Imbalance Learning

Hao Ding

Department of Electronic Engineering
College of Information Science and Engineering
Ocean University of China
Qingdao, China 266100
dinghao@stu.ouc.edu.cn

Abstract. Imbalanced learning has been a research emphasis in recent years because of the growing number of class-imbalanced classification problems. There is a growing number of real-world applications showing characteristics of class-imbalance classification suffering from severe class distribution skews and underrepresented data. EasmEnsemble samples several subsets from the majority class, trains a learner using each of them, and combines the outputs of the learners. SMOTE creates extra training data by performing certain operations on real data.AdaCost, a variant of AdaBoost, is a misclassification cost-sensitive boosting method. It uses the cost of misclassifications to update the training distribution on successive boosting rounds.

Keywords: Imbalanced Learning, EasyEnsemble, SMOTE, AdaCost, AdaBoost

1 Introduction

Highly imbalanced datasets are not uncommon in many pattern recognition tasks. For example, in medical datasets instances of diseased patients are typically rarer than instances of sane individuals. Yet, it is the rare cases that attract the most interest, as detecting them enables patients to be diagnosed and treated. Similar needs also appear in other real-world applications such as anomaly detection, fault diagnosis, email foldering, face recognition, fraud detection.

Under-Sampling method is efficient facing to imbalanced learning. The traditional algorithm of this method samples a subset from the majority class, then train the subset and the minority class at the same time. In this process the two trainers are training classes of the same size so the imbalanced problem seems to been solved.

The simplest approach to undersampling is to randomly select a fraction of records from the majority class. Unfortunately, this may lead to a loss of useful information. An interesting undersampling procedure is proposed in two methods called EasyEnsemble and BalanceCascade. [3] In EasyEnsemble, the majority class is sampled into several independent subsets that are used to train

separate classifiers, whose outputs are finally combined to produce the classification decision. In BalanceCascade, trained models are used to guide the sampling process for succeeding classifiers. The system is more focused on training patterns that are hard to classify.

Boosting and other ensemble methods, such as Bagging [4], have proved to be particularly robust at handling imbalanced data. A recent review on ensemble methods applied to handle the class imbalance problem is [5]. AdaBoost [6] for instance, is designed to reduce the bias towards the majority class by focusing on misclassified training patterns, while Bagging introduces the concept of bootstrap aggregating that consists in training several classifiers with bootstrapped copies of the original training set. Ensemble classifiers by themselves do not ameliorate the issue of imbalance if they are directly applied on data: this is due to their accuracy-oriented design. However, their combination with other techniques leads to posi- tive results. In these approaches, a datapreprocessing algorithm is applied before bagging/boosting, hence a 3-step process can be identified: resampling, ensemble building, and voting for the final classification.

Another group of classifiers is based on cost-sensitive learning. In this approach, a different cost is assigned to false negative and false positive patterns. SVM-WEIGHT implements cost-sensitive learning for SVM modeling [1]. It is implemented in LIBSVM 1 , so it is a very interesting baseline. The cost-sensitive principle is also applied in [2] to the ELM classifier.

To deal with imbalanced class, we can not easily use entire accuracy to evaluate an algorithm. this paper uses F-measure and G-mean to evaluate algorithms.

2 Related Work

Sampling, Cost-Sensitive equation and ensemble methods are three major method dealing with imbalanced learning.

2.1 Sampling Methods

Representative work in this area includes random oversampling [7], random undersampling [8], synthetic sampling with data generation [9], cluster-based sampling methods [10], and integration of sampling and boosting [11].

Under-Sampling changes the training sets by sampling a smaller majority training set and repeating instances in the minority training set respectively. For example, the synthetic minority oversampling technique (SMOTE) algorithm creates artificial data based on the feature space similarities between existing minority examples [12].

Over-Sampling copies the minority class for several times until it reaches the number of the majority class. Some studies have shown random oversampling to be ineffective at improving recognition of the minority class [13], while another study has shown that random undersampling is ineffective [14]. These two sampling methods also have significant drawbacks.

2.2 Cost-Sensitive Methods

Cost-Sensitive learning is another important class of class-imbalanced learning methods.

Cost-Sensitive methods solve the imbalanced learning problem by change the weight of different classes. The minority class gets high weight while the majority class gets low weight. Once the minority class being misclassified, the program costs a lot more. In each iteration, the program could train itself to recognize the minority class better and better.

One of the differences between methods is how the classifiers choose weights in every iteration like BMPM [15] and the other is how they iterate like AsymBoost [6], DataBoost [16] and AdaCost [17].

The deficiency of such two methods is that they ignore a large number of useful information from the majority class to pay too much attention to the minority class.

