Can PCA extract important information from not significant features? Neural Network case. final project report

Maciej Odziemczyk

Faculty of Economic Sciences, University of Warsaw

January 2021

Abstract

The main goal of this study was to check if Principal Component Analysis is a useful tool for boosting Neural Networks performance, especially when non significant features are transformed. To spot these features six methods was applied: Random Forest feature importance, Mutual information, Spearman rank correlation with target and between features, General to Specyfic procedure based on Logistic Regression and Loglikelihood Ratio test and "Lasso" Logistic Regression (L1 penalty). During experiments Random Forest feature importance score turned out to be the best and gave the best results when only one metric applied. Finally 3 types of algorithms was tested on selected dataset for bankruptcy prediction of the polish manufacturing sector companies: Random Forest, Extreme Gradient Boosting and Neural Networks with and without PCA preprocessing. It turned out that XGB outperforms all other algorithms in this task, followed by Random Forest, Neural Networks were the worst, but its performance was quite good anyway. Finally based on Wilcoxon median equality for small samples test, PCA was found to be helpful with boosting Neural Network performance for 2 layer model.

1 Introduction

The main goal of this project is to check usefulness of PCA for the Neural Network model. To do so, polish bankruptcy dataset was selected. Of course there was also a standard question: "Which algorithm is the best for this problem?", the considered ones were Random Forest, Extreme Gradient Boosting (*XGB*) and Neural Network. The next question that might be asked was about the best feature selection method. It was nice to work on real data for the real problem, because bankruptcy prediction is an extremely important ability for business. Many economic actors are ready to pay a lot of money to know about company bankruptcy prediction in advance. Machine learning methods are usually good enough for the task, which makes them useful for decision making.

2 Dataset

Selected dataset contains 64 financial indicators (continuous variables) and the binary target - 1 if the company went bankrupt in choosen horizon and 0 if not. The whole dataset is about polish manufacturing sector companies from 2007-2013 and is splitted into 5 tasks, dependent on the forecast horizon (avalible from 1 to 5), for this project one year horizon was taken. The dataset used in this study contains 5910 observations and is highly imbalanced, 410 of them has a bankrupt label, that is less than 7%. The data has a lot of missing values for many features, to handle this problem data imputation algorithm was implemented. Some of the features has too many missings for imputation and had to be dropped. Finally 59 features were left for further analysis. Another problem with this data was correlation between features, there were 37 highly correlated attributes (at least 0.8 absolute value Spearman correlation coefficient). To sum up, the data was not trivial, and the tree-based methods were the favorites because of its property for finding all necessary nonlinearities etc. The dataset was used in [3], where many machine learning methods were applied too.

3 Methods and models

For the data imputation purposes, Random Forest proximity-based algorithm was implemented by hand. It is an iterative algorithm which tries to find the best possible values for missings based on the simillarity with the other samples. Firstly the median is imputed, then Random Forest Classifier is trained and based on the samples nodes in Forest structure simillarity is created and normalized, after that weighted average for missings is computed and the whole process is repeated (excluding median imputation of course).

All the models were trained and tested in 5-fold Startified Cross Validation. For the feature selection purposes, six methods were applied:

- Random Forest impurity based feature importance score - the final score was the average of folds score,
- Mutual informiation for classification problem and continuous variables - it is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency,
- Spearman rank correlation with the target variable (in the binary problem it is not a good measure, but in some hard decission can be considered),
- General to Specyfic procedure based on the Logistic Regression and Loglikelihood Ratio test - automated procedure implemented by hand (first LR test, then GtS procedure), the econometric standard,
- "Lasso" Logistic Regression - logit penalized with L1 norm, thanks to its structure if α penalty parameter increases non significant features coeficients vanishes to the zero.
- Spearman rank correlation between features, to spot the high correlation groups for potential grouping.

Non significant features were grouped and transformed by PCA, this procedure was a part of a special Neural Network wrapper.

The models trained and optimized in this study were Random Forest, XGB and Neural Networks. Because of data imbalance, the main performance metric was folds mean area under precission-recall curve test set score and the support metric was the folds mean area under receiver operating characteristic curve test set score.

3.1 Random Forest

Firstly, model sensitivity to some hyperparameters was tested in CV, after that narrow range of these parameters values were used to find the best set in random search. Tested hyperparameters and its optimal values:

- maximum tree depth, optimal: 8,
- class weight, optimal: default one to one,
- number of trees in the forest, enough: 100,
- maximum number of features used in one tree, optimal: 53,
- minimum number of samples in node to split, optimal: 7,
- minimum number of samples in final node, optimal: 5.

3.2 Extreme Gradient Boosting

Because of the computation costs, hyperparameters sensitivity was checked on one split only, but random search was performed in CV, after that some grid search for dropout and dropout skip rate was performed in CV also. Tested hyperparameters and its optimal values:

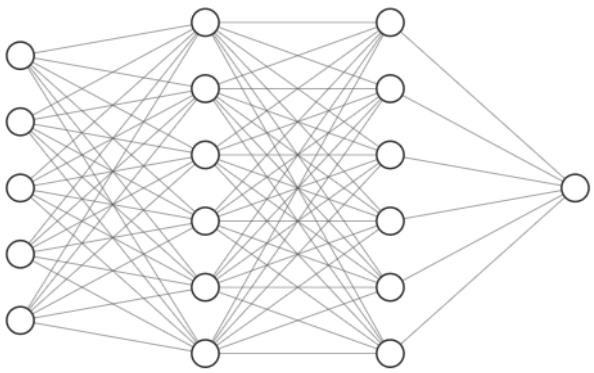
- maximum tree depth, optimal: 5,
- learning rate η, optimal: 0.08,
- subsample for a tree (by rows), optimal: 0.9,
- number of features for a tree, optimal: 0.8,
- number of features for a tree level, optimal: 1,
- regularization (L1 and L2), optimal: 0.1 L2,
- minimum loss reduction parameter γ, optimal: 4,
- dropout, optimal: 0.2,
- dropout skip probability, optimal: 0.6.

Note: tested boosters (tree-algorithms) were gbtree and dart, the second one differs in the possibility of using a dropout (last two hyperparams).

3.3 Neural Networks

Neural network optimization procedure was a bit different than previous algorithms. During NN building process the most important things are the results of course, but smoothness of the learning history is crucial also, we don't want our results to be an effect of luck. To find the best model, experiments on the single data split were performed. One experiment was a check out some hyperparameter values in the for loop and after that main metrics history were plotted. The best parameters gives the best results and the smoothest history. After that process the Cross Validation was performed. The architecture is showed below (number of hidden units were reduced on the image).

Figure 1: 2 layer Neural Network model structure

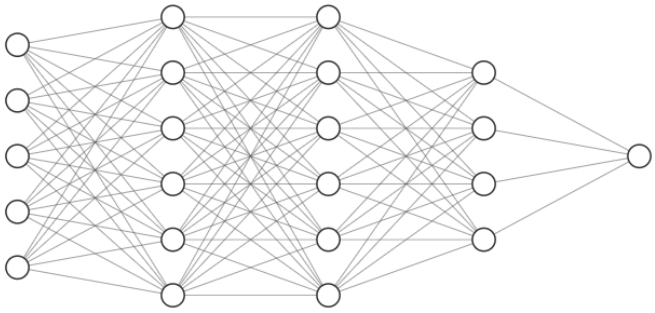


Source: Own study with the use of http://alexlenail.me application

2 layer Network details:

- number of hidden units: 60 in first and in the second hidden layer,
- no batch normalization,
- batch size: 350,
- activation functions: first hidden layer - hyperbolic tangent, second hidden layer - sigmoid,
- dropout: 0.4 in the first and the second layer,
- no regularization,
- optimizer: RMSprop with default settings,
- 400 epochs.

