BLG 454E Learning From Data

Term Project Report

**Hasan Hüseyin Kaçmaz, Ahmed Burak Gülhan, Kadir Bulut Özler**

{kacmazh17, gulhan16, ozler16}@itu.edu.tr

Abstract

In some cases where Machine Learning is applied, number of features in the dataset exceeds the number of instances. In other words, model needs to decide but it has too many information to consider. This is called “curse of dimensionality”. In this report, we explain our approach to the problem of classifying 120 patients if they have ASD or not by using 595 features of human brain. We used several methods to reduce the number of features. We also trained several classifiers and showed 65% accuracy on one half of the given test data on Machine Learning platform Kaggle.

1. Introduction

Basically we are doing supervised learning with a low number of samples and a high number of features. To model our data, since it has a high amount of features we used dimensionality reduction. Our approach is standardizing the data, using Recursive Feature Elimination and Principal Component Analysis together to reduce the number of features to a reasonable number, 12. As classifier, we applied K-Neighbors Classifier, Multi-layer Perceptron Classifier, Support Vector Classification, and Random Forest. We finally chose Random Forest as our classifier since it has much more decent scores than others.

**Kaggle names:**

abgulhan : Ahmed Burak Gulhan 150160903

Kadir Bulut Ozler : Kadir Bulut Özler 150160715

Huseyin Kacmaz : Hasan Hüseyin Kaçmaz 150170705

**Kaggle team name:** 150170705\_150160715\_150160903

**Kaggle Rank and Score:** PublicRank: 38 (tied with rank 30), Private Rank 6 (tied with rank 5). Score: 0.65 (public), 0.625 (private)

1. Dataset

Since we have relatively few data (120) with a large number of features (595), we suffer from “curse of dimensionality”. To get rid of the curse of dimensionality we must reduce the number of features. The general rule of thumb is that, number of data samples should be about equal to 10 times of the number of features [1]. Since we have 120 features we should reduce it to around 12. To do this we have to use a dimensionality reduction method. For this we used feature elimination and principal component analysis. With Recursive Feature Elimination we eliminated unnecessary data until we were left with 235 features. This was still too much, so we then used PCA to reduce the number of features to 12.

1. Methods

We used Python as our programming language with a simple text editor. We also used “numpy” and “pandas” for data structures, “sklearn” for methods.

* 1. **Recursive Feature Elimination**

The feature elimination method takes the features and labels, applies fitting to detect weakest features. In each iteration, it recursively eliminates the features it found as weakest. It basically pays attention to dependencies and co-linearities between the features. To find the best combination, it employs cross validation method.

* 1. **Principal Component Analysis**

This method is basically projecting dataset into lower dimensions. It takes existing features and extracts new features that also represent the characteristics of the data. Number of new features is lower than number of original features. [5]

* 1. **K-Neighbors Classifier**

This is a classifier which considers k neighbors of new instance in order to decide which class it belongs to. The number k is specified by user. Classifier locates the new instance in the dataset space and by using Euclidian distance it detects k nearest neighbors and checks their classes. Then it assigns new instance to the majority class. [6]

* 1. **Multi-layer Perceptron Classifier**

This classifier employs neural networks with multiple layers, uses backpropagation to update parameters, and finally classifies the given features to a binary label.

* 1. **Support Vector Classification**

Support vectors classifies given data by calculating the hyperplane borders by trying to maximize its width. The data points that are close to the hyperplane affects its position in the space.

* 1. **Random Forest (used in final code)**

This classifier is basically an ensemble system consists of decision trees. It applies bagging method. In other words, it divides the data for several sub-models (decision trees in its case), and combines their classifying abilities to increase final classification decision. [7]

1. Results

We first used a feature *selection* method for dimensionality reduction. At first we used only recursive feature elimination with cross validation (RFECV) with the estimation model SVR. However despite having acceptable results in out testing data (used 20% of testing data as training) we achieved unsatisfactory results on kaggle. We tried several different classifier algorithms. We came to the conclusion that our problem is not with the models, but with the dimensionality reduction mechanism.

We then used principle component analysis (PCA), which is a feature *extraction* mechanism, together with RFECV (only using PCA can cause performance to decrease if there is a lot of irrelevant features [2]). We thought that this might give us a better result since, due to the nature off the data, feature elimination loses us too much information, since it discards many slightly relevant accurate features. However feature extraction generates us new features based on the old features, which allows us to keep more information [3]. At the end we used RFECV and PCA together (we also used standard scaling before applying RFECV to be safe).

For our prediction model we used Random Forest algorithm, since it is a good classifier and does not suffer from over-training [4] (therefore we don’t need to fine tune the amount of trees used).

1. Conclusions

At first we did feature reduction in an incorrect way. We first thought that our algorithms that we used were insufficient, but after trying several algorithms we came to the conclusion that our feature reduction mechanism was incorrect. We then fixed this using feature selection *and* feature extraction methods.

To improve our accuracy we might use different classifiers and see which one performs the best (we only checked random forest after fixing our dimensionality reduction mechanism). We can also definitely improve our speed of our program. RFECV takes too long finish. We set RFECV to eliminate the 5 worst features, but there can still be more improvements to be made in terms of speed.

##### References

1. <https://datascience.stackexchange.com/questions/19980/how-much-data-are-sufficient-to-train-my-machine-learning-model>
2. <http://blog.explainmydata.com/2012/07/should-you-apply-pca-to-your-data.html>
3. <https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c>
4. <https://en.wikipedia.org/wiki/Random_forest>

[5] https://www.scikit-yb.org/en/latest/api/features/rfecv.html

[6] https://www.codershood.info/2019/01/25/k-nearest-neighbors-k-nn-classifier-using-python-with-example/

[7] https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd