Introduction to Machine Learning : Project 1 Report

Sourabh Majumdar
University at Buffalo, The State University of New York
Buffalo, NY, 14261
smajumda@buffalo.edu

Abstract

In this report we attempt to provide a logistic regression approach to the Wisconsin Diagnostic Breast Cancer (WDBC) Dataset. This work is done in partial fullfilment for the course "Introduction to Machine Learning, Fall 2019".

1 Introduction

Cancer Detection and Prevention is one of the most challenging problems in oncology today. Scientists have spent decades formulating tests to detect and prevent cancer. With the rise in Data Driven Approaches like Machine Learning, we present our data driven solution to the problem. In the following sections, we describe the dataset and features used for our computation and present results from our experimentation.

2 Dataset

We used the Wisconsin Diagnostic Breast Cancer (WDBC) for training our Models. 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. They can be seen in Table 1

Sr No.	Feature Name
1	radius (mean of distances from center to points on the perimeter)
2	texture (standard deviation of gray-scale values)
3	perimeter
4	area
5	smoothness (local variation in radius lengths)
6	compactness (perimeter$^2/area1.0$)
7	concavity (severity of concave portions of the contour)
8	concave points (number of concave portions of the contour)
9	symmetry
10	fractal dimension ("coastline approximation" - 1)

Table 1: Features Computed for the Dataset

3 Model

Since this problem requires us to classify our data into two classes (Malignant or Benign), we use a classification model. Specifically we use a logistic regression model, since they have been shown to have promising performance on two class classification problems.

3.1 Logistic Regression

Logistic Regression approach is a discriminative statistical model that estimates the log odds of the class probabilities as a function of input features. In two class classification problem, the equation that we are optimizing is :

$log(p/(1-p)) = \sum_{i=1}^{n} w_i x_i - (1)$

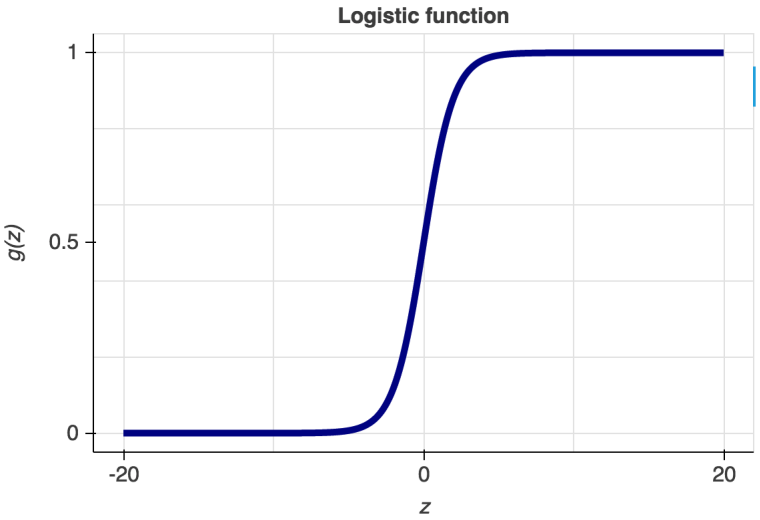
here $p/(1-p)$ is called the odds of an event with probability p, w_i are the weights given to individual feature x_i. The value computed in the right hand side of this equation $\in \mathbb{R}$. In order to convert it into a probability distribution, we pass this value through a non linear function like sigmoid (also called logistic function) to obtain a value $q \in (0, 1)$. So the entire procedure is summarized in set of equations described below.

$z = \sum_{i=1}^{n} w_i x_i - (2)$

$q = \sigma(z) - (3)$

where $\sigma(z) = 1/(1 - \exp(-z))$ is the sigmoid function.

A figure of sigmoid is given below.



In equation (1), the p is the true probability of the class, which we attempt to compute as a function of the parameters x_i.

Initially we don't know the correct values of weights w_i for the problem. Hence in logistic regression, our problem is simplified to finding the correct w_i that correctly predicts the probabilities p_i. We find w_i using a method call Gradient Descent, which is discussed in detail in the section below.