Figure 2: 3 layer Neural Networ model structure



Source: Own study with the use of http://alexlenail.me application

3 layer Network details:

- number of hidden units: 60 in first and in the second hidden layer, 40 in the third,
- no batch normalization,
- batch size: 350,
- activation functions: first and second hidden layers - hyperbolic tangent, third hidden layer - sigmoid,
- dropout: 0.4 in the first and the second layer,
- 0.001 L2 kernel regularization in the third layer,
- optimizer: RMSprop with default settings,
- 400 epochs.

3.4 Principal Component Analysis

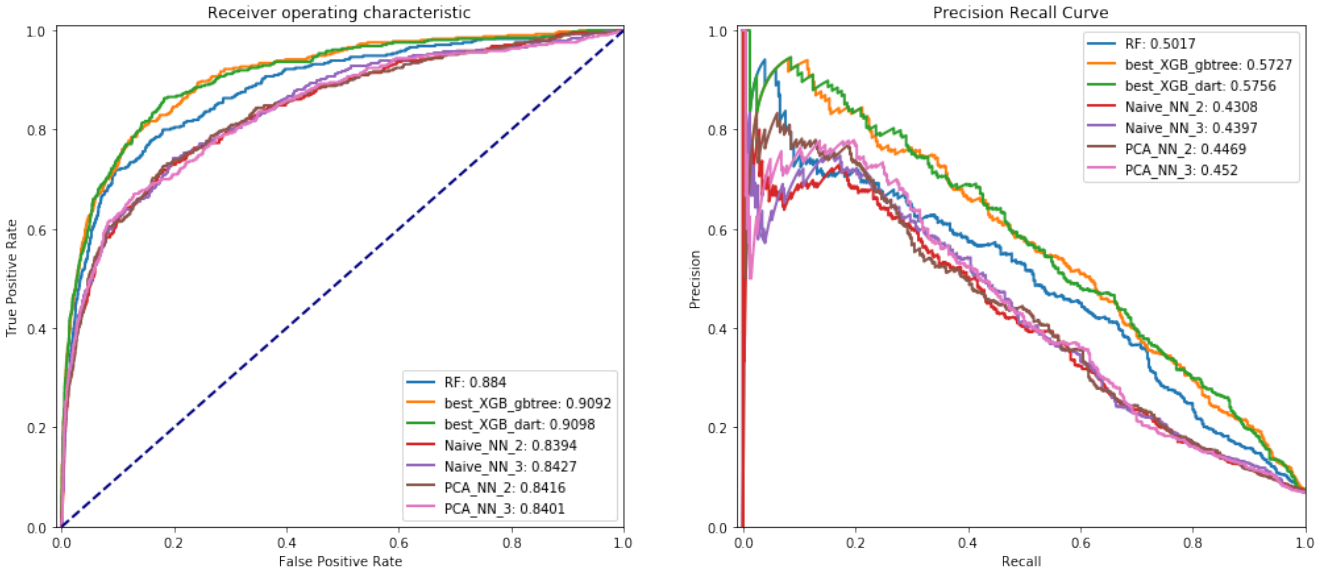
Firstly PCA was applied on the whole dataset to inspect it, 19 components explained 95% of the variance and 26 components explained 99% of the variance. After that, previously computed importance metrics were analyzed and non important features were selected and grouped to perform PCA on them, then some experiments were performed. The importance metrics analysis goes as follows (there were no hard decission rules): One big table with metrics was created, then it was sorted by RF importance score. Analysis starts from the worst by RF score criterion features. First feature were taken, then its correlation with another features were checked and after that, all of these features metrics were checked. The most important was RF score, the second was Mutual Information, nearly equal to the Lasso parameter, when feature vanished in importance, sometimes General to Specyfic procedure was checked and Spearman correlation with target. Between features correlation was a base for grouping. The experimets consisted single metrics decision rule checking, for example: all features with RF score < 0.007 were grouped into the one group. Number of components was the result of the experiments also, and the most powerful rule was based on 95% explained variance ratio. The PCA was tested in CV with Neural Network (excluding the

first PCA applied to the whole dataset). The best one was the PCA that was result of the human analysis, but RF score < 0.007 rule works pretty well too.

