Learning Minority Classes in Unbalanced Datasets

TuBao Ho
Japan Advanced Institute of Science and Technology
Tatsunokuchi, Ishikawa
923-1292 Japan
bao@jaist.ac.jp

DucDung Nguyen
Japan Advanced Institute of Science and Technology
Tatsunokuchi, Ishikawa
923-1292 Japan
dungduc@jaist.ac.jp

Saori Kawasaki
Japan Advanced Institute of Science and Technology
Tatsunokuchi, Ishikawa
923-1292 Japan
skawasa@jaist.ac.jp

ABSTRACT

Mining prediction rules for a target class whose instances occur rarely in a large unbalanced data set has high practical significant. There have been attempts to deal with this problem, mainly by rebalancing the training data sets with up-sampling or down-sampling technique, or using cost sensitive classification. In this work we develop a method called LUPC that can learn minority classes from large unbalanced datasets with high performance. The key features of LUPC are the combination of separate-and-conquer rule induction with association rule mining, the search strategies associated with adaptive thresholds, and the using of the unbalanced dataset property in eliminating non-promising hypothesis. Comparative evaluations on some large datasets show that LUPC has high performance in learning minority classes, and it can also be applied to learn effectively all classes. Moreover, LUPC has ability to find patterns according to the user's interest.

1. INTRODUCTION

Learning minority classes in unbalanced datasets is a significant problem in both research and applications. It happens that in many domains the user wants to learn only one particular class, and this target class is represented by only a few examples while the others are represented by a large number. Examples of this learning problem are the detection of fraudulent telephone call [1], the detection of oil spills [5] or learning with biological data [8]. Facing with unbalanced datasets, common learning algorithm usually produce unsatisfactory classifiers. It may be caused by the fact that the fundamental assumption of the same class distribution and measures of learning performance built in these learning algorithms are not met or suitable in unbalanced datasets.

Learning minority classes in unbalanced datasets is among issues that were not much previously considered by the machine learning community, is now coming into light [3]. The main approaches to this problem include rebalancing the training data and cost sensitive classification with or without boosting. Naturally, rebalancing techniques include up-sampling of the minority classes [6] or down-sampling of the majority classes [5]. The cost sensitive classification is usually (though not strictly) applied to the divide-and-conquer approach [11][10]. Methods in this approach typically bias decisions in different directions, as if there were more or fewer cases in a given class, by raising or lowering the cost of a misclassification.

In this paper we introduce a method called LUPC (stands for Learning Unbalanced Positive Class) that can learn minority positive classes in unbalanced datasets with high performance. The main features of LUPC are a combination of separate-and-conquer rule induction [4] with association rule mining [1]. The strength of LUPC lies in its use of the property of unbalanced datasets and adaptive thresholds on accuracy and coverage of rules associated in its different search heuristics. Careful experimental comparative evaluations show that LUPC not only be well-suited in learning a target minority class but also it can learn all classes with high performance. We also illustrate some interesting results discovered from the somach cancer data by using the LUPC system with user's interest.

2. LEARNING UNBALANCED POSITIVE CLASS

2.1 Preliminaries

Let $A_1, A_2, \ldots A_p$ denote *attributes* with domains $dom(A_1)$, $dom(A_2)$, ... $dom(A_p)$ that can be categorical or numeric. The *instance space* X defined by $A_1, A_2, \ldots A_p$ is the Catersian product $dom(A_1) \times dom(A_2) \times \cdots \times dom(A_p)$. An *instance* is denoted by $< x, c(x) >$ where $x \in X$ and c is a function defined over X and has values in a finite set C of classes. In case of supervised data, values of c over X are given *a priori*. Consider the target class in C as the *positive* class C^+ and all other classes as the *negative* class C^-, we then have the training dataset as union of the positive instance subset (denoted by Pos) and the negative instance subset (denoted by Neg). We consider the case when C^+ is minority, i.e., $|Pos| \ll |Neg|$, and the problem is seen as *learning a minority class in a unbalanced dataset.*

A prediction rule for the positive class C^+ is defined as a conjunction of m attribute-value pairs $\bigwedge_{j=1}^{m}(A_{i_j} = v_{i_j}) \rightarrow C^+$ where $i_j \in \{1, 2, \ldots p\}$ and $v_{i_j} \in dom(A_{i_j})$. Denote by $cov(R)$ the set of all training instances covered by a rule

R. This set is divided into two subsets of covered instances in Pos and Neg, denoted by $cov^+(R)$ and $cov^-(R)$, that means $cov(R) = cov^+(R) \cup cov^-(R)$. For the purpose of prediction/classification, the task is to find a set of prediction rules for the target positive class, $R^+ = \{R_1^+, R_2^+, \ldots, R_q^+\}$, so that $Pos \subseteq \bigcup_{i=1}^{q} cov(R_i^+)$ and the discovered rules are "best" in terms of high sensitivity as well positive predictive value, and low false positive rate.

