Ensemble Learning for Imbalanced E-commerce Transaction Anomaly Classification

Haiqin Yang and Irwin King

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
{hqyang, king}@cse.cuhk.edu.hk

Abstract. This paper presents the main results of our on-going work, one month before the deadline, on the 2009 UC San Diego data mining contest. The tasks of the contest are to rank the samples in two e-commerce transaction anomaly datasets according to the probability each sample has a positive label. The performance is evaluated by the lift at 20% on the probability of the two datasets. A main difficulty for the tasks is that the data is highly imbalanced, only about 2% of data are labeled as positive, for both tasks. We first preprocess the data on the categorical features and normalize all the features. Here, we present our initial results on several popular classifiers, including Support Vector Machines, Neural Networks, AdaBoosts, and Logistic Regression. The objective is to get benchmark results of these classifiers without much modification, so it will help us to select a classifier for future tuning. Further, based on these results, we observe that the area under the ROC curve (AUC) is a good indicator to improve the lift score, we then propose an ensemble method to combine the above classifiers aiming at optimizing the AUC score and obtain significant better results. We also discuss with some treatment on the imbalance data in the experiment.

1 Introduction

The 2009 UC San Diego data mining contest is a yearly competition for undergraduate students, graduate students, and postdoctoral researchers in colleges since 2004. The goal of this year' contest is to design computational methods to rank the example in the two datasets, where data are from anomalous web transactions, according to the probability each example has a positive label. The following is a description of the data and we summarize them in Table 1.

- The contest consists of two tasks, one is named "easy" and the other is named "hard". They are two datasets involving 19 features from web transaction anomaly data, where two features are the state and email information of the transaction and the other 17 features can be deemed as continuous features. The features corresponding to state and email information are categorical features.

C.S. Leung, M. Lee, and J.H. Chan (Eds.): ICONIP 2009, Part I, LNCS 5863, pp. 866–874, 2009.

Table 1. Data Description

Task	# Feature	Train			Test
		Total	Positive	Negative	
Easy	19	94,682	2,094	92,588	36,019
Hard	19	100,000	2,654	97,346	50,000

- For the task 1 (the "easy" task), the training data consist of 94,682 examples and the test set consists of 36,019 examples. The test set is drawn from the same distribution as the training set.
- For the task 2 (the "hard" task), the training data consist of 100,000 examples and the test set consists of 50,000 examples, where the test set is drawn from the same distribution as the training set.

There are difficulties encountered from the distribution of the data as well as the evaluation criterion:

- The class distribution in the datasets is highly imbalanced. There are roughly fifty times as many negative examples as positive. In this case, standard classifiers tend to have a bias in favor of the larger classes and ignore the smaller ones.
- The evaluation criterion is lift at 20% on the probability each example has a positive label of the datasets. That is, it takes the first 20% of the sorted list of predicted values with the biggest values, and makes a list of the original indices of this top 20%. The result counts the number of true positives in the list. That means a perfect classifier can get the best score as 5. The evaluation is different to the objective in standard classifiers, which aim at optimizing the error rate.

Based on the data characteristics and specific evaluation criterion, we first preprocess the data on the categorical features and normalize them. Here, we present the results of our first stage testing. Hence, our objective is to test on several popular classifiers, including Support Vector Machines (SVMs), Neural Networks (NNs), AdaBoosts, and Logistic Regression, to get their benchmark results, without much modification. These basic results can be used to choose a better classifier for further tuning. Further, after observing that the area under the ROC curve (AUC) [3,5] is a good indicator to improve the test performance, we then ensemble the above four classifiers to get a powerful classifier by optimizing the AUC score. Significant better results are obtained on both tasks.

The rest of this paper is organized as followings: In Section 2, we illustrate the test procedure and describe the methodologies adopted. In Section 3, we detail the procedure of parameter seeking on the models, the current results, observations, and our other testing. Finally, we conclude the paper in Section 4.

2 Flow and Methodologies

The test consists of three main processing steps: 1) data preprocessing, including categorical features preprocessing and data normalization; 2) classifiers building with parameters tuning and models ensemble; 3) output of test results: the probability of each sample being assigned to positive label. In the following subsections, we will describe the above procedure in details.

2.1 Data Preprocessing

The data consist of 17 continuous features and two categorical features containing the state and the email information. For the state feature, there are 54 states in the transaction datasets. Most transactions are recorded the state as "CA" and some states, e.g., "AE", "AP", etc. only appear in several transactions. Hence, we categorize the state feature based on the number of the transaction happened on the state. Concretely, we first set the state as a specific category when the number of the transaction on that state is in one digit order. Next, we set the state into a new category based on the number of transactions in the order of each 100, each 1,000, and 10,000. After that, we obtain 20 and 17 categories to represent all 54 states in the state feature for the "easy" task and the "hard" task, respectively. We then expand the state feature into a 20-dimensional and 17-dimensional features with 1 indicating the corresponding categorized state and 0 when the state does not appear in that transaction and that category.

In the email feature, some domains, e.g., 'AOL.COM', 'COMCAST.NET', etc., appear frequently. Other domains, e.g., '.MIL', etc., seldom appear in the transactions. Similarly, based on the frequency of domains appearing in the transactions, we categorize the email domains into 19 types and expand the email feature into 19-dimensional features.

After concatenating the continuous features with expanded state features and expanded email features, we obtain the corresponding training data and test data. We further use a standard method to normalize them: making the sum square for each feature in the training data to 1, and normalize the test data according to the weight in the training data. Due to the number of features is relative small comparing to the number of training data, we do not perform feature selection on both tasks further.

2.2 Classifiers

In the test, we first explore several popular classifiers, including Support Vector Machines (SVMs) [21], Adaboost [19], Neural Networks [2], and Logistic Regression [8], to get their benchmark results. These results are used to select a better classifier for further tuning.

Support Vector Machines. Highly imbalance of the data is a major difficulty in the contest. To solve the imbalance problem, in SVM, we seek a decision

boundary, $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b$, where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, by adding different weights on the cost of different label of data as follows:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C^+ \sum_{i:y_i=1} \xi_i + C^- \sum_{i:y_i=-1} \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, N. \end{aligned} \tag{1}$$

where training data are $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, with N instance-label pairs. C^+ and C^- are weights of training errors with respect to the positive and negative samples, respectively. In the test, we set C^+ to be 50 times larger than C^-. Since the task is in large scale, we just seek linear classifier for SVMs. The output scores are then re-scaled by $1/(1 + \exp(-f(\mathbf{x})))$.

Here, we adopt a very basic routine on the SVMs to get benchmark results. Other methods, e.g., support vector method for the AUC score [12] may be adopted to get better performance; probability output of svms [17] may be adopted to fit the evaluation metric of the task.

Neural Networks. Neural networks (NNs), also called artificial neural networks, are computational models that can capture the non-linear property or structural relation embedded in the data [2]. Their powerful computation ability motivates us to test the performance on the tasks. Here, we adopt a radial basis function (RBF) network, which uses radial basis functions as activation functions and combines these radial basis functions in a linear form. Here, we use an implementation of a RBF network in [16]. The drawbacks of the RBF network are that there are some parameters need to be tuned and the model is easy to seek a local optimal solution.

Adaboost. Boosting is a very efficient and effective method to find a classification rule by combining many "weak" learners, in particular when each of which is only moderately accurate. In the test, we also tried tried the AdaBoost [19].

The main idea of AdaBoost is to construct a highly accurate classifier by combining many weak learners. The weak learners are only moderately accurate but should be diverse. Currently, there are some extensions or generalizations from the basic AdaBoost algorithm first introduced by Fruend and Schapire [6]. These extensions include the Real AdaBoost [19], the Modest Adaboost [22], and etc.

Here, we choose Real AdaBoost [19] implemented by [20], which supports real-value prediction and obtains better performance. The weak learner we used is classification and regression tree (CART). This is because CART is inherently suited for imbalanced dataset since its tree is constructed according to the correct classified ratio of positive and negative examples and the model selection procedure can be done simply and efficiently by iteratively increasing the number of weak learners and stopping when the generalization ability on the validation set does not improve. In the test, we change the number of splits in the CART and the number of iterations for the Real Adaboost to get a better benchmark result.

Logistic Regression. Logistic regression (LR) [8] is a standard tool to predict the probability of occurrence of an event by fitting data to a logistic curve. It can output the probability directly, which exactly fit the evaluation criterion of the contest. The output of logistic regression is a probability in the following form:

$$P_{LR} = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}}, \tag{2}$$

where the parameters $\mathbf{w}$ and b are estimated by maximum likelihood. The advantage of the logistic regression is that there is no parameter to be tuned and the model can achieve relative better results.

2.3 Models Ensemble

Literature states that combining divergent but fairly high performance models into an ensemble can usually lead to a better generalization performance [7,13,14]. Other than combining divergency, ensemble method may also play the role of voting to help the generalization performance [18]. Here, we combine the output of the above four base classifiers in a linear form as follows:

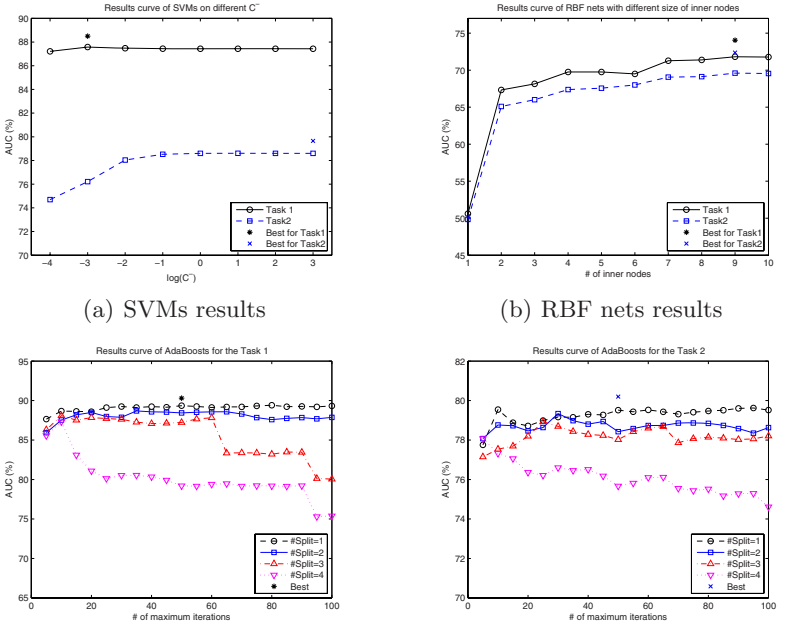
$$P_{final} = w_1 * P_{SVM} + w_2 * P_{RBF} + w_3 * P_{Adaboost} + w_4 * P_{LR} \tag{3}$$

The above weights, $w_i, i = 1, \ldots, 4$, are selected uniformly from $[0, 1]$ to tune a powerful ensemble classifier. The voting scheme is then incorporated by the values of the weights. Large value in the corresponding weight means that it votes towards the result of the corresponding classifier.

From the preliminary results on individual classifiers, we notice that the test performance, or the lift score, is proportional to the AUC score. A higher AUC score on the training data corresponds to a higher lift score on the test data. Hence, in tuning the ensemble model, we seek to optimize the AUC score of the model. Since w_i can be set to 0, some models will be automatically discarded when seeking a better ensemble model.

3 Experiments and Current Results

In the test, we first test the basic performance of individual models. In order to quickly obtain preliminary results, we use different training size on the models. More specifically, we randomly split the training data into 10 folds, where nine folds are used for training the SVMs, RBF nets, and Logistic Regression, and the rest fold is used to test the AUC score of these models. For the Real AdaBoost, we use only one twentieth of the training data in the training procedure and use the rest for test, due to the computation consideration. Since only one submission is allowed in one day for the contest and we observe that the AUC score is a good indicator to attain better test result, we apply the trained classifier corresponding to highest AUC score in each individual model for the test data to get the lift score. In the following, we detail the parameters seeking in different models:



(a) SVMs results (b) RBF nets results

(c) AdaBoosts results on the Task 1 (d) AdaBoosts results on the Task 2

Fig. 1. Validation results curve on the training data

SVMs: an SVM implemented by LibSVM [4] with linear kernel is adopted due to the computational consideration. The parameter of C^- in SVMs is tested from 10^{-4} to 10^3 and C^+ is set to 50 times larger than C^-.

AdaBoost: the Real AdaBoost implemented by [20] are tested with different number of splits in the tree and different number of maximum iteration. The number of splits in the tree is enumerated from 1 to 4 and the maximum iteration is tested from 5 to 100 with each step being 5.

NNs: RBF nets implemented by [16] are tested the number of inner nodes from 1 to 10 with other parameters being default.

Logistic Regression: no parameters need to be set.

Ensemble Model: weights, $w_i, i = 1, \ldots, 4$, are selected uniformly from $[0, 1]$ to tune a powerful ensemble classifier on the obtained above best models.

We show the results of individual models in Fig. 1 and report the best results obtained for all models in Table 2. From results, we have the following observations:

- The parameter C in SVMs is less sensitive to the task 1 and less sensitive to the task 2 when C is large.
- The AUC scores increase as the number of inner nodes increases for RBF nets and become less sensitive when the number of inner nodes is large.

Table 2. Results on different models. Lift scores are obtained when uploaded the results on the test sets. AUC scores are results on the inner test sets.

Method	Easy		Hard	
	AUC (%)	Lift	AUC (%)	Lift
SVM	88.5	3.699	79.7	2.771
RBF	74.0	1.964	72.4	2.664
AdaBoost	90.3	3.826	81.1	3.024
LR	90.1	3.82	80.1	2.984
Ensemble	**92.0**	**4.235**	**82.2**	**3.115**

- The AUC scores decrease as the number of split nodes increases for both tasks. They attain the maximum scores when the number of iterations equals 50 for both tasks.
- For individual models, Adaboost obtains the best lift score and AUC score for both tasks while Logistic Regression attains the second best lift score and AUC score. A higher AUC score on the test result of the training dataset corresponds to a higher lift score obtained from the submitted results.
- After obtained the ensemble model, we obtain a significant improvement on both AUC score and lift score for both tasks, which are the best among all the models. The experimental results indicate that the ensemble model works like as a voting scheme: a model with better performance has a larger weight.

In the above, we report the preliminary results on the contest. Since the number of given features is relative small, there may be non-linearity embedded in the data, we also use some techniques, e.g., spline [23], to expand the features and achieve better results. Finally, we find that a bottleneck is the highly imbalance in the data. Usually, standard classifiers tend to bias in favor of the larger class since by doing so it can reach high classification accuracy. Researchers usually adopt methods such as down-sampling of major class, up-sampling of minor class, or class-sensitive loss function, to tackle the imbalance data problem [15,24]. A more systematic method, the Biased Minimax Probability Machine, to solve the imbalance data problem is also proposed in the literature [11,10,9]. Due to computation consideration, for Adaboost, we modify the model by adjusting the dependent variable [1]. For other methods, we adopt a standard method, down-sampling on the negative samples, to alleviate the imbalance problem. However, the above methods are in heuristic way and data dependent. Seeking good parameters or good sub-samples is time consuming and we do not find much improvement on it. After trying several other methods, we can improve the lift score to 4.26 for the “easy” task and 3.19 for the “hard” task. Our results are ranked in top 20 for the “easy” task and top 10 for the “hard” task in one month before the deadline.

