2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Locality Linear Fitting One-class SVM with Low-Rank Constraints for Outlier Detection

Sheng Li, Ming Shao and Yun Fu

Abstract— **We propose a novel outlier detection approach in this paper, which learns the most accurate hyperspheres for the normal data through a top-down procedure. Conventional one-class support vector machine (SVM) based approaches aim to find nonlinear global solutions for all the normal data, with the benefit of kernel trick. However, those methods are intractable when data are in large-scale and inaccurate when data are under complex distributions. It's observed that high dimensional data, e.g., features of texts or images, are always sparse, and linear classifier usually performs well. A specific class of data seldom lie in one single subspace. In this paper, we propose to learn multiple discriminative hyper-spheres locally based on the data distributions, and fit them globally to formulate a more discriminative boundary for the normal data. By far, neural mechanisms used by human brain-mind for outlier detection are not known, however, the top-down strategy proposed in this paper would inspire understanding of the human neural mechanisms. The benefits of our model are two-folds. First, the distribution of each local cluster is much simpler than that in a global view, which makes the fitting processing for each individual cluster much easier and insensitive to the choice of kernel. In particular, we adopt low-rank constraints to find multiple clusters automatically. Secondly, the proposed approach trains the model linearly which tackles the large-scale problem, substantially reducing training time and memory space. Extensive experimental results on three image databases demonstrate that our approach outperforms several related methods.**

I. Introduction

OUTLIER detection is an important research topic in the areas of data mining and pattern recognition. The aim of outlier detection is to identify abnormal samples in a given database. Outlier detection is also referred to as anomaly detection, which has been widely applied to numerous real-world applications, such as fraud detection and fault identification.

A successful outlier detection algorithm should be able to discover the unseen data space. Specifically, if a data point doesn't conform the distribution of normal data, it could be denoted as an outlier. An intuitive idea is to learn a region that contains (almost) all the normal data. As a result, any new samples that couldn't be covered by this region are outliers. However, this idea is usually infeasible due to the following reasons. First, a normal region, encompassing every possible normal behaviors, is hard to define. Secondly, the boundary between normal and outlying behaviors is usually very fuzzy. Therefore, an outlying observation which is close to this boundary can be normal. Finally, in some cases, the representations of normal data may not be correct for future observations. Therefore, how to correctly and effectively define positive samples becomes the most important topic in outlier detection techniques.

Currently, there are two types of outlier detection approaches: statistical parametric approaches and non-parametric approaches [1]. Statistical parametric approaches assume that the underlying distribution of observations is known [2], [3], [4]. Other statistical techniques are based on estimating the parameters of distributions [5]. In these methods, outliers are denoted as observations that deviate from the assumptions of underlying models. The techniques are usually not suitable for the high-dimensional data due to the lack of prior knowledge of data distribution. In contrast, nonparametric methods are model-free. As distance-based methods, those methods usually make use of local distance measurements, and are suitable for dealing with large-scale databases [6], [7], [8], [9], [1].

Within the non-parametric methods, people break down complex problems into small ones through clustering, and the problem turns to be small clusters based outlier detection [10], [11], [12], [13]. Recently, one-class SVM has been introduced into outlier detection (anomaly detection) as a non-parametric method. The motivation of one-class SVM is to find a nonlinear hyper-sphere in the higher dimensional space to enclose as many samples as possible by a tight radius. Theoretically, it is possible to find a hyper-sphere to include each training samples in the higher dimensional space, which is fulfilled by kernel trick. However, in some cases, the training samples are under complicated distribution, and the cluster boundaries are fuzzy. In these cases, kernel plays a critical role to map messy data to a RKHS (reproducing kernel Hilbert space) where data are linearly separable. Nonetheless, the choice of kernel and other parameters are part of model selection and require a deep understanding of underlying distribution of training samples. Usually, obtaining this prior knowledge is expensive or even impossible.

