Predicting Malignant or Benign using Logistic Regression Algorithm

Swati Sajee Kumar
Department of Computer Science Engineering
swatisaj@buffalo.edu #50289560

Abstract

To perform Classification using Logistic Regression on Wisconsin Diagnostic Breast Cancer dataset. The cost minimization is done using gradient descent algorithm. The suspected FNA cells (or predicted output) has been classified into Benign (Class 0) or Malignant (Class 1).The prediction accuracy , precision and recall values were obtained.

1 Introduction

Logistic Regression is an algorithm used for Supervised Learning , which is used for classification purpose. In our project we use Logistic Regression for Binary Classification, that is $y \in \{0,1\}$ where y is the output.For further discussion the symbols and variables used would be as shown in Table 1:

Table 1: Variables used in the report

VARIABLE	FUNCTION
y	Actual Output Expected (0 or 1)
$h_\theta(x)$	Predicted output or Hypothesis function (0 or 1) using the training model
θ	Weights
X	Input data
m	Number of samples
n	Number of features
$J(\theta)$	Cost Function ,total cost for the m sample dataset

- The hypothesis function h_θ lies between 0 and 1 , that is $0 \leq h_\theta \leq 1$.
- For logistic regression hypothesis function can be written as :

  $h_\theta(x) = g(\theta^T x)$ …… Eq(1)

  where $g(z) = 1/1+e^{-z}$, on substituting the z value we get

  $g(\theta^T x) = 1/1+e^{-\theta T x}$ …… Eq(2)

  This is called Sigmoid Function / Logistic Function
- $h_\theta(x)$ can be interpreted using two probability functions as shown below:

  $h_\theta(x) = P(y=1 \mid x; \theta)$ or $h_\theta(x) = P(y=0 \mid x; \theta)$

  Also, $P(y=1 \mid x; \theta) = 1 - P(y=0 \mid x; \theta)$
- Together with the Sigmoid function , probability function, the decision boundary can be created with a threshold = 0.5, that is:

  Predict " y =1" if $h_\theta(x) \geq 0.5$ or

  Predict " y =0" if $h_\theta(x) < 0.5$
- Cost Function / Error Function ($J(\theta)$) for all the samples m, needs to be minimized in

order to predict the hypothesis accurately. Cost function can be represented as follows:

$$J(\theta) = 1/m \left(\sum \text{Cost} \left(h_\theta (x) , y \right) \right) \quad \text{;where summation runs from i= 1 to m}$$

Where Cost ($h_\theta (x)$, y) is cost function for one particular sample data

$$\text{Cost} \left(h_\theta (x) , y \right) = \begin{cases} -\log (h_\theta (x)) & \text{if } y = 1 \\ -\log (1-h_\theta (x)) & \text{if } y = 0 \end{cases}$$

Cost function for one sample data can be written as :

$$\text{Cost} \left(h_\theta (x) , y \right) = -y \log (h_\theta (x)) - (1-y) \log (1-h_\theta (x))$$

Thus ,

$$J(\theta) = - 1/m \left[\sum y \log (h_\theta (x)) + (1-y) \log (1-h_\theta (x)) \right] \quad \text{..... Eq(3)}$$

Furthermore, how we update the weights to minimize the Cost function ($J(\theta)$) and to obtain the hypothesis function $h_\theta (x)$ is discussed in Section-4 under Logistic Regression Algorithm.

2 Dataset Description

Wisconsin Diagnostic Breast Cancer(WDBC) dataset was used for training , validation and testing. The dataset contained 569 instances with 32 attributes (ID, diagnosis (B/M), 30 real-valued input features).That is our 'm' , number of sample data is 569 and 'n' which is number of weights/features is 30.

3 Data Preprocessing

The dataset was divided into three categories : Training dataset (80%) , testing dataset (10 %) and validation dataset (10 %).

3.1 Training Dataset

80 % of the Dataset is randomly chosen to train the model. This data is used to get the weights for minimal cost. Let total data in train set be 'tr'

3.2 Testing Dataset

10% of the dataset which wasn't chosen for training is used for testing the algorithm. This will help us to know how accurate our hyperparameters (that is learning rate and weights) are. Let total data in test set be 'tst'.

