Carl Piening, Nathan Decurnex December 2020

Machine Learning project

Introduction

After a quick glance at the data set, we can see that we are faced with a p >> n problem as we have about 4870 predictors for only 708 data points or feature as some papers call it. Due to the nature of the dataset, we expect low variance methods such as regularization to be most successful. This is mostly because with p >> n problems, one of the main concerns is overfitting.

First visualization attempts

Data visualization was a complicated task at first since we did not know where to start; which of the 5k variables to plot? We first explored basic methods to understand how to dataset behaves and, if possible, which variables were most important. We plotted the most important variables found in a random forest with boosting and we also plotted the feature that most correlated with the response. While we did not gain any useful insight into the dataset, we realized in these steps that a significant number of variables had zero variance, making them obviously useless for our prediction task. We also found out that many predictors were highly correlated with each other, and thus, from here on, we run some of our methods on the "uncorrelated" dataset (set in which we keep only one of two correlating variables). This greatly reduced the number of predictors which lowered variance error in this kind of data set because highly correlated variables contain similar information on the response. We thus avoided giving too much weight to features that have the same information.
Then, we decided to try to reduce the number of dimensions to extract a pattern. Of course, PCA is the first method that came to mind. Unfortunately, when plotting the data, no significant pattern can be seen. We decided to not even run a pattern detection algorithm like t-SNE, especially since it would not provide significant information for the task. However, one very interesting point was that the Cumulative Proportional Variance Explained graphs indicated that the first ~100 principal components explained 99% (approximation) of the variance.

Generalities about our approach to creating models

As mentioned above, we expected low variance method to work better, so we directly excluded some simple method like kNN, polynomial fit, classic neural network, and trees.
For each model we used the following general procedure. To begin with, we tried to make it work and tried how the model would perform with the different pre-processing steps (we mainly used three approaches: remove the zero variance features, remove the intra-features correlation and the low correlation of certain features to the response). We continued by performing cross validation (=CV) to get estimates of the test error and to optimize hyperparameters. We went for low number of splits to guaranty a significant change in training points, because of the low number of data points (low n). However, in some files you will see an alternative to proper cross validation where the choices of the splits were randomized but ran several times. This decreased running time by a lot but decreased reproducibility and precision on the validation error, but we could not afford to run certain models for 10+ hours.
Regarding multiple hyperparameters tuning, we sometimes tried an approach that we called recursive CV: perform CV for the first hyperparameter, then do a second CV for another parameter while fixing the first one to the best value found in the first CV. We continue by trying again the first CV but with the second parameter fixed to its optimum (found in the second CV) and do this until convergence of the two parameters. It solved the difficulty to CV one parameter while keeping others constant, because the tuning of one parameter could also depend on the choice of others. We choose the MSE as loss function because the test set is evaluated with it. When the best hyperparameters are found, we would train our final model for submission with the whole training set to increase performances.

Linear methods

We decided that our first real test would be to perform regularized linear regression as classic linear regression is already overfitting the data. We first did subset selection to optimize the regularization, but it was computationally too heavy, and we had to use forward selection. To CV the right number of variables to use, we had to reduce it before treatment by taking only about the 300 most important variables. It found around 30 variables. We then fit a lasso regularization. We realized that the three pre-processing steps (proposed in the section "generalities") were not significantly changing the results. By trying to CV the alpha parameter, we realized that the alternative CV method (randomized split ran multiple times) was not enough reproducible and did not help to reduce our validation error. To conclude, the lasso, which was our best linear method, set a baseline for our future tests. Afterall, these two methods maybe had too high bias error.

Non-linear methods

The first method tried was random forest. We believed it to be powerful because of its many trees reducing the variance. It is here where we explored massive recursive CV by running several times many cells. It turned out our estimate of the test error was very variable and made it difficult to choose the right hyperparameters. Pre-processing the data (cf. Section “generalities”) increased the performance by a little. To submit a model, we had to apply the same exact data treatment to the test set that we made for the pre-processing steps with the training set. At a point we thought that increasing the number of trees (= ntree) would reduce variance even more, but it ended up being a time waste: when you reach a certain value of ntree, the validation error converges and stays constant. This special value was found around 500. In fact, all the work in the file *RF2* ended up fruitless. However, it stayed our best model until we found our different approach described at the end.

Then we tried boosting as it is a better version of random forests. Again, recursive CV found good hyperparameters, but they turned out to have high variance in the validation error. It may be due to the alternative way to CV our parameters. We tried to run boosting on the most important features that were found earlier but it decreased performances. We did not explore further boosting as it seemed to be worse than random forest while very time consuming. We think it was overfitting the data. Also, we choose to test different parameters (validation split, #repeats in the alternative CV) to evaluate our error. This allowed us to nicely compare boosting model with each other but not with other methods.

The last non-linear approach was to train a highly regularized neural network (=NN). While we knew that a NN was very likely to be too flexible, but we nonetheless hoped that the power of these networks could find us nice features in the data which a human could not see. On top of that, part of the reason we took the machine learning class was to learn about NNs, we simply had to code one. With scaling techniques to decrease the error, we tried to search for the best hyperparameters. We read about a library called keras tuneR which automatizes the tunning, but we were unable to install it on anaconda. Thus, we ended up just testing various combinations, since unlike other methods, there is no direct way to CV and find optimal parameters due to their variety and configuration. So, we basically ran several times our network and kept track of the lowest validation error. The best network we found had low number of layer and neurons and was highly regularized with the kernel regularizer and dropout rate.

Innovative approach[1]

After having tried a lot of techniques with relatively low success, we figured the problem was really the data. And from what we explain above, especially the low correlation with response making prediction patterns non-existent, we had to try a completely new way to pre-process our data. Thanks to the incredible 3D graph of the two first principal component (= PC) versus the response and the good result of a simple PCR, we decided to try to use the principal components as feature representation of the data (change of basis: represent the data by these new coordinates). We thought it would provide the reduction of variance error needed while keeping a lot of information on the variables (= keeping low bias error). It would transform the problem to a n > p problem. A clever move was to perform also PCA on the whole data including the test set to capture more information on the predictive space. We also checked if they were no outlier in the data or big differences between test and training set.

Then we had to choose which model to fit to our new dataset. We wanted to try a GAM but selecting the node manually made us change our mind. So, we went for NN. They were no real motivation to go for NN rather than RF or boosting, but we thought a function more flexible than a simple linear one would perform better. Again, with NN and the choice of PC, it was hard to directly find the best parameters so we tested several combinations. Regarding these new predictors, 15 PCs seemed to be a good compromise between underfitting (we tested 2PCs) and overfitting (we tested 20 and 30 PCs). These numbers are not completely random, they come from the fact that they are at a remarkable level of cumulative proportion of variance explained. For instance, 15 PCs explains ~85% of the variance. We also believe that our method will stay high in the competition and we can be quite confident as our method is expected to have low variance. Being in bio-engineer major a cool thing is that a technique called PCR (also the diminutive for Polymerase Chain Reaction) performed well ☺. However, our method should rather be called PCNN.

Conclusion

Finally, this project was very difficult as nearly all the classical approaches we learned in the lesson failed. Overall, we felt like this project enabled us to understand the strengths and, mostly, weaknesses of the different methods. In the end, we managed to find an innovative way to handle the task and managed to score a good submission.

[1]path of the code where you can find this approach: ~/src/visualize data/Custom