ECAI 2014
T. Schaub et al. (Eds.)
© 2014 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-419-0-965

Post-processing a classifier's predictions: Strategies and empirical evaluation

Salem Benferhat, Karim Tabia[1] and **Mouaad Kezih, Mahmoud Taibi**[2]

Abstract. In this paper, we propose an approach allowing to revise the outputs of a classifier in order to take into account the available domain knowledge. This approach can be applied for any classifier be it probabilistic or not. We propose post-processing criteria and methods to encode and exploit different kinds of domain knowledge. Finally, we provide experimental studies on a set of benchmarks.

1 Introduction

Classification consists in predicting the class of an object given its features. Most works in classification deal either with learning efficient classifiers from data or combine multiple classifiers. Many related issues receive also much interest especially regarding learning classifiers from imbalanced datasets, classifier evaluation, reject and drift options, multiple class classification problems, etc.
In this paper, with deal with a complementary issue aiming to exploit any extra domain knowledge by post-processing the classifier predictions. We addressed this problem originally in [1] in a computer security context and we dealt only with probabilistic classifiers. The contributions of this paper are i) Providing a formalization of the problem of revising the predictions of a classifier. ii) Proposing two new revision criteria where the first criterion allows to relabel the items where the classifier's confidence is low measured in terms of entropy while the second criterion is tailored for cost-sensitive classification problems. iii) Generalizing and extending the prediction revision to any classifiers, and iv) Providing an experimental study covering most of the problems dealt with in classification tasks.

2 Post-processing a classifier's predictions

Typical domain knowledge a user may want to revise with is:
i) Knowledge about the items to classify: Assume that we have n objects to classify denoted o_1, o_2,.., o_n. Then one may have information (in general or within a specific situation) that the amount of items of a class c_i is greater than c_j (namely, the probability $p(c_i)>p(c_j)$).
ii) A user's requirements: In many prediction applications, a user may want a specific amount of instances in a given class.

Typically, three types of domain knowledge can be exploited to post-process the predictions of a classifier: i) *Knowledge about a single class,* ii) *Knowledge about the ranking over the classes* and iii) *Knowledge about the class distribution.* Knowledge about the class distribution is the most exhaustive and accurate domain knowledge. As illustrated on Figure 1, the objective is to design a post-processor to revise the predictions made by a classifier to fit the set of requirements of the user. Assume that we have a set of items to classify

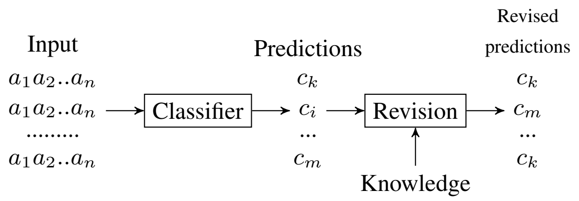

Figure 1. Post-processing a classifier's predictions

denoted $\mathcal{O}$=$\{(a_1a_2..a_n)_1$,..,$(a_1a_2..a_n)_m\}$ where $a_1a_2..a_n$ is an instantiation of the attributes $A_1A_2..A_n$. The classifier f will associate with each instance $(a_1a_2..a_n)_i$ a class instance $c_k \in D_C$, denoted c_k=$f((a_1a_2..a_n)_i)$. Without loss of generality, let us assume that the classifier f outputs a vector of scores v_i=$(s_1, s_2, .., s_k)_i$ for each instance $(a_1a_2..a_n)_i$ (here, k denotes the number of class instances). The scores $(s_1, s_2, .., s_k)_i$ denote:

i) *A posterior probability distribution* in case of probabilistic classifiers. A probability distribution allows to encode the ranking such that if c_i is ranked before c_j then $p(c_i)>p(c_j)$.
ii) *A vector of zeros and ones* in case of classifiers outputting only class labels. For example, a classifier predicting c_1 will output the vector $(1, 0, .., 0)$ where the value 1 denotes the predicted class while the remaining zeros exclude the corresponding ones.

This paper deals with revising the predictions of a classifier where a prediction c^* for an item $a_1..a_n$ is obtained according to the rule: c^*=$argmax_{i=1..k}(s_i)$, where s_i denotes the scores associated by the classifier f to the item $a_1..a_n$ for being in the class c_i.