2.3 Ensemble Methods

Ensemble methods are widely used in the last decade. The algorithms usually combine different classes of methods together to reach higher accuracy. Such as SMOTEBoost is the combination of SMOTE and Boost, AdaCost adds Cost-Sensitive attribute into AdaBoost algorithm.

There are also two algorithms named EasyEnsemble and BalanceCascade [3]. They are ensemble methods using both Under-Sampling and Cost-Sensitive.

The two ensemble algorithms are useful for increasing the accuracy based on the above methods. But they are also dichotomy algorithms. EasyEnsemble.M [18] developed from EasyEnsemble and transfered it to solve multi-class problems. Since EasyEnsemble.M just mechanically designed the iterations and used absolute equilibrium algorithm, both the F-measure and G-means come down.

3 Algorithms

3.1 EasyEnsemble

EasyEnsemble, as mentioned before, is an ensemble algorithm gathering Under-Sampling classifiers. EasyEnsemble undersamples the majority class for many times with numbers of iterations. Thus data in the majority class won't be too much wasted.

First get a minority class $\mathcal{P}$ and a mojority class $\mathcal{N}$. To solve 2-classes imbalanced problem, then randomly samples a subsets $\mathcal{N}'$ from $\mathcal{N}$, $\mathcal{N}' = |\mathcal{P}|$. As for ensemble method, independently sample several subsets $\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_t, ..., \mathcal{N}_T$ from $\mathcal{N}$. For each subset $\mathcal{N}_t (1 \leq t \leq T)$, a classifier H_t is trained using $\mathcal{N}_t$ and all of $\mathcal{P}$. All generated classifiers are combined for the final decision. AdaBoost is used to train the classifier H_t.

Algorithm 1 EasyEnsemble

Input: A set of minority class examples $\mathcal{P}$; a sets of majority class examples $\mathcal{N}, |\mathcal{P}| < |\mathcal{N}|$; the number of subsets T to sample from $\mathcal{N}$ and s_t,the number of iterations to train an AdaBoost ensemble H_t

1: initial $t = 1$;
2: **repeat**
3: Randomly sample a subset $\mathcal{N}_t$ from $\mathcal{N}$, $|\mathcal{N}_t| = |\mathcal{P}|$;
4: Learn H_t using $\mathcal{P}$ and $\mathcal{N}$. H_t is an AdaBoost ensemble with s_t weak classifiers $h_{t,d}$ and corresponding weights $\alpha_{t,d}$. The ensemble's threshold is θ_t, *i.e.* $H_t(x) = sgn(\sum_{d=1}^{s_t} \alpha_{t,d} h_{t,d}(x) - \theta_t)$;
5: **until** $t = T + 1$

Output: An ensemble:
$H(x) = sgn(\sum_{t=1}^{T} \sum_{d=1}^{s_t} \alpha_{t,d} h_{t,d}(x) - \sum_{t=1}^{T} \theta_t)$

Similar to the balanced Random Forests, EasyEnsemble generates T balanced sub-problems. The output of the ith sub-problem is AdaBoost classfier H_i, an ensemble with s_i weak classifiers. The output of EasyEnsemble is a single ensemble, but it looks like an 'ensemble of ensembles'.

3.2 SMOTE

SMOTE is an algorithm to handle class imbalance problem in data with discrete class labels. It uses a combination of SMOTE and the standard boosting procedure AdaBoost to better model the minority class by providing the learner not only with the minority class examples that were misclassified in the previous boosting iteration but also with broader representation of those instances (achieved by SMOTE). Since boosting algorithms give equal weight to all misclassified examples and sample from a pool of data that predominantly consists of majority class, subsequent sampling of the training set is still skewed towards the majority class. Thus, to reduce the bias inherent in the learning procedure due to class imbalance and to increase the sampling weights of minority class, SMOTEis introduced at each round of boosting. Introduction of SMOTE increases the number of minority class samples for the learner and focus on these cases in the distribution at each boosting round. In addition to maximizing the margin for the skewed class dataset, this procedure also increases the diversity among the classifiers in the ensemble because at each iteration a different set of synthetic samples are produced.

The algorithm flow is as follows:

- For each sample x in a few classes, the distance from the Euclidean distance to all the samples in the sample set is calculated and the k nearest neighbor is obtained.

- Set a sampling ratio based on the sample imbalance ratio to determine the sampling magnification N. For each minority sample x, randomly select several samples from its k neighbors, assuming that the selected neighborhood is xn.
- For each randomly selected neighbor xn, construct a new sample D with the original sample according to the formula: $x_{new} = x + rand(0, 1) * (\hat{x} - x)$
- from the sample set D, do not put back random sampling n1 ¡n samples, get the set D1, training weak classifier C1;
- from the sample set D, the extraction of n2 ¡n samples, which merged into the first half by C1 classification error samples, get the sample set D2, training weak classifier C2;
- extract the D sample set, C1 and C2 classification inconsistent sample, composed of D3, training weak classifier C3;
- with three classifiers to vote, get the final classification results.

In this method, the SMOTE method is used to over-sample a few classes to increase the number of classes, and a new data set is generated. Secondly, the new data sets are subjected to under-sampling in the case of keeping the sample distribution to generate the training data set. The Boosting algorithm is used to classify the training data sets after sampling, and the corresponding training model is generated. It will increase the weight of the misclassified samples in each iteration process, so that the trainer will pay more attention to the errors in the next training sample. The combination of multiple weak classifiers is promoted to a strong classifier, thereby improving the recognition rate for a small number of classes and data sets as a whole.

3.3 AdaCost

Freund and Schapires AdaBoost learns a highly accurate voted ensemble of many weak hypotheses. Typically, each hypothesis outputs both a prediction and a confidence for this prediction. Each hypothesis is trained on the same data set yet with a different distribution. Different hypotheses are produced in different rounds of boosting. At each round, AdaBoost increases the weights of wrongly classified training instances and decreases those of correctly predicted instances. AdaBoost allows for an arbitrary initial distribution. For classification error, each example is given an equal weight. To reduce cumulative misclassification costs, costly examples can be given higher weights. AdaBoost reduces the weighted error for this initial distribution. Schapire, Singer and Singhal gave different weights for false positives and false negatives to apply AdaBoost in text-filtering. In AdaCost, the weight updating rule increases the weights of costly wrong classifications more aggressively, but decreases the weights of costly correct classifications more conservatively. This is accomplished by introducing a misclassification cost adjustment function into the weight updating formula. Under this updating rule, the weights for expensive examples are higher and the weights for inexpensive examples are comparatively lower. Each weak hypothesis correctly predicts more expensive examples for such a distribution. The final voted ensemble will also correctly predict more costly instances.

Algorithm 2 AdaCost

Input: $S = \{(x_1, c_1, y_1), ..., (x_m, c_m, y_m)\}$; $x_i \in X, c_i \in \Re^+$, $y_i \in \{-1, +1\}$.
1: Initialize $D_1(i)$(such as $D_1(i) = c_i / \sum_j^m c_j$).
2: **repeat**
3: Train weak learner using distribution D_t
4: Compute weak hypothesis $h_t : X \to R$
5: Choose $\alpha_t \in R and \beta(i) \in R^+$.
6: Update $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i) \beta(i))}{Z_t}$
7: **until** $t = T + 1$
Output: The final hypothesis:
$H(x) = sign(f(x)) where f(x) = (\sum_(^T t + 1) \alpha_t h_t(x))$

3.4 Evaluation Criteria

In case of imbalanced dataset, accuracy is not the metric that should be exercised as it can prove to be sometimes deceiving. Hence, several new metrics were proposed to better evaluate the models and tell us what we are really interested in. The followings are some of the common performance metrics that are supposed to better analyze and evaluate the performance of the imbalanced methods than the inferior traditional classification accuracy [19].

TN : true reg
FP : false positives
TP : true positives
FN : false negatives

For dichotomy we used such two evaluation criterias:

$$G - means = \sqrt{\frac{TP}{TP + FN} * \frac{TN}{TN + FP}} \tag{1}$$

$$F - measure = \frac{1}{\frac{1}{2}(\frac{1}{p} + \frac{1}{r})} = \frac{2pr}{p + r} \tag{2}$$

For multi-classification we used such two evaluation criterias:

$$G - means = \sqrt{\frac{TP}{TP + FN} * \frac{TN}{TN + FP}} \tag{3}$$

$$Micro - F = \frac{1}{k} \sum_{i=1}^{k} F_i \tag{4}$$

4 Experiment

4.1 Dataset

Multiple Features Data Set, the dataset for this experiment is dataset consisting of features of handwritten numerals (‘0 ’-‘ 9’) extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized In binary images. I selected ten features in mfeat-pix (240 pixel averages in 2 x 3 windows). According to the unbalance rate of 9,4,2.3,1.5,1, the balance data set into an unbalanced data set. For the imbalance rate of 9 is 0-8 as a positive class with 0 as a label, 9 as a negative class with 1 as a label, the rest of the situation with the same rate of imbalance. For EasyEnsemble requires artificial production training set and test set, this experiment uses the training set from 10 categories were selected 120 samples that a total of 1200 samples, the remaining samples as a test set.

