Regularization in Logistic Regression

Dataset Cancer

Charles Acevedo Díaz
Ingeniería de Sistemas
Universidad Tecnológica de Bolívar
Cartagena, Colombia
chad9591@gmail.com

I. Abstract

This paper will be about the logistic regression can be implemented and improved by adding a regularization parameter, in order to avoid some common errors. In this document you will find the *documentation* of an example that was solved using logistic regression with its regularization, and we'll see how it works and the outputs we received after executing the python program we have designed.

Keywords—Machine Learning; regression; Regularization; logistic; supervised learning

I. Introduction

The main purpose of this workshop is to implement the logistic regression to generate a model which is capable of predict new samples input different from the dataset we possess, so we have to implement a regularization parameter to help the method improve the way it trains the models, notices that the word "*modules*" is in plural, as we will have to train different models in order to identify which regularization parameter has the best resulting model, it will be checked by verifying the score of each model; the scores is how good is the model when it has to handle new samples that are supplied to it during prediction.

This method is intended to avoid the overfiting problem that we get when the samples are too disperse and the model is overtrained to this samples and has a very good approach to them, but then when new samples appear, the model fails with a high rated error.

We've used a module that the programming language we've used has available to work with supervised learning, as it's the case of sklearn, it provides a wide range of functions that help us programming an intelligent agent that will solve the prediction situation, in this case we have implemented the logistic regression it provides.

II. Theory

Logistic Regression:

This is a method that helps finding discrete results, or binary outputs, which means that is intended to be used in circumstances where the results can be classified into a certain amount of groups like "Yes/No", "To buy, To sell", etc. The output here is a probability that takes a particular value based on combination of values taken by the predictor.

Its name is based on the function used at the core of the method, *the logistic function*, also called *sigmoid function* was developed by statisticians in order to describe properties of population growth in ecology. This is an S-shaped curve that take any real value number and gives as result a value between 0 and 1, but never exactly at those limits.

Regularization

Sometimes, when we are training the model, it gets too fitted to the dataset that it makes pretty good approaches to those samples, but when it's about new entries and unseen data it misbehave, it makes bad approaches and bad predictions, that's because the model has been overtrained with the data set, and we don't want it to memorize the dataset we want it to generalized the info and make good predictions with new data.

Therefore we add a regularization parameter (bias) that will try to avoid the overfitting, but we have to be careful, because if the bias is too big, we will end up having an underfitted model, which has bad performance in both, dataset and new data.

Cost Fuction

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y_i(\log h(x_i)) + (1-y_i)log(1-h(x_i)) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Regularized Gradient

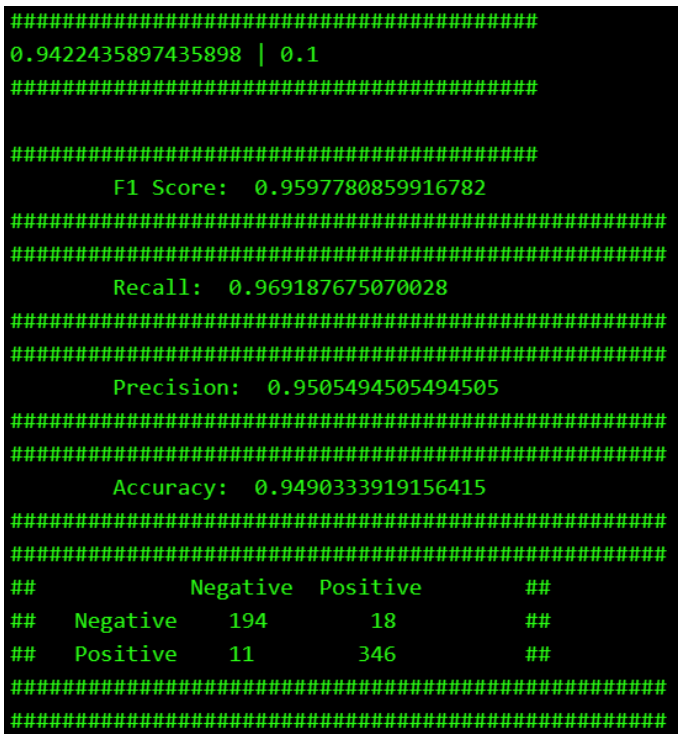
$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)x_{ij} - \frac{\lambda}{m}\theta_j$$

III. Procedure

- The dataset needs to be splitted in two groups, for training purpose; in our case we have decided to split it with 70/30 percent relationship.
- We proceed to train the data with multiple lambdas (a.k.a. bias), in order to verify the best performance. The way the model was trained used the KFold method.
- When we have the best performance (best model), we proceed to train it again.
- We test the model by predicting results with it.
- We verify if the predictions are good or not. We do this by applying some metrics to the results like: accuracy, recall, precision and f1 score.

IV. Results

The program was designed to deal with the prediction of the final model, and verify whether if it's good or not, so as outputs it shows some scores using different metrics we studied and apply early (Confusion Matrix, F1_Score, Recall, Accuracy and Precision), the outputs we get can be seen in the following figure:



It also shows the score of the model that has been chosen and the value of lambda that was used in it.

Conclusions

Based on the outputs we got from the program, we can see that the model that has been chosen has a pretty good performance, and also has a good score resulting from the cross validation, which tells you that it has well performance to unseen data and new entries.

We can conclude from this results that the fact of using a regularization parameter is a good way to make a good training for the model as it gives some good results when it's used, gives good performance with dataset (known data) and new data, which is what we want, an agent capable of predict rather than memorize.

References

[1] Jason Brownlee, "Logistic Regression for Machine Learning" Phil. machinelearningmastery.com/logistic-regression-for-machine-learning/ April 2016.

[2] Medium.com/@aprendizaje.maq/regresion-lineal-con-gradiente-decendiente-c3b5ca97e27c

[3] http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[4] http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

[5] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

[6] http://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics