Taking class importance into account

José-Luis Polo, Fernando Berzal and Juan-Carlos Cubero

Department of Computer Science and Artificial Intelligence.
University of Granada. 18071 Granada. Spain
E-mail: {jlpolo|fberzal|JC.Cubero}@decsai.ugr.es

Abstract

In many classification problems, some classes are more important than others from the users' perspective. In this paper, we introduce a novel approach, weighted classification*, to address this issue by modeling class importance through weights in the [0,1] interval. We also propose novel metrics to evaluate the performance of classifiers in a weighted classification context. In addition, we make some modifications to the ART classification model [3] in order to deal with weighted classification.*

1. Introduction

Classification is an extensively studied problem in Machine Learning research. Despite this, many classification problems exhibit specific features that render most classification models ineffective. Several models have been proposed to deal with class attribute peculiarities:

- In *imbalanced classification* [1] [9] problems, some classes are supported by a very low number of examples. Unfortunately, traditional classification models tend to ignore such classes, no matter what their importance is.
- *Cost-sensitive classification* [7] [6] [5] models take into account misclassification costs. These models are useful when the cost of a false positive is not the same for every class.
- *Subgroup discovery* [10]: In this case, there is only a class that is important for the expert. The aim of subgroup discovery is finding the most interesting subgroups of examples according to statistical criteria.

In this paper, we focus on a class attribute feature that is ignored by traditional classification models: the relative importance of each class.

2. The weighted classification problem

Each class in a classification problem may have a different degree of importance. In some situations, the user might be interested in achieving the maximum possible accuracy for specific classes while keeping the classification model complexity to a minimum, even at the cost of lower accuracy for less important classes. She might also desire a minimum model complexity while preserving a reasonable accuracy level, even for the most important classes. Moreover, different users could attach different importance degrees to each class depending on their personal goals even for the same problem.

Therefore, we need classification inducers that take class importance into account when building classification models. In order to represent the relative importance of each class, we can resort to *relative weights* w_i for each class. For the sake of simplicity, we will assume that the weights w_i are floating-point values between 0 and 1. These values can always be normalized.

In this work, we focus on getting classification models as simple as possible for the important classes without penalizing classification accuracy. If we were only interested in classification accuracy, existing models could have been used. In particular, we could have used a cost-sensitive model [7] by defining a cost matrix, which would have reflected the relative importance of each class. However, classifier complexity is also a fundamental issue in supervised learning, since complexity is closely related to interpretability. A classifier might be useless from a practical point of view if it reaches a good accuracy level but is too complex to be understood by the decision makers who need a rationale behind their decisions.

In particular, *weighted classification models* (that is, classification models built by taking class weights into account) can be useful in situations such as the ones described by the following examples from the UCI Machine Learning repository [4]:

2006 International Conference on Hybrid Information Technology (ICHIT'06)
0-7695-2674-8/06 $20.00 © 2006 IEEE

- **Extreme classes problems:** In some problems, experts could be specially interested in properly classifying 'extreme' classes, i.e. classes whose importance is paramount in the decision making process. For example, when dealing with the CAR data set, we could be interested in getting clear rules for good cars in order to recommend them (and for bad ones in order to avoid them).

  It should be noted that cost-sensitive classification could have also been used in this problem, since the expert might be interested in not mistaking a bad car for a good one, regardless of the required classifier complexity.

- **Two-class problems:** In binary classification problems, it is relatively common for the proper description of one class to be much more important than the other's for providing the rationale behind a given decision. For example, in the ADULT data set, where the class attribute is personal income, with values >50K and <=50K, a tax inspector might be more interested in people who earn more money in order to perform a financial investigation.

- **Classes and ontologies:** When the classes in a classification problem can be organized somehow, we can also resort to importance degrees in order to focus on related classes that might be specially relevant for the user. For example, a use hierarchy can be defined for the 6-class GLASS data set: three kinds of glass are used to make windows (one for vehicle windows, two for building windows), while the other three have other applications (containers, tableware, and headlamps). If we were interested in identifying glass from a broken window, we could assign high importance degrees to all the kinds of glass used to make windows.

After the definition of weighted classification and the study of potential application areas, we face the problem of evaluating weighted classification models. In this paper, we propose two metrics that take into account class importance. They might be helpful when evaluating the accuracy and complexity of weighted classifiers:

- Classifier accuracy is the main goal of any classification system. In weighted classification problems, we recommend the use of the following **weighted accuracy** measure:

$$wAcc = \sum_{i=1}^{\#classes} w_i \cdot acc(i) \quad (1)$$

  where *acc(i)* is the average accuracy for the *i*-th class, w_i is the weight for the *i*-th class, and $\#classes$ is the number of classes.

- We also propose an analogous **weighted complexity** measure for evaluating classifier complexity, which is closely related to its understandability and interpretability:

$$wOpacity = \sum_{i=1}^{\#classes} w_i \cdot opacity(i) \quad (2)$$

  where *opacity(i)* is the value of the complexity measure for the *i*-th class. For instance, *opacity(i)* might represent the average depth for nodes belonging to the *i*-th class in a decision tree. In this case, the complexity measure for *i*-th class can be defined as follows:

$$opacity(i) = depth(i) = \frac{\sum_{x \in class(i)} level(x)}{freq(i)} \quad (3)$$

  where $level(x)$ is the depth of the leave corresponding to the example x, $freq(i)$ is the number of examples belonging to *i*-th class, and $class(i)$ is the set of examples belonging to the *i*-th class.

A weighted classifier should be evaluated according to these measures. An optimal classifier would optimize all of them at the same time, although this multi-objective optimization is not always possible, so we will usually have to achieve a trade-off between accuracy and complexity.

3. Adapting ART for weighted classification

In this paper, we show how standard classification models can be adapted for dealing with class weights. In particular, we focus on the ART classification model [3]. ART, which stands for Association Rule Tree, is a Separate and Conquer algorithm that is suitable for Data Mining applications because it makes use of efficient association rule mining techniques.

The special kind of decision list ART obtains can be considered as a degenerate, polythetic decision tree. The ART algorithm outline is shown in Figure 2. Unlike traditional TDIDT algorithms, ART branches the decision tree by simultaneously using several attributes.

Internally, ART makes use of association rules in order to find good descriptions of class values. When evaluating candidate rules, the classical confidence measure used in association rule mining is employed to rank the discovered rules, even though alternative criteria might be used [2].

Once ART discovers potentially useful classification rules, they are grouped according to the attributes in their antecedents, as shown in Figure 1. A rule selection mechanism is also necessary for choosing one of the resulting rule groups. The chosen group is used to branch the decision

$$\{A_3\&B_2 \rightarrow C_1,\ B_4\&C_2 \rightarrow C_2,\ A_4\&B_1 \rightarrow C_2,$$
$$B_0\&C_1 \rightarrow C_1,\ A_2\&C_2 \rightarrow C_1\}$$
$$\Downarrow$$
$$\{A_3\&B_2 \rightarrow C_1,\ A_4\&B_1 \rightarrow C_2\}$$
$$\{B_4\&C_2 \rightarrow C_2,\ B_0\&C_1 \rightarrow C_1\}$$
$$\{A_2\&C_2 \rightarrow C_1\}$$

Figure 1. Grouping rules with compatible antecedents

tree and the whole process is repeated for the remaining examples.

In the following sections, we propose some modifications to the rule evaluation and rule selection criteria used by the ART algorithm in order to deal with weighted classification problems.

3.1. Weighted selection criterion

As we have mentioned before, a selection criterion is needed when alternative sets of rules are considered good enough to branch the tree. In ART, the criterion is based on the support of the rules belonging to the group. The best set of rules is the set that covers the maximum number of examples. However, this approach does not take the weights of the rules into account. We propose a modified criterion, which we call weighted coverage:

$$weightedCoverage(RuleSet) = \sum_{r \in RuleSet} support(r) \cdot w(r)$$

where $w(r)$ is the weight of the class in the consequent of the rule r and $support(r)$ is the number of examples supporting the rule r. In some sense, this is similar to the idea used in boosting algorithms such as AdaBoost [8].

By using weighted coverage, the set of rules that cover a larger number of more important classes is preferred over other sets. Since such a set will be selected as soon as possible, its level in the tree will tend to be lower and, therefore, the classifier opacity is expected to be reduced.

3.2. Weighted rule evaluation criterion

ART uses confidence as rule evaluation criterion by default. A rule is a good or valid rule when its confidence value is above a threshold that is determined by the minimum desirable confidence value and a tolerance margin. To be precise, a rule r is valid when:

$$evaluate(r) \geq MinEval - Tolerance$$

function ART (data, MaxSize, MinSupp, MinEval): classifier;
// **data**: *Training dataset*
// **MaxSize**: *Maximum LHS itemset size*
// *(default value = 3)*
// **MinSupp**: *Minimum support threshold*
// *(default value = 0.05 = 5%)*
// **MinEval**: *Minimum desirable rule evaluation value*
// *(1.0 confidence threshold by default)*

k = 1; *// LHS itemset size*
list = **null**; *// Resulting decision list (degenerate tree)*

while ((list **is null**) **and** ($k \leq$ MaxSize))

// Rule mining
Find all the confident rules from input data with
k items in the *LHS* and the class attribute in the *RHS*
taking the rule evaluation tolerance into account
e.g. $\{A_1.a_1$ .. $A_k.a_k\} \Rightarrow \{C.c_j\}$

if there are candidate rules to grow the list

// Rule selection
Select the best set of rules with the same set of attributes
$\{A_1..A_k\}$ in the *LHS* according to the selection criterion.

// Tree branching
list = List resulting from the selected rules
$\{A_1.a_1$ .. $A_k.a_k\} \Rightarrow \{C.c_j\}$, where
all training examples not covered by the selected
association rules are grouped into an 'else' branch
which is built calling the algorithm recursively:
data = uncovered data *// Transaction trimming*
MinEval = $max_{r \in list}$ evaluate(r)
list.else = ART (data, MaxSize, MinSupp, MinEval);

else
k = k + 1;

if list **is null** *// no decision list has been built*
list = default rule labelled with the most frequent class;

return list;

Figure 2. ART Algorithm Outline

where *evaluate(r)* represents the rule quality according to the chosen evaluation criterion.

The key idea behind the adaptation of ART rule evaluation criterion to weighted classification is adjusting the tolerance margin according to the class of the rule we evaluate. Here we discuss three heuristics we have devised while trying to solve this problem. The experimental results we have obtained with them can be found in Section 4.

Tolerance Reduction (TR). If we only accept rules for the less important classes when their accuracy is very high, rules for the important classes will be more likely to be selected. This idea is expressed in the next formula:

$$tol_{TR} = tol \cdot (1 - \frac{MaxWeight - w}{MaxWeight})$$

where *w* is the weight for the class of the rule being evaluated and *MaxWeight* is the maximum weight of the classes in the classification problem at hand. In the expression above, *tol* represents the original value for the tolerance in ART (0.1 by default) and tol_{TR} is the adjusted tolerance.

Using this heuristic criterion, when w equals the maximum weight, $tol_{TR} = tol$. That is, the original tolerance margin is preserved for the most important class. However, the tolerance margin will be reduced for any $w < MaxWeight$.

Relative Weight Premium (RWP). In this case, the tolerance margin will be increased according to the importance of the class. If the class is not important, the tolerance margin will be similar to the one used in ART. When the class is important, the tolerance margin will be higher, thus ensuring that more rules corresponding to important classes will be considered during the classifier construction.

$$tol_{RWP} = tol \cdot (1 + \frac{w - MinWeight}{MaxWeight - MinWeight})$$

When w equals to the minimum weight, $tol_{RWP} = tol$. If w corresponds to the maximum weight, the fraction becomes 1 and $tol_{RWP} = 2 \cdot tol$. For any value between MinWeight and MaxWeight, the tolerance will be proportionally increased.

Weight Premium (WP). RWP increases the tolerance margin for important classes. However, it drives tolerance to the same values $[tol, 2 \cdot tol]$ regardless of the particular weights chosen by the user.

For instance, if there is a 2-class problem, the tolerance margin becomes $2 \cdot tol$ for the most important class and tol for the less important class using RWP. These margins will be the same no matter if our weights are (0.9, 0.1) or (0.6, 0.4).

Table 1. Data sets used to evaluate ART in weighted classification.

Dataset	Size	Attributes	Classes
ADULT	48842	15	2
AUSTRALIAN	690	15	2
CAR	1728	7	4
CHESS	3196	36	2
GLASS	214	9	6
HAYESROTH	160	5	3
HEART	270	14	2
IRIS	150	5	3
MUSHROOM	8124	23	2
NURSERY	12960	9	5
PIMA	768	9	2
SPLICE	3175	61	3
TICTACTOE	958	10	2
TITANIC	2201	4	2
VOTES	435	17	2
WAVEFORM	5000	22	3
WINE	178	14	3

We can easily modify the RWP criterion if we normalize with respect to the maximum weight in absolute terms:

$$tol_{WP} = tol \cdot (1 + \frac{w - MinWeight}{MaxWeight})$$

Using this heuristics, the tolerance margin in the (0.9, 0.1) case becomes $(1.88 \cdot tol, tol)$ whereas in the (0.6, 0.4) case it becomes $(1.33 \cdot tol, tol)$.

4. Experimental results

In this section, we evaluate the performance of our modified ART algorithm by performing standard ten-fold cross-validation experiments on several data sets available from the UCI Machine Learning repository [4]. The data sets we have used in our experimentation are summarized in Table 1

We have used two kinds of weight distributions to determine the effect of a particular set of weights on classifier accuracy and complexity:

- First, we have performed experiments using extreme values for the class weights. We have selected a single class as the truly important class and we have made the others unimportant. We have assigned a 0.1 weight for all the unimportant classes and we have set the weight of the important class so that the sum of the weights equals 1. For instance, we have tested a weight distribution of (0.9, 0.1) for 2-class problems, (0.8, 0.1, 0.1) for 3-class problems, and so on.

- We have also tested a different kind of class weight distribution. Again, we have considered one class to be more important than the others but, in this case, the difference between the weight of the important class and the weight of the unimportant classes is not so extreme. Table 2 shows the particular weight distributions we have used.

Table 2. Non-extreme values for the weights

Number of classes	Weights
2	(0.7, 0.3)
3	(0.6, 0.2, 0.2)
4	(0.4, 0.2, 0.2, 0.2)
5	(0.5, 0.125, 0.125, 0.125, 0.125)
6	(0.4, 0.12, 0.12, 0.12, 0.12, 0.12)

For each data set in Table 1, we have performed $2 \cdot n_{ds}$ experiments, where n_{ds} is the number of classes in the data set. In each pair of experiments, we have selected a particular class as the important class and we have performed two cross-validation experiments, using extreme and non-extreme class weights.

Table 3 shows the experimental results we have obtained using the extreme weight distributions.

Each row in the table summarized the results obtained from a particular combination of heuristics. Each column in the table shows the overall average of each one of the measures we have used to evaluate the classifiers, as well as the number of times a particular heuristic combination matches or improves the standard ART algorithm (out of the 48 individual experiments performed for each heuristic combination).

The *wAcc* and *wOpac* measures correspond to the weighted accuracy and complexity metrics introduced in equations 1 and 2. The *Acc_mic* measure stands for the classifier accuracy with respect to the most important class, something that could be specially relevant to check the bias weights introduce in the learning algorithm. In a similar way, *Opac_mic* represents the average classifier opacity for the most important class (i.e. the average depth of the nodes corresponding to the important class in the resulting decision trees). Finally, *Acc* represents the standard cross-validation classifier accuracy, while *Opac* is the average classifier opacity (i.e. the average decision tree depth), both without taking class weights into account.

The experiments show that the best results with respect to classifier complexity (about 20% improvement) are obtained by the *RWP* heuristics when combined with the standard coverage rule selection criterion.

It is important to emphasize that we achieve a 20% reduction in complexity without significantly penalizing the

Table 3. Experimental results obtained using extreme values for the class weights.

	wOpac	wAcc	Acc_mic	Opac_mic	Acc	Opac
Standard ART	5.86	78.28	78.28	5.86	**83.87**	5.33
Confidence + WCoverage	5.90 (36)	78.19 (40)	78.23 (41)	5.78 (36)	83.82 (43)	5.63 (21)
TR + Coverage	5.81 (34)	**80.17 (30)**	**81.22 (32)**	5.69 (37)	83.22 (31)	5.33 (25)
TR + WCoverage	6.06 (34)	80.13 (30)	80.94 (34)	5.79 (38)	82.99 (33)	5.95 (22)
RWP + Coverage	**4.68 (46)**	78.26 (35)	77.89 (34)	**4.46 (47)**	83.21 (35)	**4.58 (45)**
RWP + WCoverage	4.89 (45)	78.16 (34)	77.94 (36)	4.56 (46)	82.91 (32)	5.12 (38)
WP + Coverage	4.87 (46)	78.80 (42)	78.79 (40)	4.66 (47)	83.60 (39)	4.71 (46)
WP + WCoverage	5.08 (46)	78.50 (35)	78.64 (37)	4.75 (46)	83.41 (36)	5.28 (40)

