International Journal of Advanced Smart Convergence Vol.8 No.1 75-81 (2019)
http://dx.doi.org/10.7236/IJASC.2019.8.1.75

IJASC 19-1-9

Heterogeneous Ensemble of Classifiers from Under-Sampled and Over-Sampled Data for Imbalanced Data

Dae-Ki Kang[1] and Min-gyu Han[2]

[1]*Dept. of Computer Engineering, Dongseo University, Busan, South Korea*
[2]*ICT Convergence Program, Hansung University, Seoul, South Korea*
[1]*dkkang@dongseo.ac.kr,* [2]*andyhan@hansung.ac.kr*

Abstract

Data imbalance problem is common and causes serious problem in machine learning process. Sampling is one of the effective methods for solving data imbalance problem. Over-sampling increases the number of instances, so when over-sampling is applied in imbalanced data, it is applied to minority instances. Under-sampling reduces instances, which usually is performed on majority data. We apply under-sampling and over-sampling to imbalanced data and generate sampled data sets. From the generated data sets from sampling and original data set, we construct a heterogeneous ensemble of classifiers. We apply five different algorithms to the heterogeneous ensemble. Experimental results on an intrusion detection dataset as an imbalanced datasets show that our approach shows effective results.

Keywords: *Over-sampling, Under-sampling, Heterogeneous ensemble, Imbalanced data.*

1. Introduction

Imbalanced data causes problem in machine learning process [1]. Firstly it gives biased accuracy information which is based on counting the number of correctly classified instances from cross-validation. If we have 9,999 instances of normal event and 1 instance of abnormal event, then even "always print normal" algorithm will have 99.99 % of accuracy. Secondly, it is difficult to maintain reasonable decision boundary for minor category. Finally, imbalanced data involved different cost for misclassification which is often difficult to estimate.

Usually three different approaches are adopted for mitigating this problem [2]. The first approach is a sampling based approach to reduce the majority category by under-sampling [3] or inflate the minority category by over-sampling [4]. The second approach is to emphasize minority instances or penalize misclassification of minority instances by giving different weights [1]. The third approach is to use imbalance-aware machine learning algorithms [2].

Manuscript Received: Jan. 16, 2019 / Revised: Jan. 25, 2019 / Accepted: Jan. 30, 2019
Corresponding Author: andyhan@hansung.ac.kr
Tel: +82-02-760-5881, Fax: +82-02-760-5739
ICT Convergence Program, Hansung University, Seoul, South Korea

Ensemble methods are known to reduce variances. Well known ensemble methods include bagging [5], boosting [6], arcing [7], stacking [8], etc. When we construct an ensemble with classifiers generated from the same learner, we call it homogeneous ensemble, and if we construct an ensemble with classifiers generated from different classifiers, we call it heterogeneous ensemble [9].

In this paper, we propose a heterogeneous ensemble of classifiers which are created from differently sampled datasets. From an imbalanced dataset, we create an under-sampled dataset and an oversampled dataset. Together with the original dataset, we perform five different learning algorithms to generate different classifiers. From fifteen generated classifiers, made from three different datasets and five different learning algorithms, we generate a heterogeneous ensemble. We empirically validate the effectiveness of our approach on intrusion detection dataset, which is one of well-known data imbalance problem.

2. Data imbalance problem

Data imbalance problem arises when there is a huge difference between the number of one category instances and the other category instances. Since most learning algorithms try to maximize accuracy and use cross-validation or leave-one-out to measure the accuracy, data imbalance can cause misleading results. For example, when you have 9,900 normal instances and 100 abnormal instances, using stratified ten-fold cross-validation generates 10 data chunks, each of which has 990 normal instanced and 10 abnormal instances. And we use nine chunks to "generate" a classifier and use the remaining one chunk to "test" the classifier. For each different combination of choosing nine chunks from ten chunks (which is ten), we go through the "generate and test" step. Usually cross-validation is reasonable, but, for this imbalanced data, since there is a big difference among the instance counts, counting correctly classified and misclassified instances can generate misleading accuracy result. That is, even for a dumb classifier that always output "normal", its accuracy is 99.00%.

Unfortunately, this skewness among the numbers of instances naturally happens in most real-world problems. One well known example is in the problems which describe abnormal events, such as surveillance application, intrusion detection [10], bankruptcy prediction [2], etc. As aforementioned, there are many approaches to solve this problem.

2.1 Over-sampling

Over-sampling increases the number of instances, so when over-sampling is applied in imbalanced data, it is applied to minority instances. If you naively copy an instance into multiple identical instances (aka minority over-sampling with replacement [11], it can cause a serious problem. Since most evaluation approaches use cross-validation approach, multiple identical instances can be included into training instances and test instances. This cause inadvertent information leak, which lets the learner peek some instances in the training process. Another problem reported for minority over-sampling with replacement is making minority class decision boundary more specific, which causes over-fitting [4].

More refined way to relieve this information leak problem is to sample a new instance from training instances in a reasonable way, so that we believe the new instance is probably from the same distribution of the original training instances. One widely used sampling technique for this purpose is Synthetic Minority Over-sampling Technique (SMOTE) algorithm [4]. In SMOTE, to create a new instance, expressed in terms of a feature vector, by taking majority vote of a certain training instance and its k neighbors. For example, let a certain training instance X1=(A,B,C,D,E,1.0) and its two neighbors as X2=(A,I,C,J,G,1.5) and X3=(A,B,H,J,F,1.1). Note that the sixth attribute is a numeric attribute. Then, SMOTE will generate XN=(A,B,C,J,F,1.2). Note that for the fifth attribute of XN, we choose F, but E or G can be chosen too. And

for the case when attribute is numeric (the sixth attribute of XN), we can simply generate a mean value of 1.0, 1.5 and 1.1.

2.2 Under-sampling

Under-sampling is basically a technique for reducing instances. This involves removal of instances, which usually is performed randomly. In short, random under-sampling (RUS) is a widely used technique for under-sampling. There is a big question regarding random under-sampling. The question is that the under-sampling should be objected to minimize dissimilarity between original data and under-sampled data with respect to the task under consideration (usually classification or regression task).

In imbalanced data, under-sampling is usually performed on majority data. As its name implies, the sampling algorithm randomly remove instances until the distribution between major category and minor category is nearly uniform.

2.3 Fractional instances

In fractional instance setting, every instance is associated with a weight. In fractional instance setting, the weight is included to real number set. This fractional instance setting is sometimes used to fill the missing value in the data. That is, instead of filling the missing value with statistical estimate such as mean value, median value or mode value, the missing value can be replaced with distribution of values. For example, if the bag (or multiset) of attribute values (generated from training data) for the attribute of the particular missing value is {1:3, 2:2, 3:2}, then mean value is 1.86, median value is 2 and mode value is 1. In fractional instance setting, the missing value is replaced with distribution {1:3/7, 2:2/7, 3:2/7}. This literally means the missing value is 1 with 3/7 weight, 2 with 2/7 weight, and 3 with 2/7 weight. One popular usage of this setting is AdaBoost [6] which maintains a distribution for each instance.

For data imbalance problem, we can set higher weight to minority category than to majority category. For example, if majority category has 1,000 instances and minority class has 100 instances, then majority instance will have the weight 1 and minority class will have the weight 10.

2.4 Cost for misclassification

If the machine learning algorithm can aware cost for classification [12], assigning higher weight for misclassifying minor category can be a solution too. In the loss function for the learning algorithm can accommodate cost vector, instead of assigning different weights for instances which is sometimes complicated, we can assign cost vector (or matrix) as hyper-parameters of the algorithm. This is natural that, for example, mistaking an enemy as a friend is usually more dangerous than mistaking a friend as an enemy.

In imbalanced data, usually this, so called, interesting event (e.g. enemy, intrusion, bankruptcy, etc.) is rarer than normal event (e.g. friend, safe state, healthy company, etc.). Therefore, usually we assign higher cost to misclassifying abnormal as normal in data imbalance situation. One problem is that it is usually hard to deciding proper cost vector for misclassification.

2.5 Imbalance-aware learning algorithms

One fundamental way for solving data imbalance problem is to devise a machine learning algorithm that perform optimization with the consideration of data imbalance situation. For example, GMBoost [2] is a natural extension of AdaBoost to accommodate data imbalance situation using geometric mean.

3. Heterogeneous ensemble

Ensemble approach is an approach to construct a committee of classifiers (or models). It is well known that ensemble may not reduce bias but surely reduce variance. If the same learning algorithm is used for each classifiers in the ensemble, it is homogeneous ensemble, otherwise it is heterogeneous ensemble. There are many different approaches for generating ensemble.

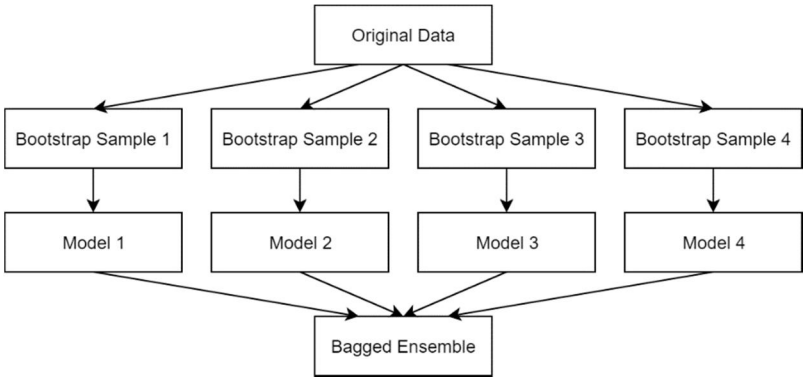

Figure 1. Bagging

3.1 Bagging

Bagging stands for Bootstrap Aggregating [5]. For a given data bootstrap techniques is applied to generate a new data. Bootstrap technique is to sample an instance with replacement from the original data. Figure 1 shows the diagram of Bagging.

3.2 Boosting

In Boosting [6], we use a distribution vector to maintain instance weights. The instance weight is initialized with 1.0 in the beginning. After applying a machine learning algorithm, the instance weight is updated according to the classification error. That is, when the previous classifier misclassify a certain instance, the weight for the instance is incremented, other the weight is decremented. Figure 2 shows the diagram of Boosting.

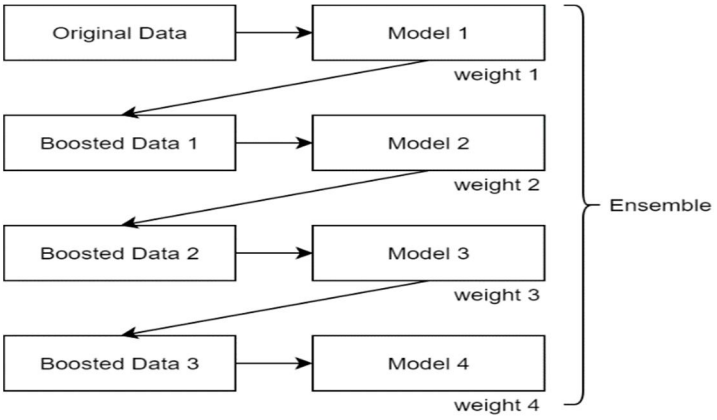

Figure 2. Boosting

3.3 Arcing

Adaptively resample and combine (Arcing) is similar to Boosting. Arcing is a sequential procedure where the next classifier is constructed based on the performance of the previous classifier. The difference between Arcing and Boosting is that Arcing uses the same weight for each classifier and Boosting uses different weights for classifiers.

3.4 Stacking

Note that Bagging can be a parallel process because there is no dependence among the classifiers, and so bootstrap can be applied in parallel to a dataset. Boosting and Arcing, to the contrary, adjust weights of instances according to the output of previous classifier, thus is a serial process.

Stacking is also a serial process, but instead of adjusting the weights of instances, the input of the next classifier is generated from the output of the previous classifier. That is, the output of the previous classifier is directly fed as an input to the next classifier.

4. Our approach

In our approach, we generate heterogeneous ensemble of models (or classifiers) which are generated from three datasets ad five different algorithms. The five algorithms are C4.5 decision tree learner, Naïve Bayes, support vector machines (SVM), repeated incremental pruning to produce error reduction (RIPPER), and logistic regression.

5. Experiments

We choose intrusion detection task for evaluating our approach on data imbalance problem. We use University of New Mexico (UNM) dataset [13] for intrusion detection.

5.1 Datasets

Each dataset in UNM data is a collection of abstracted system call sequences which corresponds to a specific exploit. We use 'UNM live lpr MIT' dataset for testing our algorithm. In 'UNM live lpr MIT' dataset, the number of normal sequences is 2,704 and the number of abnormal sequences is 1,001 [10]. We divide the data into a training set (1,803 normal sequences and 667 abnormal sequences) and a test set (901 normal sequences and 334 abnormal sequences). After under-sampling of the training set, the new training set has 667 normal sequences and 667 abnormal sequences. After over-sampling of the training set, the training set has 1,803 normal sequences and 1,803 abnormal sequences. For each sequence, we generate a bag of system calls following Kang's approach [10].

5.2 Experimental Results

Table 1 shows the results of applying five algorithms and our ensemble to 'UNM live lpr MIT' dataset. This experimental task is basically misuse detection where the learning algorithm is exposed to both normal instances and abnormal instances in the training stage.

Table 1. Experimental results on UNM live lpr MIT

Algorithm	Naïve Bayes	C4.5	RIPPER	SVM	Logistic Regression	Heterogeneous Ensemble
Original						

Accuracy	99.43	99.35	99.84	99.84	100.00	**100.00**
Detection rate	99.10	100.00	100.00	100.00	100.00	**100.00**
False positive rate	0.40	0.90	0.20	0.20	0.00	**0.00**
Under-Sampling						
Accuracy	99.51	98.30	99.43	99.68	99.60	
Detection rate	99.10	100.00	100.00	100.00	100.00	
False positive rate	0.30	2.30	0.80	0.40	0.60	
Over-Sampling						
Accuracy	99.51	99.83	99.84	99.84	100.00	
Detection rate	99.10	100.00	100.00	100.00	100.00	
False positive rate	0.30	0.20	0.20	0.20	0.00	

From the result in Table 1, it can be seen that our heterogeneous ensemble shows superior or comparable results in terms of accuracy, detection rate, and false positive rate. Note that our heterogeneous ensemble in the last column is a committee of fifteen classifiers shown in the first five columns of Table 1, therefore there are no results of under-sampling and over-sampling for heterogeneous ensemble.

