Classification using Machine Learning

Aditya Bansal
UB Person No.-50291692
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
adityaba@buffalo.edu

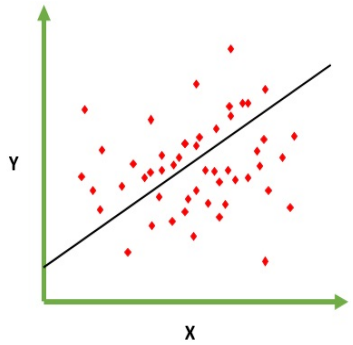
Abstract

The aim of this project is to perform classification using machine learning. It is for a two-class problem and we are doing it using the dataset of Wisconsin Diagnostics Breast Cancer (WDBC). We are applying Logistic Regression, which is one the mostly preferred regression model for implementing binary dependent variables modelling. For this project we are using discriminative classifiers type of logistic regression.

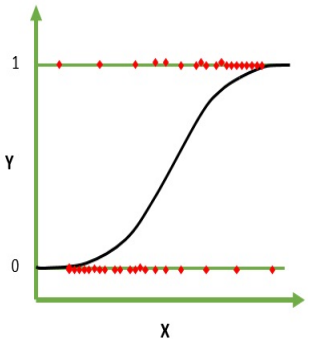
1 Introduction

Roots of logistic regression can be found in nineteenth century where it was used to explain the growth rate of populations by Verhulst. Today it is widely used in various fields like biology, medicine and Epidemology.For understanding logistic regression first we need to understand what is. Regression, Regression model is a process of relationship estimation between dependent variables and independent variables. Majorly regression is used for forecasting an effect, predictors and trend forecasting. Various types of regression models are polynomial regression, stepwise regression, linear regression, logistic regression and many more. Most commonly used and basic form of regressions are liner and logistic regression. The basic difference between linear and logistic regression is that dependent variables used in linear regression are of continuous nature and regression line is linear but in contrast in logistic regression variables which are dependent are of binary nature.

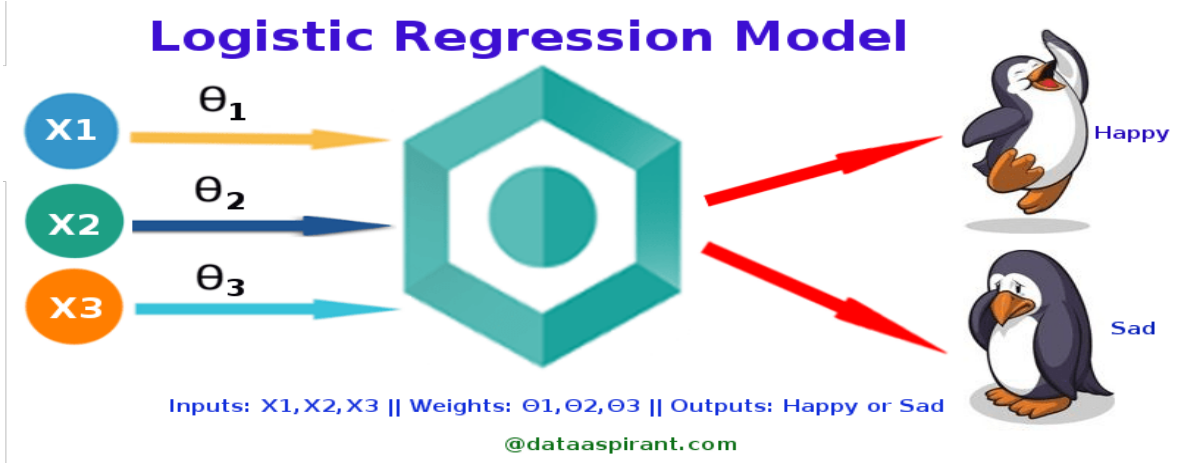
Linear Regression



Logistic Regression

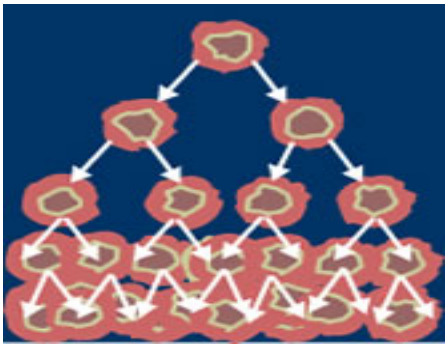


Major types of Logistic Regression are: -
Binomial Logistic Regression: -Response has only two possible outcomes, Example: - Spam or not.
Multinomial Logistic Regression: -Three or more categories without any order.
Ordinal Logistic Regression: -Three or more categories with ordering.



