Classification with imperfect supervisor: an iterative approach

Luciano Nieddu*
Faculty of Economics
UNINT University, Rome, Italy - l.nieddu@unint.eu

Donatella Vicari
Dipartimento di Scienze Statistiche
Sapienza, Universitá di Roma, Rome, Italy - donatella.vicari@uniroma1.it

Abstract

In this paper we present a possible extension of the well known k-means classification algorithm to tackle the problem of supervised classification in the case where the training set is affected by error or it has been classified by an imperfect supervisor.
Keywords: Classification; Imperfect Supervisor; Discriminant Analysis; k-means.

1. Introduction

In the last decades the problem of assigning a class to an object based on a set of measurements obtained on that object has gathered a new burst with the introduction of kernel methods and support vector machines (Shawe-Taylor and Cristianini [2004]) and with the almost unlimited computational power that the new computers provide.
In a classification problem a set of measurements on a set of entities is usually available. These measurements are used to train a classification algorithm. If the data at hand is composed of entities of known class then the problem is known as problem with supervisor (Watanabe [1985]), otherwise it is known as unsupervised classification and it is usually tackled via clustering or finite mixture models. A finer taxonomy divides the problem of supervised classification into classification with perfect supervisor, when the data have been

Therefore the focus should be moved from looking for the best classifier to looking for the most rubust classifier that can perform fairly well under various assumptions.
Some of the most robust techniques for supervised classification are those based on classification trees (Breiman et al. [1984]) that make no assumption on the parametric distribution of the data and are based on recursively partitioning the feature space into homogeneous subsets of units according to the class the entities belong to. This approach has been recently developed incorporating simple parametric models into the terminal nodes of the tree. Research in this direction was motivated by considering the fact that a constant value in the terminal node either produces a tree with a mediocre performance or tends to produce large and thus hard to interpret trees (Chan and Loh [2004]). Allowing to fit a simple function in the nodes permits some flexibility and therefore gives a good fitting of the tree on the data, without requiring to grow a very large tree.
One of the shortcomings of these approaches is that the recursive partitioning of the data, i.e. the growing of the tree, is done considering only one variable at a time and, although it makes the trees pretty simple to build rules it also makes them hard to interpret (for instance some variables might show up more than once in the tree path)
Building on these ideas, we carry the integration of parametric models into trees one step further and propose a supervised classification algorithm where the recursive partitioning is based on the whole feature vector.
The outline of the paper is as follows: in Section 2 the algorithm will be presented, while in Section 3 some features of the proposed techniques will be discussed. In Section 4 some conclusions and future agenda will be presented.

2. The Algorithm

In this section the algorithm proposed in this paper will be presented.
Given a data set of n elements on which p measurements are available (pattern vectors) in $\Re^p$, let us assume a partition defined on the dataset, i.e. each pattern vector is assigned to one and only one of K known classes. Let us assume a Euclidean norm defined on the dataset and let be a function from $\Re^p$ onto the set $\mathcal{C} = \{1; 2; :::; k\}$ which maps each pattern vector $\mathbf{x}_j; j = 1; :::; n$ into the class $c \in \mathcal{C}$ that it belongs to.
The proposed algorithm works as follows:

- compute the barycentre of each class and compute the distance of each vector from each barycentre;
- if each vector is closer to the barycentre of its class the algorithm stops, otherwise there will be a non empty set $\mathcal{M}$ of pattern vectors which belong to a class and are closer to a barycentre of a different class. In $\mathcal{M}$ select the pattern vector $\mathbf{x}_w$ that is farthest from the barycentre of its class. This pattern vector will be used as a seed for a new barycentre for class $(\mathbf{x}_w)$;
- a k-means algorithm (MacQueen [1967]) will then be performed for all the pattern vectors in class $(\mathbf{x}_w)$ using, as starting points, the set of barycentres for class $(\mathbf{x}_w)$ and the vector $\mathbf{x}_w$. Once the k-means has been performed the set of barycentres will be composed of $k + 1$ elements. The barycentres at the new iterations need not be computed for all classes, but only for class $(\mathbf{x}_w);$ since the barycentres for the other classes have remained unchanged. In the following step the distance of each pattern vector from all the barycentres is computed anew, and so is the set $\mathcal{M}$ (see figure 1);
- if $\mathcal{M}$ is not empty then the pattern vector in $\mathcal{M}$ which is farthest from a barycentre of its own class is once again selected to serve as a seed for a new barycentre. This procedure iterates until the set $\mathcal{M}$ is empty.

