Report on
Logistic Regression

Team

1. Sai Satvik Vuppala (2017B4A71449H)
2. Shreya Kotagiri (2017B3AA0296H)
3. Dhanush Karupakula(2017B3A71011H)

Introduction

We implement logistic regression for binary classification. We train the model using Gradient Descent and Stochastic Gradient Descent with three different learning rates. We made 10 independent random 70:30 splits on the given data.

Libraries Used

1. Numpy
2. Pandas
3. Random
4. Matplotlib

Implementation

This algorithm tries to solve the binary classification problem. We find the posterior probability directly instead of from the prior probability, thus reducing the number of parameters needed for estimation. The probability is modeled by a logistic function(sigmoid function), which models the probability of a given tuple given its features.

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

In the case of M mature features,

$$P(T = 1/w) = \frac{1}{1+e^{-(w1.\Phi(x1)+\ldots+wM.\Phi(xM)}} = y_1$$

$$P(T = 0/w) = 1 - \frac{1}{1+e^{-(w1.\Phi(x1)+\ldots+wM.\Phi(xM)}} = 1\text{-}y_1$$

In general $P(T=t1)=y_1^{t1}.(1-y_1)^{1-t1}$

So as part of this, we estimate the parameters w_1, w_2....w_M such that the probability of each example reaching its correct class(ground truth) is maximised.

i.e, maximise $y_1^{t1}.(1-y_1)^{1-t1}$

We maximised this combined probability using gradient descent and stochastic gradient descent, using three different learning rates of

We proceed through the whole process in the following steps:

1) Train-test split-

   The dataset with 4 attributes and a target class has been split into 10 independent 70-30 splits for training and testing purposes.

2) Defining the loss(cost function) and their derivatives for GD and SGD to find weights that best estimate the probability(sigmoid function)-

The cost function E for GD and SGD(negative of which need to be minimized) and their derivatives(equal to 0) are $-\sum_{n=1}^{N}(t_n.\log y_n+(1-t_n).\log(1-y_n))$ and $\sum_{n=1}^{N}(y_n-t_n).\Phi(xn)$

For the subsequent iteration, w would be updated as,

$$w^{k+1} = w^{k} + \eta(\nabla(E(x_n)))$$

We find this cost function using three different learning rates of 0.65, 0.3, 0.2

3) Modeling the logistic regression-

   From the cost function defined above, we implement the logistic regression using the w values predicted above using either GD or SGD to find the final probabilities.

Most important feature

Optimal η for GD is 0.1 and its corresponding features for 10 iterations are as below:

[-2.5053395011441206, -1.5859682701822717, -1.6346605115341555, -0.8388766163489728]

[-2.704049604206519, -1.683888043377677, -1.7638308643054328, -0.8467033073400532]

[-2.6526237059416333, -1.6181826054762563, -1.7271110903989826, -0.8950377356359108]

[-2.524579160389126, -1.5631203616973501, -1.6466537279776574, -0.7626993925337276]

[-2.722970481588964, -1.6885775231958045, -1.7752314676777556, -0.9238333671031972]

[-2.8958976783612576, -1.7385664316342644, -1.8496177662312232, -0.7900708434591098]

[-2.45459460653867, -1.5217658499684765, -1.6160546448458375, -0.7481905270663857]

[-2.589987756938105, -1.6873025162930757, -1.7750083464320852, -0.8168952518146475]

[-2.704034096737561, -1.5961025801815683, -1.6847759903775472, -0.7462002326631328]

[-2.6336624084455997, -1.6580176602160357, -1.7612392389452245, -0.8039344817461458]

Optimal η for GD is 0.9 and its corresponding features for 10 iterations are as below:

[-1.09359777659756, -0.5494752390214777, -0.543920584722556, -0.30457573398466076]

[-1.0710210380749527, -0.528814934452572, -0.4997054546712983, -0.2878070625031467]

[-1.077726338612576, -0.5572348452348411, -0.509781174466149, -0.31663463429538896]

