Report on the Kaggle competition

The Free Riders

17 March 2016

Introducion

We describe the results and approaches undertaken by the Free Riders team for the Kaggle competion done in the context of the Advanced Computational Methods and Machine Learning classes at the Barcelona Graduate School of Economics.

This competition is about predicting accurately the popularity of Mashable.com's news articles. The original target variable, the number of shares, has been transformed into a categorical variable, ranging from 1 (least popular) to 5 (viral). The whole data set is divided into two parts - the training set (30 000 obsrevations) and the test set (9 644 observations). In order to predict the popularity of new articles, we use computational statistics and machine learning. More precisely, we implement algorithms ranging from a single model type (e.g. Random forest) to different combinations of models. A more detailed discussion regarding these methods can be found in the Review of Models and Predictions part of the report. Before this, we shall discuss the way we explored the data.

Data Exploration

The data, acquired and preprocessed by K. Fernandes et al., is composed of 60 variables. In order to acquire more extensive knowledge of these features, we conduct data exploration. This consists of summarizing not only the data distribution, but also the conditional distribution with respect to the target variable.

Additionally, we tried to identify "ambiguous" data points that would influence our inference. For example, observation 16546 has a "kw_max_avg" value that is considered senseless regarding the rest of the variable distribution. In fact, many of these particular observations were easily detected as they had the same extreme value for some variables. We store the ID's of such points into a data set in order to later exclude the problematic observations. We evaluated our performance using the full data, and the reduced data. This led us to prefer the second option for most of of the algorithms.

Also we have noticed that the data set is imbalanced. We used Synthetic Minority Over-sampling Technique to counter this by augmenting classes 4 and 5. However, this approach did not increase our prediction power. Therefore, we focused on classifiers dealing well with imbalanced data like Random Forest.

At the end of our data explortion, we thought about creating new features from the original ones. We scraped google news popularity results based on the article's title and the date in the URL feature. This approach unexpectedly only inconsistently improved the random forest's accuracy.

Review of Models and Predictions

The plot in the appendix displays the performance of the methods on the public and on the private leaderboard. Now, rather than describing all the techniques we tried, we will concentrate on the ones which allowed us to achieve a notable prediction power, and the ones that seemed relevant for our competition.

Single Method Approach

We first tried multinomial GLMs penalized model with Ridge, Lasso and Elastic Net. They performed rather poorly, achieving around 50% on the leaderboard. But we used them as benchmarks for the future methods.

We then tried the e1071 package that provided a very tunable implementation of Support Vector Machines, notably in terms of kernel types, classification methods (c and nu) and arguments for class imbalance. Additionally, a grid search was performed to tune both the weights on the misclassification costs and the margin. However, we stopped this procedure after about 10 hours of running. Also we made a submission, using a radial kernel and a basic configuration. It obtained a score of 0.506. Mainly due to its expensiveness, SVM wasn't considered anymore as a relevant alternative.

The best off-the-shelf methods turned out to be random forest and boosting. In particular, since random forests are fast, they are relatively easy to tune. Based on the private-leaderboard result of the interim model, this algorithm would be used as the new benchmark.

Ensemble of Methods Approach

Instead of running isolated models, we considered ensemble learning. This machine learning technique consists of letting different 'strong' learners voting for a particular class. We expect such 'generalizer' method to have smaller variance than the variance of every submodel separately, which would increase the predictive power.

SuperLearner

The first variant of the ensemble approach uses the superLearner package. It allows to combine several submodel class predictions within a meta-learner algorithm. Thanks to its proper mechanisms, the meta learner selects the relevant predictions in order to output finally their best combination. Furthermore, the package provides an internal cross validation option, as well as a parallelized implementation for some models. However, since it doesn't support multiple classification, we had to transform the target variable to a binary matrix and train the model using a mixture of GBM, random forest and Lasso GLM.

This approach yielded an accuracy of 54.3% which is relatively good, but lower than our best random forest. Also, a full training of the model and prediction lasts approximately six hours.

'Home-Made' Ensemble

We still strongly believed that ensembling could improve our accuracy. Therefore, we tried a better tailored ensemble method for our data. The main difference is that we take the mean of the predicted probabilities (of belonging to a particular class) of all the methods in the ensemble; we believe this is a better way than averaging over the raw predictions.

The models are run into a loop. In order to uncorrelate them and introduce randomness, we train them on varying size random subsamples. Also, at each iteration, the models' parameters are randomly set, within a reasonable range of values according to multiple trials.

The submodels in question are H2O's implementation of random forest, gradient boosting machine, and deep learning.. These were chosen because of the parallelized computing option and the tuning possibilities. We also included extremely randomized trees, xgboost, and the ensemble rule C5 algorithm. They were run in loops of up to 40 iterations (so 240 models in total), which takes a reasonable 2 hours running time, and it is possible to observe the up-to-date accuracy after each new model.

We saw our performance improve when we removed classes 4 and 5 from the training sample. At best, this procedure gave us an accuracy of 55.7% on the public leaderboard, 53.32% on the public leaderboard (we indeed selected our second best), and 54% on cross-validation.

Conclusion and possible improvements

So far we saw that the best approach was the probability blending ensemble approach. Of course it is much less optimized than, for instance, each submodel, but, to its credit, it has no built-in alternative for multi-class classifcation.

Around the end of the competition we tried Poisson distribution based models. We thought it would be benificial to take into account the order of the target variable. Such models did not improve our performance, even in the ensemble.

We also could probably have tried an implementation of SVM using randomized parameter search.

In our opinion, the way to increase the performance is in fact by constructing new features. Out attemps in that regard didn't lead to much improvement, but we are still thinking about it!

Appendix - accuracy on the leaderboard