CSE 574 Fall 2019 Project 1

Logistic Regression based Classification of Breast Cancer Tumors

Srisai Karthik Neelamraju
UBID - 50316785
University at Buffalo
Buffalo, NY 14260
neelamra@buffalo.edu

Abstract

Predicting cancer at an early stage is vital in the field of medicine for curing the disease. A program that can predict malignant tumors based on some attributes of the tumor can be of immense help. This can be achieved using machine learning. Logistic regression is a technique used in machine learning to solve 2-class or binary classification problems. For this project, a machine learning model to identify benign and malignant breast cancer tumors was built using logistic regression. Each instance in the dataset used for this project consisted of 30 features describing the tumor. These were subjected to preprocessing before training the model. Gradient descent was then used to optimize the loss function. Training and validation losses were analyzed for different settings of the model parameters. Performance of the model on validation set was instrumental in tuning the hyperparameters. Finally, the logistic classifier was evaluated on the test set using different performance metrics like F1 Score, accuracy, precision and recall.

1 Introduction

Logistic regression is a widely used algorithm in the field of machine learning. It is often employed for addressing prediction problems involving two classes (called the binary classification problems). It associates a weight to each of the input features and then computes the probability of the input belonging to a certain class out of the two possible classes using a sigmoid function [1]. Most implementations of logistic regression have the loss function defined by binary cross entropy, which increases as the distance between the predicted probability and the actual label increases. The weights are then updated using an omnipresent gradient descent, which is an optimization algorithm used to find the global minimum of a function. Gradient descent computes the gradient of the loss function with respect to each weight and then updates each weight accordingly. The primary task of this project is to use logistic regression for predicting and classifying breast cancer tumors into two classes - benign and malignant. In the subsequent sections, the dataset and the procedure for implementing the logistic regression classifier is detailed.

2 Dataset

The dataset used for this project was obtained from the Wisconsin Diagnostic Breast Cancer (WDBC) Database. It contains 569 instances with each instance having 32 attributes. Of these, the attribute ‘ID’ is a unique number that identifies a tumor and the attribute ‘diagnosis’ has a B or M label corresponding to class benign or class malignant. The other 30

attributes are real-valued input features. These features, as described in the dataset description [2], were computed from a digitized image of a fine needle aspirate (FNA) of a breast mass and represent the radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry and fractal dimension of the cell nuclei present in the image. For each image, the mean, standard error and largest of each of these characteristics were computed. Consequently, there are 30 features that describe each tumor.

3 Preprocessing

Before the commencement of the training phase, the dataset was subjected to preprocessing. Firstly, since the purpose of the attribute 'ID' was only to differentiate two cancer tumor instances and it did not have any impact on the tumor being benign or malignant, this attribute was dropped from the dataset. Besides, the labels 'B' and 'M' in the 'diagnosis' attribute defining the benign and malignant tumors of the dataset were mapped to integers 0 and 1 respectively. Thus, the problem essentially was a two-class classification problem with class 0 representing benign tumors and class 1 representing malignant tumors.

In addition to these, not all of the attributes in each instance belonged to the same range. Some of the attributes lied in the range (0, 1), whereas some others were of the order of thousands. To minimize the adverse effect these values can have on the performance of the model, the 30 features for each of the 569 instances were standardized and then normalized using the StandardScaler and normalize methods of scikit-learn's preprocessing module [3].

The dataset was then split into three partitions – the training, the validation and the test sets. 80% of the dataset, which amounted to 455 instances, was used for training the machine learning model. The remaining 20% of the dataset was equally split into the validation and the testing sets of 57 instances each. The intention behind creating the validation set was to tune the hyperparameters pertaining to the implementation of logistic regression on the validation set, and the test set would then be used to evaluate the performance of the model.

4 Architecture

To tackle the aforementioned two class problem of predicting benign and malignant breast cancer tumors, logistic regression with batch gradient descent was used. The architecture and implementation method are described in this section.