[-1.0036077279114015, -0.5320222790995524, -0.48190764735818, -0.3041124582625317]

[-1.0917563399312118, -0.5568593475262846, -0.5406071599122001, -0.3327525418414671]

[-1.1181635463575674, -0.5580775086938273, -0.5434280932957295, -0.28700128979609957]

[-1.1212129413772254, -0.5236994901895218, -0.5452486724694245, -0.3068212307910015]

[-1.058791439785771, -0.5575179273747184, -0.5155531951420471, -0.31960944188470997]

[-1.060837382261309, -0.5713716856333497, -0.5431167848399537, -0.2964942383857477]

[-1.0957980531556661, -0.5354275497449095, -0.5593892385923712, -0.29521189301530915]

The above are the feature values for the 10 independent random splits for the models GD and SGD respectively. Large positive values of the feature indicated higher importance of the ith feature in positive class and the large negative values of features indicates higher importance of ith feature in the prediction of negative class. And it can be clearly observed from the above that **feature 1** is observed to be of highest magnitude thereby making it the most important feature.

Final train and test metrics

Gradient Descent:

It is observed that for the algorithm we implemented the ideal ηvalue is observed to be 0.1 for the gradient descent algorithm. And the final test and train metrics for the model are as follows:

```
____TRAIN DATA____

Accuracies over each iteration: [95.83333333333334, 95.9375,
96.04166666666667, 95.3125, 96.04166666666667, 95.9375, 95.0,
96.04166666666667, 95.72916666666667, 95.625]

Precisions over each iteration: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0]

Recalls over each iteration: [0.9071925754060325, 0.9088785046728972,
0.9075425790754258, 0.8960739030023095, 0.9107981220657277,
0.9093023255813953, 0.8865248226950354, 0.9086538461538461,
0.9026128266033254, 0.9011764705882352]

Average Accuracy: 95.750

Average Cost: 0.099

____TEST DATA____

Accuracies over each iteration: [95.87378640776699, 95.63106796116504,
95.3883495145631, 97.0873786407767, 95.63106796116504, 95.63106796116504,
97.81553398058253, 95.14563106796116, 96.35922330097088,
96.60194174757282]

Precisions over each iteration: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0]

Recalls over each iteration: [0.9050279329608939, 0.9010989010989011,
0.9045226130653267, 0.9322033898305084, 0.9021739130434783, 0.9,
0.9518716577540107, 0.8969072164948454, 0.9206349206349206,
0.9243243243243243]

Average Accuracy: 96.117
```

Stochastic Gradient Descent:

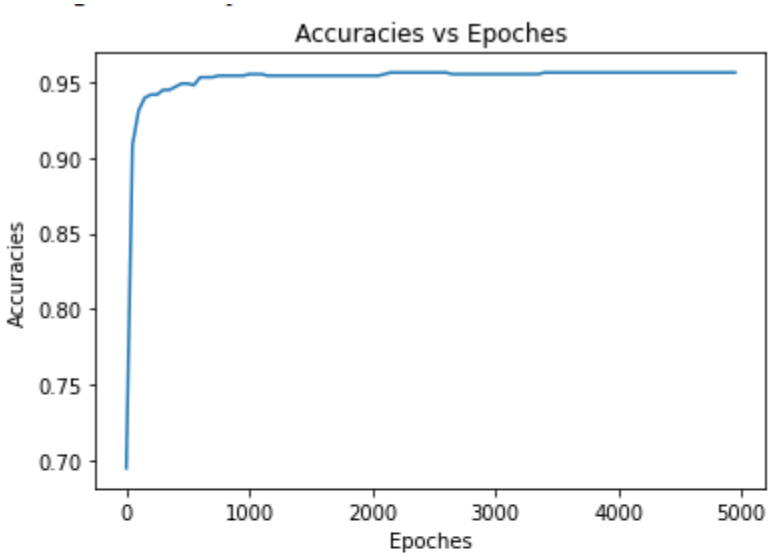
It is observed that for the algorithm we implemented the ideal ηvalue is observed to be 0.9 for the stochastic gradient descent. And the final test and train metrics for the model are as follows:

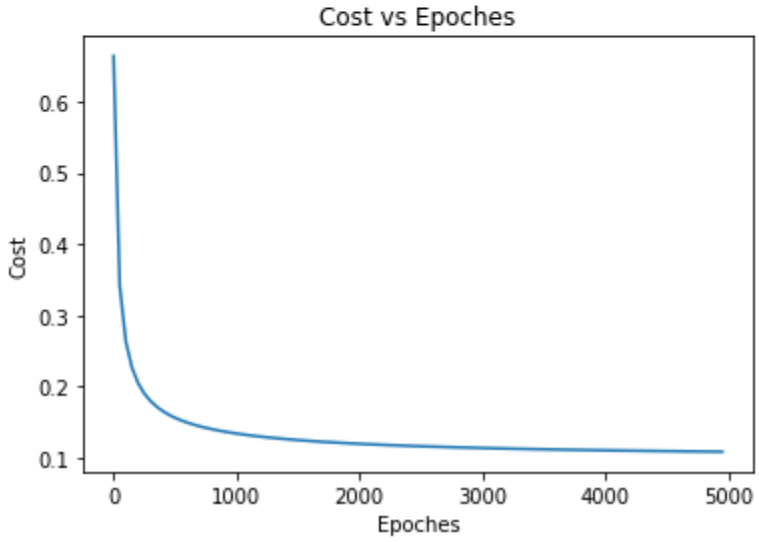
```
____TRAIN DATA____

Accuracies over each iteration: [95.52083333333333, 95.3125, 95.0,
94.27083333333334, 95.3125, 95.20833333333333, 95.10416666666667, 94.6875,
94.6875, 95.625]

Precisions over each iteration: [1.0, 0.9895287958115183,
0.9950617283950617, 0.9794871794871794, 0.9947643979057592, 1.0,
0.9892761394101877, 0.9923273657289002, 0.9923076923076923,
0.9944598337950139]

Recalls over each iteration: [0.9029345372460497, 0.9021479713603818,
0.8975501113585747, 0.8904428904428905, 0.8983451536643026,
0.892018779342723, 0.8956310679611651, 0.8899082568807339,
0.8896551724137931, 0.899749373433584]

Average Accuracy: 95.073

Average Cost: 0.145

____TEST DATA____

Accuracies over each iteration: [95.63106796116504, 94.66019417475728,
95.14563106796116, 95.63106796116504, 95.3883495145631, 95.87378640776699,
94.66019417475728, 96.35922330097088, 96.60194174757282,
94.41747572815534]

Precisions over each iteration: [1.0, 0.9941520467836257,
0.9795918367346939, 0.9822485207100592, 0.9941176470588236, 1.0,
0.9888888888888889, 0.9877300613496932, 0.9878787878787879,
0.9947368421052631]

Recalls over each iteration: [0.8922155688622755, 0.8900523560209425,
0.8944099378881988, 0.9171270718232044, 0.9037433155080213,
0.907608695652174, 0.898989898989899, 0.9252873563218391,
0.9314285714285714, 0.8957345971563981]

Average Accuracy: 95.437
```

