KAGGLE COMPETITION: BNP PARIBAS CARDIF CLAIMS MANAGEMENT

Peidong Wang, Ke Tan

Department of Computer Science and Engineering
The Ohio State University, Columbus, OH 43210

ABSTRACT

This paper is a summarization of the machine learning techniques we applied in the Kaggle competition BNP Paribas Cardif Claims Management [1]. We propose three methods to extract features from the original dataset and four statistical models to analysis the features. Then we compare the performances of different combinations of the features and models and show our ranking in the competition.

***Index Terms*—** Extended Features, Auto-Encoder, Linear Regression Features, DNN, Bagged and Boosted Trees

1. INTRODUCTION

Kaggle competitions are famous for their real-world datasets and highly competitive participants. The competitions can test how well the machine learning techniques we know can do in real-world problems. There are so many theories and methods in the area of machine learning, but what really matter are the ones that can achieve good results in real-world problems.

The Kaggle competition we pick is the BNP Paribas Cardif Claims Management. The duration of this competition is from February 3rd to April 18th. This competition is supported by an insurance company called BNP Paribas Cardif. The company receives many claims from its customers everyday, and the claims need different levels of check. So that the company wants to build a system which can automatize the process of determining to which level a claim belongs.

An anonymized dataset with two categories of claims is provided by BNP Paribas Cardif. The first category contains claims for which approval could be accelerated, and the second one contains claims for which additional information is required (in other words, they cannot be accelerated). Each data sample is a 131-dimension vector. The output is the probability for a sample to be in the first category. The training set is consisted of 114322 samples, and the test set has 114394 samples. The performance is evaluated using Log Loss function.

The organization of the following paper is as follows. First we will show the three feature extraction methods we used in this competition. Then we will illustrate the four statistical models. After that, the experimental results are shown and analyzed. Finally, we will show our ranking in the competition and make a conclusion.

2. FEATURE EXTRACTION

2.1. Data Preprocessing

Since the dataset provided by BNP Paribas Cardif derives from the real world, some values of the sample vectors are missing. Due to the data loss, preprocessing is needed to restore the data before feature extraction is performed. In addition, some values of the data are characters or strings, which are extremely difficult to utilize directly.

To address these difficulties, we perform the preprocessing on the original dataset as follows:
1. We set the missing values to zero.
2. We "encode" the character values following the alphabetical order. For example, we replace the characters 'A', 'B', 'C' and 'D', with the numbers, 0, 1, 2 and 3, respectively.
3. We simply set the "string" values to zero.

Considering the data loss is possibly useful information, we create a new binary-valued dimension that we call *loss indicator* for each original dimension to indicate whether the corresponding original value is missing or not. In other words, for a particular dimension in a sample, we set its loss indicator to 1 if it is missing, and 0 otherwise. Hence, we end up with a set of 254-dimensional "new" samples.

2.2. Feature Generation

In our project, we explore the following three different types of features:
1. We simply use the 254-dimensional samples generated by the preprocessing module as the input of the models. We name them *extended features (EF)*.
2. We construct a deep auto-encoder (DAE) to derive a set of 512-dimensional features. The DAE has three hidden layers and 512 sigmoidal neurons for each hidden layer. We feed the DAE with EF as both input and target. Subsequently, we try to minimize the minimum mean squared error (MMSE) between the target and the output using back-propagation (BP) with mini-batches. Finally we derive the features by performing the well-trained DAE on EF and extracting the output vec-

tor of the hidden layer closest to the output layer. This vector can be considered as a representation of the input vector, i.e., EF vector, in higher dimensional space. We call these features *DAE features (DAEF)*.

3. We aim to create features by linear regression of small groups of EF features. Instead of randomly selecting the features to take, only the first features of each group are selected randomly, then we try each group with every new feature and assign the feature to the group offering the greatest improvement. We obtain a set of features that we call *linear regression features (LRF)*.

3. MODEL SELECTION

As far as we know, there are two kinds of models which have shown their ability of scaling to large datasets, Deep Neural Networks (DNNs) [2] and Bagged or Boosted Trees. So that the model selection part is mainly focused on these two kinds of models.

3.1. DNN

DNN becomes very popular after its successful uses in computer vision and speech recognition. So that the first model we choose is DNN. The details of the DNN model are as follows. It has three hidden layers, with 512 sigmoidal nodes per layer. The output layer is consisted of two nodes, using softmax and the cross entropy loss function.

3.2. Bagged or Boosted Trees

Bagging is short for Bootstrap Aggregation. It is mainly about sampling and generating data. Boosting, on the other hand, is more about making use of multiple models. These two methods are usually used in combination. There are two widely used tree methods in real-world problems, Random Forest and Gradient Boosted Trees. In addition to these two methods, a modified version of Random Forest, Extremely Randomized Trees, will also be discussed in the following subsections.