Motivated by the above considerations, we propose a novel outlier detection approach in this paper. Our approach follows a top-down schedule, which breaks down a complex data set into several smaller sets, and then finds the proper boundary with respect to each individual group. In order to obtain stable and kernel insensitive results, we only consider the training samples group by group, and low-rank constraints are employed to identify the cluster membership automatically. For each individual group, the best radius is determined merely based on data within this group. Intu-

Sheng Li, Ming Shao and Yun Fu are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA (email: {shengli, mingshao, yunfu}@ece.neu.edu).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

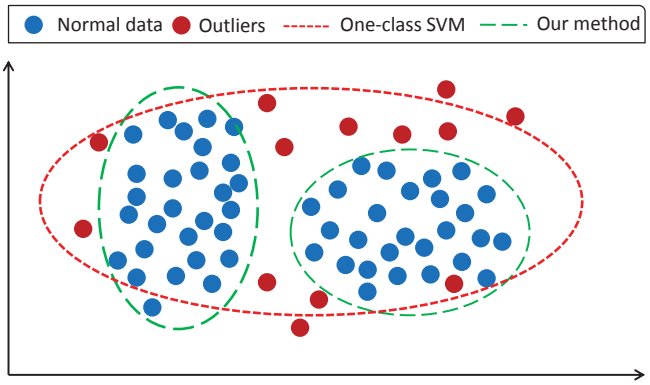

Fig. 1. Difference between the boundary found by one-class SVM (red dot lines) and the proposed method (green dot line). Blue points indicate the positive training samples, while red points represent the negative samples, i.e., outliers.

itively speaking, the original one-class SVM computes the best radius based on the global fitting whereas the proposed method finds the optimized radius for each local group and updates them globally.

By far, neural mechanisms used by human brain-mind for outlier detection are not known, however, the top-down strategy proposed in this paper would inspire understanding of the human neural mechanisms. Compared with one-class SVM, the proposed method benefits in two aspects. First, we partition the complex set of training sample into several simpler clusters, leading to several tractable problems. In ideal case, we can easily find the linear solution for the boundary in the original space. Secondly, the computation for finding the optimal boundary could be accelerated through parallel computing. It should be notified that, when the nonlinear solution is reduced to a set of linear solutions, the kernel trick becomes unnecessary, and it enables our framework to potentially scale up to large scale data sets. Recently, multi-sphere support vector data description (SVDD) methods [14], [15], [16] have also been introduced to extend the traditional one-class SVM. Our method differs from the Multi-sphere SVDD approach in that we employ the low-rank constraint to group sample set into different clusters automatically, and we also present two different optimization algorithms to address this problem.

The rest of the paper is organized as follows. We review the related works and discuss how they differ from our proposed method in the next section. In Section III, the methodology is presented, including the problem statement and optimization algorithms. The experimental results and discussions are reported in Section IV. Section V concludes this paper.

II. Preliminary

One-class SVM was first proposed in [17], which estimates a small set S that contains most normal data in the input space. Meanwhile, the abnormal data should lie outside of the set S. Recently, one-class SVM becomes more and more popular in outlier detection [18], [19], [20]. When the data distributions are similar between different clusters, it is easy to find the optimized radius and center. However, in some cases, when the distribution of the data set is very complex, one-class SVM may not converge well and be trapped by some local minima. This becomes obvious when data distributions in distinct clusters are significantly different and clusters are far from each other. As a result, one-class SVM fails to seek the best hyper-sphere. Usually, one-class SVM has a strong assumption for the distribution of data. It performs best on Gaussian or Gaussian-like distributed data, which makes the center's location critical. Typically, when the data are not contaminated by outliers, the mean and variance of samples could provide good estimations for the data distribution. When the data set contain outliers, the performance of outlier detection would be significantly affected by those parameters.

Moreover, one-class SVM is also very sensitive to the choice of kernel functions, because one-class SVM needs to project the original data into a high dimensional feature space in order to find a perfect hyper-ball by kernel techniques. However, this is not necessarily useful for the linearly separable or sparse data, especially when the data already lie in high dimensional space, e.g., images. The outcome is a huge cost of memory in the order of $\mathrm{O}(n^2)$, where n is the number of samples in the data set. In addition, the choice of kernels depends on the actual distribution of data, adding more burden to the model selection process.

In Fig.1, there are two clusters with different distributions. We can observe that the left cluster is denser than the right one. Since one-class SVM always tries to find the global solution for all the training data, the final solution needs to cover most of the positive data, which enlarges the radius of the hyper-sphere (the red dot lines in Fig. 1). As a result, many outliers are contained in the hyper-sphere incorrectly. Compared with one-class SVM, our proposed method could divide this problem into two separate problems, and find the optimal hyper-spheres incrementally (green lines).