4.1 Computational Flow of Simple Logistic Regression

Let us suppose an arbitrary instance has exactly one feature 'x' and its output label is 'y'.

The first forward propagation step of logistic regression is defined by,

$$z = wx + b$$

Here, 'w' is the weight associated with the feature 'x' and 'b' is the bias term.

Later, taking the logistic sigmoid of 'z' gives the activation value 'a'.

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

The loss 'L' for this instance is defined using the cross-entropy loss or log loss function as,

$$L = -\Big(y \log a + (1 - y) \log (1 - a)\Big)$$

After computing the loss, gradient descent algorithm with learning rate 'α' is used to modify the weight and the bias terms according to these update equations.

$$w = w - \alpha\left(\frac{dL}{dw}\right)$$

$$b = b - \alpha\left(\frac{dL}{db}\right)$$

Thus, the values of the weight and the bias change after every iteration and over a period of time, the logistic regression model fits the training data.

4.2 Finding Gradients

In this subsection, finding the gradients of loss function with respect to the weight and the bias that were used in the update equation of gradient descent is detailed.

Using chain rule in calculus, the gradients can be expressed as,

$$\frac{dL}{dw} = \frac{dL}{da}.\frac{da}{dz}.\frac{dz}{dw}$$

$$\frac{dL}{db} = \frac{dL}{da}.\frac{da}{dz}.\frac{dz}{db}$$

The derivative of L with respect to a is,

$$\frac{dL}{da} = \frac{d}{da}\Big[-\Big(y\log a + (1-y)\log(1-a)\Big)\Big]$$

$$\Rightarrow \frac{dL}{da} = -\Big[\frac{y}{a} - \frac{1-y}{1-a}\Big]$$

$$\Rightarrow \frac{dL}{da} = -\Big[\frac{y - ay - a + ay}{a(1-a)}\Big]$$

$$\Rightarrow \frac{dL}{da} = \frac{a-y}{a(1-a)}$$

The derivative of a with respect to z is,

$$\frac{da}{dz} = \frac{d}{dz}\Big(\frac{1}{1+e^{-z}}\Big)$$

$$\Rightarrow \frac{da}{dz} = \frac{-1\big(-e^{-z}\big)}{\big(1+e^{-z}\big)^2}$$

$$\Rightarrow \frac{da}{dz} = \frac{e^{-z}}{\big(1+e^{-z}\big)^2} = \Big(\frac{1}{1+e^{-z}}\Big)\Big(\frac{e^{-z}}{1+e^{-z}}\Big)$$

$$\Rightarrow \frac{da}{dz} = \Big(\frac{1}{1+e^{-z}}\Big)\Big(1 - \frac{1}{1+e^{-z}}\Big)$$

$$\Rightarrow \frac{da}{dz} = a(1-a)$$

The derivative of z with respect to w is,

$$\frac{dz}{dw} = \frac{d}{dw}\Big(wx + b\Big) = x$$

The derivative of z with respect to b is,

$$\frac{dz}{db} = \frac{d}{db}\Big(wx + b\Big) = 1$$

Consequently, the required gradients are,

$$\frac{dL}{dw} = \frac{a - y}{a\big(1 - a\big)} \times a\big(1 - a\big) \times x = \big(a - y\big)x$$

$$\frac{dL}{db} = \frac{a - y}{a\big(1 - a\big)} \times a\big(1 - a\big) \times 1 = a - y$$

4.3 Implementation

In the cancer dataset used for this project, there were 455 training instances with 30 features each and hence, 30 weights and a bias were required. If we represent the 30 features of the input by x_1, x_2, x_3, …., x_{30}, the weights associated with these features by w_1, w_2, w_3, …., w_{30}, and the bias term by b, then the computation graph of logistic regression for one training instance is depicted in figure 1. Vectorized implementation of logistic regression can easily be achieved for datasets with multiple number of instances and multiple features using explanation similar to section 4.1.

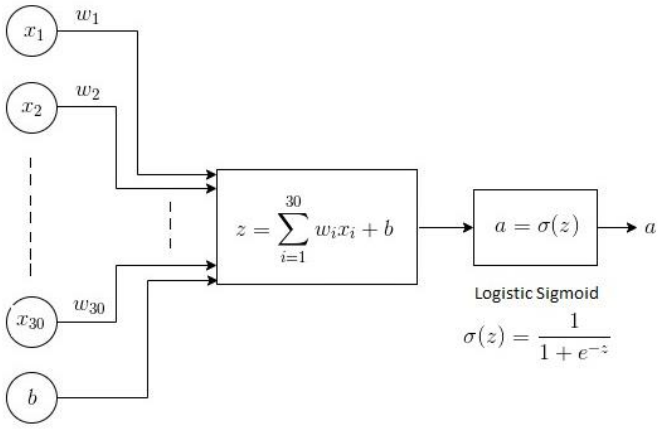


Figure 1: Forward propagation in logistic regression for one training instance

For this model, batch gradient descent was used for optimizing the weights and the bias. Therefore, the logistic regressor was trained with all of the training instances in every epoch. The loss function for the regressor was defined by the average binary cross entropy loss averaged over the number of training instances (say m).

$$L = -\frac{1}{m}\sum_{i=1}^{m}\Big[y^{(i)}\log a^{(i)} + \big(1 - y^{(i)}\big)\log\big(1 - a^{(i)}\big)\Big]$$

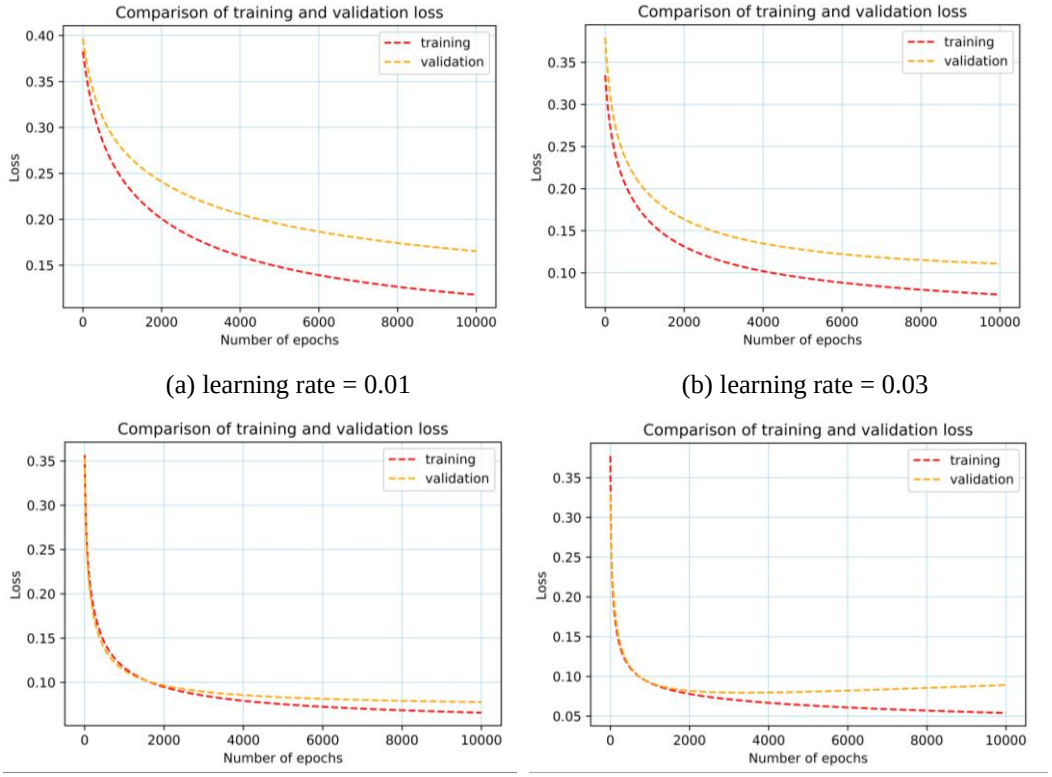
To update the weights and the bias, the formulae for finding gradients in section 4.2 were generalized for performing vectorized implementation of gradient descent. Arranging the training instances along the columns of matrix 'X' and the corresponding output labels in matrix 'Y', the gradients with respect to the weights vector and the bias can be defined as (here, 'A' is the activations matrix and 'W' is the matrix containing weights),