When evaluating the classifier performance, there will be some a positive instances correctly classified as positive, but some b positive instances incorrectly classified as negative. On the other hand, some c negative instances will be incorrectly classified as positive and some d negative instances will be correctly classified as negative. These quantities a, b, c, and d form the *confusion matrix*. Different functions using these four basic quantities have been used to measure performance of classifiers, typically $sensitivity = \frac{a}{a+b}$, $specificity = \frac{d}{c+d}$, $positive\ predictive\ value = \frac{a}{a+c}$, $negative\ predictive\ value = \frac{d}{b+d}$. Performance of learning from supervised datasets is customarily evaluated by $error\ rate = \frac{b+c}{a+b+c+d}$. However, the unbalance of datasets often hinders performance of standard classification methods. When checking the confusion matrix learned by standard methods, we can observe that classifiers induced from unbalanced datasets usually have false negative rate much higher than average error rate, and sensitivity as well positive predictive value are considerably lower than average accuracy.

The accuracy $acc(R)$ of a rule $R \in R^+$ is estimated by $acc(R) = \frac{|cov^+(R)|}{|cov(R)|}$. The error of rule R is defined as $err(R) = 1 - acc(R) = \frac{|cov^-(R)|}{|cov(R)|}$. This accuracy $acc(R)$ and the positive cover ratio $\frac{|cov^+(R)|}{|D|}$ in fact are *antecedent confidence* and *support* of the rule R in the terminology of association rule mining, respectively [1]. Association mining algorithms work with two given thresholds α and β on accuracy and cover ratio of rules, $0 \leq \alpha, \beta \leq 1$. A rule R is $\alpha\beta$-strong if $acc(R) \geq \alpha$ and $\frac{|cov^+(R)|}{|D|} \geq \beta$.

The following property that we skip the proof will be used to reduce time of scanning the large Neg in generating and selecting candidate rules for C^+. A candidate rule will be eliminated without continuing to scan though large set Neg if at some point this condition holds. Note that $cov+(R)$ can be quickly determined because the size of Pos.

Proposition 1. *Given threshold α, a rule R is not $\alpha\beta$-strong for any arbitrary β if*

$$cov^-(R) \geq \frac{1-\alpha}{\alpha} cov^+(R)$$

2.2 Algorithm LUPC

LUPC is a separate-and-conquer rule induction method that follows the generic separate-and-conquer scheme [4] with improvements to learn minority classes in unbalanced datasets. Firstly, it carries out a search biasing alternatively on accuracy and/or cover ratio with adaptive thresholds. Secondly, it focuses on doing separate-send-conquer induction in the target class with exploitation of the unbalanced property of datasets that allows trying the beam search with a large beam search parameter and one-sided selection. Additionally, it integrates prepruning and postpruning in a way that can avoid overpruning.

2.2.1 Basic ideas and search bias

As the computational quality of rules is measured by two independent measures of accuracy (antecedent confidence) and support, there is not any total order among rules that we can use to guide the search. However, if we fix a threshold for one measure (for example 90% for accuracy) and let the other varies, we can partially order the goodness of rules. An $\alpha\beta$-strong rule R_i is said better than an $\alpha\beta$-strong rule R_j with respect to α if R_i has support higher than that of R_j. Similarly, we define that an $\alpha\beta$-strong rule R_i is better than an $\alpha\beta$-strong rule R_j with respect to β if R_i has accuracy higher than that of R_j

The task of LUPC is not to find all rules satisfying given min_acc and min_cov but to find a subset of "best" rules that cover Pos. The iterative process of finding the best rule among $\alpha\beta$-strong rules and selecting promising non $\alpha\beta$-strong rules depends on the search heuristics specified by the user or by a default strategy. If both min_acc and min_cov are set to high values we can generally find high accuracy and cover ratio rules but they can often cover a small part of Pos. If both min_acc and min_cov are set with low values then discovered rules from the huge set of acceptable rules can cover Pos completely but they are often of high redundancy. LUPC distinguishes three alternatives that occur in practice and that lead to the three corresponding types of search heuristics:

1. *Bias on rule cover ratio.* It is to find sequentially rules with accuracy equal and greater than min_acc but the cover ratio is as large as possible. LUPC sets parameter α as the user specified min_acc and varies adaptive parameter β from highest possible value to the user specified min_cov.
2. *Bias on rule accuracy.* It is to find sequentially rules with cover ratio equal and greater than min_cov but accuracy is as large as possible. LUPC sets parameter β as the user specified min_cov and varies adaptive parameter α from highest possible value to the user specified min_acc.
3. *Alternative bias on rule cover ratio and accuracy.* In this case, LUPC starts with two highest values of α and β and alternatively reduced each of them when fixing the other and applying the corresponding procedure in the above two cases until reaching the stopping condition. This search heuristic is applied when the user do not specify any bias on accuracy and cover ratio.

2.2.2 The Algorithm

Figure 1 presents the scheme of algorithm LUPC that learns consequently a rule set from Pos and Neg given user-specified minimum accuracy threshold (min_acc) and minimum cover ratio (min_cov).

The procedure **Learn-positive-rule**

The procedure **Learn-positive-rule** starts with an empty $RuleSet$ (line 1) and two adaptive parameters α and β on

Learn-positive-rule$(Pos, Neg, min\_acc, min\_cov)$

1. $RuleSet = \phi$
2. $\alpha, \beta \leftarrow$ **Initialize**(Pos, min_acc, min_cov)
3. while $(Pos \neq \phi$ **and** $(\alpha, \beta) \neq (min\_acc, min\_cov))$
4. $\quad NewRule \leftarrow$ **BestRule**$(Pos, Neg, \alpha, \beta)$
5. $\quad$ if $(NewRule \neq \phi)$
6. $\quad\quad Pos \leftarrow Pos \setminus Cover^{+}(NewRule)$
7. $\quad\quad RuleSet \leftarrow RuleSet \cup NewRule$
8. $\quad$ else **Reduce**(α, β)
9. $RuleSet \leftarrow$ **PostProcess**$(RuleSet)$
10. return$(RuleSet)$

BestRule$(Pos, Neg, \alpha, \beta)$

11. $CandidateRuleSet = \phi$
12. **AttributeValuePairs**$(Pos, Neg, \alpha, \beta)$
13. while **StopCondition**$(Pos, Neg, \alpha, \beta)$
14. $\quad$ **CandidateRules**$(Pos, Neg, \alpha, \beta)$
15. $BestRule \leftarrow$ First $CandidateRule$ in $CandidateRuleSet$
16. return$(BestRule)$

Figure 1: The scheme of algorithm LUPC

rule accuracy and rule cover ratio, initialized by subroutine **Initialize** (lines 2). This subprogram specifies α or β given the thresholds min_acc, min_cov, and the user specified bias on accuracy or cover ratio. They are set as big as possible (1 for α and the maximum cover ratio of single attribute-value pair available on Pos) and always bigger than or equal to min_acc, min_cov. If the bias is on one accuracy, then β will be set to min_cov, and vise-versa. If the bias in on both accuracy and cover rate or there is no bias specified by the user, then both α or β are set as the biggest value. Lines 3-8 describe a recursive procedure to learn one the best rule among $\alpha\beta$-strong rules, to add it to the $RuleSet$, to remove positive instances covered by this rule under some conditions, and to change adaptively thresholds α and β.

If there are any instances remain in Pos, and α and β are still equal or greater than min_acc and min_cov (line 3), **Learn-positive-rule** calls the subroutine **BestRule** to learn a new rule that is "best" with respect to the user-specified search bias (line 4). If such a rule can be learned successfully (line 5), some positive instances covered by it will be removed from Pos (line 6) and the learned rule is added to the $RuleSet$ (line 7).

The instances removed from Pos, which are covered by the new rule, are those previously covered by $\delta - 1$ learned rules in $RuleSet$. If this rule is unsuccessfully learned, α and/or β will be adaptively reduced by the subroutine **Reduce** (line 8). The loop between lines 4-8 is repeated until the stopping condition (line 3) holds. The obtained $RuleSet$ can be optionally post-processed by **PostProcess**$(RuleSet)$ (line 9) before the procedure returns the final $RuleSet$ (line 10). The removing of only positive instances covered by new rule (line 6) is an *one-sided selection*. This one-sided selection differs from that in [5] as it removes instances from the minority target class satisfied constraints while leaving the majority class untouched. This strategy is an appropriate adaptation of the separate-and-conquer scheme in learning a minority class as it can keep the accuracy of candidate rules generated from Pos by **CandidateRules** as all information of the negative majority class is kept.

The procedure **Reduce**(α, β) gradually reduces α or β by ratio Δa and Δc that can be changed in applications. The default quantity Δa for reducing α is set as 2%. The default quantity Δc for reducing β is set as 1% of the biggest value of $cov^{+}(A_{ij} = v_{ij})$ for all available attribute-value pairs (A_{ij}, v_{ij}) in Pos.

The procedure **BestRule**

The procedure **BestRule** conducts a search in the rule space to find the "best" in the subset of generated $\alpha\beta$-strong rules given α and β. The **BestRule** is composed of the subroutine **AttributeValuePairs** (line 12) for determining the ordered set $AttributeValuePairSet$ of candidate attribute-value pairs to be used for generating candidate rules, and the subroutine **CandidateRules** (line 14) for determining the ordered set $CandidateRuleSet$ of candidate rules that is set empty at the beginning (line 11). The **CandidateRules** is repeated until **StopCondition** holds and returns the first $CandidateRule$ in the ordered $CandidateRuleSet$ as the $BestRule$.

The **AttributeValuePairs** determines candidate attribute-value pairs from instances remained in Pos as follows. Each attribute-value pair available on Pos is considered as condition part of a rule with a single premise whose conclusion part is C^{+}. This pair will be considered as a *candidate* attribute-value pair only if it covers more than $\alpha \times \beta \times D$ instances of the actual Pos (a necessary constraint for being an $\alpha\beta$-strong rule). All candidate attribute-value pairs will be then checked on Neg to see if some of them correspond to $\alpha\beta$-strong rules. Lastly, η candidate attribute-value pairs that form $\alpha\beta$-strong rules, ordered by their accuracy or cover ratio depending on the specified search bias, will be selected and added to the ordered set $AttributeValuePairSet$ where η is a parameter specified by the user. If the number of such attribute-value pairs is still less than η, the attribute-value pair corresponding to a non $\alpha\beta$-strong rule but with the highest accuracy or cover ratio will be selected. The process is repeated until obtaining η attribute-value pairs in $AttributeValuePairSet$ or no more attribute-value pairs can be considered.

The subroutine **CandidateRules** generates the ordered set $CandidateRuleSet$ of γ rules from $AttributeValuePairSet$ as follows. The order of rules depends on the specified search bias on their accuracy and/or cover ratio. The set $CandidateRuleSet$ in fact consists of two changeable parts, one contains ordered $\alpha\beta$- strong rules and the other contains ordered non $\alpha\beta$-strong rules. Of course, non $\alpha\beta$-strong rules with cover lower than β will no longer be considered in next steps and so cannot be considered to add to the non $\alpha\beta$-strong rule part. When **StopCondition** does not holds LUPC continues to improve $AttributeValuePairSet$ in the following way: (1) generating new rules by combining each non $\alpha\beta$-strong rules in $CandidateRuleSet$ with attribute-value pairs in $AttributeValuePairSet$, (2) if any of new generated rules becomes $\alpha\beta$- strong it will be inserted to the

Table 1: Comparing on the minority target class with misclassification costs

Data	Size	*Pos* (%)	Overall Accuracy			Sensitivity			Pos. Pred. Value		
			See5	See5C	Lupc	See5	See5C	Lupc	See5	See5C	Lupc
anneal	898	0.9	0.90	0.92	**0.94**	0.00	**0.50**	**0.50**	0.00	**1.00**	**1.00**
glass	214	4.2	0.69	0.49	**0.79**	0.33	0.00	**1.00**	0.33	0.00	**0.60**
headache	10,000	10.0	**0.90**	0.84	0.86	0.52	0.53	**0.86**	**0.83**	0.39	0.76
hypothyroid	3,613	4.1	0.93	0.97	**0.98**	**0.98**	**0.98**	**0.98**	**0.89**	0.65	0.69
inf	2,380	1.1	**0.95**	0.94	0.86	**1.00**	0.76	**1.00**	**1.00**	**1.00**	**1.00**
satellite	6,435	9.7	0.83	0.74	**0.84**	0.57	0.64	**0.68**	**0.71**	0.62	0.62
segmentation	2.070	14.1	**0.94**	0.93	0.83	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
smoking	28,550	5.2	**0.87**	0.67	0.77	0.49	**0.63**	0.45	0.92	0.27	**0.95**
sick	2,800	6.1	**0.98**	0.96	0.97	0.77	**0.84**	0.81	**0.88**	0.71	0.72
flare	1,666	3.6	**0.81**	0.79	**0.81**	0.40	0.13	**0.47**	0.75	**1.00**	**1.00**
education	10,000	6.1	**1.00**	0.97	0.99	**1.00**	**1.00**	**1.00**	**1.00**	0.95	0.98
adult	16,281	23.6	**0.86**	0.81	**0.86**	0.59	**0.86**	0.59	**0.78**	0.57	0.75
stomach	6,773	4.5	**0.69**	0.51	0.66	0.01	0.00	**0.12**	0.25	0.00	**0.27**
Avg			**0.88**	0.82	0.87	0.60	0.59	**0.72**	0.71	0.63	**0.80**