4 Conclusions

In this paper, we summarize our on-going work on the 2009 UC San Diego data mining contest. We have preprocessed the categorical features and tested on several standard classifiers, e.g., SVMs, RBF nets, AdaBoost, and Logistic Regression, without much modification, to get the preliminary results. The results reported in this stage can be used as reference to select classifiers for further tuning. Further, after notice that the AUC score on the training data is a good indicator to improve the lift score on the test data, we propose an ensemble model to optimize the AUC score and achieve significant better results.

References

1. Bell, R.M., Haffner, P.G., Volinsky, J.C.: Modifying boosted trees to improve performance on task 1 of the 2006 kdd challenge cup. ACM SIGKDD Explorations Newsletter 2, 47–52 (2006)
2. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1996)
3. Bradley, A.: The use of the area under the ROC curve in the evaluation of machine learning algorithm. Pattern Recognition 30(7), 1145–1159 (1997)
4. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
5. Fawcett, T.: An introduction to roc analysis. Pattern Recognition Letters 27, 861–874 (2006)
6. Freund, Y., Schapire, R.E.: Game theory, on-line prediction and boosting. In: Proc. of the Ninth Annual Conference on Computational Learning Theory, pp. 325–332 (1996)
7. García-Pedrajas, N., García-Osorio, C., Fyfe, C.: Nonlinear boosting projections for ensemble construction. Journal of Machine Learning Research 8, 1–33 (2007)
8. Hosmer, D.W., Lemeshow, S.: Applied logistic regression, 2nd edn. Wiley-Interscience Publication, Hoboken (2000)
9. Huang, K., Yang, H., King, I., Lyu, M.R.: Imbalanced learning with biased minimax probability machine. IEEE Transactions on System, Man, and Cybernetics Part B 36, 913–923 (2006)
10. Huang, K., Yang, H., King, I., Lyu, M.R.: Maximizing sensitivity in medical diagnosis using biased minimax probability machine. IEEE Transactions on Biomedical Engineering 53, 821–831 (2006)
11. Huang, K., Yang, H., King, I., Lyu, M.R., Chan, L.: The minimum error minimax probability machine. Journal of Machine Learning Research 5, 1253–1286 (2004)
12. Joachims, T.: A support vector method for multivariate performance measures. In: ICML, pp. 377–384 (2005)
13. Juditsky, A., Rigollet, P., Tsybakov, A.B.: Learning by mirror averaging. Annals of Statistics 36, 2183–2206 (2008)
14. Kégl, B., Busa-Fekete, R.: Boosting products of base classifiers. In: ICML, p. 63 (2009)
15. Maloof, M.A., Langley, P., Binford, T.O., Nevatia, R., Sage, S.: Improved rooftop detection in aerial images with machine learning. Machine Learning 53, 157–191 (2003)

16. Nabney, I.T.: Netlab: Algorithms for Pattern Recognition. Springer, Heidelberg (2004)
17. Platt, J.C., Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in Large Margin Classifiers, pp. 61–74. MIT Press, Cambridge (1999)
18. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. Annals of Statistics 26, 1651–1686 (1998)
19. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37(3), 297–336 (1999)
20. G.A.M. Toolbox, http://research.graphicon.ru/machine-learning/gml-adaboost-matlab-toolbox.html
21. Vapnik, V.: The Nature of Statistical Learning Theory, 2nd edn. Springer, New York (1999)
22. Vezhnevets, A., Vezhnevets, V.: Modest adaboost – teaching adaboost to generalize better. Graphicon (2005)
23. Wahba, G.: Spline Models for Observational Data, volume 59. In: CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 59. SIAM, Philadelphia (1990)
24. Weiss, G.M.: Mining with rarity: a unifying framework. SIGKDD Explorations 6(1), 7–19 (2004)