Upon convergence, the algorithm yields a set of barycentres which, in the worst case, are in a number equal to the number of elements in the dataset and which has a lower bound in the number of classes.
The set of barycentres so obtained can be used to classify new entities using the class of the barycentre the query point is closest to (ties can be broken randomly).

3. Discussion

If elements from the training set are used as query points, then the algorithm always classify them correctly because, once converged, all pattern vectors in the training set are closer to a centroid of their own class.

```
Step1 Let
        { $\mathsf{x}_j$, $j = 1, \ldots, n$ be the pattern vectors in the training set
        { $\mathsf{B}_0$ be the set of $k$ initial barycentres $\mathsf{b}_i$, $i = 1, \ldots, k$
Step2 Compute the distances of each $x_j$ from all the $b_i \in B_t$
      Let $\mathcal{M}$ be the set of $\mathsf{x}_w$ that are closer to a barycentre of a class different from their own.
      $t \leftarrow 0$
Step3 while $M \neq \emptyset$
        { Let $\mathsf{x}_s$, $s \in \mathcal{M}$ be the vector with the greatest distance from its own barycentre.
        { $c \leftarrow \psi(\mathsf{x}_s)$
        { Let $B_{t+1} \leftarrow B_t \cup \mathsf{x}_s$
        { for all the elements of class $c$ perform a $k$-means routine using as starting points the barycentres of $B_{t+1}$
          that belong to class $c$
        { $t \leftarrow t + 1$
        { Compute the distances of each $\mathsf{x}_j$ from all the $\mathsf{b}_i \in B_t$
        { Let $\mathcal{M}$ be the set of $\mathsf{x}_w$ that are closer to a barycentre of a class different from their own.
  • end
```

Figure 1: : Algorithm in meta-language

Although the algorithm may be slow in training, especially if the classes occupy overlapping regions of the parameter space, the set of baricentres must be computed only once. In classification only b distances need to be computed, where b is the number of barycentres obtained from training.
The aim of this algorithm is to find subclasses in the dataset which can be used to classify new vectors of unknown class. It is worth noticing that if the partition defined on the dataset is consistent with the features considered, i.e. if the pattern vectors are linearly separable, then the algorithm generates a number of barycentres equal to the number of classes. On the other hand, if the dataset is not linearly separable, then the algorithm continues splitting the classes until the subclasses obtained are linearly separable. It is obvious that it can continue splitting until all the subclasses are composed of only one vector (singleton).
It must be stressed that it is possibile that the algorithm will not reach convergence if two entities in the dataset belong to different classes and are represented by the same pattern vector. In such a situation either one of the entities has been given the wrong class, or the measurements on the objects are not sufficient enough to discriminate the objects. This problem can be easily overcome increasing the dimension of the vector space.
The algorithm can be generalized allowing for impurity in the result, i.e. the recursive partitioning of the feature space can be performed until the percentage of elements that are closer to a barycentre of another class than to one of their own has decreased under a certain threshold which can be set to a value different from zero.
This can be useful to avoid overfitting of the data, since the algorithm, as detailed in figure 1, iterates until all the elements in the training set are closer to a barycentre of their own class and creating a new barycentre at each iteration.
The proposed algorithm has been tested on an extensive simulation yielding very promising results when compared to standard classification algorithms.

4. Conclusions

The idea for a new classification algorithm that builds on the concept of model based recursive partitioning Zeileis et al. [2008] has been presented in this paper. The algorithm has already been tested on its form without allowing impurity in the classification yeilding very good results when compared with standard classification algorithms. The version with impurity is currently being tested and the results are promising.

References

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classi cation and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

Kin Y. Chan and Wei Y. Loh. Lotus: An algorithm for building accurate and comprehensible logistic regression trees. *Journal of Computational and Graphical Statistics*, 13(4):826–852, 2004.

Richard O. Duda and Peter E. Hart. *Pattern Classi cation and Scene Analysis*. John Wiley & Sons Inc, 1 edition, February 1973. ISBN 0471223611. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471223611`.

U.A. Katre and T. Krishnan. Pattern recognition with an imperfect supervisor. *Pattern Recognition*, 22(4):423 – 431, 1989. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/0031-3203(89)90051-4. URL `http://www.sciencedirect.com/science/article/pii/0031320389900514`.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972.

S. Watanabe. *Pattern Recognition: Human and Mechanical*. Wiley, New York, 1985.

David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390, October 1996. ISSN 0899-7667. doi: 10.1162/neco.1996.8.7.1341. URL `http://dx.doi.org/10.1162/neco.1996.8.7.1341`.

Achim Zeileis, Torsten Hothorn, and Kurt Hornik. Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, 17(2):492–514, 2008. doi: 10.1198/106186008X319331.