An Imbalanced Data Rule Learner

Canh Hao Nguyen and Tu Bao Ho

School of Knowledge Science,
Japan Advanced Institute of Science and Technology,
1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan
{canhhao, bao}@jaist.ac.jp

Abstract. Imbalanced data learning has recently begun to receive much attention from research and industrial communities as traditional machine learners no longer give satisfactory results. Solutions to the problem generally attempt to adapt standard learners to the imbalanced data setting. Basically, higher weights are assigned to small class examples to avoid their being overshadowed by the large class ones. The difficulty determining a reasonable weight for each example remains. In this work, we propose a scheme to weight examples of the small class based solely on local data distributions. The approach is for categorical data, and a rule learning algorithm is constructed taking the weighting scheme into account. Empirical evaluations prove the advantages of this approach.

1 Introduction

It a practical sense, applying standard machine learning methods to real world tasks when their class distribution is imbalanced is problematic. This is the case of a data set, in which the number of examples of one class is substantially smaller than the others. For instance,if one class accounts for only 2% of the total number of examples in the data set, a classifier can get a high accuracy of 98% just by assigning all its examples to the large class. However, in such a case, the classifier completely fails to learn the small class, which is usually of interest. In practice, researchers have encountered this problem in many domains, including the detection of fraudulent transactions [1], network intrusion detections [2] and oil spills in satellite radar images [3].

The reason that standard classifiers can no longer give a satisfactory performance on such data sets is because they make the fundamental assumption that frequencies of classes are equally distributed. Adaptations to imbalanced data sets are usually made by giving small class examples higher weights. One simple way is resampling, which duplicates small class examples or selects only a subset of large class ones. Such approaches do not have high performance, as reported in [4], because various examples are affected differently by the class imbalance problem. SMOTE [5] combines synthetic example generation with downsampling, but the resampling degree is not specified. Resampling to reflect relative weights between examples or classes still remains an art.

It is believed to be more promising to weight examples differently, and various approaches have been proposed. Kubat et al. [3] insist that large class examples

A. Jorge et al. (Eds.): PKDD 2005, LNAI 3721, pp. 617–624, 2005.

in a mixed region should be weighted zero as long as that increases performance measure. Learning on a cluster basis is used [6] to weight examples accordingly. Learning decision trees (DT) [7] is made independent of class frequencies by using the Area Under the ROC Curve (AUC) as a splitting criterion, which is equivalent to weighting examples according to their distribution in the set of examples covered by the splitting nodes. A general way to optimally weight examples (in Bayes risk sense) is using MetaCost [8], by bagging and then probability estimation. However, in highly imbalanced data sets, examples of small class are rarely learned, making their optimal costs extremely high. Again, it is still a challenge to weight examples optimally for the imbalanced data problem.

We propose a method to estimate the optimal weight of each small class example basing solely on local data distributions. The intuition is that by looking more closely into local data distribution, we have more chance to reveal useful information about the effect of class imbalance. To this end, we first define the concept of *vicinity*, which characterizes local data distribution and then determines examples' weights with the aiming of maximizing AUC in the vicinity. The weight is integrated into a rule induction algorithm at the rule pruning step.

The paper is organized as follows. Section 2 is the foundation and formulation of our locally adaptive weighting scheme. Integration of the weighting scheme into our rule learning algorithm is described in Section 3. In Section 4, we show experimental evaluations of the scheme to other imbalanced data classifiers. Conclusions are presented and future work is discussed in section 5.

2 Locally Adaptive Weighting Scheme

We approach the imbalanced data problem by giving a weight adaptively for each small class example, while keeping the weights of large class examples at a default value (i.e. 1). The key idea is to weight each small class based on its local neighborhood (hence, it is locally adaptive), which is defined as the vicinity of the rule covering it. This section will define the concept of vicinity and derive the formulation of example weighting based on vicinity using AUC as the criterion.

Vicinity: The idea behind vicinity is as follows. Consider two rules $R_i, i = 1, 2$ with the same coverage for every class (R_i covers n_i, p_i examples from large and small classes respectively, $n_1 = n_2, p_1 = p_2$). Conventionally, the two rules are evaluated as the same goodness (e.g., precision for small class $\frac{p_i}{p_i+n_i}$). Assume that we have some way to define the surround of a rule, called a neighborhood. If R_1 is likely to be pruned to a better one, then in its neighborhood there must be some examples of the same class as the predicting class of R_1. On the other hand, R_2 is surrounded by examples from other classes, hence it cannot be pruned to a better one. Our idea is to evaluate the two rules differently, R_1 to be higher than R_2, reflecting their ability to be pruned. This different evaluation is based on the fact that there is a set of examples in each rule's neighborhood, which creates the difference in pruning ability. By vicinity, then, we mean this set of examples. We define vicinity based on the concept of k-vicinity.

Definition: *The distance from a rule to an example is the minimum number of attribute value pairs in the body of the rule that must be removed in order to make the rule cover the example.*
Definition: *The k-vicinity of a rule R for a training data set D is the set of examples in D that are less than or equal to a distance of k to the rule.*

$$k\text{-}vicinity(R) = \{x \mid x \in D, Distance(R, x) \leq k\} \tag{1}$$

K-vicinity is a subset the training data set, which is potentially covered by the rule after k steps of generalization. The smaller k is, the higher the influence the examples in k-vicinity may have on the generalization (pruning) ability of the rule. For example, 0-vicinity is the set of examples covered by the rule, m-vicinity is the whole data set if m is the number of attribute-value pairs in the rule body. The set of all k-vicinities is a nested chain of subsets of the data set, meaning: *0-vicinity* $\subseteq$ *1-vicinity* $\subseteq ... \subseteq$ *m-vicinity*. We define vicinity using this chain with weights. Formally, vicinity is a function f over k-vicinities.

$$vicinity = f\{1\text{-}vicinity, 2\text{-}vicinity, ..., m\text{-}vicinity\} \tag{2}$$

Estimating vicinity is a difficult task. However, the way around the problem is to let the vicinity of a rule remain a virtual concept. We only need to calculate the *ratio of class distribution* in the vicinity as in the weighting scheme discussed in the next section.
Example Weighting and Rule Evaluation: As a vicinity is expected to contain examples that influence the pruning ability of a rule, we use this assumption to define the *best* rule as the one giving optimal classification within its vicinity. Our idea is to weight examples in the vicinity such that the optimal classification coincides with the lowest misclassification cost. Defining optimal classification on a vicinity results in a locally adaptive weighting scheme.

We define optimal classification as the one that gives the largest AUC. AUC is a popular metric for comparing classifiers' performance [9, 10] when the misclassification costs are unknown. When a classifier is a set of rules, as in Figure 1 (a), the ROC curve contains a set of line segments. Here, the classifier is assumed to have four rules, sorted in decreasing order of their precision for a class. For simplicity, we assume that there is only one rule for small class in a vicinity. Then, the ROC curve of a classifier (by R or $R2$) in its vicinity would look like Figure 1 (b). The classifier here consists of a rule (say R) and the default rule predicting the large class. Suppose R covers p small and n large class examples, and the vicinity contains P small and N large class examples. The rule evaluation metric, defined to be AUC above, is calculated [7] as:

$$AUC(R) = \frac{p}{2P} - \frac{n}{2N} + \frac{1}{2} \tag{3}$$

The above formula implies that the weight of a small class example in this vicinity is $\frac{N}{P}$ when the weight of a large class example is the default value 1.

The rule evaluation metric is used to compare different rules for search bias. However, it is not natural to compare AUC in different contexts (vicinities).

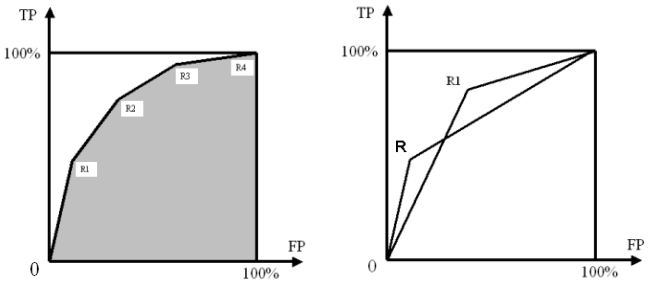

Fig. 1. The ROC space: (a) plot of a rule learner. (b) for rule comparison in a vicinity.

Hence, we propose a comparison strategy that rule R_1 is considered better than rule R if and only if it gives a higher AUC in the vicinity of R. Equivalently, R_1 is considered better than R if their AUC difference (in formula 4) is positive.

$$AUC(R_1) - AUC(R) = \frac{1}{2}(\frac{p_1 - p}{P} - \frac{n_1 - n}{N}) = \frac{1}{2}[(p_1 - p)\frac{N}{P} - (n_1 - n)]\frac{1}{N} \quad (4)$$

For the purpose of comparing rules for search bias, it is sufficient to know $\frac{N}{P}$. This is the reason we allow vicinity to remain an abstract concept, while we heuristically estimate $\frac{N}{P}$ directly. From equation 2, we propose to estimate class distribution ratio $\frac{N}{P}$ in the vicinity to be:

$$\frac{N}{P} = \sum_{k=1}^{m} w_k * \frac{N_k}{P_k} \quad (5)$$

where m is the number of vicinities. In this formula, $\frac{N_k}{P_k}$ is class distribution ratio in k-vicinity and w_k is its associated weight, with $\sum w_k = 1$. This just smooths out the class distribution ratios of different k-vicinities to estimate that of the vicinity, in similar fashion to a shrinkage estimator, to make it robust. The definition of vicinity is tunable by its weighting scheme, namely the set $\{w_k\}$. If w_k is large for small k-s, vicinity reflects more local information. If w_k is large for large k-s, vicinity is more global. If we want to define vicinity to be the whole data set, then set $w_m = 1$. Having tunable set of $\{w_k\}$ is a generalization of a simple cost sensitive classification. In this algorithm, we fix the default as:

$$w_k = \frac{1}{m}, k = \overline{1, m} \quad (6)$$

Such weights can also be set adaptively by users.

Discussion: The key point which makes this example weighting scheme suitable for imbalanced data is the use of a local neighborhood. Having a myopic view around a rule gives us a better picture of how much the imbalance may hinder classification rules. Examples far away from the boundaries of classes may not participate in any vicinity, also not affecting class discrimination (this is similar to the idea behind SVMs).

IDL
1. Generate a candidate rule set
2. Prune rules from high coverage to low

GenerateRuleSet
1. Generate a decision tree
2. Stop when leaf nodes contain only example from one class
3. Extract the set of leaf nodes that contain only examples of small class
4. Convert those nodes into rules and return

PruneRules
1. Sort rules according to coverage
2. From high to low coverage rule do
3. Remove *best* attribute value pair
4. Until no more AUC is gained
5. Return pruned rules

Fig. 2. IDL algorithm

3 IDL: Imbalanced Data Learner

IDL is a rule induction algorithm, which uses an one-sided selection strategy, taking example weights into account to learn rules for a small class. IDL consists of two steps. First, it generates a set of candidate rules for the small class by growing a decision tree, which are meant to be complete and of high precision. Then it prunes these rules by greedily removing attribute value pairs to make them robust. Example weighting is used in rule pruning step using Formula 3 as the rule evaluation metric. The overall strategy is depicted in Figure 2.

In the candidate rule set generation step, IDL grows a decision tree and only stops when the leaf nodes contain examples from one class. As recommended in [11], IDL takes the impurity ($2\sqrt{p(1-p)}$) gain as the splitting criterion. After the decision tree is fully grown, the set of leaf nodes that contain only small class examples are collected and turned into a set of rules. In the second step, the collected rules for the small class are sorted in decreasing order of coverage. Starting from the highest coverage one, each rule is pruned by removing the *best* attribute value pair (the one having the highest AUC difference), according to formula 4. It stops when removing does not improve either AUC or when the precision of the rule (calculated without taking weights into account) falls under a certain threshold. The examples covered by a rule are marked so that if they are covered again, only half of their weights are retained. This makes the rules overlap, and also greatly improves the recall of the classifier. The threshold represents the minimum precision a rule should achieve, reflecting the amount of noise in the data. This threshold is generally set by the users, and is estimated in IDL as follows. First, set it to 80%, then do a 10-fold stratified cross-validation

on the data set to estimate its difficulty to learn. Taking the F-measure on the small class, say f (percent), then the threshold takes the value $max(50, f - 10)$.

In the first step, IDL constructs an unpruned decision tree, which is of $O(ea)$ time complexity, where e is the number of examples and a is the number of attributes. In the second step, suppose it generates k rules, each has maximum n_k attribute value pairs. As each pruning operation requires a pass of the data set to calculate the class distribution ratio in the vicinity, the time complexity of this step is at most $O(ekn_k)$.

4 Experimental Evaluation

We evaluated IDL on its ability to learn a small class, and compared it to other approaches. The first of these was SMOTE-NC (over C4.5) [5], the nominal categorical version of what is arguably best method (SMOTE) for learning imbalanced data. Since SMOTE is sensitive to its degree of sampling parameters, we ran it on three degrees of small class upsampling, namely N=100%, N= 300% and N=700%. We also compared IDL to a general classifier of C4.5, with and without cost sensitive setting. In cost sensitive setting (C.S.), the relative cost is just the ratio of class distribution of the data set. Boosting is also capable of enhancing imbalanced data learners [12], so AdaBoost over C4.5 was also compared. We used these classifiers from WEKA[1]. All algorithms ran with their default parameters. We used F-measure on the small class as the performance criterion, instead of the AUC measure (since we learn only small class rules).

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{7}$$

We evaluated those algorithms on selected fifteen UCI data sets [2], where smallest class was chosen to be small class, and the other classes were merged to become the large class. As the algorithm is for categorical data, all data sets were discretized. We split data sets with a ratio of 75-25 randomly in a stratified manner, the large parts were used for training and the small parts for testing. Table 1 shows the result of testing on the small part of the data. The columns are names, percentage of small class proceeded with class index and then classifiers (SMOTE is tested with three parameters). All numbers are in percentage. The last line shows average performance of on all data sets.

The table shows that our approach outperforms general classifiers by a large margin, and is competitive to all three parameters for SMOTE. IDL shows an improvement of 11.74% in terms of F measure on the small class compared to a standard classifier of C4.5. For the cost sensitive setting of C4.5 (C.S), it also improves by 3.81%. Compared to AdaBoost, IDL's accuracy is 2.85% higher. This means that IDL is more suitable for imbalanced data than general classifiers. Comparing IDL to a imbalanced data learner of SMOTE (SMOTE-NC version), IDL is also competitive to the three parameter settings; the average

[1] www.cs.waikato.ac.nz/ml/weka/

[2] http://www.ics.uci.edu/ mlearn/MLRepository.html

Table 1. Comparison of Classifiers on UCI data

Name	%	C4.5	C.S.	SMOTE				A.Boost	IDL
				100	300	700	average		
annealing1	11.0	73.9	62.3	77.4	71.0	66.7	71.7	70.6	96.2
car3	3.7	66.7	84.2	77.4	80.0	80.0	79.1	80.0	76.9
flare4	8.0	0.0	36.9	30.4	36.1	38.6	35.0	32.7	29.0
glass3	13.5	93.3	73.7	93.3	82.4	82.4	86.0	80.0	85.7
hypo0	5.0	85.7	81.9	85.7	83.1	83.1	84.0	84.6	84.6
inf0	6.3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
krkopt16	0.9	85.7	82.8	88.4	88.4	90.1	89.0	84.1	87.1
krkopt4	0.7	58.0	69.1	66.7	71.8	66.7	68.4	61.3	75.9
led7	8.4	59.8	59.9	63.9	61.6	50.5	58.7	50.7	62.4
letter0	3.9	91.0	90.0	92.0	91.8	89.7	91.2	96.0	92.0
satimage3	9.7	51.5	52.8	57.9	51.8	51.3	53.7	57.9	50.3
segmentation5	14.1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
sick1	6.5	82.8	69.6	82.8	81.8	73.8	79.5	82.8	80.9
vowel5	9.1	0.0	75.2	67.8	69.4	65.5	68.2	80.0	60.0
yeast4	3.4	0.0	30.8	33.3	44.4	40.0	39.2	21.1	43.5
Average	**6.94**	**63.23**	**71.16**	**74.59**	**74.24**	**71.89**	**73.57**	**72.12**	**74.97**

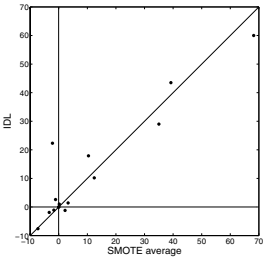

Fig. 3. Improvement of IDL versus SMOTE

performance of SMOTE is 1.40% lower than that of IDL. It is noteworthy that there is no systematical way to determine the resampling degree for SMOTE.

It is interesting to look at the improvement of IDL over C4.5 compared to the average improvement of that of SMOTE over C4.5 in Figure 3. X axis is the performance improvement of SMOTE (averaging all parameters), while y axis is for IDL. The set of points shows a near linear relation. This means that improvement of IDL is proportionate with that of SMOTE, meaning that IDL is consistently similar to SMOTE.

5 Conclusion

We have proposed a method to weight examples for a small class based on their local neighborhood. Neighborhood is defined as the virtual concept of vicinity, while computation is based on k-vicinities. The algorithm is clearly more

accurate than general classifiers, including Adaboost and MetaCost, and is competitive to SMOTE while having the advantage of not requiring resampling parameters. From this, we can conclude that the information of neighborhood of an example is useful for weighting it, in order to compensate imbalanced data.

The clear limitation of this method is how to define the weighting scheme for a vicinity. For the moment, computational complexity is its main problem, which should be reduced for large data sets. Applying the weighting scheme to other classifiers for imbalanced data is a natural extension. Whether local data distribution can be used to improve classifiers in general is an open question.

References

[1] Fawcett, T., Provost, F.: Combining data mining and machine learning for effective user profiling. In Simoudis, Han, Fayyad, eds.: The Second International Conference on Knowledge Discovery and Data Mining, AAAI Press (1996) 8–13

[2] Lazarevic, A., Ertoz, L., Ozgur, A., Srivastava, J., Kumar, V.: "evaluation of outlier detection schemes for detecting network intrusions". In: Third SIAM International Conference on Data Mining. (2003)

[3] Kubat, M., Holte, R.C., Matwin, S.: Machine learning for the detection of oil spills in satellite radar images. Machine Learning **30** (1998) 195–215

[4] Japkowicz, N.: The class imbalance problems: Significance and strategies. In: Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000). Volume 1. (2000) 111–117

[5] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sapling technique. Journal of Artificial Intelligence Research **16** (2002) 321–357

[6] Nickerson, A., Japkowicz, N., Milios, E.: Using unsupervised learning to guide re-sampling in imbalanced data sets. In: Eighth International Workshop on AI and Statitsics. (2001) 261–265

[7] Ferri, C., Flach, P., Hernandez-Orallo, J.: Learning decision trees using the area under the roc curve. In Sammut, C., ed.: Nineteenth International Conference on Machine Learning ICML'02, Morgan Kaufmann (2002)

[8] Domingos, P.: Metacost: A general method for making classifiers cost-sensitive. In: Knowledge Discovery and Data Mining. (1999) 155–164

[9] Provost, F., Fawcett, T.: Robust classification for imprecise environments. Machine Learning **42** (2001) 203–231

[10] Furnkranz, J., Flash, P.: An analysis of rule evaluation metrics. In: The Twentieth International Conference on Machine Learning (ICML'03), AAAI Press (2003) 202–209

[11] Elkan, C.: The foundations of cost-sensitive learning. In: Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01). (2001) 973–978

[12] Joshi, M.V., Agarwal, R.C., Kumar, V.: Predicting rare classes: can boosting make any weak learner strong? In: Proceedings of the eighth ACM international conference on Knowledge discovery and data mining, ACM Press (2002) 297–306