In our method, we will first partition the training samples into different segments, then find the best boundary for each cluster independently. Compared with one-class SVM, our method divides the complex problem into several simpler problems. In some cases, these optimization even can be solved linearly, which means we do not even need to use kernel trick to find the hyper-sphere in higher dimensional space. Without using kernel, our method could have more stable results and is efficient for large-scale problems.

III. Methodology

In this section, we describe the proposed approach for learning the centers and their corresponding radius automatically. Then we describe the optimization algorithms.

A. Problem Statement

Let X denote a set of observations in the D-dimensional sample space, i.e., $X = [x_1, x_2, \cdots, x_N] \in \mathbb{R}^{D\times N}$. N represents the number of observations. $C = [c_1, c_2, \cdots, c_M] \in \mathbb{R}^{D\times M}$ is a center matrix for the clusters and M is the number of clusters. $R = [r_1, r_2, \cdots, r_M] \in \mathbb{R}^{M}$ represents

the radius of the hyper-sphere for the corresponding cluster, i.e., r_i represents the radius for c_i. In order to associate each observation with its center and the center's radius, we employ an indicator matrix $U = [u_1, u_2, \cdots, u_N] \in \mathbb{R}^{M\times N}$ to determine which cluster each sample x_i belongs to.

Our strategy is to put most of the data into the hyper-spheres. We want to use fewer hyper-balls with the smallest radius, including most of the training data at the same time. Intuitively, all the samples in the sample space could be included by only one hyper-sphere with a large radius. This is the most simplest way to use the hyper-sphere and its radius to represent the data. However, the large radius will also include undesired false positive samples (i.e., outliers) in the testing phase. Therefore, we need to "tighten" the hyper-ball. Since it's expensive to know the distribution of data in most cases, in order to reduce the radius which include the most of the data, it is reasonable to introduce more clusters instead of a single one. In the most extreme case, we could set all the samples as the clusters' centers themselves with radius 0. However, this representation over-fits the training samples, which causes a lot of false negative in the testing. Furthermore, when the training samples are corrupted, this representation will be even worse. Therefore, we need to balance the number of the clusters and the radius for each cluster.

In the training stage, we only consider the normal data. Based on the above observations, we propose an objective function as follows:

$$\begin{aligned} &\min_{C,U,R,\xi} \mathrm{Rank}(U) + \tfrac{1}{\lambda_1 N}\|U^{\mathrm{T}}R\|^2 + \tfrac{1}{\lambda_2 N}\sum_i \xi_i \\ &\text{s.t.,} \quad \forall x_i \ \|x_i - Cu_i\|^2 \le \|u_i^{\mathrm{T}}R\|^2 + \xi_i, \quad \xi_i \ge 0. \end{aligned} \tag{1}$$

Our method always starts with over complete clusters, which means all the clusters will be selected by some samples around the clusters' centers via matrix U. This makes the indicator matrix U in Eq.(1) be a full rank matrix. In our cost function, we encourage the rank of U be smaller. This means some of the centers will not be selected by any of the training samples, which enables our framework to not only select the centers automatically but also change the number of clusters based on the distribution of training data. Note that we use $u_i^{\mathrm{T}}R$ and Cu_i to represent the cluster's radius and the center of x_i correspondingly. $\|U^{\mathrm{T}}R\|^2$ in Eq.(1) sums up all the radius of the selected clusters, which decides the tightness of the hyper-sphere globally. In order to put all the samples into the corresponding hyper-sphere, we add one more constraint into the objective function. Basically, what the constraint does is to assure that the distance between each sample and the center of the corresponding cluster is smaller than the cluster's radius with a small error tolerance ξ. The trade-off between the number of samples it can hold and the radius of hyper-sphere is controlled by the parameters λ_1 and $\lambda_2 \in [0, 1]$.

B. Optimizing Hyper-Sphere from Convex Problem

The optimization problem in Eq.(1) is difficult to solve because of the non-convexity of rank function. Usually, the rank function can be replaced by the nuclear norm (or trace norm), and we obtain the following optimization problem:

$$\begin{aligned} &\min_{C,U,R,\xi} \|U\|_* + \tfrac{1}{\lambda_1 N}\|U^{\mathrm{T}}R\|^2 + \tfrac{1}{\lambda_2 N}\sum_i \xi_i \\ &\text{s.t.,} \quad \forall x_i \ \|x_i - Cu_i\|^2 \le \|u_i^{\mathrm{T}}R\|^2 + \xi_i, \quad \xi_i \ge 0. \end{aligned} \tag{2}$$