6. Conclusion

In this paper, we propose heterogeneous ensemble of fifteen classifiers generated from the combination of three datasets (original, under-sampled, and over-sampled) and five learning algorithms (naïve Bayes, C4.5, RIPPER, SVM, and Logistic Regression). Experimental results from 'UNM live lpr MIT' dataset show that our proposed heterogeneous ensemble outperform other learning algorithms.

Possible future work include the application of this approach for deep learning methodologies [14,15].

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2018R1D1A1A02050166).

References

[1] P. Kang, and S. Cho. "EUS SVMs: Ensemble of Under-Sampled SVMs for Data Imbalance Problems." Lecture Notes in Computer Science, Vol. 4232, 2006.
DOI: https://doi.org/10.1007/11893028_93.

[2] M.-J. Kim, D.-K. Kang, and H. B. Kim. "Geometric Mean Based Boosting Algorithm with Over-Sampling to Resolve Data Imbalance Problem for Bankruptcy Prediction." Expert Systems with Applications, Vol. 42, No. 3, pp. 1074–1082, 2015.
DOI: https://doi.org/10.1016/j.eswa.2014.08.025 .

[3] C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse, and A. Napolitano, "RUSBoost: Improving Classification Performance When Training Data Is Skewed," in Proc. 19th International Conference on Pattern Recognition, pp. 1-4, 2008.

[4] N. V. Chawla, W. B. Kevin, O. H. Lawrence, and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-Sampling Technique." J. Artif. Int. Res., Vol. 16, No. 1, pp. 321–357, June 2002.

[5] L. Breiman, "Bagging Predictors." Machine Learning, Vol. 24, No. 2, pp. 123–140, August 1996.
DOI: https://doi.org/10.1023/A:1018054314350 .

[6] Y. Freund, and R. E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." Journal of Computer and System Sciences, Vol. 55, No. 1, pp. 119–39, 1997.
DOI: https://doi.org/10.1006/jcss.1997.1504 .

[7] L. Breiman, "Arcing Classifier (with Discussion and a Rejoinder by the Author)." The Annals of Statistics, Vol. 26,

No. 3, pp. 801–849, 1998.
DOI: https://doi.org/10.1214/aos/1024691079 .

[8] A. I., Naimi and L. B. Balzer. "Stacked Generalization: An Introduction to Super Learning." European Journal of Epidemiology, Vol. 33, No. 5, pp. 459–464, May 2018.
DOI: https://doi.org/10.1007/s10654-018-0390-z .

[9] M. P. Sesmero, A. I. Ledezma, and A. Sanchis. "Generating Ensembles of Heterogeneous Classifiers Using Stacked Generalization." Wiley Int. Rev. Data Min. and Knowl. Disc., Vol. 5, No. 1, pp. 21–34, January 2015.
DOI: https://doi.org/10.1002/widm.1143.

[10] D.-K. Kang, D. Fuller, and V. G. Honavar. "Learning Classifiers for Misuse Detection Using a Bag of System Calls Representation." In Proceedings of Intelligence and Security Informatics, IEEE International Conference on Intelligence and Security Informatics, ISI 2005, pp. 511–516, Atlanta, GA, USA, May 19-20, 2005.
DOI: https://doi.org/10.1007/11427995_51..

[11] B.X. Wang, N. Japkowicz, "Boosting Support Vector Machines For Imbalanced Data Sets," in Proceedings of the 17th international conference on foundations of intelligent systems, , pp. 38-47, Springer-Verlag, Berlin, Heidelberg, 2008.

[12] C. Elkan, "The Foundations Of Cost-Sensitive Learning," Proceedings of the 17th international joint conference on artificial intelligence, vol. 2, pp. 973-978, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2001.

[13] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstafi, "A sense of self for unix processes," in Proceedings of the 1996 IEEE Symposium on Security and Privacy, p. 120, IEEE Computer Society, 1996.

[14] Ho, J., and Kang, D.-K., "Improvement of the Convergence Rate of Deep Learning by Using Scaling Method," International Journal of Advanced Smart Convergence (IJASC), 6(4):67-72, December 2017.

[15] Pratama, K., and Kang, D.-K., "The Effect of Hyperparameter Choice on ReLU and SELU Activation Function," International Journal of Advanced Smart Convergence (IJASC), 6(4):73-79, December 2017.