Table 2: Comparing on minority target class with up and down sampling

Data	Sensitivity			Pos. Pred. Value		
	See5up	See5down	Lupc	See5up	See5down	Lupc
anneal	**1.00**	**1.00**	0.50	**1.00**	0.33	**1.00**
glass	0.33	**1.00**	**1.00**	0.33	**1.00**	0.60
headache	**0.94**	0.88	0.86	0.61	0.52	**0.76**
hypothyroid	**1.00**	**1.00**	0.98	**0.74**	0.46	0.69
inf	**1.00**	**1.00**	**1.00**	**1.00**	0.78	**1.00**
satellite	0.61	**0.76**	0.68	0.58	0.53	**0.62**
segmentation	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
smoking	**0.98**	**0.98**	**0.45**	0.55	0.34	**0.95**
sick	0.82	**0.84**	0.81	**0.72**	**0.72**	**0.72**
flare	**0.80**	0.71	0.46	0.48	0.33	**1.00**
education	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	9.98
adult	0.84	**0.87**	0.59	0.59	0.56	**0.75**
stomach	0.28	**0.63**	0.12	0.16	0.14	**0.24**
Avg	0.82	**0.86**	0.72	0.68	0.60	**0.80**

ordered part of $\alpha\beta$-strong rules in $CandidateRuleSet$. The **StopCondition** holds if one of the following constraints is matched: (1) there is no change in non $\alpha\beta$-strong rules in the previous run of **CandidateRules**, (2) there are already γ $\alpha\beta$-strong rules in $CandidateRuleSet$.

The subroutine **CandidateRules** may require a lot of checks on Neg to see if a generated candidate rule is $\alpha\beta$-strong. Fortunately, thanks to the property in Proposition 1, many candidate rules are quickly rejected if they are found to match the condition $cov^-(R) \geq \frac{1-\alpha}{\alpha} cov^+(R)$ during the scan of Neg. It is easy to count $cov^+(R)$ for each candidate rule R as Pos is small, and we need only to accumulate the count of $cov^-(R)$ when scanning Neg until either we can reject the candidate rule as the constraint holds or we completely go throughout Neg and find the rule has a satisfied accuracy.

Two parameters η and γ can influence on the findings of LUPC. Generally, the higher value η and γ the higher chance to discover better rules.

3. EVALUATION

This section reports our experimental comparative evaluation of LUPC. It aims at evaluating the performance of LUPC on (i) learning one target minority class in a unbalanced dataset; and (ii) learning all classes as a classification method.

After inducing a set of rules, LUPC matches testing instances (or unknown instances) in the following way: If a testing instance matched rules of more than one class, LUPC then predicts its class label by the majority vote weighted by the rule confidence. If a testing instance does not match any rule, it will be classified into the default class that is taken as the class of the majority among unclassified training instances.

3.1 Performance in Learning Minority Classes

We carried out an comparative evaluation of performance in learning one target minority class from a unbalanced dataset between LUPC and other methods using cost-sensitive and rebalancing techniques. The aims is to show that LUPC can improve the sensitivity and positive predictive value when learning minority classes and competes with other techniques.

The experiments were designed as follows. LUPC is compared with See5, the PC commercial descendent of C4.5 [9], and See5 using cost-sensitive and rebalancing (up and down) techniques. The experiments were done on 13 datasets containing minority classes. These datasets include 12 UCI datasets (in parentheses are names of the positive classes): anneal (1), glass (tableware), headache (2), hypothyriod

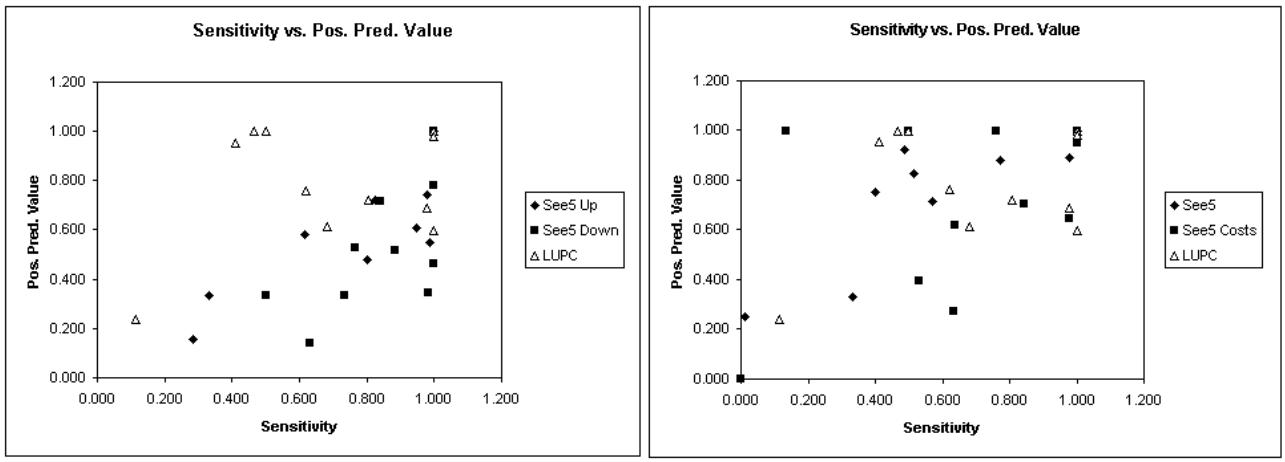

Figure 2: Scatter plot of sensitivity vs. positive predictive value.