In our project we have taken WDBC dataset in which features taken are computed from pictures of pre-computed from images of a fine needle aspirate (FNA) of a breast mass. Our primary task is to classify suspected FNA cells to Benign (class 0) or Malignant (class 1), so we are using Binomial logistic regression.

2 Dataset Definition



Wisconsin Diagnostic Breast Cancer (WDBC) dataset will be used for training, validation and testing. The dataset contains 569 instances with 32 attributes (ID, diagnosis (B/M), 30 real-valued input features). Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. Computed features describes the following characteristics of the cell nuclei present in the image:

2.1. Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.

3 Pre-Processing

Preprocessing of the CSV format(use to store tabular data in form of databases or spreadsheet), WDBC dataset is done before using it into our logistic regression model. Following steps were followed for pre-processing of our dataset: -

1.Processed original CSV datafile to Pandas Data frame, 'df'.

2. Converted B=Benign to 0 and M=Malignant to 1 in second column which is diagnosis so that we can compare it to our predicted results

3.Converted B=Benign to 0 and M=Malignant to 1 in second column which is diagnosis so that we can compare it to our predicted results.

4.For Logistic Regression we don't require IDs of images present in column 1 so we have sliced our data and saved column1 as 'redundant' and all other columns as 'x'.

5.Now we needed to split our data into three categories which are train data, validation data and test data for which we are using 'train_test_split()' function present in library sklearn.

5.1 First we split our data into x_train i.e. training data and x_set in the ratio of 8:2.

5.2 Secondly we divided x_set into x_val i.e validation data and x_test i.e test data in ratio 1:1.

6.Our next step is to slice all three data sets into solution and instances i.e we sliced first column of all three data sets and saved it as train_sol, val_sol and test_sol respectively and saved our instances as train_Inst, val_Inst, test_Inst.

7.Now we are normalizing all three of our instances data set. We can use various functions to normalize data like 'normalize()', 'min_max_scaler()' and 'standard_scaler()' each of them use different formulas which we will discuss in our architecture all three are present in 'sklearn' library. We are primarily using 'standard_scaler()' for our code.

StandardScaler : It transforms the data in such a manner that it has mean as 0 and standard deviation as 1. In short, it **standardizes the data**. Standardization is useful for data which has negative values. It **arranges the data in normal distribution.** It is more **useful in classification than regression**.

$$f'_i = \frac{f_i - \text{mean}(f_i)}{\text{std}(f_i)}$$

Normalizer : It squeezes the data between 0 and 1. It performs **normalization**. Due to the decreased range and magnitude, the **gradients in the training process do not explode** and you do not get higher values of loss. Is **more useful in regression than classification**. You can read this blog of mine.

$$x'_n = \frac{x_n}{\text{size}(x_n)}$$

$$L_2\ form.\ \ ||x_n|| = \sqrt[2]{f_{n,1}^2 + \ldots + f_{n,d}^2}$$

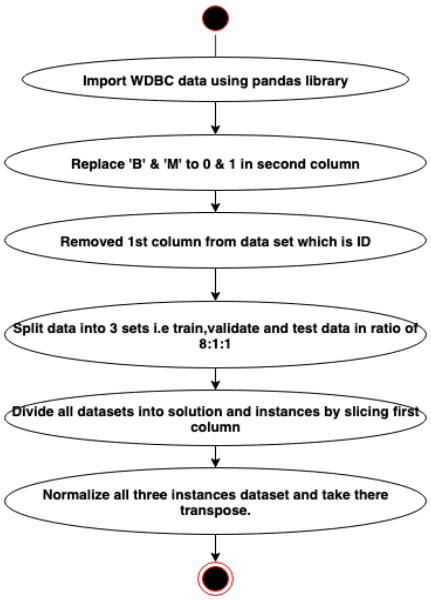
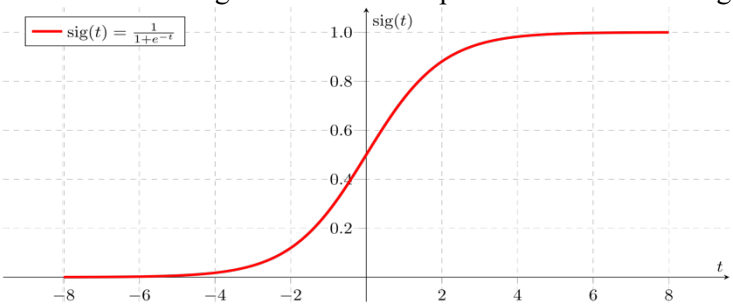