3.3 Validation Dataset

10% of the dataset which wasn't chosen for training or testing the algorithm is used for validation. Validation set is also used like testing data set, but here with this data we change hyperparameters to get accurate results. Let total data in validation set be 'val'.

Thus, m = 569 = tr +tst + val .

4 Logistic Regression Algorithm

Step -1 : Read the data file- using Pandas library

Step -2 : Process the data file – (i)drop the column id/ patient id (ii) Map the labels M/B to 1/0 respectively.

Step -3 : Split the dataframe into train , validation and testing as discussed in Section-3 using sklearns's tarin_test_split

Step -4 : Normalize the dataset except the first colum. Between the range of 1 to 20. Using the formula –

```
data_norm= ((data-data.min())/(data.max()-data.min()))*20
```

********** Training step starts from Step-5 to Step-7 **********

Step -5 : Initialize the Learning Rate (lr),also pass the training dataset to the Logistic Regression Model.

*Step -6 :Inside the model, i*nitialize the weights and bias, in this project weights and bias were initialized to zero.

*Step -7 :*The loop for gradient descent begins here:

for i in range(10000):

$$\mathbf{Z = \theta^T x + b}$$; However we keep the bias as zero always

- Now we calculate the sigmoid function using the formula below

$$\mathbf{Sigmoid = 1/1+e^{-z}}$$

- Now lets calculate the Loss function using the formula obtained from Eq(3), h_θ in Eq(3) is the sigmoid here.

$$\mathbf{J(\theta) = -[\ y \log(sigmoid) + (1-y) \log(1- sigmoid)\ ].mean()}$$

- Now we find the partial derivative of the cost function using gradient descent as shown below .
- Partial derivative of cost function is written as, $\partial J(\theta) / \partial \theta_j = [\ ^{i=1 \text{ to } tr}\sum (h_\theta(x^{(i)}) - y^{(i)})\ x^{(i)}{}_j\ ]$ where i is from 1 to tr (total data in train set) and j is the number iterations used for obtaining the updated weights. ***The derivation is shown in Appendix***

- $\theta_j = \theta_j - lr\ [\partial J(\theta) / \partial \theta_j\ ]$ is the formula for updating the weights

$$\mathbf{\theta_j = \theta_j - lr\ [(Sigmoid - y\_train).\ x^T\ ]}$$

- In the above equation we are updating the weights using the formula $\theta_j = \theta_j - lr\ [\ ^{i=1 \text{ to } tr}\sum (h_\theta(x^{(i)}) - y^{(i)})\ x^{(i)}{}_j\ ]$ where h_θ is our sigmoid function , and y is the training data set. Also 'lr' here is the learning rate.
- This loop runs until i – 10000

********** Testing Model in Step-8 **********

*Step -8 :*Create another module for prediction, where we would pass the training / validation dataset and the updated weights obtained from training the module from Step 7

- Use the genesis equation and find the Z , the weights here are the ones which we obtained from training the module

$$\mathbf{Z = \theta^T x + b}$$

- Now find the sigmoid function using the updated weights

$$\mathbf{Sigmoid = 1/1+e^{-z}}$$

- Now start the loop for prediction

```
for i in range(size of sigmoid)
    if(sigmoid <0.5):
      y_predict = 0
    else:
     y_predict = 1
```

- The above step classifies the output value to Benign (y_predict = 0) or Malignant (y_predict = 1) on the basis of sigmoid value 0.5

The True Positive, True Negative , False Positive and False Negative values are checked. The results for it are mentioned in Section-5 , Results and Discussion section.

5 Results and Discussion

- The number of iterations for every epoch was 10000
- The length of training dataset was 455
- The length of testing dataset was 57
- The length of validation dataset was 57
- The different values of Learning Rate used to obtain the maximum testing accuracy and minimum loss are as follows:

lr_list=[0.00004,0.0001,0.0002,0.0003,0.0004,0.0005,0.0006,0.0007,0.0008]

At the end of Results and Discussion we have mentioned for which value of LR we obtained the best Testing Accuracy and Loss Function.

5.1 **Learning Rate vs Testing Accuracy**

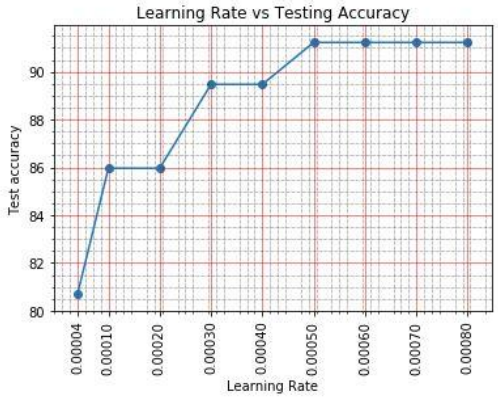


Figure 1: Learning Rate vs Testing Accuracy

From Fig 1. , we can see that as the learning rate increases from 0.00004 to 0.0005 the Testing Accuracy increases. Also we can notice that after 0.0005 the testing accuracy becomes constant. Furthermore, if we keep changing the values of LR it is more likely that after a point the testing accuracy will decrease again. This is because of the convex nature of the graph, ones we reach the local minima the accuracy is the highest, and it again decreases after a certain point.

5.2 **Learning Rate vs Validation Accuracy**

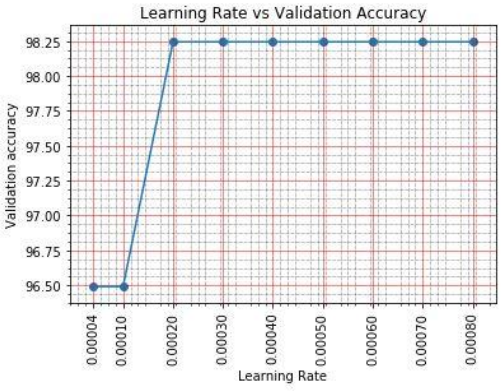


Figure 2: Learning Rate vs Validation Accuracy

We could see that the validation accuracy increases linearly from 0.0001 to 0.0002, after which the accuracy becomes constant.

5.3 **Learning Rate vs Training Accuracy**

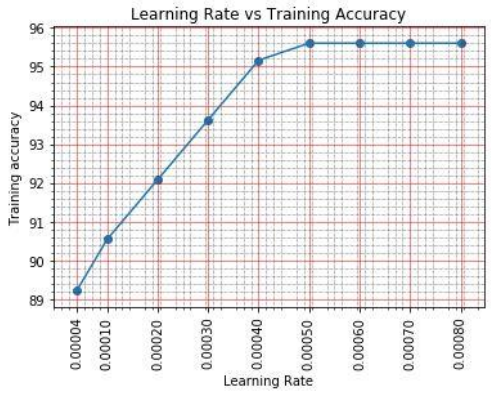


Figure 3: Learning Rate vs Training Accuracy

We can see that the training accuracy increases as we go from Learning Rate = 0.00004 to 0.0005 almost linearly, later as seen in testing accuracy's graph here as well the accuracy becomes constant after a 0.0005. . Furthermore, if we keep changing the values of LR it is more likely that after a point the training accuracy will decrease again. This is because of the convex nature of the graph, ones we reach the local minima the accuracy is the highest, and it again decreases after a certain point.

