15D012 Advanced Computational Methods

Denitsa Panova, Felix Gutmann, Thomas Vicent

17.03.2016

Introduction

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Models and Predictions

Table 2 gives a broad overview of the applied methods with the corresponding results on the leaderboard and on cross validation (if available). In order to be concise we will only discuss a few of them in more detail. In the following we give some

Some basic results and ”off the shelf” results

Given the size of the data some of the first approaches was to apply some penalized GLM’s using Ridge, Lasso and Elastic Net regularization. We hopped to benefit from the easy set and performance. However isolated it didn't perform that well.[1]

SVM's from the e1071 package seem to be flexible in case of tuning. It provides several kernel types and classification methods (c and nu) and for class imbalances. A grid search was perforemd to tune both C (cost for missclassifitaion) and γ (margin). However, we stopped it after 10 hours without a result using only two parameters for each argument. A submission with radial kernel and basic config obtained a score of 0.506 on the leaderboard. Mainly due to the lag of speed, but also because of tuning expenses and off the shelf performance SVM wasn't considered anymore for the following process.

Random Forest and boosting

The best "off the shelf" turned out to be random forest and boosting. Especially random forest turned out to perform extremly well. Due to the speed fo the model it is very easy to tune. Table 2 showes some of the results for random forest in different settings. It turned out that the second overall second best performance was based on a random forest. This was achieved quite early in the working process. Therefore we consider the random forest as the benchmark model.

Manuall ensemble approach.

In the following we briefly describe our so far most promising approach in more detail. Instead of running isolated models, we used different models and average results.

The first try in that direction uses the superLearner package of R, which allows to run several model types combined. Furthermore, it provides internal cross validation and paralized implementation for some models. However, it doesn't supports multiple classes. Therefore, we transformed popularity to a binary matrix and trained a model using gbm random forest and a lasso glm for each binary class. Despite of the rather good perfomance on the leaderboard (0.543), it couldn't beat the random forest specifiaction so far. Despite parallelising there is still a speed issue. A full training of the model and prediction lasts approximatly six hours.

We wanted to exploit this more diverse prediction power in a more flexible model and time setting (FIX ME). We use the H_20 package, which provides an optimized performance of several different models. Furthermore, the supported methods are also parallized, which brings again computational gains. Concerning the data we droped the minority classes 4 and 5 to let the models focus more on the core classes. Furthermore we droped observations, which in out opinion where questionable (as discussed in the introduction). In this setting we applied the following models for out ensembling:

- random forests
- gbm
- (deep-) neural networks
- extremely randomized trees
- xgboosts
- rule-based models

Using such a quite extensive number of models a grid search of optimal parameters didn't seem feasible in this framework. Therefore, we re-run all models several times and randomly choose

[1]Note that we used it again in one ensemble approach, see later on

parameters out of a reasonable parameter range [2]. To monitor our perfomance randomly subset roughly 66% of the training data in each iteration as a input set the other part as a test set to validate our models. This procedure turned out to be a solid working basis, because one good result could produced within less than an hour. Instead of direct predictions we use the probabilities for each classes, that the models predict. Since we average over the particular predictions of each model, we believe this provide a more granular outcome. This procedure gave us the best result on the public leaderboard with an accuracy of 0.557.

Optional

In the late process we experimented with some more "unorthodox" approaches. This involved tranforming the data using unsupervised techniques (principle components, clustering etc.) and perform prediction on that data set. The idea was try to condence and rearrange informations in a different setting. However using a random forest gave an (5-fold) cv-average accuracy of only around 0.51.

Experience with stacking

Conclusion and possible improvements

So far we saw that the best approach so far was the manual ensemble approach. Despite the fact it performed quite good, it also seem to have an "upper bound". So far in our opinion the only thing that can be done to increase the prediction performance is construct new features, which add more predictive power.

[2] Note that, while using the superLearner we set up and configured an instance on AWS to outsource some computational work and tuning efforts. However, it turned out that the cost benefit ratio here was not very appropiate due to stability of the instance and monetary conditions.

Table 1: Overview of Methods and results

Method	Specification	CV	Leaderboard	Package
Penalized multonomial GLM	-	-		glmnet
... Lasso	-	-	0.506	
... Elastic Net	-	-		
... Ridge	-	-		
Support Vector Machines	Kernel: Radial		0.496	e1071
Random Forest				randomForest
... Off the shelf	-	-		randomForest
... Balanced DS	-	-	0.506	randomForest, Smooth
Boosting				randomForest
Super Learner	RandomForest, Boosting, LassoGLM,		0.543	SuperLearner
Manual Ensemble	RandomForest, LassoGLM,		0.543	$H_2 0$
Unsupervised		0.516		Several

Table 2: Overview of Methods and results

Method	Specification	CV	Leaderboard	Package
Penalized multonomial GLM	-	-		glmnet
... Lasso	-	-	0.506	
... Elastic Net	-	-		
... Ridge	-	-		
Support Vector Machines	Kernel: Radial		0.496	e1071
Random Forest				randomForest
... Off the shelf	-	-		randomForest
... Balanced DS	-	-	0.506	randomForest, Sm
Boosting				randomForest
Super Learner	RandomForest, Boosting, LassoGLM,		0.543	SuperLearner
Manual Ensemble	RandomForest, LassoGLM,		0.543	H_20
Unsupervised		0.516		Several