Applied Soft Computing 38 (2016) 10–22

Contents lists available at ScienceDirect

Applied Soft Computing

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

Evolutionary wrapper approaches for training set selection as preprocessing mechanism for support vector machines: Experimental evaluation and support vector analysis

Nele Verbiest[a,*], Joaquín Derrac[b], Chris Cornelis[a,c], Salvador García[c,d], Francisco Herrera[c]

[a] Department of Applied Mathematics and Computer Science, Ghent University, Belgium
[b] Affectv Limited, London, United Kingdom
[c] Department of Computer Science and AI, Research Center on Information and Communications Technology (CITIC-UGR), University of Granada, Spain
[d] Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 31 January 2014
Received in revised form 2 September 2015
Accepted 3 September 2015
Available online 30 September 2015

Keywords:
Support vector machines
Training set selection
Data reduction

ABSTRACT

One of the most powerful, popular and accurate classification techniques is support vector machines (SVMs). In this work, we want to evaluate whether the accuracy of SVMs can be further improved using training set selection (TSS), where only a subset of training instances is used to build the SVM model. By contrast to existing approaches, we focus on wrapper TSS techniques, where candidate subsets of training instances are evaluated using the SVM training accuracy. We consider five wrapper TSS strategies and show that those based on evolutionary approaches can significantly improve the accuracy of SVMs.

1. Introduction

In many real-world applications, datasets can contain noisy or wrong information. Even the best classifiers might not be able to deal with these datasets. Training Set Selection (TSS, [1–3]) is a good way to alleviate this problem. It is a preprocessing technique that only selects relevant instances before applying the classifier. The objective of TSS is twofold: on the one hand, the accuracy of the classifier can be improved, while on the other hand, the efficiency can be enhanced.

TSS has mainly been investigated for the K Nearest Neighbor (KNN) classifier [4], in that context it is referred to as Prototype Selection (PS, [5]). There are two main groups of PS techniques. Wrapper techniques use the KNN classifier to evaluate entire candidate subsets of instances, while filter techniques do not make use of the KNN classifier or only use it to carry out partial evaluations.

In this work we want to study if TSS techniques can also improve the accuracy of Support Vector Machines (SVMs). As wrapper PS techniques explicitly use the KNN classifier, they cannot be applied meaningfully to improve SVM classification. It is clear, however, that filter PS techniques can be directly applied to SVMs as they are less dependent on the KNN classifier. On the other hand, filter methods are in general less suited to improve the accuracy of a classifier.

To the best of our knowledge, only two filter TSS techniques have been proposed to specifically improve SVMs. In [6], the Multi-Class Instance Selection (MCIS) method is proposed, which selects instances near the boundary between one and the other classes of datasets. This method focuses on reduction of the dataset to improve the efficiency of the SVMs. Another approach is presented in [7], where only training instances that are likely to become support vectors are selected. This Sparsifying Neural Gas (SNG) algorithm was developed to improve the efficiency of the SVM while maintaining or slightly improving the accuracy.

Unfortunately, these filter TSS techniques are unable to improve the accuracy of the SVMs. Therefore we attempt to improve the accuracy of SVMs using wrapper approaches. We adapt the five most important wrapper TSS techniques by plugging the SVM into the TSS methods. The wrapper techniques we consider evaluate candidate subsets based on the so-called training accuracy, which is the accuracy obtained when building the classifier at hand based on the candidate subset and using this model to classify the entire training data. In our case we use SVMs to calculate the training

* Corresponding author at: Krijgslaan 281 (S9), 9000 Gent, Belgium. Tel.: +32 92644770; fax: +32 92644995.
E-mail address: Nele.Verbiest@UGent.be (N. Verbiest).

http://dx.doi.org/10.1016/j.asoc.2015.09.006
1568-4946/

accuracy, and as a result subsets with a high training accuracy will be well-suited for SVM classification.

The remainder of this paper is organized as follows: In Section 2 we provide the necessary background on SVMs. In Section 3 we present existing filter TSS techniques and in Section 4 we present the design of the wrapper TSS techniques for SVMs. Then, we set up an experimental framework to evaluate the approaches' performance in Section 5. By means of an experimental evaluation on 43 real-life datasets, we show that wrapper TSS techniques can indeed significantly improve SVM classification. Evolutionary approaches, and the Generational Genetic Algorithm (GGA,[8,9]) in particular, seem to be especially well-suited for our purpose. In order to get more insight into the operation of the evolutionary wrappers, we provide a more detailed analysis for the latter, investigating the effect of TSS on the SVM's support vectors, and illustrating their behavior graphically on a two-dimensional artificial dataset. Finally, we conclude in Section 6.