3.2 Gradient Descent

Gradient Descent/Ascent is a first-order derivative based optimization method, where we find optimal value(s) of a given function $f(x)$ by using it's derivative with respective to its parameters x. Then to find the optimal values, we move the parameters in the direction of the gradient. In Gradient Descent, we move in the opposite direction of the gradient.

Hence if we are attempting to optimize a function $f(w)$ with respective to it's parameters w. At each iterative step we update the parameters, w by a factor of the gradient. The update is given below.

$$w^{(new)} = w^{(old)} - \eta \nabla f(w) - (4)$$

In the case of logistic regression, we are optimizing the error function $J(w)$, between the estimated probabilities q and the true probabilities p to obtain the minimum possible error. The error function for the error logistic regression (also called cross entropy loss function) is given below.

$$J(w) = -\sum_{i=1}^{n} [p_i log(q_i) + (1 - p_i) log(q_i)] - (5)$$

For gradient descent, the gradient of $J(w)$ with respect to parameters w is :

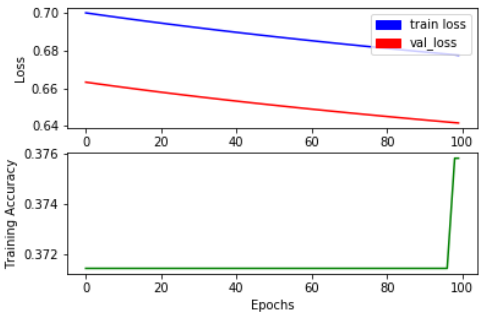
$$\nabla J(w) = (\sigma(w, x) - p)x - (6)$$

Hence the update equation of gradient descent for logistic regression is :

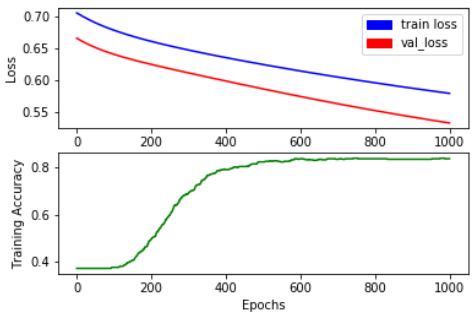
$$w^{(new)} = w^{(old)} - \eta(\sigma(w, x) - p)x - (7)$$

4 Experiments and Results

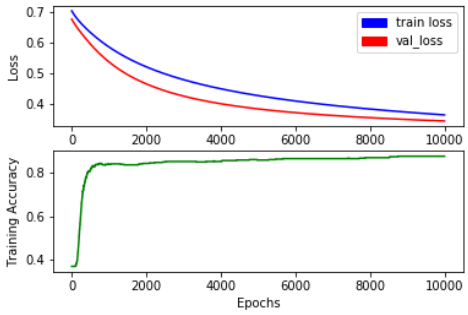
We trained our model on a training set of 455 samples, which were validated on a validation set of 57 samples. Finally we tested our model on another set of 57 samples. There are only two hyper-parameters pertinent to this problem, they are the learning rate η and the number of epochs. The compiled results along with the loss curves and training accuracies are presented below.