5.4 **Cost vs Number of Iterations**

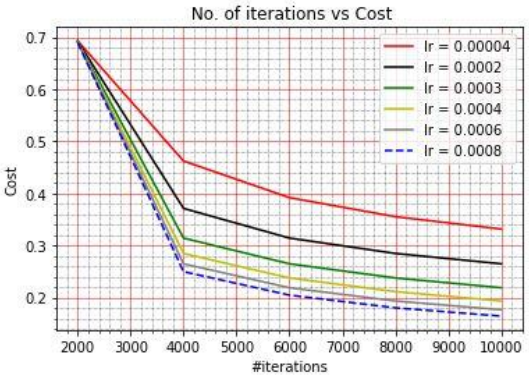
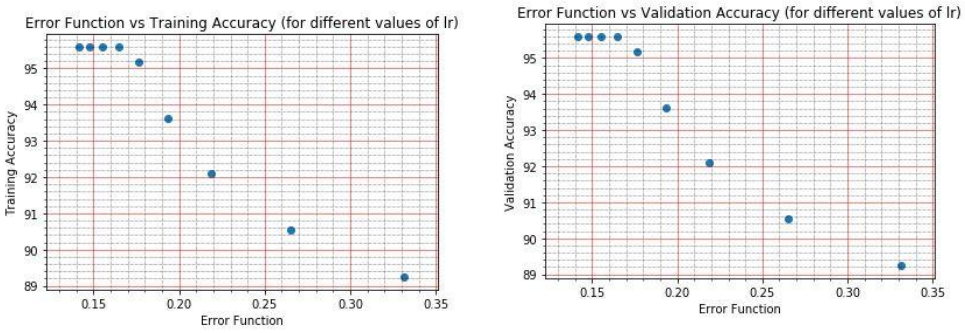


Figure 4: Cost vs Number of Iterations

Here we check the Cost/ Error function with respect to the number of iterations. We can note that as the number of iterations increases from 2000 to 10000 keeping LR as constant the Cost decreases. Also we can note that for LR=0.0008 the cost decreases rapidly. Also we can see that we get a minimum Loss/Cost function at LR = 0.0008 and iterations = 10000

5.5 **Error function vs Accuracy**



(a) (b)

Figure 5: (a) Error function vs Testing Accuracy (b)Error function vs Validation Accuracy

The above two graph depicts that the Accuracy in general decreases if the Error Function/Cost/Loss function increases. Thus our aim needs to include not only an accurate value , but accuracy with minimum Error. Both of them are inversely proportional to each other.

5.6 **Predicting the output for testing data**

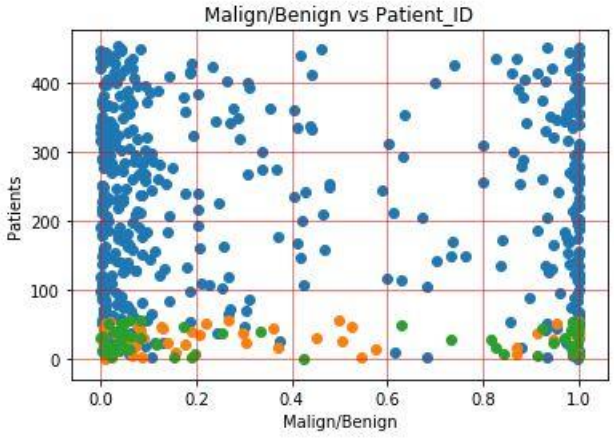


Figure 4: Predicting the output for training data

There were 57 data points in testing data, this scattered image was obtained before we classified our sigma values as malignant or benign. We can notice that the sigmoid function in prediction model along with the weights calculated in training section got us allocate the features between 0 and 1 , as shown above. It is after this that we classify the points lying above 0.5 as malignant and below 0.5 as benign.

5.7 **Accuracy, Precision and Recall** were calculated for Training dataset. Loss Function was also noted down. Along with training and validation accuracy for different values of LR as shown in Table 2.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

$$\text{Precision} = TP / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

Where TP = True Positive , TN = True Negative, FP = False Positive and FN = False Negative

The best accuracy and minimum loss function was obtained at **LR = 0.0008** and **iterations = 10000** the corresponding values are mentioned in the tabular column below:

Table 2: Precision Parameters for LR = 0.0008

LR	TP	TN	FP	FN	Testing Accuracy	Testing Precision	Testing Recall	Training Accuracy	Validation Accuracy	Loss Function
0.00004	12	34	7	4	80.701%	63.1578%	75.0%	89.23%	96.49%	33.1613%
0.0001	13	36	5	3	85.964%	72.221%	81.25%	90.549%	96.491%	26.493%
0.0002	13	36	5	3	85.964%	72.225%	81.25%	92.087%	98.24%	21.8956%
0.0003	13	38	3	3	89.47%	81.25%	81.25%	93.626%	98.245%	19.3533%
0.0004	13	38	3	3	89.47%	81.25%	81.25%	95.16%	98.24%	17.66%
0.0005	14	38	3	2	91.22%	82.35%	87.5%	95.60%	98.24%	16.45%
0.0006	14	38	3	2	91.22%	82.35%	87.5%	95.60%	98.24%	15.2278%
0.007	14	38	3	2	91.22%	82.35%	87.5%	95.60%	98.24%	14.7844%
0.0008	**14**	**38**	**3**	**2**	**91.228 %**	**82.3529%**	**87.5%**	**95.60439%**	**98.2456%**	**14.179%**

7 Conclusion

The best epoch for which the Testing Accuracy was maximum and the loss function was minimum was for **learning rate = 0.0008** and **iterations = 10000**. The Test Accuracy obtained was **91.228 %** , Precision obtained was **82.3529 %** and Recall value was **87.5%.**

Acknowledgments

I am extremely grateful to Professor Sargur Srihari for elucidating all the necessary concepts related to Logistic Regression. I am also thankful to all the Teaching Assistants, their recitation sessions have been really helpful.

References

[1] Andrew Ng's course on Youtube : https://www.youtube.com/watch?v=TTdcc21Ko9A&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN&index=36

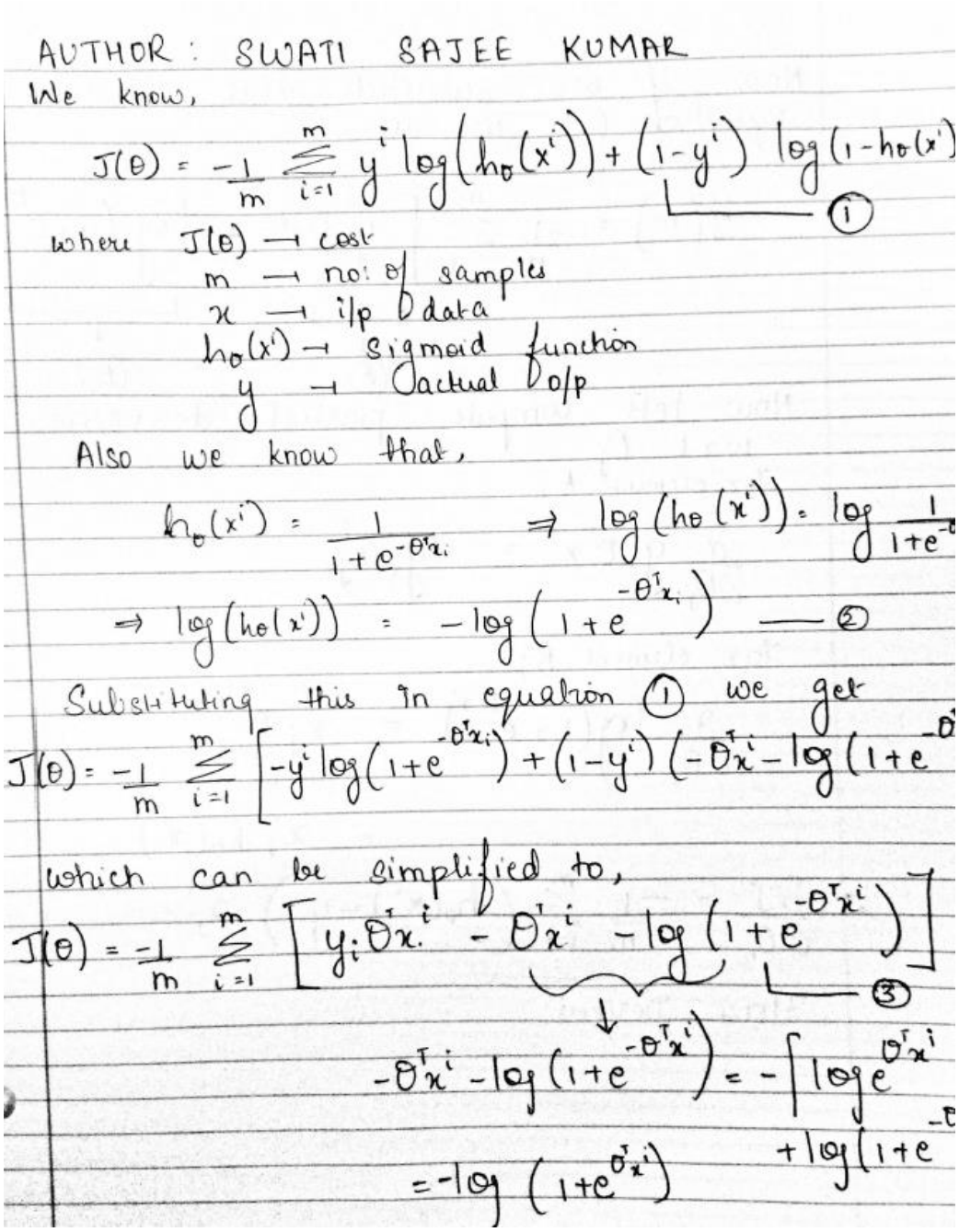
[2] CSE474LECB: Machine Learning – Week 3: Linear Regression --→ Gradient Descent