Table 4. Experimental results obtained using non-extreme values for the class weights.

	wOpac	wAcc	Acc_mic	Opac_mic	Acc	Opac
Standard ART	5.86	78.28	78.28	5.86	**83.87**	5.33
Confidence + WCoverage	6.01 (36)	78.23 (43)	78.27 (43)	5.81 (40)	83.87 (45)	5.58 (25)
TR + Coverage	5.36 (34)	**79.90 (33)**	**82.04 (40)**	4.65 (42)	83.35 (35)	5.02 (31)
TR + WCoverage	5.64 (33)	79.81 (33)	81.92 (40)	4.65 (43)	83.36 (34)	5.48 (30)
RWP + Coverage	**5.03 (43)**	78.22 (36)	78.85 (38)	**4.46 (47)**	83.33 (35)	**4.75 (44)**
RWP + WCoverage	5.33 (42)	78.47 (36)	79.40 (38)	4.50 (46)	83.40 (36)	5.18 (38)
WP + Coverage	5.33 (46)	78.23 (42)	78.39 (40)	4.98 (47)	83.84 (42)	4.99 (44)
WP + WCoverage	5.47 (43)	78.40 (37)	78.43 (41)	4.93 (47)	83.85 (42)	5.22 (37)

classical measures (*Acc* and *Opac*) nor their weighted counterparts (*wAcc* and *wOpac*).

As mentioned above, we have also performed some experiments using non-extreme weights distribution. Table 4 summarizes the results we have obtained. Once again, the classifier opacity (wOpac) is reduced by most of the heuristics we had devised. In addition, accuracy does not get worse and even improves in some cases.

Regarding to the heuristics we have developed to adapt ART to weighted classification, TR plus coverage tends to get the best results for accuracy measures, with minor improvements in classifier complexity. On the other hand, RWP plus coverage behaves better with respect to complexity measures without penalizing classifier accuracy, one of the main goals behind our work.

5. Conclusions and future work

Balancing complexity and accuracy in classification models is a difficult trade-off. We have devised a decision-list-oriented algorithm that finds shorter descriptions for the most important classes (as defined by a class weight assignment), and we have done so without penalizing classifier accuracy.

Several measures have been proposed to evaluate the behavior of supervised learning techniques in the context of weighted classification. In the experiments, we show that our method behaves well with respect to these novel measures, as well as with respect to the traditional measures of accuracy and complexity.

We expect to see a growing interest in complexity-oriented classification models in the near future. In particular, we intend to extend the work we have presented in this paper to general decision tree classifiers by introducing the appropriate complexity-oriented heuristics in the decision tree building process.

References

[1] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *Sigkdd Explorations*, 6(1):20–29, 2004.

[2] F. Berzal, J. C. Cubero, N. Marín, and J. L. Polo. An overview of alternative rule evaluation criteria and their use in separate-and-conquer classifiers. In *Proceedings of 16th International Symposium on Methodologies for Intelligent (ISMIS 2006)*, 2006.

[3] F. Berzal, J. C. Cubero, D. Sánchez, and J. M. Serrano. Art: A hybrid classification model. *Machine Learning*, 54(1):67–92, 2004.

[4] C. Blake and C. Merz. Uci repository of machine learning databases, 1998. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[5] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. Brodley. Pruning decision trees with misclassification costs. In *Proceedings of the European Conference on Machine Learning*, pages 131–136, 1998.

[6] C. Drummond and R. Holte. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Proceedings of The Seventeenth International Conference on Machine Learning*, pages 239–246, 2000.

[7] C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[8] 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*, 55(1):119–139, August 1997.

[9] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal*, 6(5):429–449, 2002.

[10] N. Lavrac, B. Kavsek, P. Flach, and L. Todorovski. Subgruop discovery with cn2-sd. *Journal of Machine Learning Research*, 5:153–188, 2004.