4.2 Results

Table 1. EasyEnsemble

Imbalance rate	5/5	6/4	7/3	8/2	9/1
F-Measure	0.4096	0.4825	0.5525	0.5098	0.5445
G-Means	0.7453	0.6956	0.6856	0.5334	0.5982

Table 2. SMOTEBoost

Imbalance rate	5/5	6/4	7/3	8/2	9/1
F-Measure	0.9045	0.8649	0.7385	0.6954	0.6452
G-Means	0.6278	0.6956	0.6012	0.5834	0.5985

Table 3. AdaCost

Imbalance rate	5/5	6/4	7/3	8/2	9/1
F-Measure	0.7458	0.7937	0.7239	0.6028	0.5734
G-Means	0.7397	0.7304	0.6932	0.6347	0.6823

We ran three algorithms on Multiple Features Data Set. Our evaluation criterias are F-Measure and G-Means.

In Table 1, EasyEnsemble performs better on higher imbalance rate using F-Measure, and gets lower values on G-Means.

In Table 2, as the imbalance rate goes down, SMOTEBoost gets lower values both on F-Measure and G-Means

In Table 3, AdaCost gets lower values both on F-Measure and G-Means when facing higher imbalance rate.

5 Conclusion

Using Multiple Features Data Set, for the same imbalance rate, AdaCost is the most robust algorithm comparing with EasyEnsemble and SMOTEBoost. SMOTEBoost is the best one dealing with low imbalance rate problems while AdaCost gets best values on high imbalance rate.

References

1. T. Yu, T. Jan, S.J. Simoff, J.K. Debenham, A Hierarchical VQSVM for imbalanced data sets, in: Proceedings of IJCNN, 2007, pp. 518523.
2. Weiwei Zong, Guang-Bin Huang, Yiqiang Chen, Weighted extreme learning machine for imbalance learning, Neurocomputing 101 (2013) 229242.
3. X.-Y.Liu, J.Wu, and Z.-H.Zhou. "Exploratory under-sampling for class-imbalance learning." In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM'06), Hong Kong, China, pp. 965-969. 2006.
4. G.-Z. Li, H.-H. Meng, W.-C. Lu, J.Y. Yang,and, M.Q. Yang, Asymmetric bagging and feature selection for activities prediction of drug molecules, BMC Bioin- form. 9 (S6) (2008) S7.
5. Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman & Hall/ CRC, Boca Raton, FL, 2012.
6. P.Viola and M.Jones. "Fast and robust classification using asymmetric AdaBoost and a detector cascade," In: Advances in Neural Information Processing Systems 14, T.G.Dietterich, S.Becker, and Z.Ghahramani, Eds.Cambridge, MA: MIT Press, pp. 13111318. 2002.
7. A. Estabrooks, T. Jo, and N. Japkowicz, A multiple resampling method for learning from imbalanced data sets, Computational Intelligence, vol. 20, no. 1, 1836, 2004.
8. C. Drummond and R. C. Holte, C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling, in Proceedings of International Conference Machine Learning, Workshop on Learning from Imbalanced Data Sets II , 2003.
9. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research, vol. 16, pp. 321357, 2002.
10. T. Jo and N. Japkowicz, Class imbalances versus small disjuncts, ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 4049, 2004.
11. H. Guo and H. L. Viktor, Learning from imbalanced data sets with boosting and data generation: The DataBoost-IM approach, ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 3039, 2004.

12. N.V.Chawla, K.W.Bowyer, L.O.Hall, and W.P.Kegelmeyer. "SMOTE: Synthetic minority over-sampling technique." In: Journal of Artificial Intelligence Research, vol.16, pp. 321357, 2002.
13. C. Ling and C. Li, Data mining for direct marketing problems and solutions, in Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), pp. 7379, AAAI Press, 1998.
14. N. Japkowicz and S. Stephen, The class imbalance problem: A systematic study, Intelligent Data Analysis, vol. 6, no. 5, pp. 429450, 2002.
15. K.Huang, H.Yang, I.King, and M.R.Lyu. "Learning classifiers from imbalanced data based on biased minimax probability machine." In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, pp. 558563. 2004.
16. H.Guo and H.L.Viktor. "Learning from imbalanced data sets with boosting and data generation: The DataBoost-IM approach," In: ACM SIGKDD Explorations, vol.6, no.1, pp. 3039, 2004.
17. W.Fan, S.J.Stolfo, J.Zhang, and P.K. Chan. "AdaCost: Misclassification cost-sensitive boosting," In: Proceedings of the 16th International Confernece on Machine Learning, Bled, Slovenia, pp. 97105. 1999.
18. Q.-Q.Li, and X.-Y.Liu. "EasyEnsemble.M for multiclass imbalance problem." In: Pattern Recognition and Artificial Intelligence 27.2:187-192. 2014.
19. Kaur P, Negi V. Techniques based upon boosting to counter class imbalance problemA survey[C]//Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on. IEEE, 2620-2623.2016.