MC886 - Machine Learning Assignment 1 - Linear Regression

Fabio Tanniguchi
RA 145980
ftanniguchi@gmail.com

Felipe Caminada
RA 140604
caminadaf@gmail.com

I. Introduction

The motivation behind this assignment is to reinforce the contents studied in class about Linear Regression and the basics of Machine Learning.

The task at hand is to find a Linear Regression based model to predict the release year of songs ranging from 1922 to 2011.

The songs included at the dataset are mostly western, commercial tracks ranging from 1922 to 2011, with a peak in the year 2000s, and they are a subset from the Million Song Dataset, a collaboration between LabROSA (Columbia University) and The Echo Nest.

II. Activities

First of all, there was an effort in order to understand the usage of Python library SciKit-Learn. The given training dataset was used to train a sklearn model, which was used to predict the year of the songs of the testing dataset.

Then, the same steps were made but using Descendant Gradient. On the first attempt, the prediction results were strange, with bigger numbers of years than expected. Searching on the Web, it was found [2] that Feature Scaling was required to use the sklearn's Descendant Gradient implementation and there was a sklearn implementation of Feature Scaling. So, using Feature Scaling, the training dataset was used to determine the model and the prediction results were finally coherent.

After these basic experiments, the group was ready to work harder with the specific given problem.

III. Proposed Solutions

The first step to improve the learning results was to understand the dataset that was being worked on, and filter out features that wouldn't aggregate any meaningful result.

After some analysis of the data distribution and finding the ones that best represented the whole model, a slightly better data distribution graphic was plotted. However, it was noticed that there was a certain tendency for the timbre features to be symmetrical in relation with the horizontal axis.

With the knowledge that the Linear Regression model would have a hard time learning from this data distribution, we've found that by using the absolute value for each feature, we could increase our results, as it can be noticed on 2.

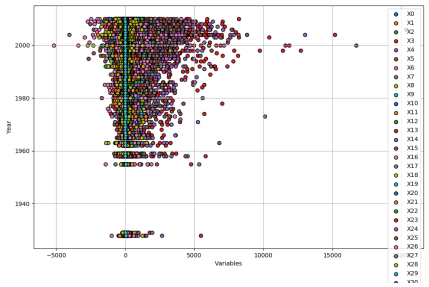

Figure 1. Graphic of data distribution per feature per year.

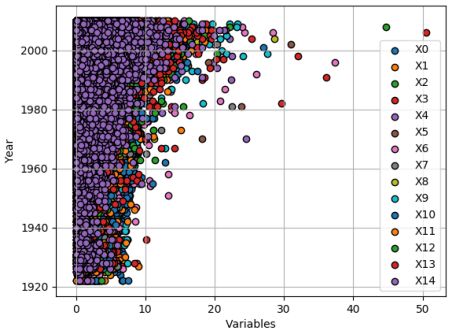

Figure 2. Graphic of data distribution per feature per year with abs applied.

IV. Experiments and Discussion

The experiments on the following sections were performed using the training database split into two sets: training and validation. This division was performed by first shuffling the whole database and then separating a percentage of 30% it for the validation process.

The metrics that were used to compare the models were Mean Squared Error and R^2 regression score function.

After each test, a cross-validation graphic was plotted to check if there was over or under fitting.

Ordinary Least Squares Linear Regression

The default Linear Regression implementation present on the sklearn library is the OLS Linear Regression, and we used it as-is on this task, more information on [3].

The first run of this model was with the original unmodified dataset, and even though the results were fairly high in comparison with the rest of the numbers presented on this report, the cross-validation test gave out that an over fitting of the data was happening:

Figure 3. Cross-validation graphic of the linear regression model on the original data.

After the data normalization, scaling and feature filtering, the result was as below:

Mean squared error: 104.81

Variance score: 0.12

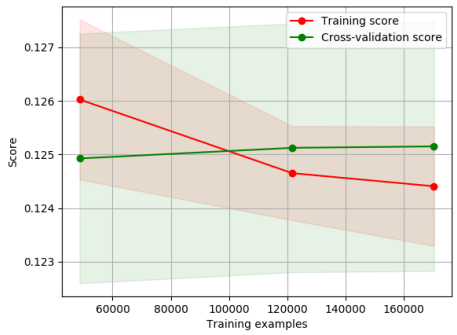

Figure 4. Cross-validation graphic of the linear regression model on the final database.

As it can be noticed, the variance score was 25% lower, but the cross-validation results were much closer to normalization.

Stochastic Gradient Descent

The second try to improve the model was by using a Gradient Descent on the linear regression to obtain the best parameters for each feature. Sklearn provides us with the Stochastic Gradient Descent model for this, as it can be seen on [4]

Additionally to the cross-validation, a score-iterations graphic was generated to check the complexity of the model.