Strategies for revising a classifier's predictions

Let o_1,..,o_m denote the set of objects to classify with o_i=$(a_1a_2..a_n)_i$. Let also v_1,..,v_m denote the set of predictions made by the classifier f such that $f(o_i)$=v_i. Similarly, let us use f_i (resp. r_i) to denote the class label predicted by f (resp. the post-processor) for the object o_i. Assume also that we have a set of constraints $\mathcal{K}$=$\{K_1$,..,$K_w\}$ representing the extra domain knowledge and requirements to satisfy. Then there are three situations to be considered:

1. **Case 1:** $\forall K_i \in \mathcal{K}$, $p_f(c_i)$=α_i meaning that all the constraints K_i (namely $p_K(c_i)$=α_i) are already satisfied by the classifier f.
2. **Case 2:** $\exists K_i \in \mathcal{K}$, $p_f(c_i)>\alpha_i$. This happens when the classifier f classifies more objects in a class c_i than required by the domain knowledge. To satisfy the constraint K_i, some of the objects predicted as c_i have to be relabeled in the other classes c_k with $k \neq i$.
3. **Case 3:** $\exists K_i \in \mathcal{K}$, $p_f(c_i)<\alpha_i$. This situation happens if the classifier f has not predicted enough objects in class c_i meaning that some objects predicted by f in the other classes c_k with $k \neq i$ have to be revised and predicted by the post-processor in the class c_i.

[1] CRIL CNRS UMR 8188, Artois University, France. {benferhat,tabia}@cril.fr
[2] Badji Mokhtar University of Annaba, Algeria. {mouaad.kezih, mahmoud.taibi}@univ-annaba.dz

For *Case 2* and *Case 3*, many strategies can be adopted to select the objects to relabel while satisfying the set of constraints $\mathcal{K}$. The principles that our revision strategy follows i) minimize miss-classification cost and ii) minimize relabelings. We use a heuristic algorithm to minimize the number of relabelings. It first satisfies the constraint K_i requiring the largest items in class c_i, then it continues with the following constraints in a decrementing order. Note that it is enough to deal only with constraints of *Case 3* to satisfy the set of constraints $\mathcal{K}$. In order to minimize relabelings, an item predicted in the class c_i will not be relabeled if the corresponding constraint K_i requires more items in c_i than predicted by the classifier f.

Criteria for post-processing the predictions

In case a constraint $\mathcal{K}_i$ is not satisfied then we need to relabel some items predicted by the classifier f in the other classes and predict them in the target class c_i. We propose five revision criteria:

- *MCTC (Maximize Confidence in the Target Class):* This criterion interprets the scores v_i=$(s_1, s_2, .., s_k)_i$ associated with an object o_i by the classifier f as the confidence of f that the right class of o_i is $argmax((s_1, s_2, .., s_k)_i)$. The selected object $\hat{o_j}$ using the *MCTC* criterion is defined as follows:
$$\hat{o_j} = argmax_{j=1..m}(v[i]_j), \tag{1}$$
where $v[i]_j$ is the score s_i of the target class c_i in the vector v_j of the scores associated by the classifier f to o_j.
- *MCPC (Minimize Confidence in the Predicted Class):* MCPC selects to relabel the object classified with the lowest confidence.
$$\hat{o_j} = argmin_{j=1..m}(\max((s_1, .., s_k)_j)), \tag{2}$$
where $\max((s_1, .., s_k)_j)$ denotes the highest score among the ones associated by the classifier f to the object o_j.
- *MPTCD (Minimize the Predicted-Target Class Confidence Difference):* MPTCD combines *MCTC* and *MCPC* and aims at minimizing the gap between the predicted class and the target one.
$$\hat{o_j} = argmin_{j=1..m}(\max((s_1, .., s_k)_j) - v[i]_j) \tag{3}$$
- *ME (Maximize the Entropy):* ME aims to select to relabel the object where the classifier f is less confident in terms of entropy.
$$\hat{o_j} = argmin_{j=1..m}(entropy(s_1, .., s_k)_j)) \tag{4}$$
- *MMCC (Minimize Miss-Classification Cost):* This criterion allows to take into account both the scores output by the classifier f and the miss-classification costs.
$$\hat{o_j} = argmin_{j=1..m}(\sum_{h=1}^{k} s_h * cost(f(o_j), c_i)), \tag{5}$$
where $cost(f(o_j), c_i)$ is the cost of miss-classification of c_i in the class predicted by the classifier $f(o_j)$.

3 Experimental studies

All the used datasets in the experimental studies are publically available (from the UCI [3] and KEEL[4] imbalanced dataset repositories). Note that we selected different types of datasets with different characteristics (size, dimension, etc). We carried out experiments on both probabilistic and non probabilistic ones. As domain knowledge, we use two kinds of knowledge obtained only from training datasets: i) *Training Dataset Distribution (TDD)* (we use as domain knowledge the frequencies of the different classes) and ii) *Miss-Classification Rates (MCR)* (the domain knowledge we exploit here is relative to the miss-classification rates obtained by evaluating the classifier on the training dataset). We provide in Table 1 the results of the Naive Bayes NB classifier [2] as example of probabilistic classifiers. The first five result columns of Table 1 denote respectively the PCC (Percentage of Correct Classification) obtained with the NB classifier without any post-processing (column NB) while the remaining columns denote the results of post-processing the NB predictions using the criterion in the header of each column. In each cell, we give the results of revising with TDD knowledge and the results of revising with MCR knowledge between brackets. For the *MMCC*, we provide results obtained using a cost-matrix generated randomly. Finally, the results are obtained through a 10-fold cross-validation on the training datasets.

Dataset	NB	MCTC	MCPC	MPTCD	ME	MMCC
spambase	79.22%	79.33% (80.42%)	78.77% (79,46%)	77.80% (81.83%)	76.61% (77.63%)	76% (82.13%)
dbworld	89.06%	84.37% (87.50%)	87.50% (90.63%)	96.68% (96.87%)	85.94% (90.62%)	90.62% (90.62%)
column 2c	77.74%	49.67% (65.80%)	49.67% (68.06%)	73.22% (80.96%)	73.22% (79.67%)	80.64% (80.96%)
column 3c	83.22%	48.70% (61.61%)	48.06% (79.03%)	83.54% (83.54%)	82.90% (83.22%)	80.64% (83.54%)
AU	52%	46.35% (48.62%)	47.69% (52.36%)	54.55% (54.81%)	54.46% (54.76%)	52.78% (54.81%)

Table 1. Results of NB classifier evaluation

In Table 2, we evaluate a non probabilistic C4.5 decision tree classifier [3]. As for domain knowledge, we use only the class labels predicted by these classifiers. The results of Table 1 and Table 2 show two main trends: The first trend is that on most the datasets using the MCR knowledge performs better than the classifier alone and better than the classifier with the post-processor exploiting the TDD knowledge. The second trend is that on most the datasets the criteria *ME* and *MMCC* perform better than the *MCTC*, *MCPC* and *MPTCD* both when using TDD knowledge or the MCR knowledge. Note that the results of the other probabilistic classifier TAN [2] and a probabilistic classifier k-NN comply with these main trends.

Dataset	C4.5	MCTC	MCPC	MPTCD	ME	MMCC
spambase	92.97%	92.91% (92.97%)	92.97% (92.78%)	92.91% (93%)	92.91% (93.04%)	93.28% (93.52%)
dbworld	71.87%	62.5% (70.31%)	64.06% (70.31%)	65.63% (71.87%)	62.5% (71.87%)	84.37% (85.94%)
column 2c	81.61%	80.64% (83.22%)	78.06% (78.7%)	82.58% (83.22%)	82.58% (83.22%)	83.22% (84.84%)
column 3c	81.61%	75.48% (81.93%)	49.67% (80.96%)	76.77% (81.29%)	71.61% (80.64%)	82.90% (83.23%)
AU	64.25%	64.08% (64.32%)	63.96% (64.18%)	64.35% (65.3%)	64.37% (65.58%)	64.1% (65.22%)

Table 2. Results of the $C4.5$ classifier evaluation

4 Conclusions

This paper dealt with a novel and important issue in classification consisting in exploiting the available domain knowledge. The selection criteria proposed in this paper are based on natural ideas and aim at minimizing miss-classifications while fitting all the considered domain knowledge. Our approach is designed as a plug-in to be combined with any prediction model be it a probabilistic or non probabilistic classifier or even any prediction model.

REFERENCES

[1] S. Benferhat, A. Boudjelida, K. Tabia, and H. Drias, 'An intrusion detection and alert correlation approach based on revising probabilistic classifiers using expert knowledge', *Appl. Intell.*, **38**(4), 520–540, (2013).

[2] N. Friedman, D. Geiger, and M. Goldszmidt, 'Bayesian network classifiers', *Machine Learning*, **29**(2-3), 131–163, (1997).

[3] J. R. Quinlan, 'Induction of decision trees', *Mach. Learn.*, **1**(1), 81–106, (1986).

[3] `https://archive.ics.uci.edu/ml/datasets.html`
[4] `http://sci2s.ugr.es/keel/imbalanced.php`