Otto Group Product Classification Challenge

Amal Targhi Mihaela Sorostinenan Mohamed Mohamed

December 2, 2016

1 Introduction

The purpose of our project is to analyze the OTTO Product Classification Challenge on Kaggle and try to propose a solution that yields good results. We first explored various classification methods in order to see which one is more fit (gives the best results) for this specific classification problem. Thereafter, we tried to see if dimensionality reduction can help us obtain a higher accuracy. Finally we report about the results we obtained and the conclusions we have drawn while trying to solve this challenge.

2 The data set

The OTTO Group Challenge is a competition which has as target to build a classifier, able to distinguish, for each product, a specific category to which it belongs. The data set provided by the organizers contains more than 200,000 products, where each product is described by 93 features. As mentioned before, the objective was to classify these products in one of the 9 categories. For this purpose, 30% of the data set was used for training and validation, while the remaining 70% was used to test the classifier.

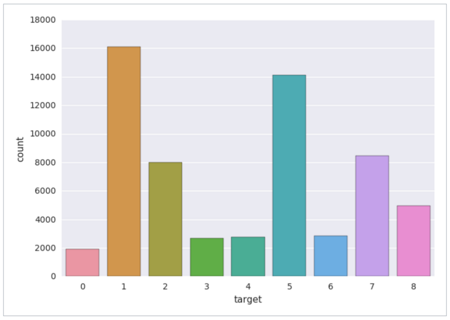

Figure 1: The partition of classes in the data set (class imbalanced problem).

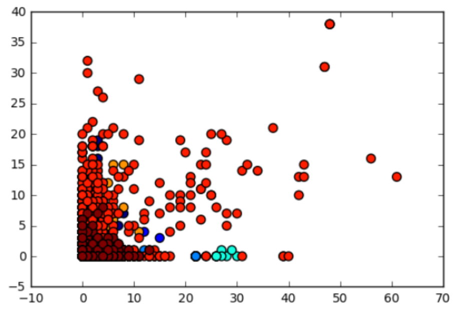

Figure 2: The classes have very similar characteristics. (class overlapping problem)

3 Classification algorithms

In order to find the algorithm which best fits to our problem, we used the data set to train several different classifier. In this section we will present some details on the algorithms which showed best results for our problem, while in the last subsection we mention the ones that didn't show good results.

3.1 Random Forest

Random forest is a tool which takes a subset of observations and a subset of variables to build decision trees [1]. It builds multiple such decision trees and merges them together to get a more accurate and stable prediction.
In order to improve the parameters of our predictive model we have used **PCA** to extract the best features, we deducted the number of dimensions using Tree-based feature selection.The dimensions was reduced to **21** according to Select From Model selection. There are primarily 2 parameters which we modified: **1.n_estimators:** represent is the number of the built trees. Higher number of trees a better performance but it make the code slower. this parameters is chosen depending to the machine capacities. In our case we had chosen **800.**
2.max_features: represents the maximum number of features Random Forest is allowed to try in an individual tree. Increasing max_features generally improves the performance of the model as at each node now we have a higher number of options to be considered. However, this is not always the case. In order to decide the number of features we had used Cross validation for many parameters. The best result we obtained was **3** (figure 3).

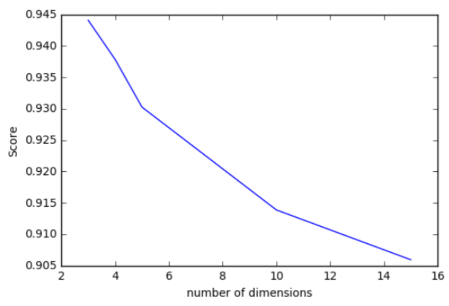

Figure 3: Cross validation for choosing the number of features

3.2 XGBoost

Xgboost "Extreme Gradient Boosting" algorithm is a supervised algorithm [2] , working based on the ensemble trees, where the decision depends on summation of the scores of multiple optimized trees. The leaves values aren't the target but they are real numbers, called scores. The objective function consists of two functions (training loss and regularization). These functions are optimized at each iteration in the training phase. The training loss represents the difference between the desired and the actual output. Regularization represents the model complexity and this function is optimized to avoid over-fitting. This algorithm aims to learn the gradient, where at each iteration, a new tree is optimized based on the gradient which was learned using the trees created in the previous iterations. Finally the output can be considered the summation of the output of the trees which are created through all the iterations. The trained parameters, which are changed at each iteration for the created tree, are the tree's leaves scores and the tree structure. Learning the tree structure depends on the technique of pruning. This technique represent a way of the dimensions reduction based on the dimensions correlations with the target. And this explains why using techniques like PCA for dimensions reduction may lead in a drop in the accuracy. what The selection of the number of iterations is important for the final XgBoost algorithm accuracy. k-fold validation is used to measure the convergence at each iteration till found the best number of the iteration. We used 10 folds. Since the distribution of classes in the train data-set is uneven, we used the stratified k-fold, which guarantees to maintain approximately the same percentage of classes in each set.

3.3 Other Algorithms

In the initial phase of our project we also trained an Naive Bayes classifier on the data set, using both Gaussian and Multinomial distributions. While the result using Multinomial distribution was better than Gaussian, both results where bad. We hade also test Knn and SVM with multi-parameters but the results was not pertinent.

4 Results

Table 1 shows the best accuracy we obtained for each algorithm we tried, according to the Kaggle platform. As it can easily be seen, the classifier based on Xgboost obtained the highest accuracy, which is considerably better than all the others. Random forest algorithm, which is also based on trees obtained the second based accuracy in our trials, while the accuracy of Naive Bayes classification was significantly low, compared with the others.

Table 1: Accuarcy of classification for each algorithm

Algorithm	Accuracy
SVM (RBF kernel)	**7.1**
Random Forrest	**1.1499**
Knn (k=6)	**7.36**
Artificial neural networks	**7.36**
Multinomial Naive Bayes	**11.68843**
Xgboost(180 Iterations)	**0.46606**

5 Conclusion

Based on results analysis, we can conclude that there are correlations between some features and the target. Therefore, when PCA was used to do dimension reduction for the features, this led to accuracy dropping of the Xgboost algorithm, although we tried with 70 dimensions, where the accuracy of Xgboost depends on constructing the trees using the correlation between the features and the target. Also it is clear that Xgboost reached the best accuracy for the OTTO Kaggle Challenge between the algortihms that we tried. We believe that this is due to the fact that Xgboost successfully selects among the numerous features in the dataset, the ones which are the most important at each step. Although there are big variations between the number of the samples per each class, the accuracy dropped when we tried to do equality for the number of the samples per each class by selecting the first 1929 sample per each class. This indicates that there are variations between the samples per each class in the training set. Finally, the architecture proposed by the winners of this competition is based on a layered approach, in which various algorithms were used in the first layer in order to generate meta features for the second layer. In the second layer three classifiers (Xgboost, NN, and AdaBoost) were trained using these features and the final result is computed in the third layer as a weighted mean of the previous one.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *arXiv preprint arXiv:1603.02754*, 2016.