Linear Regression Project

Benjamin Borden
Phuong Truong

December 15 2017

Abstract

This project had us implement a linear regression model which will analyze data gathered from patient medical information. The goal was find some pattern or correlation in the data which will aid in predicting if the patient has an invasive form of breast cancer while the cancer is still in it's early stages so patient has the best chance of living. We implemented and minimized a cost function which can be found in the appendix which allowed us to find the best set of theta values to predict the likelihood of the patient having cancer. We then converted that probability into a Boolean value and determined our accuracy based of the true values of whether the patient had cancer or not. The conclusion of our project was that our model was able to predict cancer with an accuracy of 0.9416, a sensitivity of 0.8864, a specificity of 0.9474, and a precision of 0.9286. We feel confident these results make this model a valuable tool in the medical field.

Introduction

Machine learning is a field related to both probability and computer science which has rapidly expanded in recent times. We will be dipping our toes into the field of machine learning by creating a linear regression model. When practicing the creation of linear models, its better to begin with an application which demonstrates to students the real world impact of linear models and their predictive abilities. Using Python, we will be implementing a sigmoid and cost function which will determine the probability of an individual having breast cancer, predict functions which determine cut off points in order to divide the patients into cancer vs no cancer, divide our data into training, validation, and splitting sections in order to build and test our model, build models for several different lambda values, and classify our model based on it's score of accuracy, sensitivity, specificity, and precision.

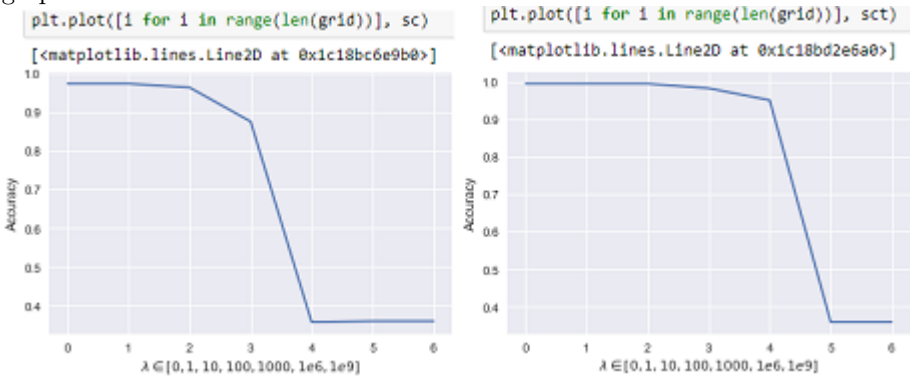
Background

It is estimated that 252,710 women will have been diagnosed with a new case of invasive breast cancer at the end of this year. Therefore, the problem is to make a prediction on who will have a risk of having breast cancer in the future. We're using data from the Wisconsin Breast Cancer Data Set, which contains 683 entries, each with 10 features with a class identifier, to distinguish between benign and malignant tumors. The term "benign" describes a tumor, condition, or growth that is not cancerous and unlikely to spread, while the term "malignant" describes a tumor that has a potential to, or already has spread to other parts of the body. Taking the values from the dataset as input, we design a predictive model to determine whether or not a patient has a breast cancer and distinguish between benign and malignant tumors. Our data is contained in a database with 10 columns with each columns representing a value of a characteristic of every patient and 683 rows with each row representing an individual patient. The columns which contain our data represent the patients' Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin,Normal Nucleoli, Mitoses, and a tenth column of 1s to compensate for the linear regression method. These columns will be referred to as X. We also have a separate column vector Y which contains the true value for whether or not the patient had cancer which we will consider our target values for our model.

Model and Analysis

In this project, we have built a logistic regression model to predict if a patient has a breast cancer. Logistic regression is adapting linear regression to a situation where the outcome has only two values, 0 or 1. Thus, it will follow a Bernoulli distribution, and the goal is to find the probability that the result is 1, which means the patient has breast cancer. In order to use linear regression on a Bernoulli outcome, we use the sigmoid function, which is defined as: $g(z) = \frac{1}{1+e^{-z}}$ and in our code as the function g[1]. Next, we implement h[2] and computeCost[3], which returns the value of the hypothesis function and the cost. In order to utilize our function g, we built the function h which performs dot multiplication to compile our theta values and X values into a single z value to be fed into g. Our computeCost method replaces the traditional sum of squared errors function as the value which we attempt to minimize in order to increase accuracy. After implementing cost function, we find the optimized theta that minimizes the cost function through optimizeRegularizedTheta. Then the fit function will implement optimize regularized theta to find the optimal collection of parameters and return the score of accuracy that the model obtained using the score function. The score[4], predict class[5], and predict[6] work together to predict the probability of an individual having breast cancer(predict), determine a cut off point and file each individual as a 0 or 1 based off the likelihood that individual has cancer(predict class), and compare our predictions with the