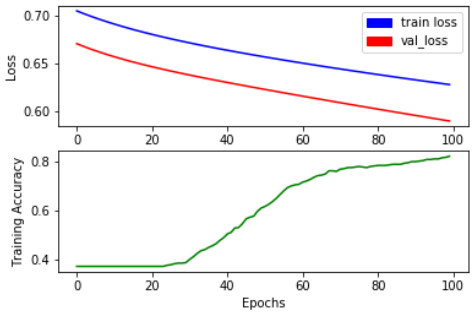
learning rate 0.01 and epochs 100



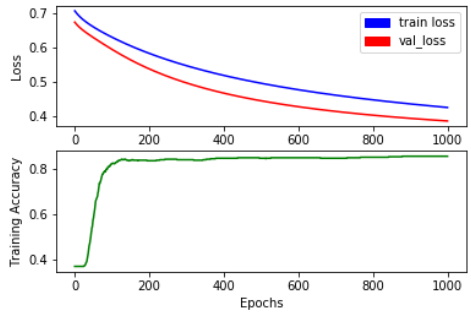
learning rate 0.01 and epochs 1000



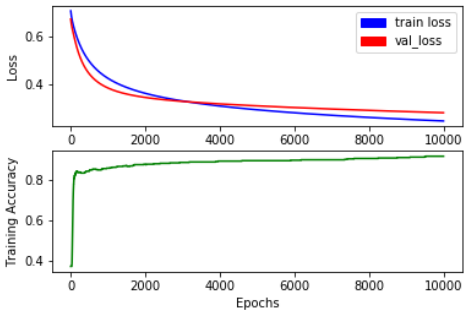
learning rate 0.01 and epochs 10000



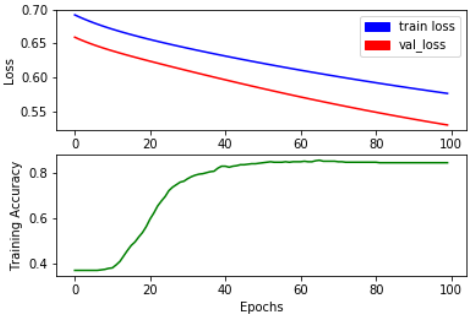
learning rate 0.05 and epochs 100



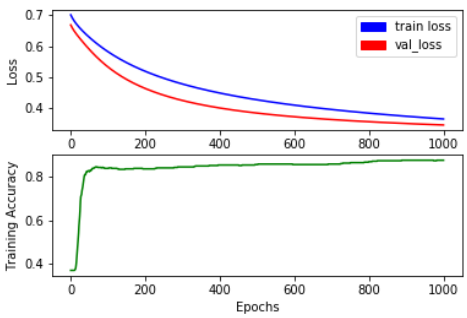
learning rate 0.05 and epochs 1000



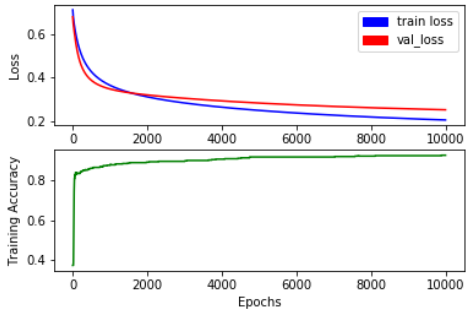
learning rate 0.05 and epochs 10000



learning rate 0.1 and epochs 100



learning rate 0.1 and epochs 1000



learning rate 0.1 and epochs 10000

Figure 1: Plots with different learning rates and epochs

In order to get a better view of the hyper-parameters, we also tracked the minimum validation loss for each setting of the hyper-parameters. They are presented in Table 2 below.

Learning rate/Epoch	**100**	**1000**	**10000**
0.01	0.67	0.54	0.35
0.05	0.59	0.38	0.28
0.1	0.54	0.35	0.25

Table 2: Minimum validation losses for each setting of the hyper-parameters

As we can see that minimum validation loss is obtained for choosing $\eta = 0.1$ and $epochs = 10,000$. So we use this setting to perform evaluation on the test data. We now present the compiled statistics of this model on the train, validation and test data.

Class	Precision	Recall	F1-Score	Support
Benign	0.91	0.98	0.94	286
Malignant	0.97	0.83	0.89	169
Average	0.94	0.91	0.92	455

Table 3: Training Statistics

Class	Precision	Recall	F1-Score	Support
Benign	0.81	0.97	0.88	30
Malignant	0.95	0.74	0.83	27
Average	0.88	0.85	0.86	57

Table 4: Validation Statistics

Class	Precision	Recall	F1-Score	Support
Benign	0.93	1.00	0.96	41
Malignant	1.00	0.81	0.90	16
Average	0.97	0.91	0.93	57

Table 5: Test Statistics

As we can see, that our model performs optimally on all training, validation and test set. We see high f1-scores not only on average but for individual classes as well. This indicates that our model has found the distinguishing features for both the classes of cells i.e Malignant as well as Benign.