Plots

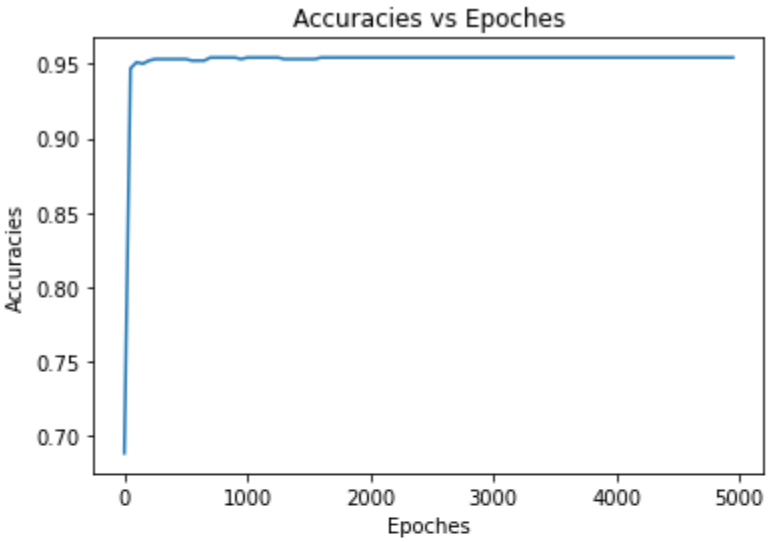
Gradient Descent:

