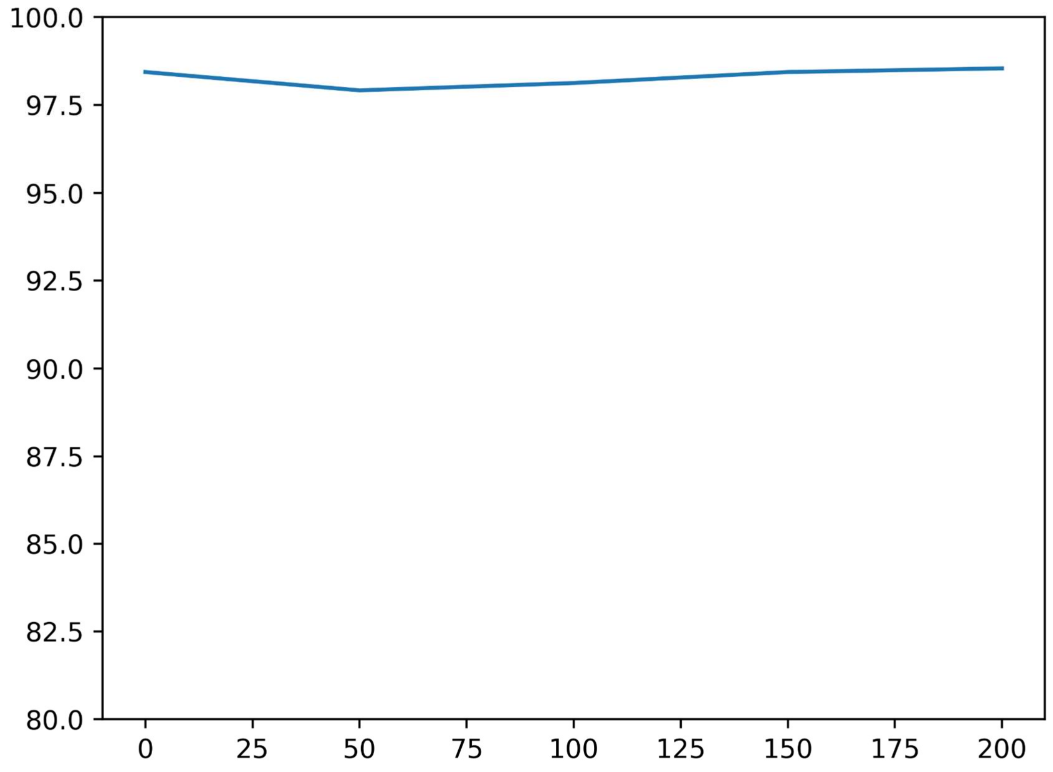
Plot of Accuracy for SGD with learning rate = 0.0001



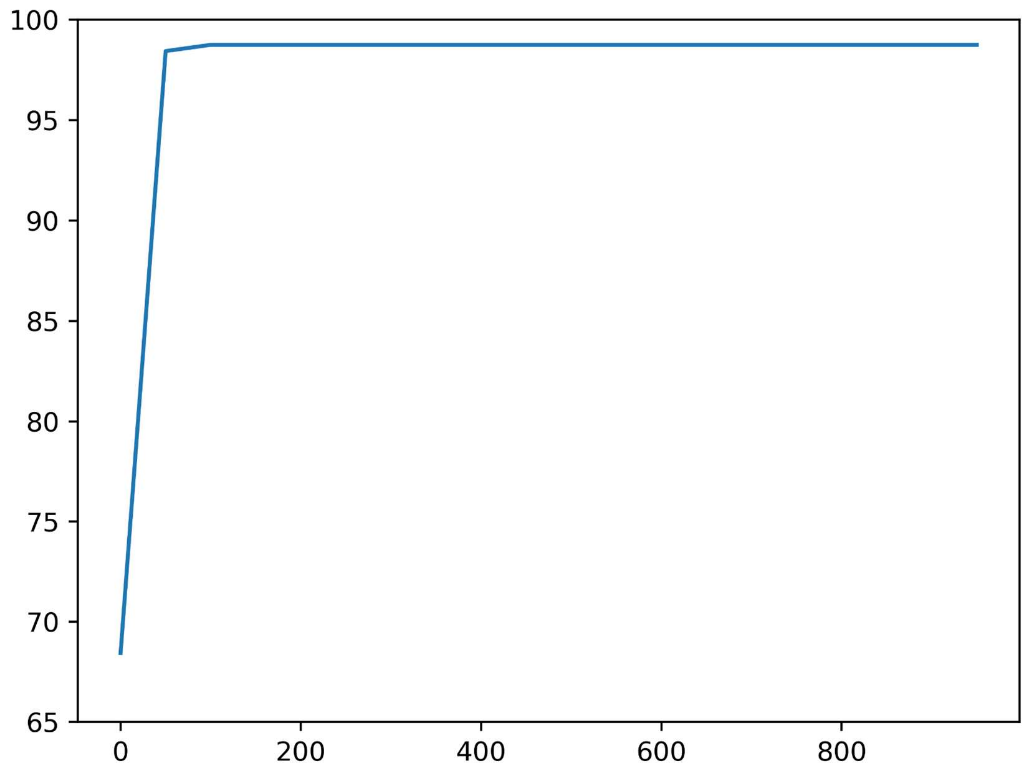
Plot of Accuracy for GD with learning rate = 0.1