The solution to the optimization problem in Eq.(2) is not straightforward, since there are multiple parameters to be optimized. For efficiency, we adopt the inexact Augmented Lagrange Multiplier (ALM) [21], [22] to solve the above problem due to its successful application in low-rank representation in [22], [23], [24]. We first rewrite Eq.(2) to the following equivalent formulation:

$$\begin{aligned} \min_{J,C,U,R,\xi} \quad & \|J\|_* + \tfrac{1}{\lambda_1 N}\|U^{\mathrm{T}}R\|^2 + \tfrac{1}{\lambda_2 N}\sum_i \xi_i \\ \text{s.t.,} \quad & \forall x_i \ \|x_i - Cu_i\|^2 \le \|u_i^{\mathrm{T}}R\|^2 + \xi_i, \\ & U = J, \ \ \xi_i \ge 0. \end{aligned} \tag{3}$$

The augmented Lagrange function of (3) is:

$$\begin{aligned} \mathcal{L} = \ & \|J\|_* + \tfrac{1}{\lambda_1 N}\|U^{\mathrm{T}}R\|^2 + \tfrac{1}{\lambda_2 N}\sum_i \xi_i \\ & + \sum_i \alpha_i(\|x_i - Cu_i\|^2 - \|u_i^{\mathrm{T}}R\|^2 - \xi_i) \\ & + \mathrm{tr}(Y^{\mathrm{T}}(U - J)) + \tfrac{\mu}{2}\|U - J\|_{\mathrm{F}}^2 - \sum_i \beta_i \xi_i. \end{aligned} \tag{4}$$

The problem shown in Eq.(4) is unconstrained. Therefore, we can separately minimize the variables J, U, R, C, when fixing the other variables. The Lagrange multipliers α_i and Y could also be updated in this procedure, where $\mu > 0$ is a penalty parameter.

Here we show how to update J_{k+1}, C_{k+1}, R_{k+1} and U_{k+1} iteratively. After dropping the irrelevant terms w.r.t. J, (4) can be rewritten as

$$J_{k+1} = \min_{J_k} \ \tfrac{1}{\mu_k}\|J_k\|_* + \tfrac{1}{2}\|J_k - (U_k + (Y_k/\mu_k))\|_{\mathrm{F}}^2 . \tag{5}$$

Problem (5) could be effectively solved by using the singular value thresholding (SVT) algorithm [25]. Let $U_J\Sigma_J V_J$ denote the SVD of matrix $U_k + (Y_k/\mu_k)$, where $\Sigma_J = \mathrm{diag}(\{\sigma_i\}_{1\le i\le r})$, r is the rank, and σ_i are the singular values. Then, the optimal solution $J_{k+1} = U_J\Omega_{(1/\mu_k)}(\Sigma_J)V_J$, where $\Omega_{(1/\mu_k)}(\Sigma_J) = \mathrm{diag}(\{\sigma_i - (1/\mu_k)\}_+)$, and t_+ means the positive part of t.

By ignoring terms independent of C in (4) and doing some derivations, we have

$$\frac{\partial \mathcal{L}}{\partial C} = 2\sum_i \alpha_i(Cu_i - x_i)u_i^{\mathrm{T}}. \tag{6}$$

In a similar way, we have

$$\frac{\partial \mathcal{L}}{\partial R} = 2(\frac{1}{\lambda_1 N}UU^{\mathrm{T}} - \sum_i \alpha_i u_i u_i^{\mathrm{T}})R. \tag{7}$$

$$\begin{aligned} \tfrac{\partial \mathcal{L}}{\partial U} = \ & \tfrac{2}{\lambda_1 N}RR^{\mathrm{T}}U + Y + \mu(U - J) \\ & -2\sum_i (C^{\mathrm{T}}x_i - C^{\mathrm{T}}CUv_i - RR^{\mathrm{T}}Uv_i)u_i^{\mathrm{T}}. \end{aligned} \tag{8}$$

Algorithm 1. *Solving Eq.*(3) *by Inexact ALM*

Input: Sample set X, parameters λ_1, λ_2, η, $J = 0$, $Y = 0$, $\xi = 0$, $\mu_{\max} = 10^6$, $\mu = 10^{-6}$, $\rho = 1.1$, $k = 1$, $\epsilon = 10^{-8}$