Three attempts were done on this model, each with a different learning rate, as below:

Inverse Scaling:

$$\eta^{(t)} = \frac{eta_0}{t^{power\_t}}$$

For the inverse scaling, the following parameters were used: epsilon$\bar{0}$.1, eta0$\bar{0}$.01, power$\bar{t}$0.25.

Immediately SGD proved to perform better than the Linear Regression model, even in the same conditions:

Mean squared error: 100.82

Variance score: 0.16

Figure 5. Cross-validation graphic of the SGD model with inversed scale learning rate.

Constant Rate: The last attempt was with a constant rate of 0.0001, and the results where still very similar to the Inverse Scaling.

Mean squared error: 101.06

Variance score: 0.16

Figure 6. Cross-validation graphic of the SGD model with constant learning rate.

Linear Regression + Polynomial Interpolation

The last model worked on this activity was an experiment done by pipelining the Linear Regression model with a Polynomial Interpolation of degree 2. The reasoning here was that since the data had a tendency to sparse on the later years, a polynomial function would be able to fit better than a simple linear model. For more details on how this is implemented on sklearn, please see [5].

Mean squared error: 98.26
Variance score: 0.18

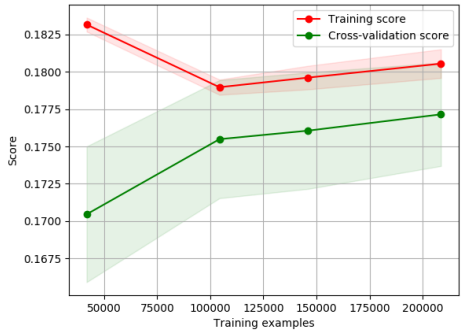

Figure 7. Cross-validation graphic of the Linear Regression model with Polynomial Interpolation.

Since we are still using the Linear Regression as base, this did not work as well as we initially though, but even so it was still better than SGD, which is more optimized than the basic Linear Regression model. The cross-validation results also prove that it is more efficient at learning the model without over fitting.

With this final result, we've taken the decision of running our final model with the actual test database provided:

Test Results

Mean squared error: 99.60
Variance score: 0.16

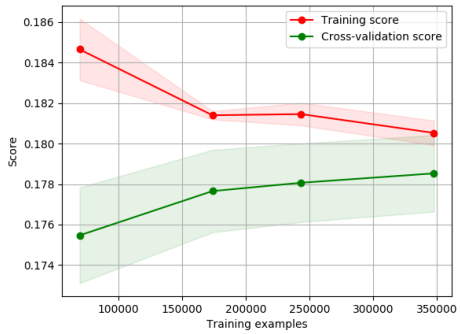

Figure 8. Cross-validation graphic of the Linear Regression model with Polynomial Interpolation using the test database.

As expected, the model performed slightly worse on the test database since it may contain different data than what our model was optimized for. But still, in the end it proved to be more efficient than the rest of the models on the training database, and may be an interesting approach to follow in the future.

Additionally to the cross-validation graphic, a Prediction graphic was plotted to check the general behavior of the predictions.

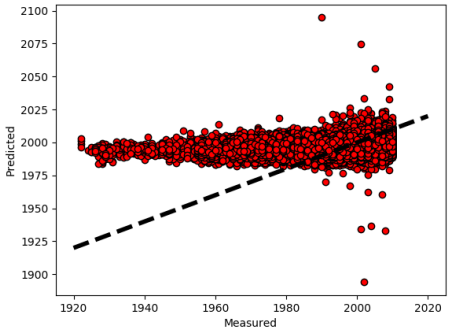

Figure 9. Prediction graphic of the Linear Regression model with Polynomial Interpolation using the test database.

Figure **??** clearly shows that there the huge imbalanced database impacted on the performance of the model, since most of the predictions resulted on late 90's or 2000's dates.

V. Conclusions and Future Work

The assignment was very interesting because the group could put in practice some linear regression knowledge and understand better the techniques involved.

People interested to continue this work in the future are suggested to study more techniques and approaches in order to achieve better test results. It is important to note that models based on Linear Regression may not perform so well in this kind of model, as the proved by the results here presented.

Also, a good observation is that the problem is really difficult because the dataset has some limitations (put a whole song into some segments can be a poor representation, for example). Also, further studies may want to consider the imbalance present in this dataset, and how to counter this effect on the model.

So, trying to solve the problem with a different dataset with much more features can result in better test results. But the trade-off is that more features will result in a bigger computational cost.

References

[1] http://scikit-learn.org/stable/
[2] https://stackoverflow.com/questions/31443840/sgdregressor-nonsensical-result
[3] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.h
[4] http://scikit-learn.org/stable/modules/sgd.html#regression
[5] http://scikit-learn.org/stable/auto_examples/linear_model/plot_polynomial_interpolation