4 Results

The model performance comparison is summed up by the Figure 3. The models are: RF - optimal Random Forest, best_XGB_gbtree - optimal gbtree based XGB, best_XGB_dart - optimal dart based XGB (with dropout), Naive_NN_2 - 2 layer neural network, Naive_NN_3 - 3 layer neural network, PCA_NN_2 and PCA_NN_3 are the PCA combos versions of the standard networks (PCA parameters based on human analysis). AUC scores are reported in the charts legends. Note the AUC score is the mean of the CV folds scores - it is the best approach [2].

Figure 3: Models comparison



Source: Own study.

Overfitting results:

- RF 0.0843 at AUC-ROC, 0.3352 at AUC-PR,
- best_XGB_gbtree at AUC-ROC 0.0892, 0.4098 at AUC-PR,
- best_XGB_dart 0.0882 at AUC-ROC, 0.4034 at AUC-PR,
- Naive_NN_2 0.0447 at AUC-ROC, 0.1441 at AUC-PR,
- Naive_NN_3 0.0527 at AUC-ROC, 0.1725 at AUC-PR,
- PCA_NN_2 0.0414 at AUC-ROC, 0.1364 at AUC-PR,
- PCA_NN_3 0.0515 at AUC-ROC, 0.1457 at AUC-PR.

XGB outperforms all the other algorithms in all the metrics, next is Random Forest, especially in AUC-ROC. In AUC-PR there are thresholds for which NNs beat RF but overall even in AUC-PR Random forest outperforms NNs. On ROC, NNs results are close to each other, on PR bigger differences may be observed. PCA_NNs looks better than its naive versions, there is also a visible difference (in score) between 2 and 3 layers, but not so big. PCA transformation helps NNs work better on lower thresholds, it is easy to see that the brown and pink curves are much higher on PR for < 0.2 threshold. Wilcoxon test for samples median equality null hypotesis was rejected (p value 0.0216) only for Naive_NN_2 and PCA_NN_2 median of the PR results (tested: PR results ROC overfitting and PR overfitting between Naive_NN_2 and PCA_NN_2, Naive_NN_3 and PCA_NN_3) that means PCA boosted 2 layer network performance, in contrast to [1] where PCA causes worse results in random forest and logistic regression. It is nice to see that NNs built in this study has better preformance than NNs build in [3] by Zieba et al.

5 Conclusions

PCA can extract important information from not imporant features in Neural Network case, but it is not a magic trick. The results are slighlty better (wilcoxon test rejected null hypotesis of median equality for 2 layer Neural Network PR results and accepted the null hypotesis of median equality for 3 layer Neural Network PR results and AUC-ROC and AUC-PR overfitting median equality). However PCA should be considered during predictive modelling of high dimensional data, especially when there are correlated features.

Random Forest impurity based feature importance score gave the best results in terms of feature selection among considered ones: Mutual Information, Spearman correlation, General to Specyfic Logistic Regression LR test, L2 Logistic regression, but all of this methods are important and may be used successfully during analysis.

If overfitting is not a problem, tree-based methods outperforms Neural Networks in current task. Extreme Gradient Boosting with gbtree or dart booster performance was the best, following by Random Forest (the easiest to implement).

General Advice: If you don't have time, just use the Random Forests, tune it and it'll be fine.

References

[1] Mu-Yen Chen. Predicting corporate financial distress based on integration of decision tree classification and logistic regression. *Expert Systems With Applications*, 38:11261—11272, 2011.

[2] Martin Scholz George Forman. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12, 2010.

[3] Jakub M. Tomczak Maciej Zieba, Sebastian K. Tomczak. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. *Expert Systems With Applications*, 58:93–101, 2016.