Figure: Flow Chart for preprocessing of data

4 Architecture

4.1 Hypothesis

We have taken hypothesis that if we can represent z that is our solution as linear equation $\boldsymbol{z = \emptyset^T x + bias}$ then if we substitute this equation in the sigmoid function then we can classify our data into binary form. Now to investigate further it is important to know what sigmoid function is.



In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

Sigmoid Function: -

$$h_\theta(x) = \frac{1}{1 + e^{-(x)}}$$

Logistic regression hypothesis expectation: -

$$0 \leq h_\theta(x) \leq 1$$

Decision Boundary We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1.

For Example, We have 2 classes, let's take them like cats and dogs (1 — dog, 0 — cats). We basically decide with a threshold value above which we classify values into Class 1 and of the

value goes below the threshold then we classify it in Class 2.

Pseudo Code

- Preprocess the data.
- Define sigmoid function

$$\boldsymbol{h_\theta(x) = \frac{1}{1 + e^{-(x)}}}$$

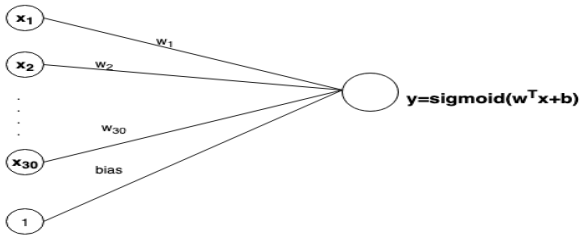
$def\ sigmoid(z)$: *return 1 / (1 + np.exp(-z))*

- Initialize hyperparameters epochs (no. of iterations) and learning rate(amount that the weights are updated during each iteration) and tune them by comparing results of validation data with training data.
- Initialize weights with some random values and bias with zero. The idea of having bias is about model giving importance to some of the features in order to generalize better for the larger dataset with various other attributes. Bias in ML does help us generalize better and make our model less sensitive to some single data point.

- Now for all epoch in epochs
    1. Find solution set by passing data in linear equation for both training and validation data.

$$\boldsymbol{z = \emptyset^T x + bias}$$

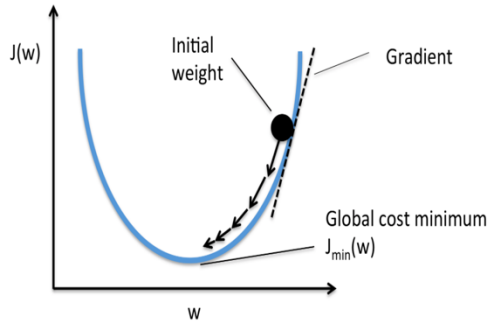
ztrain=np.dot(w.T,train_Inst)+bias



    2. Find predicted values by passing z in sigmoid function for both training and validation data.

p_train = sigmoid(z_train)

    3. Now we are finding gradient of error function with respect to weights.



$$\theta = \frac{1}{m}\sum_{m=1}^{n}(h_\theta(x^i)\text{-}y^i)x^i$$

dz = p_train-train_Sol.

*dw = (1 / m) * np.dot(train_Inst, dz.T)*

4. Now we are finding change in bias for every epoch. Since bias is constatnt so we can calculate difference by talking

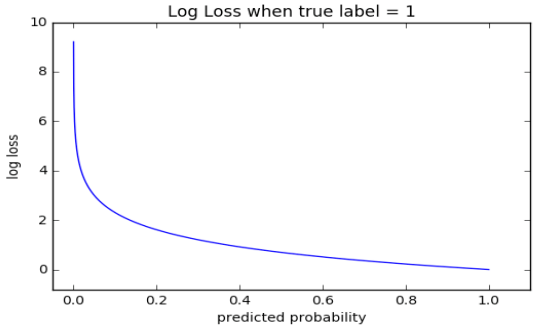
*dbias = (1 / m) * np.sum(dz)*

5. Adjust weight by subtracting weight with product of cost function derivative θwe found above with learning rate, and adjust bias by subtracting bias from mean of error in solution for all the instances.