$$\frac{dL}{dW} = \frac{1}{m} X(A - Y)^T$$

$$\frac{dL}{db} = \frac{1}{m} np.sum(A - Y)$$

5 Results and Inferences

The performance of the logistic regressor on training and validation sets was analyzed for different values of the learning rate of the gradient descent algorithm. Particularly, four values of learning rate – 0.01, 0.03, 0.1 and 0.3 were considered and to compare their performance, the number of epochs was chosen to be 10,000 by default. Plotting the training and validation losses against the number of epochs for each of these four scenarios revealed that using a learning rate of 0.1 was the appropriate choice for this problem.



(a) learning rate = 0.01

(b) learning rate = 0.03

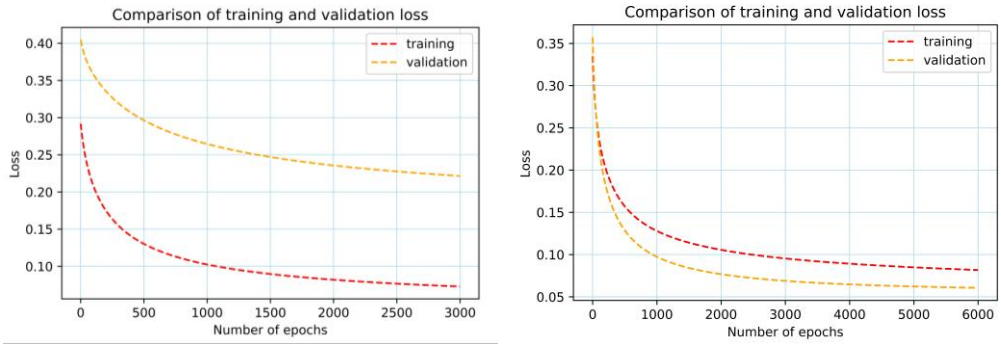
(c) learning rate = 0. 1

(d) learning rate = 0.3

Figure 2: Comparing training and validation loss against the number of epochs for different values of learning rates; (a) and (b) show underfitting whereas (d) shows overfitting

While using the learning rates of 0.01 and 0.03 provided a decent test set accuracy, the model still underfitted the training data because of non-convergence of training loss as can be evidenced from the plots 2(a) and 2(b). On the other extreme, the validation loss curve in plot 2(d) counterintuitively starts increasing after around 2000 epochs. This behavior is seen because the model has overfitted the training data. This shows that a machine learning model that overfits the training data does not tend to perform well on the validation and test sets. However, using a learning rate of 0.1 provided very small values of training and validation

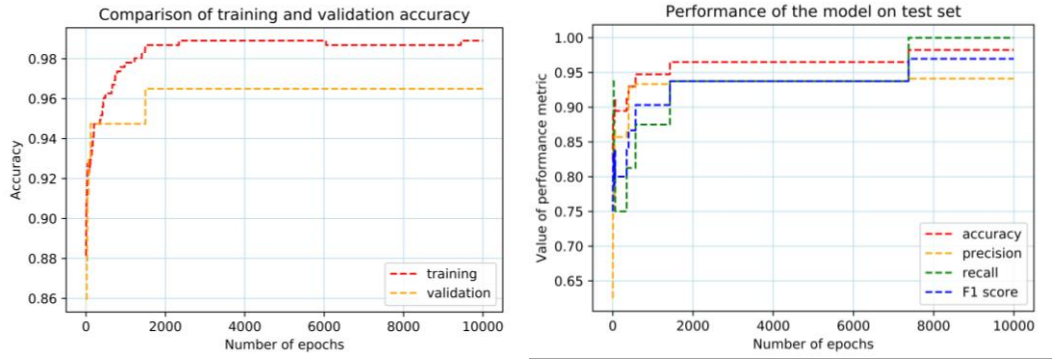
losses with neither underfitting nor overfitting the training set. Consequently, the learning rate was finalized to 0.1 for the rest of the implementations that follow.