2. Preliminaries

In this subsection we provide a general background on SVMs to make the paper self-contained. We denote instances by their feature vectors x. For now, we consider two-class problems, the class of an instance is either -1 or 1. At the end of this section we discuss the multi-class case.

The most basic form of SVMs are separating hyperplanes, where one aims to separate the two classes linearly by a hyperplane. The hyperplane can be represented by a linear function $f(x)$ that is optimized such that the distance from the hyperplane to the closest instances from both classes is maximal. To classify a new instance t, the value $f(t)$ is calculated. When $f(t) > 0$, t is classified to class 1 and else to the negative class -1.

In practice, the data is often not linearly separable, which led to the introduction of support vector classifiers, which allow for overlap between the classes. The idea is to find a hyperplane that maximizes the margin between the two classes, but allows for some data points to fall on the wrong side of the margin. Of course the number of misclassified training points is bounded. It can be shown that the resulting hyperplane is a linear combination of a set of instances, these lie on the classification margin and are called support vectors.

To allow for even more flexibility, kernel-based SVMs were introduced. Before constructing the separating hyperplane, the feature space is enlarged using a function h such that for two feature vectors x_1 and x_2

$$h(x_1)^t h(x_2) = K(x_1, x_2) \tag{1}$$

where K is a kernel function. A well-known example is the Radial Basis Function (RBF) kernel (x_1, x_2 feature vectors):

$$K(x_1, x_2) = \exp\left(-\frac{||x_1 - x_2||^2}{2\sigma^2}\right). \tag{2}$$

The separating hyperplanes are represented by a function f that takes values in $(-\infty, \infty)$. However, it is more useful to obtain probabilities. Therefore, a sigmoid model can be used to calculate the probabilities $P(y = 1|f)$. This scaling, referred to as Platt's scaling [10], can also be seen as training the model to find a better threshold: instead of using the standard 0 as threshold to classify test instances, we can train the model based on the class probabilities to find a better threshold.

The discussed methods apply to two-class problems. A traditional approach to handle multi-class problems is pairwise coupling [11–13], where the multi-class problem is decomposed in all possible two-class problems and the majority voting principle is applied. For instance, when there are K classes, for each pair of classes i and j with $i, j \leq K$ and $i \neq j$, the binary SVM is constructed. A new instance is classified by all classifiers, and each class gets a vote if the new instance is classified to that class. The class with the highest number of votes is the final class returned for that instance. Another approach is the so-called one-versus-all technique. In this case, K training datasets are considered, where in each dataset one class is the positive class and the remaining classes form the negative class. The SVM is trained on each of these training datasets and the target instance t is classified by each SVM. Each SVM returns a probability value p expressing the confidence that t should be classified to the positive class. Finally, t is classified to the class for which this probability is maximal.

3. Related work: filter TSS techniques for SVMs

In [5], a comprehensive overview of TSS techniques is provided. Apart from the already mentioned distinction between wrapper and filter approaches, TSS techniques can also be categorized as edition, condensation or hybrid methods: while edition (or editing) methods remove noisy instances in order to increase classifier accuracy, condensation methods compute a training set consistent subset, removing superfluous instances that will not affect the classification accuracy of the training set. Finally, methods that eliminate both noisy and superfluous are called hybrid ones.

As we are mainly interested in improving the accuracy of SVM classification, we only consider the nine editing filter techniques discussed in [5]. They are reviewed in Section 3.1.

In addition to the nine TSS techniques from [5], which were originally developed to improve KNN, we also consider two filter TSS techniques that were specifically developed for SVMs. These are discussed in Section 3.2.

3.1. Editing filter TSS methods

A basic method is the Edited Nearest Neighbor (ENN, [14]) algorithm, which considers every instance in the training set and removes it whenever the nearest neighbor rule classifies it incorrectly using the remaining instances as training data. Many methods are derived from ENN, including:

- ENN with Estimation of Probabilities of Threshold (ENNTh, [15]): proceeds like ENN, except that the removal criterion is based on probabilities.
- All-KNN ([16]): applies ENN for different numbers of neighbors and removes an instance whenever any of the ENN runs marked an instance for removal.
- Modified ENN (MENN, [17]): takes into account the fact that multiple instances can be at the same distance from the target instance.
- Nearest Centroid Neighborhood Edition (NCNEdit, [18]): is very similar to ENN, but uses an alternative definition to determine the neighbors of an element based on centroids.
- Multi-Edit [19]: randomly divides the training data in blocks, applies ENN to them and merges the resulting sets.

We also consider three methods that are not derived from ENN:

- Relative Neighborhood Graph (RNG, [20]): constructs a proximity graph and removes instances that are misclassified by the neighbors in the graph.
- Model Class Selection (MOCS, [21]): uses a feedback system to incorporate knowledge about the dataset in a tree-based classifier.