3.2.1. Random Forest

Random Forest [3] uses an ensemble of trees (weak learner) to build a strong learner. Each tree in the ensemble is trained by samples drawn with replacement from the training set. In addition, the Random Forest method selects a random subset of features at each candidate split in the tree learning process.

3.2.2. Extremely Random Trees (ExtraTreesClassifier)

Compared with Random Forest, Extremely Randomized Trees [4] goes one step further in the way splits are computed. For each candidate feature, instead of looking for the most discriminative splitting point (threshold), the splitting point is drawn at random.

3.2.3. eXtreme Gradient Boosting (XGBoost)

XGBoost [5] is an efficient and scalable implementation of Gradient Boosting [6]. During each iteration in the boosting process, the Gradient Tree Boosting method adds a new tree to fit the difference between the desired output and the output of current model. Denoting the model at time step m as $F_m(x)$, the desired output as y and the difference between the desired output and the output of the model as $h(x)$, we have the following equation.

$$h(x) = y - F_m(x) \tag{1}$$

Then we use a new tree to fit $h(x)$, and thus updating the model as follows.

$$F_{m+1}(x) = F_m(x) + h(x) \tag{2}$$

Note that the residual $y - F(x)$ can be viewed as the negative gradient of the squared error loss function $\frac{1}{2} \times (y - F(x))^2$ over $F(x)$. And this is where the name "gradient" comes from.

4. EXPERIMENTAL RESULTS

By performing feature extraction, we derive 254-dimensional EF, 512-dimensional DAEF, and 131-dimensional LRF. The expected output is the probability with which the sample belongs to the first category. The evaluation metric of the performance adopted by the competition Log Loss function given by

$$logloss = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \tag{3}$$

where N is the number of samples, y_i is the binary target, and p_i is the predicted probability that y_i equals 1.

We explore the following six different combinations of features and models:

(1) DNN + EF
(2) DNN + DAEF
(3) XGBoost + EF
(4) XGBoost + DAEF
(5) XGBoost + LRF
(6) ExtraTreesClassifier + LRF

Table 1 presents the performance of these model-feature combinations on the test set. As we can see, ExtraTreesClassifier+LRF performs the best. Comparing the result involving random forest model, we find that the log loss is much larger than the others. With the same features, XGBoost provides slight improvement in the prediction performance, relative to DNN. In addition, when LRF is used as input, a log loss gain

Table 1. Performance of Different Model-Feature Combinations

Model + Features	Logloss
DNN + EF	0.47470
DNN + DAEF	0.49768
XGBoost + EF	0.47320
XGBoost + DAEF	0.48231
XGBoost + LRF	0.48547
ExtraTreesClassifier + LRF	**0.45037**

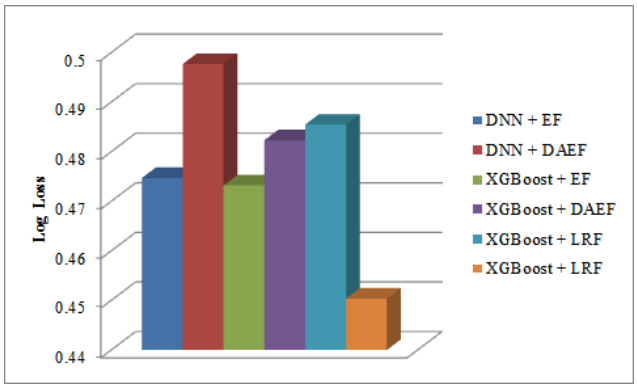

Fig. 1. Performance of Different Model-Feature Combinations

of 0.0351 is achieved by employing ExtraTreesClassifier instead of XGBoost. Fig. 1 illustrates a clearer comparison.

With ExtraTreesClassifier and LRF, we rank the 436th (top 15%) among 2947 teams on the Kaggle leaderboard.

5. CONCLUSION

In this paper, we explore six model-feature combinations involving four different models and three types of features. The experimental results indicate that a significant improvement in prediction performance can be achieved by using ExtraTreesClassifier as the model and LRF as the features. However, further research on both the models and the features is demanded since the prediction performance can be impacted by model configuration, such as parameter setting and selection of activation function.

6. REFERENCES

[1] "Kaggle competition: Bnp paribas cardif claims management," `https://www.kaggle.com/c/bnp-paribas-cardif-claims-management`, Accessed: 04-26-2016.

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[3] Andy Liaw and Matthew Wiener, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[4] Pierre Geurts, Damien Ernst, and Louis Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

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

[6] Jerome H Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.