*w = w - learningrate * dw*

*bias = bias - learningrate * dbias*

6. Now we will find the *cross-entropy* function. **Cross-entropy loss**, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.



Cross Entropy function: -

$$\mathbf{C} = \frac{-(\mathbf{y}\log(\mathbf{a}) + (\mathbf{1}-\mathbf{y})\log(\mathbf{1}-\mathbf{a}))}{\mathbf{m}}$$

C=Coast, y= solution given in data set, a=predicted solution we found from sigmoid

m =

total number of instances.
We are finding cost-entropy function for both training data and validation data also we are saving value of cost for each iteration so that we can plot graph of cost vs epochs for both data sets.

```
cost = -np.sum(np.multiply(np.log(p_train), train_Sol) + np.multiply((1 - train_Sol),
                                                    np.log(1-p_train)))/m
losstrack.append(np.squeeze(cost))
```

7. Finding accuracy score for both training and validation data set and appending results for every epoch so that we can plot graph of accuracy score vs epoch, but to find accuracy score we first converted our predicted value in binary 0 and 1.That between 0-0.5 will be converted to 0 and 0.5-1.

```
q=p_train
q[q>=0.5]=1
q[q<0.5]=0
q=q.astype(int)
train_Accuracy.append(accuracy_score(y_true= train_Sol[0],y_pred= q[0]))
```

5 Results

1. We have taken results for different value of hyperparameters after comparing results of loss function and accuracy score of train data and validation data. Also we tried to observe the difference in results by using different types of normalization factions.
2. We have evaluated our test set using Accuracy, Precision, Recall and F1 score.
   Accuracy-It is the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision= it is the number of positive predictions divided by the total number of positive class values predicted.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall-it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate.

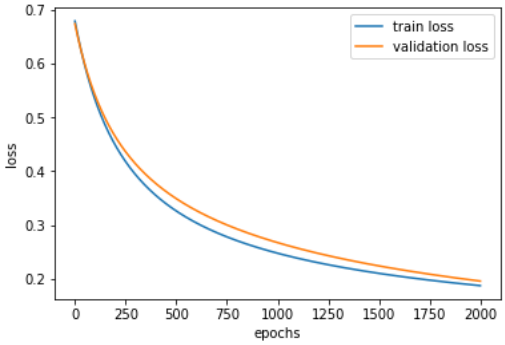
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

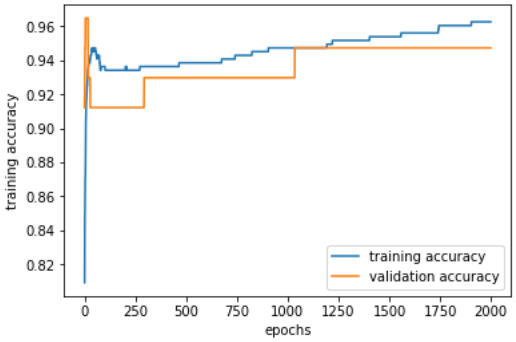
F1 score-It conveys the balance between the precision and the recall.

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

True Positive (TP)= Results which are given positive are true.
True Negative (TN)= Results which are given negative are true.
False Positive (FP)= Results which are given positive are false.
False Negative (FN)= Results which are given negative are false.

