Pattern Recognition 47 (2014) 2165–2177

Contents lists available at ScienceDirect

Pattern Recognition

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

Covariance-guided One-Class Support Vector Machine

Naimul Mefraz Khan [a,*], Riadh Ksantini [b], Imran Shafiq Ahmad [b], Ling Guan [a]

[a] Ryerson University, 350, Victoria Street, Toronto, ON, Canada M5B 2K3
[b] University of Windsor, 401, Sunset Avenue, Windsor, ON, Canada

ARTICLE INFO

Article history:
Received 26 March 2013
Received in revised form
1 October 2013
Accepted 1 January 2014
Available online 15 January 2014

Keywords:
Covariance
Support Vector Machine
One-class classification
Outlier detection

ABSTRACT

In one-class classification, the low variance directions in the training data carry crucial information to build a good model of the target class. Boundary-based methods like One-Class Support Vector Machine (OSVM) preferentially separates the data from outliers along the large variance directions. On the other hand, retaining only the low variance directions can result in sacrificing some initial properties of the original data and is not desirable, specially in case of limited training samples. This paper introduces a Covariance-guided One-Class Support Vector Machine (COSVM) classification method which emphasizes the low variance projectional directions of the training data without compromising any important characteristics. COSVM improves upon the OSVM method by controlling the direction of the separating hyperplane through incorporation of the estimated covariance matrix from the training data. Our proposed method is a convex optimization problem resulting in one global optimum solution which can be solved efficiently with the help of existing numerical methods. The method also keeps the principal structure of the OSVM method intact, and can be implemented easily with the existing OSVM libraries. Comparative experimental results with contemporary one-class classifiers on numerous artificial and benchmark datasets demonstrate that our method results in significantly better classification performance.

1. Introduction

One-class classification, novelty detection, outlier detection or concept learning [1,2] is the process of separating one particular type of data from the others. The difference with traditional two-class or multi-class classification is that only one class of data is available for training (deemed as *targets*). The objective is to distinguish any other data points from the targets. The “other” data points are typically called *outliers*. One-class classification is necessary for several reasons: (1) Outlier data may not be available or very costly to measure. For example, it is possible to measure the necessary features for a nuclear plant operating under normal circumstances. However, it is too dangerous or impossible to measure the same features in case of an accident. In this case, a classifier has to be trained based only on the data for normal circumstances (the target class). (2) In some cases, the available outlier data might be too small or badly sampled with unknown priors and ill-defined distributions [3]. (3) Another scenario where one-class classification can be of importance is the comparison of two datasets [3]. Usually, a classifier is trained on a dataset by a complex procedure. It will be beneficial if this training information can be reused. To solve a similar problem, the new dataset can be compared with the old dataset instead of repeating the whole training process.

For these reasons, one-class classification has found its application in several fields such as engine fault detection [4], medical diagnosis [5], nuclear testing [6], web page classification [7] and network intrusion detection [8].

The key limitation in one-class classification is that only one class of labeled dataset is available during training. The existing approaches use different techniques to estimate the necessary parameters to classify future data points as targets or outliers. Based on the techniques used, they can be divided into three categories; reconstruction based, density-based and boundary-based methods [9].

In reconstructive classifiers, a model regarding the data generation process is assumed first. In the training phase, the parameters of this model are estimated according to the target dataset. During classification, a reconstruction error for the incoming data point is calculated. The less the error, the more accurate is the model. Examples of reconstructive classifiers are K-Means [10], Self-Organizing Map (SOM) [11], and so on.

Density-based one-class classifiers are based on the estimation of the probability density function (PDF) of the target class [12]. The PDF is estimated through the training data using statistical

* Corresponding author. Tel.: +1 647 9685762; fax: +1 416 979 5280.
E-mail addresses: n77khan@ee.ryerson.ca (N. Mefraz Khan), ksantini@uwindsor.ca (R. Ksantini), imran@uwindsor.ca (I. Shafiq Ahmad), lguan@ee.ryerson.ca (L. Guan).

0031-3203/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2014.01.004

methods. For example, a Parzen density estimator is used in [13,14]. Gaussian distribution is used in [15]. After density estimation, thresholding is used to label data points as belonging to the target class or the outliers.

Vapnik [16] suggested that instead of using the complete training data to estimate the distribution (as done in density-based classifiers), only the boundary data points should be used to guess the area where the future target data points could reside. This is precisely what is done in boundary-based classifiers. The boundary points around the target class are used to classify an incoming data point. In [17], neural networks are trained with boundary data points to generate a target class border. But this approach has the inherent limitations of neural networks [3], i.e. size of the network, initialization, stopping criterion and so on. In [18], it was proposed that boundary estimation can be formulated into a convex optimization problem. The Support Vector Machine (SVM) [19] is a popular two-class classification method based on this philosophy, where the target is to maximize the distance margin between the two classes using support vectors. Most of the current boundary-based one-class classifiers are based on the principal of SVM. Examples of popular boundary-based one-class classifiers are the Support Vector Data Description (SVDD) [3] and One-class Support Vector Machines (OSVM) [20].