On compiling results on the test set, we see a perfect recall score (see Table 5) for Benign cells. It means that our model perfectly captures all benign cells. However it calls some malignant cells also as benign which leads to a lower precision score. In oncology terms this is little dangerous as our model will classify some malignant cells as benign leading to incorrect diagnosis of some patients leading them to suffer. However this precision is still quite high compared to present doctor performed diagnosis, which means that our model will result in less number of people suffering.

We also note a perfect precision score on Malignant cells (see Table 5). It means that all the cells that are predicted to be malignant are in actual malignant. Hence our system is quite reliable for actual cancer patients. So we can always trust the prediction made by the model and if the test is positive then the patient should immediately consult his/her doctor.

In conclusion, if our model predicts a patient to have a malignant cancerous cells, then the patient should immediately consult his/her doctor. However, if the test is negative, the patient can still have cancerous cells, which can lead in great suffering.

5 Conclusion and Future Work

In this work, we presented a logistic regression solution to the Wisconsin Diagnostic Breast Cancer Dataset (WBDC). We compiled and presented our results and it's interpretations. We hope that this work can be influential in designing better diagnostic tools for oncology as well as other healtcare domains. In future we would like to improve the performance of our model, specifically find that handle the malignant cases that are being identified as benign.

Appendix

Sourabh Majumdar

September 2019

1 Maximum Likelihood Estimate for Logistic Regression

In this section we derive the Maximum Likelihood Estimate for logistic regression.

Before we begin, we have to define the conditional likelihood function. Assume that $Pr(Y = C_i|X = x_i) = p(x; w)$, for some function p parameterized by w and further assume that the observations are independent of each other. Then the (conditional) likelihood function is

$$\prod_{i=1}^{n} Pr(Y = y_i|X = x_i) = \prod_{i=1}^{n} p(x_i; w)^{y_i}(1 - p(x_i; w)^{1-y_i})$$

Recall that in a sequence of Bernoulli trials $y_1, ...y_n$ where there is a constant probability of success p, the likelihood is

$$\prod_{i=1}^{n} p^{y_i}(1 - p)^{1-y_i}$$

Now in logistic regression, we have a binary output variable Y, and we want to model the conditional probability $Pr(Y = C_1|X = x)$ as a function of x; any unknown parameters in the function are to be estimated by maximum likelihood.
If x is characterized by a set of independent features, then we can write the maximum likelihood estimate as

$$L = \prod_{i=1}^{n} p(x; w)^{y_i}(1 - p(x; w))^{1-y_i}$$

By taking logarithm on both sides, we obtain

$$log(L) = \sum_{i=1}^{n} [y_i log(p(x_i)) + (1 - y_i)log(1 - p(x_i)]$$

The right hand side is the cross entropy loss between the original probabilities and the predicted probabilities. Hence maximizing the likelihood is equivalent to minimizing the cross entropy error.

$$E(w) = -\sum_{i=1}^{n} [y_i log(p(x_i)) + (1 - y_i)log(1 - p(x_i)]$$

2 Gradient Calculation of Cross Entropy Function

To simplify our calculations, we will use the following result

$\frac{\delta\sigma(z)}{\delta z} = \sigma(z)(1-\sigma(z))$

We know that the Cross Entropy Error is

$E(w) = -\sum_{i=1}^{n}[y_i log(p(x_i)) + (1-y_i)log(1-p(x_i))]$

Derivative of gradient for one datapoint (x,y) is :

$\frac{\delta E(w)}{\delta w} = \frac{\delta}{\delta w} y log(\sigma(wx)) + \frac{\delta}{\delta w}(1-y)log[1-\sigma(wx)]$ derivative of sum of terms

$= [\frac{y}{\sigma(wx)} - \frac{1-y}{1-\sigma(wx)}]\frac{\delta}{\delta w}\sigma(wx)$ derivative of log $log(f(x))$

$= [\frac{y}{\sigma(wx)} - \frac{1-y}{1-\sigma(wx)}]\sigma(wx)[1-\sigma(wx)]x$ chain rule + derivative of $\sigma(z)$

$= \frac{y-\sigma(wx)}{\sigma(wx)[1-\sigma(wx)]}\sigma(wx)[1-\sigma(wx)]x$ algebraic manipulation

$= [y-\sigma(wx)]x$ cancelling terms