(a) number of epochs = 3000 (b) number of epochs = 6000

Figure 3: Comparing training and validation loss against the number of epochs for different values of the number of epochs; (a) and (b) both show underfitting

After tuning the learning rate to 0.1, the next step was to find the appropriate number of epochs required to fit the training data. The plots in figure 3 represent a comparison of the training and validation losses when the model was trained with learning rate 0.1 for two values of epochs – 3000 and 6000. These plots clearly suggested that the convergence of gradient descent algorithm was not achieved at the end of the training phase. However, from figure 2(c), it can be observed that the training loss converges at the end of 10,000 epochs. Consequently, the number of epochs was finalized to 10,000 for the rest of the implementations that follow.



(a) (b)

Figure 4: (a) Comparing training and validation accuracy, (b) progression of evaluation metrics on the test set against the number of epochs

Finally, after tuning both hyperparameters of the logistic regression classifier, the training and the validation accuracies were plotted as shown in the figure 4(a). A straightforward observation from the plot is the increase in the accuracy over time. The figure 4(b) shows the performance of the model on the test set by computing the metrics accuracy, precision, recall and F1 score defined as,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

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

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

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

where TP, TN, FP and FN indicate the true positives, true negatives, false positives and false negatives respectively [4]. At the end of 10,000 epochs, the accuracy was found to be 98.24%, the precision was found to be 0.94, the recall was found to be 1.00 and the F1 score was found to be 0.97. These metrics suggested that the regression model performed really well on the test set. In addtion, an important observation was that the model with the aforementioned hyperparameter setting often produced perfect results on the test set as shown in figure 5. A possible explanation to this behaviour is the fact that the initialization of weights and the distribution (partitioning) of the data into training, validation and test sets was completely random. In the scenarios where the training instances and the test instances are similar, this kind of performance can be expected.

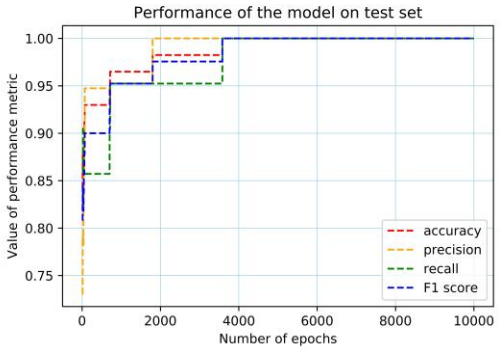


Figure 5: Performance of the classifier on test set depicting perfect metrics

6 Summary

In this project, a machine learning model to classify breast cancer tumors as benign or malignant was built from scratch using logistic regression with gradient descent optimizer. The dataset which contained 569 instances of cancer tumors with 30 features each underwent scaling and normalization before getting divided into training, validation and testing sets in 8:1:1 proportion. While training, the performance of the model on the validation set was analyzed for different settings of the algorithm learning rate. The best model was then chosen based on the validation accuracy and loss incurred over the epochs. The found model was evaluated on the test set and performance metrics like F1 Score, Accuracy, Precision and Recall were calculated. Results indicated that the logistic regression classifier performed very well on the test set with good metrics.

References

[1] Bishop, C.M., 2006. *Pattern Recognition and Machine Learning*. Springer.

[2] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[3] https://scikit-learn.org/stable/modules/preprocessing.html

[4] https://scikit-learn.org/stable/modules/model_evaluation.html

[5] CSE 574 Slides by Prof. Sargur Srihari

[6] Andrew Ng's 'Machine Learning by Stanford University' on Coursera