JMLR: Workshop and Conference Proceedings 7: 35-43 KDD cup 2009

A Combination of Boosting and Bagging for KDD Cup 2009 - Fast Scoring on a Large Database

Jianjun Xie JXIE@IDANALYTICS.COM
Viktoria Rojkova VROJKOVA@IDANALYTICS.COM
Siddharth Pal SPAL@IDANALYTICS.COM
Stephen Coggeshall SCOGGESHALL@IDANALYTICS.COM
ID Analytics
15110 Avenue of Science
San Diego, CA, 92128, USA

Editor: Gideon Dror, Marc Boullé, Isabelle Guyon, Vincent Lemaire, David Vogel

Abstract

We present the ideas and methodologies that we used to address the KDD Cup 2009 challenge on rank-ordering the probability of churn, appetency and up-selling of wireless customers. We choose stochastic gradient boosting tree (TreeNet®) as our main classifier to handle this large unbalanced dataset. In order to further improve the robustness and accuracy of our results, we bag a series of boosted tree models together as our final submission. Through our exploration we conclude that the most critical factors to achieve our results are effective variable preprocessing and selection, proper imbalanced data handling as well as the combination of bagging and boosting techniques.

Keywords: KDD Cup, bagging, boosting, data mining, ensemble methods, imbalanced data

1. Introduction

The task of the KDD Cup 2009 is to build three Customer Relationship Management (CRM) models to predict three different wireless customer behaviors: 1) loss of interest toward current provider (churn), 2) propensity to purchase new products or services (appetency), and 3) tendency for upgrades or add-ons (up-selling).

Several aspects of data mining and statistical modeling have been addressed in this challenge:

- Handling a large dataset. The organizers provided 15000 variables, 14740 numerical and 260 categorical, to test the ability of participants in handling a large dataset. Although a downsized version with only 230 variables was made available in the second phase of the challenge, all the top teams also descrambled the variable mapping and used information from the large set.

- Rapidity of model building. Participating teams were required to complete all three models in 5 days in order to win the fast track, which has a higher priority than the slow track.

- Variable preprocessing and selection. Variables were populated by unnormalized values. Missing entries and outliers are significant. Some categorical variables have a huge number of distinct entries. Effective variable preprocessing and selection is a must for any modeling algorithm to achieve the best results.

All three tasks are binary classification problems. There are several well established modeling algorithms available and suitable: Logistic Regression, Neural Networks, Decision Trees, SVM etc. Nowadays, ensemble learning schemes are widely used to enhance the overall performance of a single classifier by combining predictions from multiple classifiers (Breiman, 1996; Dietterich, 2000). In the family of decision tree classifiers, Random Forest uses bagging to combine many decision trees by bootstrap sampling (Breiman, 2001). TreeNet® uses the stochastic gradient boosting (Friedman, 1999a,b) which constructs additive regression trees by sequentially fitting a base learner to current pseudo-residuals by least squares at each iteration. The pseudo-residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data point evaluated at the current step.

In this competition we use TreeNet® as our main classifier. The log-likelihood loss function has been chosen since all 3 tasks are binary classification problems. In order to further enhance the results, we combine bagging and boosting together. We bag a total of 5 boosted tree models for each task and take the average of all scores as the final prediction.

The results of each model are evaluated by area under receiver operating characteristic (ROC) curve, so called AUC. The AUC measurement does not require the models to produce the true probability of the predicted class as long as the model score can rank order the positive class and negative class effectively. This gives us more freedom of using sampling techniques to tackle the imbalance issue without worrying about converting the score back to a real probability.

We organize this paper as follows. Data analysis and preprocessing on training and testing datasets are described in Section 2. After establishing a gradient boosting tree as the main classifier, we proceed with variable selection and sampling the imbalanced data. These steps together with final bagging of different boosting decision tree models are described in Section 3. In Section 4 we explain our exploration on the small dataset. Finally, we summarize our final submissions and how they are compared with others in Section 5.

2. Data Analysis and Preprocessing

Data analysis is an important step of any data mining and modeling task. It helps for deep understanding the modeling task and selecting the proper modeling technique. One illustrative example is KDD Cup 2008. The Patient ID was found to be a predictive variable: essentially a target leakage which was introduced into the training data by mistake. It was overlooked by everybody except for the winner (Perlich et al., 2008).

2.1 Histogram Analysis

The frequency distributions of all 15000 variables of the training and testing datasets are analyzed to establish the "equality" between training and testing samples. The binning for

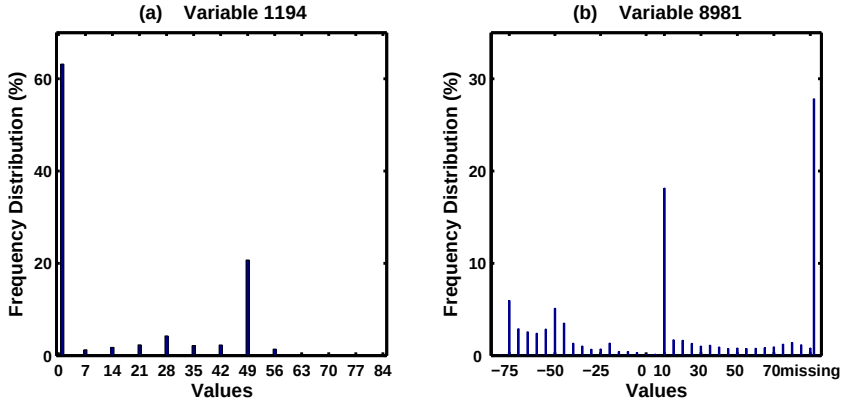

Figure 1: Example of histogram analysis: variable 1194 and variable 8981.

Table 1: Histogram analysis on labels of churn, appetency and up-selling.

Churn	**Appetency**	**Up-selling**	**Frequency**	**Percentage**
-1	-1	-1	41756	83.51%
-1	-1	1	3682	7.36%
-1	1	-1	890	1.78%
1	-1	-1	3672	7.34%

the histogram is performed around every discrete entry. Based on the histogram results we conclude that there is no substantial sampling bias between the training and testing data sets. In the slow track, we utilize histogram results of the large and small datasets to unscramble the small dataset. More details of the unscrambling process will be described in Section 4.1.

We find that most of the numerical variables have skewed distributions as shown in Figure 1. By checking the histograms of the variables, we discover that many of the numerical variables are populated by values that have a common factor, for example in Figure 1(a) the data values are all multiples of 7. This could be an indicator that these variables were artificially encoded.

Another observation is that many variables only have a few discrete values. For example, about 50% of all numerical variables have 1 or 2 discrete values as shown in Figure 2(a). Nearly 80% of all categorical variables have fewer than 10 categories, as shown in Figure 2(b). It can also be seen in Figure 2 that 12% of numerical variables and 28% of categorical variables are constant (only have a single entry). Furthermore, numerical values are heavily populated by 0s. We find that 80% of the numerical variables have more than 98% of their population filled by 0. These results suggest that a large number of variables can be removed since they are constant or close to constant.

Label frequencies for all three tasks are presented in Table 1. We can see that all of them are highly unbalanced. Appetency has extremely low positive rate. The imbalance of the class distribution has to be taken into account in the model building step. There is no overlap between any pair of labels, they are exclusive to each other. This motivated us to

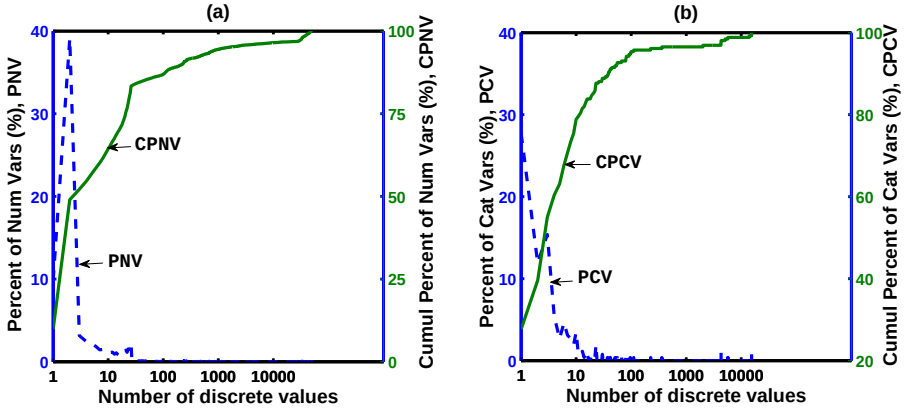

Figure 2: Distribution of discrete value frequencies in (a) numerical variable and (b) categorical variable.

incorporate the other 2 scores for a given model to improve the performance. However, we did not see significant improvement as described in Section 4.3.

2.2 Discretization and Grouping (Binning)

Even though most of numerical variables are populated by a limited number of discrete values, the population on each value differs significantly. We discretize 10 selected numerical variables that have strong correlation with the target. We consolidate the extremely low populated entries (having fewer than 200 examples) with their neighbors to smooth out the outliers. Similarly, we group some categorical variables which have a large number of entries (> 1000 distinct values) into 100 categories. This procedure of univariate groupings is frequently referred to as binning. Every category is replaced by a numerical risk factor (mean positive rate).

2.3 Missing Value Handling

A significant amount of variables are poorly populated, that is, for some variables many of the input values are missing. There are known techniques to approach the missing value problem which include mean substitution, multiple regression, maximum likelihood estimation, multiple imputation etc (Little and Rubin, 1987). In this work, we perform a simple substitution. We either replace them by a risk factor (for binned numerical variables and categorical variables) or treat them as a standalone entry ("missing" is simply another category of the input variable).

3. Modeling on Fast Track

3.1 Variable Selection

First, we removed 1531 constant variables and 5874 quasi-constant variables (where a single value occupies more than 99.98% population) based on our data analysis step. This left us

a dataset with 7595 variables which is still a quite large number. We then went on with a multi-round wrapper approach. We first split the reduced training set into 3 chunks for each label and built 3 preliminary models for each task. The parameters used in TreeNet model were set as the following: learning rate = 0.02, number of nodes = 6, number of trees = 600. At every step TreeNet uses exhaustive search by trying all 7595 variables and split points to achieve the maximum reduction of impurity. Therefore, the tree construction process itself can be considered as a type of variable selection and the impurity reduction due to a split on a specific variable could indicate the relative importance of the variable in the tree model.

For a single decision tree a measure of variable importance can be calculated by (Breiman et al., 1984)

$$VI\left(x_i, T\right) = \sum_{t \in T} \triangle I\left(x_i, T\right),$$

where $\triangle I\left(x_i, T\right) = I\left(t\right) - p_L I\left(t_L\right) - p_R I\left(t_R\right)$ is the decrease in impurity to the actual or potential split on variable x_i at a node t of optimally pruned tree T. p_L and p_R are proportion of cases sent to the left or right by x_i.

The variables entered into the model based on their contribution to the impurity reduction at each split. We removed all variables with relative importance less than 2.0% for each model. We then merged all variables selected by these preliminary models for each task together as a pool. We got a total of 1720 variables for the next round selection. The final variable set was obtained by using 75% of all training data. The rest was reserved as testing. We kept removing variables that have the least importance until the model performance on the 25% test dataset started dropping. We narrowed down our final variable set to less than 300 for all 3 labels.

3.2 Down Sample Negative Population

As listed in Table 1, the distribution of labels for the 3 models are highly unbalanced. There are several popular ways to handle such an unbalanced dataset, for example, cost sensitive learning (Domingos, 1999), and a variety of sampling techniques (Van Hulse et al., 2007). Here we take the approach of down sampling the negative population. Table 2 lists the down sampling rate we used in our model. Since the model performance is measured by AUC which is the rank order of each record, the absolute value (scaling) of the score will not affect the results. Therefore, we directly used the raw score without any sampling correction.

3.3 Build the Best Boosting Tree Model via Cross Validation

We started building the best boosting tree models by adjusting the training parameters after variable selection. The factors we considered include the down sampling rate on the negative population, learning rate, number of trees, and minimum number of nodes. The model performance was evaluated by 5-fold cross validation. We also used the feedback on 10% test data as a reference. Not always was the 10% test feedback in agreement with the cross validation results. Upon checking the AUC results for each fold validation, we discovered large variations. Table 3 lists the AUC results for each fold and total of 5 folds

Table 2: Down-sampling rate for each modeling task.

Label	**Sampling rate of negative records**	**Positive rate after sampling**
Churn	70%	10.17%
Appetency	20%	8.31%
Up-selling	90%	8.12%

Table 3: AUC results for Upsell model in 5-fold cross validation.

Fold Number	1	2	3	4	5	Total
AUC	0.8956	0.9112	0.9037	0.9182	0.9085	0.9071

for Upselling model. We can see that the variation among each fold can be as large as 0.02 which is big enough to drop your ranking by more than 20 in this very close competition. This makes us believe that the 10% feedback is not reliable to judge the performance of a model. We did not change our strategy even though we saw other team's results were better than ours based on the 10% test feedback.

3.4 Bagging the Selected Boosting Tree Models

After we determined the final variable set, learning rate, and other tree parameters we were convinced that the best strategy for improvement of model performance was consistent bagging. This is an essential part of our solution, since we take a down sampling approach on the negative class to improve the label balance, so as a result some of the records are never seen in the training set. By creating the training dataset 5 times using different random seeds, we built a total of 5 boosted tree models for each label. Our final model is a simple average of all these boosted tree models. We noticed after the competition that an ensemble of TreeNet classifiers won the 2003 Duke/NCR Teradata Churn model contest (Cardell et al., 2003). Ensembles of multiple TreeNet models usually outperform a single model.

4. Modeling on the Slow Track

We processed the small dataset in a similar way as we did for the large. We quickly realized that unscrambling the small dataset, mapping and combining it with the large might be the most beneficial strategy in the slow track. First, the rule of this year's KDD Cup competition requires the results on small dataset to compete with the large. Next, after experimenting in this direction we discovered that results on small dataset did not outperform the results on the large. Finally, a combination of small and large datasets gives us an additional 10% testing feedback on the same model.

4.1 Unscramble the Small Dataset

It turns out that unscrambling the small dataset is quite straightforward. There are two unscrambling steps: first is to unscramble the variable mapping, second is to unscramble the

example order. Step 1 can be done by comparing the frequency distribution of each variable in the small and large datasets. This simple comparison maps 194 out of 230 variables in the small dataset to the large dataset. It is found that most of the numerical variables of the small dataset are simply scrambled as a constant fraction of the corresponding numerical variable in the large. The remaining variables typically have a one-to-many or many-to-many correspondence, which can be solved after Step 2.

In Step 2, we selected some mapped variables from the small and large and place them together as a key in the format of var_i, var_j, var_k ... var_n. We then cut them out from the small file in their original order. To ensure the uniqueness of the key enough variables have to be pulled. We create a file that has 2 fields, a sequence ID_{small} and a key. Then we created the same file using the mapped variables from the large data in the format of the sequence ID_{large} and key (consists of the corresponding variables from large dataset). The value is unscrambled so that the key will be same for both the large and small. Sort both files by key and then paste them together, and you get a map for the sequence IDs between the small and large data.

With this mapping table we converted one set of score files of the large dataset into the small data order and submitted for evaluation. Results on the 10% test feedback confirms that 1) there is a large variation of the AUC on the 10% test sample (we saw the same file geting 0.88 AUC on one 10% test sample and 0.79 AUC on another 10% test sample), and 2) the small dataset does not have all the information needed to beat the large dataset. Therefore we stopped building models on the small dataset and focused on the large dataset only.

4.2 Combine Small and Large

We compared the variables selected from the small dataset and the large dataset, then added back several variables not present in the model on the large dataset and rebuilt the models on the large data. To improve the robustness of the model we binned all the top 10 numerical variables (if they were not discretized in the previous steps) for each task and then added them back to the variable list. We made 5 iterations of bagging on the final model. The final results are obtained by averaging the final score from the slow track and the score from the fast track.

4.3 Test of Using Scores from Other Models

As discussed in Section 2, the exclusive nature of the 3 tasks motivated us to incorporate scores from the other 2 tasks as variables in the 3rd task. We explored this strategy in the appetency model which has the most imbalanced distribution of target class. We did find a little improvement over the model without using other scores. However, this nested structure complicates the modeling process: update of the other models then requires the rebuilding of the underlying model.

5. Results and Discussions

Table 4 lists our final submissions for both the fast track and the slow track. The winner's results are also listed for comparison. Looking back to our submission history, we find

Table 4: AUC results of our final models on KDD Cup test dataset.Winner's final results are also listed for comparison.

Dataset	**Churn**		**Appetency**		**Up-selling**		**Scores**	
	10%	100%	10%	100%	10%	100%	10%	100%
Large (fast)	0.7333	0.7565	0.8705	0.8724	0.9308	0.9025	0.8354	0.8448
Large (slow)	0.7390	0.7614	0.8714	0.8761	0.9023	0.9061	0.8376	0.8479
Small (slow)	0.7612	0.7611	0.8544	0.8765	0.9155	0.9057	0.8437	0.8478
Winner's fast	-	0.7611	-	0.8830	-	0.9038	-	0.8493
Winner's slow	-	0.7651	-	0.8819	-	0.9092	-	0.8520

that we did submit our best model as final, which confirms the correctness of our model improvement process. The difference between the 10% test and the 100% test is significant, which is in line with what we found in our cross validation process. The results on the small dataset are results of the model built on the large dataset bagged with one set of scores obtained from the real small data model. It actually decreases the overall performance from 0.8479 to 0.8478. In fact, all the top teams in the slow track descrambled the small dataset. In our view, the large dataset has more predictive information than the small.

We tried two other modeling techniques, logistic regression and SVM (Chang and Lin, 2001). Both of them require converting all categorical variables to proper numerical values. Logistic regression gives slightly worse results than boosted decision tress on same dataset we prepared. SVM can achieve similar accuracy but with much slower computing speed.

Comparing our results with the others in the contest, our overall performance ranked second in the fast track and third in the slow track. Our weakest model comparing with the winner was Appetency which has the most imbalanced class label. This might be an effect of our aggressive down-sampling and not enough number of bagging iterations.

In conclusion, we find through our practice that effective variable preprocessing and selection, proper imbalanced data handling and the combination of bagging and boosting are the important factors for achieving our results on the KDD Cup 2009 challenge.

Acknowledgments

We would like to thank the KDD Cup organizers for setting up this year's contest with a challenging problem, a nice website that participants can see online feedbacks and exchange ideas. Author J. Xie would like to thank his wife Rui Zhang for the support and encouragement she provided during her maternity leave for taking care of their baby girl Joann.

References

L. Breiman, Jerome H. Friedman, R.A. Olson, and C.J. Stone. *Classification and Regression Trees*. Wadsworth Int'l Group, Belmont, Calif., 1984.

Leo Breiman. Bagging predictors. In *Machine Learning*, volume 24, pages 123–140, 1996.

Leo Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

N. Scott Cardell, Mikhail Golovnya, and Dan Steinberg. Churn modeling for mobile telecommunications, 2003. URL `http://www.salford-systems.com/doc/churnwinF08.pdf`.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Thomas G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 1999a.

Jerome H. Friedman. Stochastic gradient boosting, 1999b. URL `http://www-stat.stanford.edu/~jhf/ftp/stobst.ps`.

R. J. A. Little and D. B. Rubin. *Statistical analysis with missing data*. John Wiley & Sons, New York, 1987.

Claudia Perlich, Prem Melville, Yan Liu, Grzegorz Świrszcz, Richard Lawrence, and Saharon Rosset. Breast cancer identification: Kdd cup winner's report. *SIGKDD Explor. Newsl.*, 10(2):39–42, 2008. ISSN 1931-0145.

Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 935–942, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3.