(hypothyriod), inf (1), satellite (4), segmentation (sky), smoking (1), sick (sick), flare (F), education (2), adult (>50K), and the stomach cancer dataset (dead<90days) collected at the National Cancer Center in Tokyo. The first three columns of Table 1 summaries the name of datasets, the size of datasets, and the percentage of size of the target minority class in each dataset. Each dataset is randomly divided into a fixed stratified training data (2/3) and a testing data (1/3) for the *common* use in See5 and LUPC.

There different ways to assign costs of misclassification. We tried three common ones and chosen the following that gave the best performance of See5 with cost sensitive: the cost of misclassifying members of class C_i to class C_j is set as $\frac{|C_i|}{\sum_k |C_k| - |C_j|}$. For comparing LUPC with rebalancing techniques, we run See5 on each training dataset and its two derived ones by down-sampling and up-sampling. For a fair comparison, See5 and LUPC are run on all datasets with their default parameters: See5 with min CF = 25%, $min\_cover = 2$, and LUPC with $min\_acc = 80\%$, $min\_cover = 2$, the number of candidate attribute value pairs $\eta = 100$ and number of candidate rules $\gamma = 20$. it is easy to try LUPC

Table 1 shows experimental results concerning cost sensitive learning. Columns 4-6 present the overall accuracy on all classes obtained by each method. Columns 7-9 and 10-12 present the sensitivity and the positive predictive value, respectively, of each method on the minority positive class.

Table 2 shows experimental results concerning rebalancing techniques. Columns 2-4 present the sensitivity obtained on the minority positive class by LUPC and See5 with up and down sampling. Columns 5-7 present the positive predictive value obtained on the minority positive class by LUPC and See5 with up and down sampling.

Recall that the sensitivity is probability that a positive instance will be classified to the positive class (recall), and the positive predictive value is probability that an instance is positive if it is classified to the positive class (precision).

The following conclusions can be drawn:

- Columns 7 and 9 in Table 1 show that the overall accuracy of See5 and LUPC are very comparable (and they are both somehow higher than that of See5 with misclassification costs, column 8), but columns 7, 10, 13 show that the sensitivity and positive predictive value in minority target classes are decreased considerably with See5 (as with some other classification systems in our experiments). It is particularly clear in some datasets such as stomach cancer data (stomach).
- Columns 10-12, 13-15 in Table 1 show that See5 with costs improves the sensitivity in minority target classes but it is not clear for positive predictive value. LUPC, however, achieves in many domains sensitivity and positive predictive value higher than those of See5 and See5 with costs.
- Columns 2-4 in Table 2 show that the rebalancing techniques can generally give sensitivity higher than those of LUPC, and down sampling attains somehow sensitivity for minority target class higher up sampling in many domains, but vise-versa for positive predictive value. Columns 5-7 show that LUPC can generally achieve a positive predictive value higher than that of See5 with up or down sampling in the minority target class. LUPC somehow can improve these two measures and balance well these two measures.
- From both Table 1 and Table 2 we can see that LUPC deals well with a minority target class in comparison with cost sensitive learning and rebalancing techniques

Figure 2 gives intuitively views on sensitivity versus positive predictive value of LUPC and See5Cost (left), LUPC and See5 with up-sampling (right) by the scatter plot of these values taken from Table 1 and Table 2, respectively.

3.2 Performance in Learning All Classes

The task of the experiments is to show that although it is developed to learn minority classes in unbalanced datasets