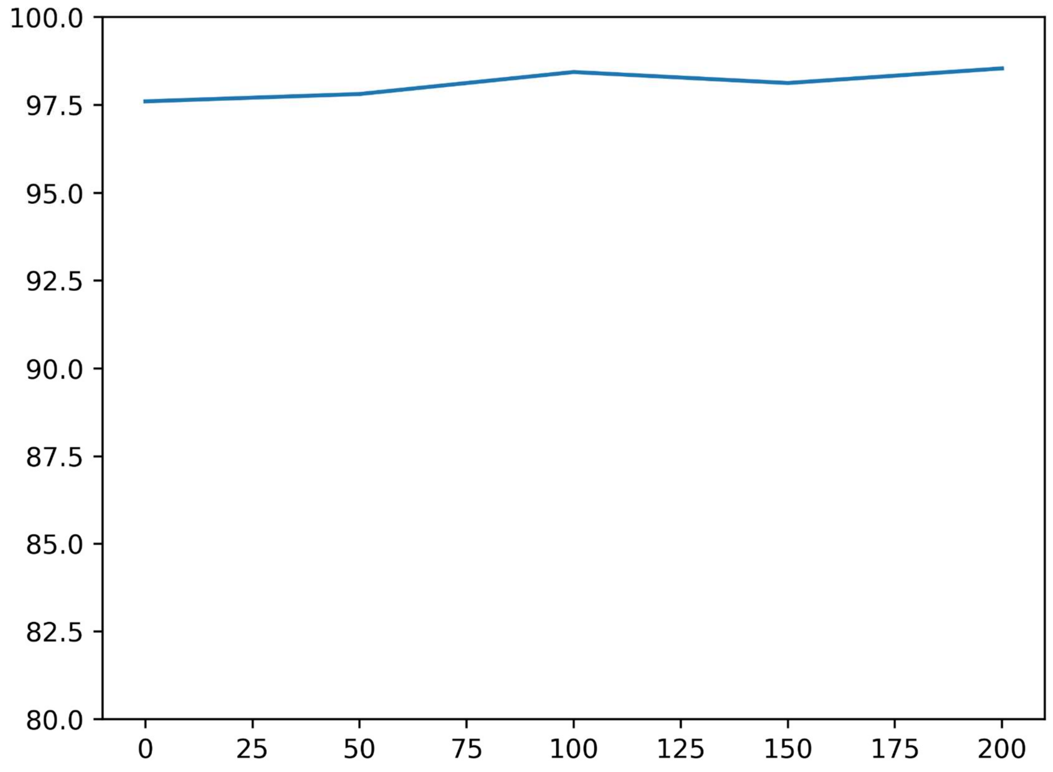
Plot of Accuracy for SGD with learning rate = 0.1



Plot of Accuracy for GD with learning rate = 5



Plot of Accuracy for SGD with learning rate = 5



100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
0
25
50
75
100
125
150
175
200