Neurocomputing 138 (2014) 316–324

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Adaptive training set reduction for nearest neighbor classification

Juan Ramón Rico-Juan *, José M. Iñesta [1]

Dpto. Lenguajes y Sistemas Informáticos, Universidad de Alicante, E-03081 Alicante, Spain

ARTICLE INFO

Article history:
Received 26 April 2013
Received in revised form
20 November 2013
Accepted 21 January 2014
Communicated by R. Capobianco Guido
Available online 15 February 2014

Keywords:
Editing
Condensing
Rank methods
Sorted prototypes selection
Adaptive pattern recognition
Incremental algorithms

ABSTRACT

The research community related to the human-interaction framework is becoming increasingly more interested in interactive pattern recognition, taking direct advantage of the feedback information provided by the user in each interaction step in order to improve raw performance. The application of this scheme requires learning techniques that are able to adaptively re-train the system and tune it to user behavior and the specific task considered. Traditional static editing methods filter the training set by applying certain rules in order to eliminate outliers or maintain those prototypes that can be beneficial in classification. This paper presents two new adaptive rank methods for selecting the best prototypes from a training set in order to establish its size according to an external parameter that controls the adaptation process, while maintaining the classification accuracy. These methods estimate the probability of each prototype of correctly classifying a new sample. This probability is used to sort the training set by relevance in classification. The results show that the proposed methods are able to maintain the error rate while reducing the size of the training set, thus allowing new examples to be learned with a few extra computations.

1. Introduction

The research community related to the human-interaction framework is becoming increasingly more interested in interactive pattern recognition (IPR), taking direct advantage of the feedback information provided by the user in each interaction step in order to improve raw performance. Placing pattern recognition techniques in this framework requires changes to be made to the algorithms used for training.

The application of these ideas to the IPR framework implies that adequate training criteria must be established [16,20]. These criteria should allow the development of adaptive training algorithms that take the maximum advantage of the interaction-derived data, which provides new (*data*,*class*) pairs, to re-train the system and tune it to user behavior and the specific task considered.

The *k*-nearest neighbors rule has been widely used in practice in non-parametric methods, particularly when a statistical knowledge of the conditional density functions of each class is not available, which is often the case in real classification problems.

The combination of simplicity and the fact that the asymptotic error is fixed by attending to the optimal Bayes error [6] are important qualities that characterize the nearest neighbor rule. However, one of the main problems of this technique is that it requires all the prototypes to be stored in memory in order to compute the distances needed to apply the *k*-nearest rule. Its sensitivity to noisy instances is also known. Many works [13,18,4,11,1] have attempted to reduce the size of the training set, aiming both to reduce complexity and avoid outliers, whilst maintaining the same classification accuracy as when using the entire training set.

The problem of reducing instances can be dealt with two different approaches [14], one of them is the generation of new prototypes that replace and represent the previous ones [5], whilst the other consists of selecting a subset of prototypes of the original training set, either by using a *condensing* algorithm that obtains a minimal subset of prototypes that lead to the same performance as when using the whole training set, or by using an *editing* algorithm, removing atypical prototypes from the original set, thus avoiding overlapping among classes. Other kinds of algorithms are hybrid, since they combine the properties of the previous ones [12]. These algorithms try to eliminate noisy examples and reduce the number of prototypes selected.

The main problems when using condensing algorithms are to decide which examples should remain in the set and how the typicality of the training examples should be evaluated. Condensing algorithms place more emphasis on minimizing the size of the training set and its consistence, but outlier samples that harm the accuracy of the classification are often selected for the prototype set. Identifying these noisy training examples is therefore the most important challenge for editing algorithms.

* Tel.: +34 96 5903400x2738; fax: +34 96 5909326.
E-mail addresses: juanra@dlsi.ua.es (J.R. Rico-Juan), inesta@dlsi.ua.es (J.M. Iñesta).
[1] Tel.: +34 96 5903698; fax: +34 96 5909326.

http://dx.doi.org/10.1016/j.neucom.2014.01.033
0925-2312

Further information can be found in a review [10] about prototype selection algorithms based on the nearest neighbor technique for classification. That publication also presents a taxonomy and an extended empirical study.

With respect to the algorithms proposed in this paper, all the prototypes in the set will be rated with a score that is used to establish a priority for them to be selected. These algorithms can therefore be used for editing or condensing, depending on an external parameter η representing an a posteriori accumulated probability that controls their performance in the range $[0, 1]$, as will be explained later. For values close to one, their behavior is similar to that of an editing algorithm, while when η is close to zero the algorithms perform like a condensing algorithm.

These methods are incremental, since they adapt the learned rank values to new examples with little extra computations, thus avoiding the need to recompute all the previous examples. These new incremental methods are based on the best of the static (classical) methods proposed by the authors in a previous article [15]. The basic idea of this rank is based on the estimation of the probability of an instance to participate in a correct classification by using the nearest neighbor rule. This methodology allows the user to control the size of the resulting training set.

The most important points of the methodology are described in the following sections. Section 2 provides an overview of the ideas used in the classical and static methodologies to reduce the training set. The two new incremental methods are introduced in Section 3. In Section 4, the results obtained when applying different algorithms to reduce the size of the training set, their classification error rates, and the amount of additional computations needed for the incremental model are shown applied to some widely used data collections. Finally, some conclusions and future lines of work are presented.

2. Classical static methodologies

The *Condensed Nearest Neighbor Rule* (CNN) [13] was one of the first techniques used to reduce the size of the training set. This algorithm selects a subset S of the original training set T such that every member of T is closer to a member of S of the same class than to a member of a different class. The main issue with this algorithm is its sensitivity to noisy prototypes.

Multiedit Condensing (MCNN) [7] solves this particular CNN problem. The goal of MCNN is to remove the outliers by using an editing algorithm [18] and then applying CNN. The editing algorithm starts with a subset $S=T$, and each instance in S is removed if its class does not agree with the majority of its k-NN. This procedure edits out both noisy instances and close to border prototypes. The MCNN applies the algorithm repeatedly until all remaining instances have the majority of their neighbors in the same class.

Fast Condensed Nearest Neighbor (FCNN) [2] extends the classic CNN, improving the performance in time cost and accuracy of the results. In the original publication, this algorithm was compared to hybrid methods, and it was shown to be about three orders of magnitude faster than them, with comparable accuracy results. The basic steps begin with a subset with one centroid per class and follow the CNN criterion, selecting the best candidates using nearest neighbor and nearest enemy concepts (Voronoi cell).

2.1. Static antecedent of the proposed algorithm

Now, the basic ideas of the static algorithm previously published by the authors [15] are briefly introduced. The intuitive concepts are represented graphically in Fig. 1, where each prototype of the training set, a, is evaluated searching its nearest enemy (the nearest to a from a different class), b, and its best candidate, c. This *best candidate* is the prototype that receives the vote from a, because if c is in the final selection set of prototypes, it will classify a correctly with the 1-NN technique.

In the *one vote per prototype farther* algorithm (1_FN, illustrated in Fig. 1a), only one vote per prototype, a, is considered in the training set. The algorithm is focused on finding the best candidate, c, to vote for that it is the farthest prototype from a, belonging to the same class as a. It will, therefore, be just nearer than the nearest enemy, $b = \arg\min_{x \in T \setminus \{a\}} \{d(a,x) : class(a) \neq$

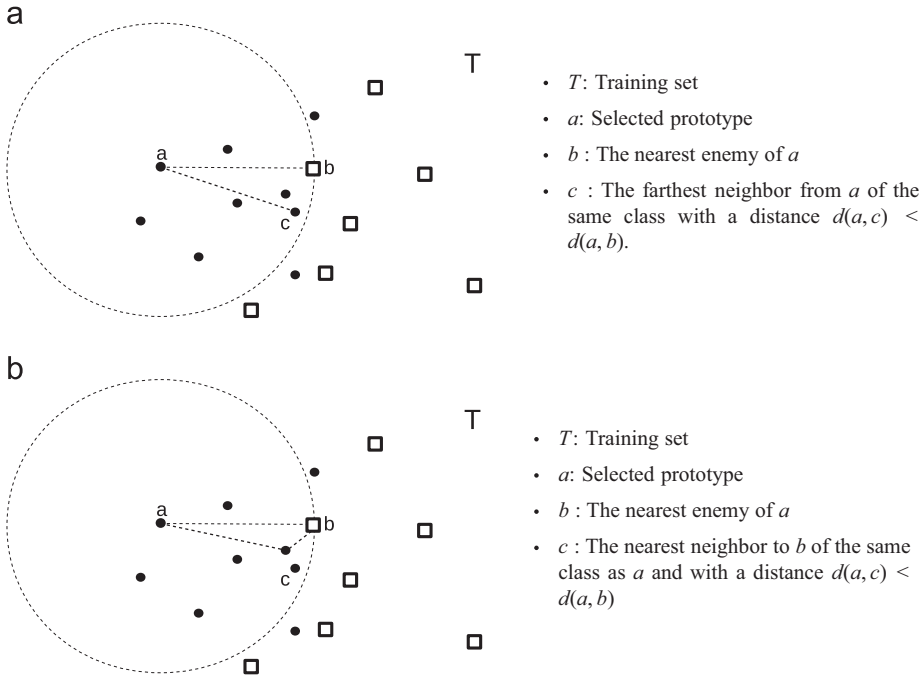

Fig. 1. Illustrations of the static voting methods used: (a) The 1 FN scheme: the voting method for two classes to one candidate away from the selected prototype and (b) The 1 FE scheme: the voting method for two classes to one candidate near the enemy class prototype.