[3] Regarding TP, TN, FP and FN I refered https://stackoverflow.com/questions/31324218/scikit-learn-how-to-obtain-true-positive-true-negative-false-positive-and-fal

[4] PRML, Section 4.3.2 for Logistic Regression

APPENDIX

Partial derivation of Cost Function w.r.t Weights :



AUTHOR : SWATI SAJEE KUMAR

We know,

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^i \log\left(h_\theta(x^i)\right) + (1-y^i)\log(1-h_\theta(x^i)) \quad \text{①}$$

where $J(\theta)$ → cost
m → no: of samples
x → i/p data
$h_\theta(x^i)$ → sigmoid function
y → actual o/p

Also we know that,

$$h_\theta(x^i) = \frac{1}{1+e^{-\theta^T x_i}} \Rightarrow \log\left(h_\theta(x^i)\right) = \log\frac{1}{1+e^{-\theta^T x_i}}$$

$$\Rightarrow \log\left(h_\theta(x^i)\right) = -\log\left(1+e^{-\theta^T x_i}\right) \quad \text{②}$$

Substituting this in equation ① we get

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[-y^i \log\left(1+e^{-\theta^T x_i}\right) + (1-y^i)\left(-\theta^T x^i - \log\left(1+e^{-\theta^T x^i}\right)\right)\right]$$

which can be simplified to,

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \theta^T x^i - \theta^T x^i - \log\left(1+e^{-\theta^T x^i}\right)\right] \quad \text{③}$$

$$-\theta^T x^i - \log\left(1+e^{-\theta^T x^i}\right) = -\left[\log e^{\theta^T x^i} + \log\left(1+e^{-\theta^T x^i}\right)\right]$$

$$= -\log\left(1+e^{\theta^T x^i}\right)$$

Now if we substitute this value in equation (3) we get

$$J(\theta) = \frac{-1}{m} \sum_{i=1}^{m} \left[y_i \theta^T x^i - \log\left(1 + e^{\theta^T x^i}\right) \right]$$

(A) (B)

Now lets compute partial derivative w.r.t θ_j

for element A;

$$\frac{\partial}{\partial \theta_j} y_i \theta^T x^i = y_i x_j^i$$

for element B;

$$\frac{\partial}{\partial \theta_j} \log\left(1 + e^{\theta^T x^i}\right) = \frac{x_j^i e^{\theta^T x^i}}{1 + e^{\theta^T x^i}}$$

$$= x_j^i h_\theta(x^i)$$

$$\therefore \frac{\partial J}{\partial \theta_j} = \frac{-1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y_i \right) x_j^i$$

Hence Derived