Table 3: Comparison of accuracy in learning all classes

Data	Lupc		See5		See5Rules		CBA	
anneal	96.68	± 2.68	91.86	± 0.7	92.78	± 0.8	**97.9**	± 1.32
australian	82.34	± 4.42	84.6	± 1.28	**86.52**	± 1.2	85.94	± 4.19
auto	80.10	± 13.6	**81.20**	± 2.62	79.46	± 2.48	75.60	± 9.4
breastcancer	94.57	± 2.74	95.04	± 0.6	**95.96**	± 0.68	94.4	± 6.23
cleve	**83.77**	± 6.24	78.6	± 2.36	80.08	± 2.24	81.1	± 6.71
crx	84.37	± 4.32	86.8	± 1.08	**86.94**	± 1.16	83.91	± 4.65
diabetes	**78.55**	± 4.41	78.18	± 1.22	76.56	± 5.2	75.61	± 5.51
german	**73.9**	± 3.76	72.58	± 1.02	72.46	± 1.32	73.58	± 4.36
glass	75.74	± 14.7	76.0	± 2.56	74.02	± 3.18	**77.48**	± 6.09
heart	**84.82**	± 2.18	81.26	± 2.74	82.82	± 2.28	82.58	± 10.87
hepatitis	85.3	± 8.19	79.74	± 2.84	83.5	± 2.5	**85.58**	± 7.18
horse	81.4	± 4.37	85.16	± 1.86	**85.48**	± 1.66	82.00	± 3.79
hypothyroid	97.79	± 0.93	**99.2**	± 0.16	99.18	± 0.16	99.05	± 0.61
ionosphere	92.23	± 3.7	89.74	± 1.86	91.5	± 1.46	**93.12**	± 3.88
iris	93.33	± 6.4	**93.44**	± 1.66	94.0	± 1.44	93.25	± 7.12
labor	89.25	± 13.36	83.58	± 5.62	84.14	± 4.76	**94.99**	± 8.07
pima diabetes	**78.5**	± 3.53	77.94	± 1.32	76.8	± 1.22	75.35	± 4.82
sick	97.11	± 1.81	97.92	± 0.26	**97.98**	± 0.26	96.92	± 1.84
sonar	**83.2**	± 7.46	81.12	± 2.43	81.12	± 2.2	80.22	± 9.76
vehicle	70.71	± 3.44	71.6	± 1.32	71.22	± 1.38	**72.67**	± 3.28
waveform	**84.55**	± 1.45	75.48	± 0.52	77.86	± 0.52	81.88	± 0.99
wine	97.82	± 3.15	92.66	± 1.94	85.62	± 1.33	**98.33**	± 3.75
KDD'99(10%)	99.14	± 0.07	**99.9**	± 0.00	**99.9**	± 0.00	-	± -
Avg	**86.31**	± 5.08	84.94	± 1.65	85.04	± 1.71	85.52	± 5.20

LUPC can also achieve performance comparable to the other methods.

We do an experimental comparative evaluation of four systems LUPC, See5, See5Rules and CBA (Liu et al., 1998, http://www.comp.nsu.edu.sg/~dm2) [7] on 22 UCI datasets and the KDD'99 cup data set (10%), some of them are unbalanced as used in the previous experiments, and others are not necessarily unbalanced. The datasets include anneal, Australian, auto, Wisconsin breast cancer (breastcancer), cleve, cxr, diabetes, german, glass, heart disease (heart), hepatitis, horse, ionosphere, iris, labor, pima diabetes, sick, sonar, vehicle, waveform, and wine. The experiments were carried out with 10-fold stratified cross validation. Note that the accuracy of a classification system estimated by stratified cross validation is a random variable, and its value varies with different trials. To have a good estimation of accuracy, we run 10 times of stratified cross validation of See5 and LUPC for each dataset, and take their average as the estimation of accuracy. Though CBA does not make a random partition of a dataset into subsets, and we could not obtain a better estimation of its accuracy on each domain, the objective of our experiments is to show that LUPC can achieve a very comparable performance evaluated on all classes as these well-known classification systems.

Table 3 summarizes the accuracy of four systems on these 22 domains. It can be observed that these systems have a comparable performance. The accuracy of LUPC and CBA are lightly higher than those of See5 and See5Rules but See5 and See5Rules are somehow more stable standard deviation.

3.3 Mining interested patterns with LUPC

The stomach cancer dataset collected at the National Cancer Center in Tokyo during the period 1962-1991 is a very important resource for research. It contains data of 7,520 patients described originally by 83 numeric and categorical attributes. These include information on patients, symptoms, type of cancer, longitudinal and circular location, serosal invasion, metastasis, pre-operative complication, post-operative complication, etc. One problem to be investigated is to use attributes containing patient information before operation to predict the patient status after the operation. The domain experts are particularly interested in finding predictive and descriptive rules for the class of patients who "dead within 90 days" after operation among totally 5 classes.

It is common known that the patients will be died when liver metastasis occur aggressively, and consequently, other learning methods when applied to this datasets yield most rules for class "dead within 90 days" containing "Liver_metastasis" they are considered acceptable but nut useful by domain experts. Also, these discovered rules do not cover the whole set of patients of this class. It means that a number of patients of the class are not concerned with "liver_metastasis" and they are difficult to be detected.

Using visual interactive LUPC, we mined rules with specified constraints to find only rules that do not contains the characterized attribute "Liver_metastasis" and/or its combination with other two typical attributes "Peritoneal_metastasis", "Serosal_invasion". Below is a rule with accuracy 100% discovered by LUPC that can be seen as rare and irregular in the class.