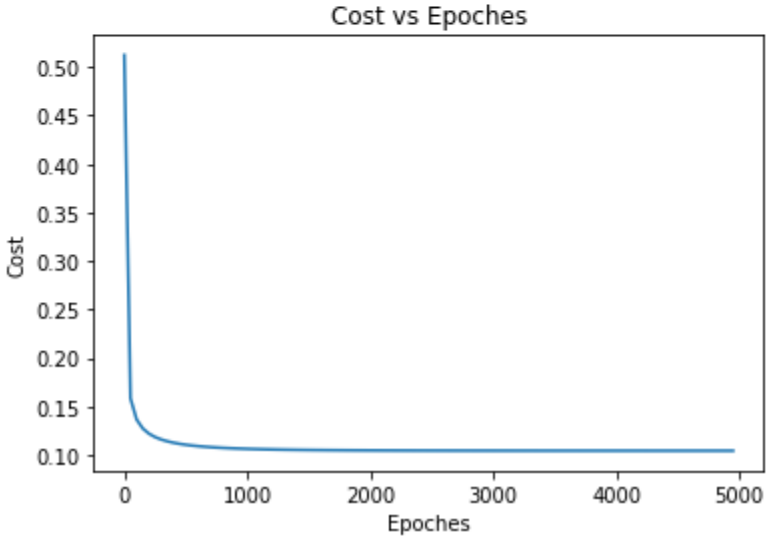
Case1 : $\eta 1 = 0.01$



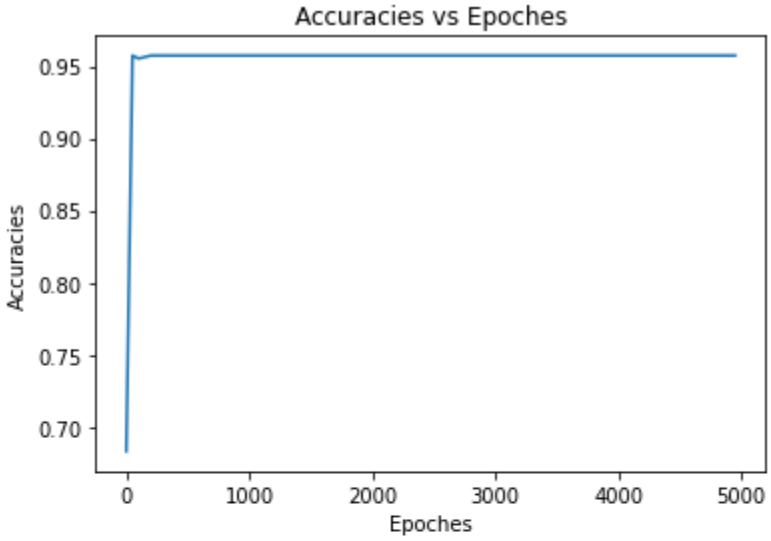


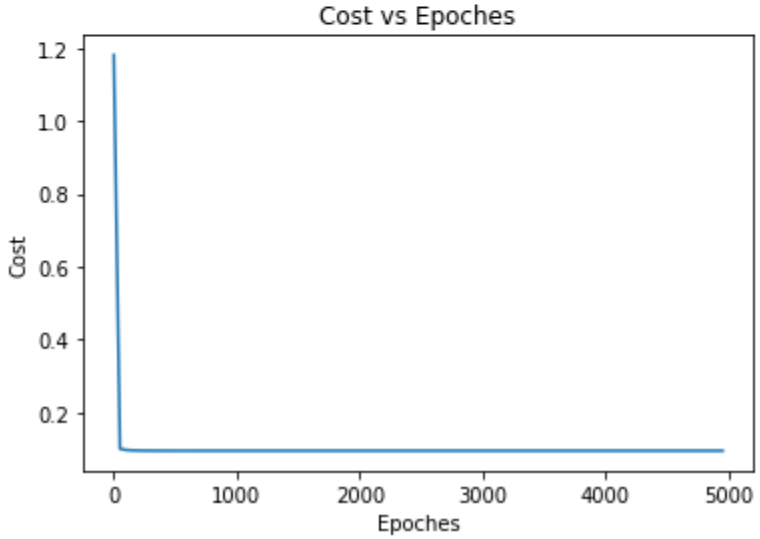
Case2 : $\eta 2 = 0.1$





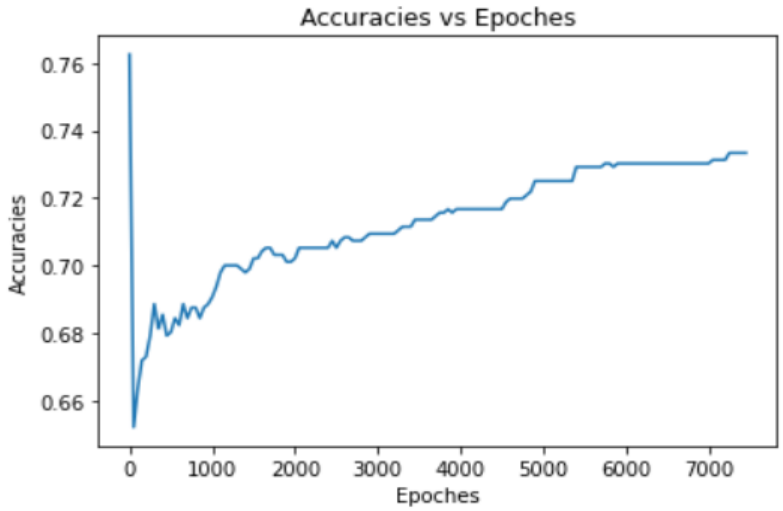