3. Let's take hyperparameters, epochs =2000 and learning rate =0.001

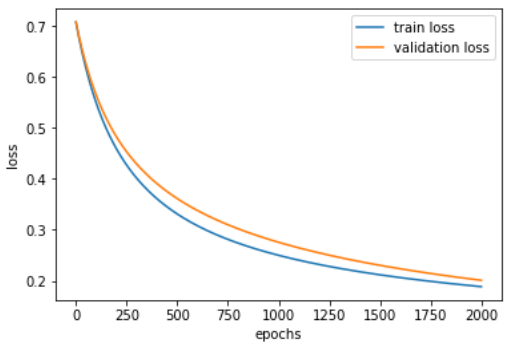


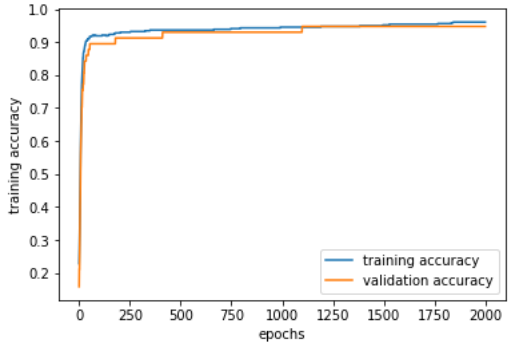


Test accuracy score= 0.947

Train Data Results: -			
	precision	recall	f1-score
0	0.97	0.97	0.97
1	0.95	0.95	0.95
avg / total	0.96	0.96	0.96
Validation Data Results: -			
	precision	recall	f1-score
0	0.94	0.97	0.94
1	0.96	0.92	0.95
avg / total	0.95	0.95	0.95
Test Data Results: -			
	precision	recall	f1-score
0	0.94	0.97	0.95
1	0.96	0.92	0.94
avg / total	0.95	0.95	0.95

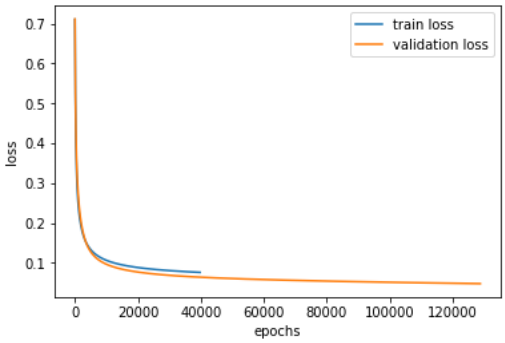
4. Let's take hyperparameters, epochs =200 and learning rate =0.001

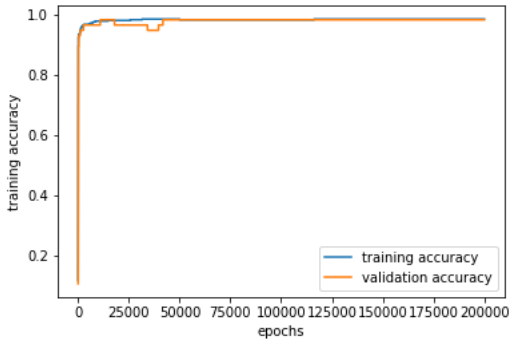




Test accuracy score= 0.947

5. Let's take hyperparameters, epochs =200000 and learning rate =0.001





Test accuracy score= 0.982

6. Conclusion

Using logistic regression and taking sigmoid function hypothesis in our program we saw that we are able to predict result with very high accuracy when we compare it to the result given to us in dataset. By this we can conclude that the hypothesis taken by us was right. After tuning hyperparameters we got accuracy score of 0.945 for 2000 epochs and 0.001 learning rate.

7. References

[1] THE IMPORTANCE OF LOGISTIC REGRESSION IMPLEMENTATIONS IN THE TURKISH LIVESTOCK SECTOR AND LOGISTIC REGRESSION IMPLEMENTATIONS/FIELDS Murat KORKMAZ1 *, Selami GÜNEY2, Şule Yüksel YİĞİTER3. J.Agric. Fac. HR.U., 2012, 16(2): 25-36

[2] On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes . Andrew Y. Ng Computer Science Division University of California, Berkeley Berkeley, CA 94720

[3] https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148, Introduction to Logistic Regression Ayush Pant.

[4] https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

[5] Machine Learning week 1: Cost Function, Gradient Descent and Univariate Linear Regression, Lachlan Miller

[6] An Introduction to Logistic Regression Analysis and Reporting CHAO-YING JOANNE PENG KUK LIDA LEE GARY M. INGERSOLL Indiana University-Bloomington.

[7] Pattern Recognition and Machine Learning, Christopher M Bishop.

[8] Accuracy, Precision, Recall or F1? Koo Ping Shung, https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9