Rule 8:	accuracy = 1.0 (4/4), cover = 0.001 (4/6712)
IF	category = R AND
	sex = F AND
	proximal_third = 3 AND
	middle_third = 1
THEN	class = death within 90 days

In KDD the prediction of rare events is coming to be of particular interest. LUPC allow us examine effectively the

hypothesis space and identify the rule even its support or confidence are very small. An example is to find rules in class "alive" that contain the symptom 'lever_metastasis". Such events are certainly rare and influence the human decision making. We found events in the class "alive" such as male patients getting "liver_metastasis" at serious level 3 can still survive with the accuracy of 50%.

Rule 1: accuracy = 0.500 (2/4); cover = 0.001(4/6712)

IF sex = M AND
type = B1 AND
Liver_metastasis = 3 AND
middle_third = 1
THEN class = alive

3.4 Discussion

There are two main reasons among others that make LUPC performs well the task of learning positive class in unbalanced datasets.

First is the focus of LUPC on learning positive class and its exploitation of the unbalanced dataset property in different ways: (1) The thresholds on accuracy and cover ratio are determined adaptively to the "hardness" and size ratio of the positive class during the learning process. This contributes to the avoiding of the phenomenon that small classes are blocked by big classes. (2) LUPC selects candidate attribute-value pairs and generates rules from positive instances and uses negative instances to evaluate and select these candidates. The necessary constraint on attribute-value pairs to cover more than $\alpha \times \beta \times D$ instances in Pos can be verified efficiently. The constraint on $\alpha\beta$-strong rules allows LUPC to eliminate considerably candidate rules as $cov^+(R)$ is often small. Recall that it has been experimentally verified that generated rules discovered by beam search with CN2 (Clark & Niblett, 1989) are often superior to those found by an exhaustive best-first-search. (3) The conditional removal of only positive instances covered by learned rules (one-sided selection) allows LUPC to reduce errors of rules found in further steps as all information on negative class is kept. This cannot be done when learning classes all together. RIPPER [2] learns classes in the order of their prevalence but differs from LUPC as it removes all instances covered by each learned rule.

Second is the effective search of LUPC based on its combination of separate-and-conquer learning and association rule mining. Along the process of separate-and-conquer induction, LUPC changes adaptively thresholds α and β according to the remaining positive instances so that in each step the best rule is mined in a promising part of the rule space. The chosen heuristics plays an important role in allowing LUPC to vary thresholds on accuracy and cover ratio alternatively from maximum values until the user specified min_accuracy and min_coverage in order to discover a rule as good as possible in each step. This feature is different from other separate-and-conquer induction methods. Moreover, LUPC does not generate all possible rules as CBA in its first phase [7], and this makes LUPC applicable to large datasets.

4. CONCLUSIONS

We have introduced the method LUPC developed to learn target minority classes from unbalanced datasets. LUPC is a separate-and-conquer rule induction method using dynamic multiple thresholds on accuracy and cover ratio and the property of unbalanced datasets. Experimental comparative evaluation show that LUPC can learn well a target minority class as well all classes.

5. REFERENCES

[1] I. T. Agrawal, R. and A. Swami. Mining association rules between sets of items in large databases. In *Inter. Conf. Management of Data SIGMOD'93*, pages 207–216.

[2] W. Cohen. Fast effective rule induction. In *Twelfth Inter. Conf. on Machine Learning, (San Francisco: Morgan Kaufmann)*, pages 115–123.

[3] P. F. Fawcett, T. Combining data mining and machine learning for effective user profiling. In *Inter. Conf. on Knowledge Discovery and Data Mining KDD'96, (ACM Press)*, pages 8–13.

[4] J. Furnkranz. Separate-and-conquer rule learning. In *Journal Artificial Intelligence Review*, pages 3–54.

[5] M. Kubat and S. Marvin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proc. of the Fourteenth Inter. Conf. on Machine Learning*, pages 179–186.

[6] C. Ling, C. X. Li. Data mining for direct marketing: Problems and solutions. In *Inter. Conf. on Knowledge Discovery and Data Mining KDD-97*, pages 258–267.

[7] H. W. M. Y. Liu, B. Integrating classification and association rule mining. In *Fourth Conf. on Knowledge Discovery and Data Mining, (New York: ACM)*, pages 80–86.

[8] B. C. H. S. A. Muggleton, S. H. Measuring performance when positive are rare: Relative advantage versus predictive accuracy – a biological case study. In *Proc. of European Conf. on Machine Learning*, pages 300–312.

[9] J. Quinlan. C4.5: Programs for machine learning, (san francisco: Morgan kaufmann).

[10] K. M. Ting. A comparative study of cost-sensitive boosting algorithms. In *Seventeenth Inter. Conf. on Machine Learning*, pages 983–990.

[11] P. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. In *Journal of Artificial Intelligence Research 2*, pages 369–409.