The problems with these existing categories of one-class classification methods are that none of them consider the full scale of information available for classification. In density-based methods, solely the overall class probability distribution is used. The first problem with this approach is that a large number of samples are required for density estimation. Also, these kinds of methods estimate the density based on the training data. But the latter represents the area of available targets only, not the complete distribution. The true distribution is unknown and can be unpredictable for a real-world problem. Density-based methods only focus on high density areas [3], and reject areas with lower training data density, although they represent valid targets. This limitation can reduce the performance of a classifier.

On the other hand, in boundary-based methods, only boundary data points are considered to build the model. These points do not completely represent the overall class. Solutions to boundary-based methods are only calculated based on the points near the decision boundary, regardless of the spread the remaining data. In [21], it has been shown that solutions to boundary-based methods like OSVM can be misled by the spread of data, since these methods tend to separate the data along large spread directions. A more reasonable method would be to simultaneously make use of the maximum margin criterion [19] while controlling the spread of data [21].

Also, unlike multi-class classification problems, the low variance directions of the target class distribution are crucial for one-class classification. In [22], it has been shown that projecting the data in the high variance directions (like PCA) will result in higher error (bias), while retaining the low variance directions will lower the total error. Boundary-based methods do not put any special emphasis on these low variance directions. In fact, these methods preferentially separate data along large variance directions [21]. However, finding the optimal number of directions to retain is also not possible because of the basic bias-variance dilemma [23]. This dilemma arises from the fact that the estimated covariance is not accurate due to the limited number of training samples. Hence, we need to reduce the estimation error by taking projections along some variance directions. However, taking these projections before training will increase the total error (bias) since we are losing important characteristics from the training data. It is nearly impossible to find out the perfect projectional directions which will generate the lowest total error in case of all datasets.

The principal motivation behind our proposed method is to use the robustness of the boundary-based classifiers while emphasizing the small variance projectional directions. We want to use the maximum margin based solution while optimally controlling the projectional direction. However, we do not want to lose any data characteristics by directly taking projections in specific directions before training. Generally, the estimated covariance matrix represents necessary information regarding the required variational directions. Hence, our approach incorporates the covariance matrix into the well-known One-class Support Vector Machine (OSVM) method [20]. The OSVM method is essentially a one-class interpretation of the classic SVM problem. We call our proposed method the Covariance-guided One-Class Support Vector Machine (COSVM) method.

Some recently proposed methods try to incorporate the covariance matrix or some other form of data in SVM that takes the overall structural information of the training set into account. The Relative Margin Machines (RMM) method [21] tries to control the spread of the data with SVM by applying an additional constraint to the optimization problem. The Discriminant Analysis via Support Vectors (SVDA) [24] only uses the support vectors to calculate the between-class and within-class scatter matrices and finds the optimal projectional directions through them. The Manifold Regularization approach [25] tries to exploit the geometry of the overall distribution of the training data in SVM. Wang and Niu [26] extend the manifold regularization technique to be more useful in case of one-versus-all SVM.

In the one-class domain, the most relevant work is the Mahalanobis One-class SVM (MOSVM) [27], where instead of the traditional Euclidean distance of OSVM, the Mahalanobis distance is used. However, calculating the Mahalanobis distance and incorporating it into the aforementioned method involve inversion of both the covariance matrix and the kernel matrix. Real datasets are often high-dimensional, and the sample points are limited. These lead to the so-called small sample size problem, which can provide suboptimal results when matrix inversion is involved [28]. Also, the method in [27] is based on a different philosophy and does not provide any guidance as to how much emphasis should be put on the covariance matrix. Merely using Mahalanobis distance does not ensure that the low variance directions will be provided special importance, which is important for one-class classification [22]. Unlike this method, the focus of our method is to put more emphasis on the low variance directions while keeping the basic formulation of OSVM untouched, so that we still have a convex optimization problem with no suboptimal calculation like matrix inversion involved.

In our proposed COSVM method, we plug the covariance matrix into the optimization problem of OSVM. The covariance matrix is estimated in the kernel space [29]. The estimated covariance matrix has the required information for all projectional directions, both along high variance and low variance. However, the OSVM optimization problem is a *minimization* problem. Hence, we can intuitively assume that incorporating the covariance matrix into the optimization problem of OSVM as an additional term will result in emphasizing the low variance directions. The additional term will essentially act as a penalty factor. The high variance directions will be penalized more than the low variance directions. Hence, after training, the weight vector will be adjusted in such a way that the low variance directions are assigned more weight (importance) than the high variance directions. This technique does not increase the overall computational complexity of the OSVM method. COSVM still results in a convex optimization problem with one global optimum solution which can be found efficiently using existing numerical methods. The degree of emphasis on the covariance matrix can be elegantly controlled through one parameter only (details in Section 3). This provides a quick control over the bias-variance dilemma. The performance of our classifier can be tuned very easily by changing this single