Case3 : $\eta 3 = 0.9$

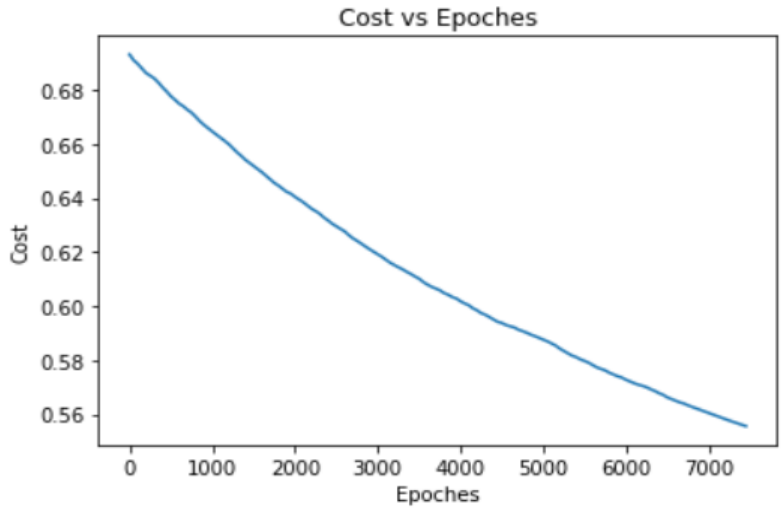




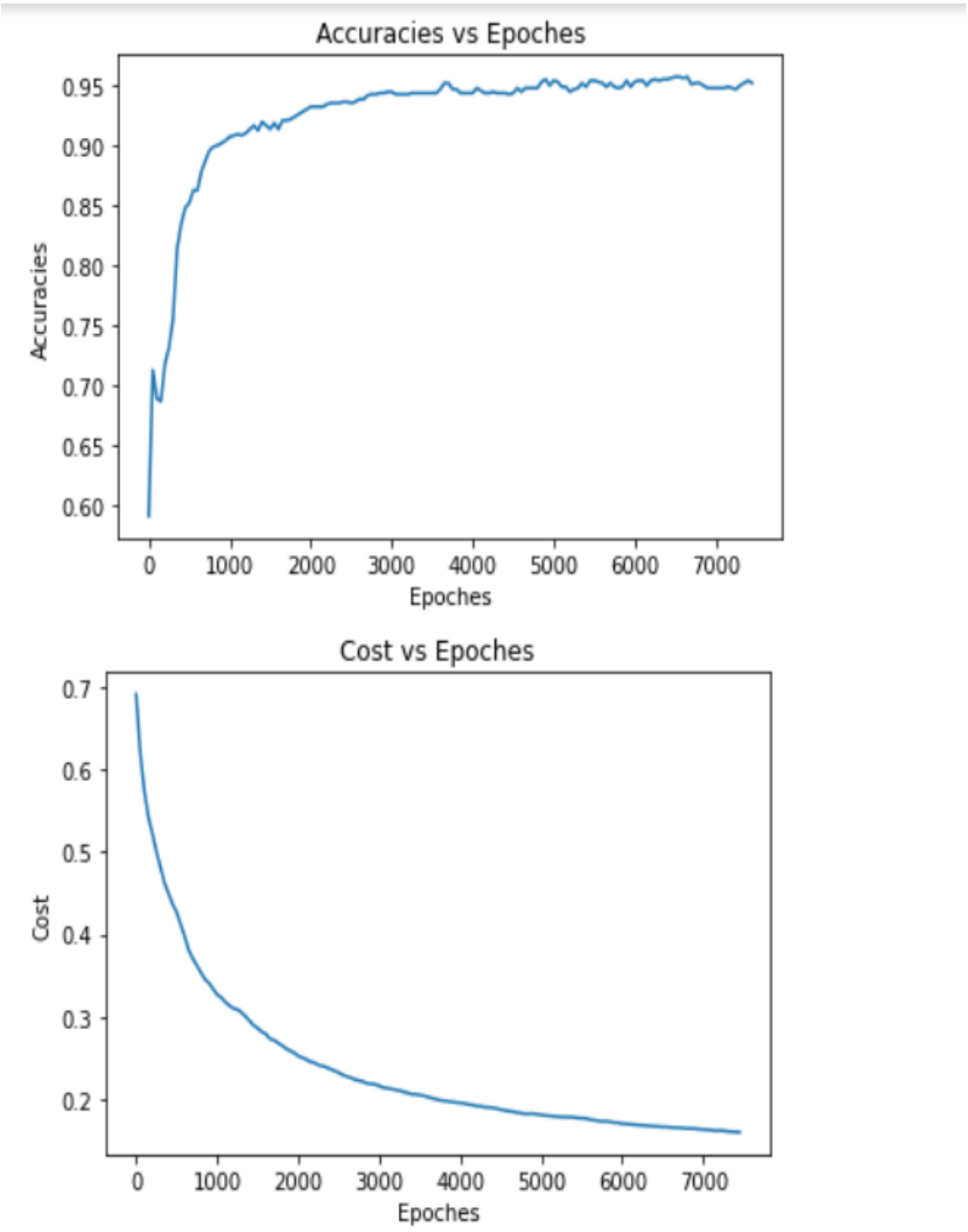
Stochastic Gradient Descent:

Case1 : $\eta 1 = 0.01$





Case2 : $\eta 2 = 0.6$



Case3 : $\eta 3 = 0.9$