true values Y which contain the answer of whether that individual did actually have cancer(score). It should be noted that the division point that we chose for determining whether an individual has cancer or not was 0.4 rather than 0.5 as 0.4 provided higher accuracy, sensitivity, specificity, and precision. The hypothesis for a logistic regression is the data is linearly separable, however, in our case, the data is not. In order to achieve a nonlinear decision boundary, transform the feature space with a non-linear feature mapping of degree 2. As a result, our vector of features will be transformed into a n'- dimensional vector. We then split our dataset into training, validation, and testing sets of sizes 60%, 20%, and 20% in order to avoid bias when choosing our model. Next, we build the model for each lambda in [0,1,10,100,1000,1e6,1e9] using the find lambda function to find the model with lambda that has the highest validation accuracy as per instruction. The accuracy of each lambda was plotted on the graphs below.



According to the plot, the smaller the lambda, the higher the accuracy. Thus, the model with lambda = 0 has the highest accuracy. Finally, we build the confusion matrix and calculate the accuracy, sensitivity, specificity and precision of our model to measure how well our model performs.

Prediction

The result of our prediction function when using our model is that among 683 patients, 665 are predicted to have breast cancer, and 18 are not, with the accuracy of 97.4%. After polynomial feature transformation, 671 patients are predicted to have breast cancer, and 12 are not, with the accuracy going up to 98.2%. According to the confusion matrix, our model's accuracy is 0.9416, sensitivity is 0.8864, specificity is 0.9474, and precision is 0.9286.

Discussion

According to the prediction result, our model successfully achieves the goal of the project. The true positive rate and true negative rate are all above

90%, which are acceptable in evaluating whether or not a patient has a breast cancer. To reduce the uncertainty and improve the reliability of the model, another algorithm of optimizing theta could be used. Based on our model, physicians can predict if the patient has a breast cancer, with the accuracy rate of 94%. However, there is a fair and well reasoned argument that the sensitivity ie "Of the patients who had cancer, how many were caught by the model" is the most important score of the model. By that logic, our model was less than ideal as sensitivity was our lowest score. However, we feel the high accuracy, specificity, and precision more than make up for the sensitivity. Doctors and other individuals implementing this model should be aware of the sensitivity score however so they can factor that knowledge into their medical recommendations to patients. Despite that, this model remains a valuable tool in the modern world for predicting breast cancer sufferers. Because of the fast running time, medical professionals can use this model to evaluate a large number of patients.

Appendix

1

```
# Solution
def g(z):
    """Sigmoid function"""
    return 1/(1+np.exp(-z))
```

2

```
def h(theta,X): # Logistic hypothesis function
    return g(np.dot(theta,X))
```

3

```
def computeCost(theta,X,Y, Lambda = 0.):
    """
    theta is an n-dimensional vector of parameters
    X is data matrix
    Y is a matrix with m-rows and 1 column of target values
    This includes regularization if you set Lambda to not be zero (make sure it is non-negative)
    """
    m = Y.size
    n=len(theta)
    X=np.matrix(X)
    return ((1/m)*sum([(-Y[i][0]*np.log(h(theta,X[i].transpose()))-(1-Y[i][0])
    *np.log(1-h(theta,X[i].transpose()))) for i in range(m)]))+((Lambda/(2*m))*sum([(theta[j]**2) for j in range(n)]))
```

4

```
    def score(self, X, Y):
        X = self.predict_class(X)
        count = [X[i]==Y[i][0] for i in range(len(Y))]
        count = Counter(count)
        print(count)
        ac=(count[True])/(sum(count.values()))
        print("Training Accuracy with Lambda = "+str(self.Lambda)+" "+str(ac))
        return ac
```

5

```
    def predict_class(self, X):
        X=self.predict(X)
        return [1 if x>0.4 else 0 for x in X]
```

6

```
    def predict(self, X):

        Xval=X.values
        th = (np.matrix(self.theta))
        return [h(Xval[i],self.theta)[0] for i in range (len(Xval))]
```