Output: C_k, U_k, R_k, ξ_k

1:**while** *not converged* **do**
2: update J_{k+1} using (5), given others fixed
3: update C_{k+1} using (6), given others fixed
4: update R_{k+1} using (7), given others fixed
5: update U_{k+1} using (8), given others fixed
6: update ξ_{k+1} by gradient descent using $\frac{\partial \mathcal{L}}{\partial \xi_i} = \frac{2}{\lambda_2 N} - \alpha_i - \beta_i$.
7: update β_{k+1} by gradient descent using $\frac{\partial \mathcal{L}}{\partial \beta_i} = -\xi_i$.
8: update the multipliers Y_{k+1} and α_{k+1}
$Y_{k+1} = Y_k + \mu_k(U - J)$
$\alpha_{k+1} = \alpha_k + \mu_k(\|x_i - Cu_i\|^2 - \|u_i^{\mathrm{T}} R\|^2 - \xi_i)$
9: update the parameter μ_{k+1} by $\mu_{k+1} = \min(\rho\mu_k, \mu_{\max})$
10: check the convergence conditions $\sum_i \|x_i - Cu_i\|^2 - \|u_i R\|^2 - \xi_i < \epsilon$ and $\|U_{k+1} - J_{k+1}\|_\infty < \epsilon$.
11: $k = k + 1$
12: **end while**

The detailed algorithm of our optimization is outlined in Algorithm 1.

In Algorithm 1, both step 2 and step 3 are convex problems, and we can solve them using closed-form solutions. Specifically, Step 3 is solved by traditional optimization method in order to avoid potential non-invertible matrix $u_i u_i^{\mathrm{T}}$, which contains in its closed-form solution. In addition, we set $\epsilon = 10^{-8}$ due to the assumption that all the samples have been normalized in the range of $[0 \sim 1]$. In order to simplify the solution in step 5 in Algorithm 1, we introduce Uv_i to replace the original u_i for matrix derivatives.

C. Dual Form Solution

Even Eq.(4) can be trivially solved through Algorithm 1, its convergence is hardly guaranteed especially when the training samples are in large-scale and high dimensional space. Notably, in Algorithm 1, there are 8 variables need to be updated: C, U, J, R, ξ and other three Lagrange multipliers. Inspired by one-class SVM, we find that it is reasonable to use the dual form to optimize the objective function instead of the primal form. There are two major reasons. First, the dual problem of SVM reveals the kernel structure, which provides more flexibility. Secondly, the dual form solution has an advantage of representing some parameters by Lagrangian multipliers, and therefore removing the slack variable if linear penalty term is adopted. This also allows the solution to open to the high-dimensional space.

In the dual form of one-class SVM, the center C is represented by a weighted combination of observations in kernel space $C = \sum_i \alpha_i \Phi(x_i)$, where $\Phi(\cdot)$ is the kernel function. In addition, the original parameter R and ξ are converted into constraint in the new dual form formulation. In the dual form, only two parameters need to be optimized, namely, α_1 and α_2. And the α_1 can be determined once α_2 is found. Therefore, they reduced the problem into a simpler optimization problem.

As we discussed above, our primal objective function has 8 variables, imposing a huge or even intractable solution space. We will reduce the number of objective variables by virtue of the similar dual form formulation. By introducing kernel $\Phi(\cdot)$, the original Lagrange function turns to be:

$$\begin{aligned}\mathcal{L} = \quad & \|J\|_* + \tfrac{1}{\lambda_1 N}\|U^{\mathrm{T}} R\|^2 + \tfrac{1}{\lambda_2 N}\sum_i \xi_i \\ & + \sum_i \alpha_i(\|\Phi(x_i) - Cu_i\|^2 - \|u_i^{\mathrm{T}} R\|^2 - \xi_i) \\ & + \mathrm{tr}(Y^{\mathrm{T}}(U - J)) - \sum_i \beta_i \xi_i.\end{aligned} \tag{9}$$

From the derivatives respect to R, we have:

$$(\frac{1}{\lambda_1 N} U U^{\mathrm{T}} - \sum_i \alpha_i u_i u_i^{\mathrm{T}}) R = 0. \tag{10}$$

We know that R can only be real positive value, and therefore we get the constraint:

$$\hat{\alpha} = \frac{1}{\lambda_1 N}, \tag{11}$$

where $\hat{\alpha} = \sum_i \alpha_i v_i v_i^T$, $u_i = U v_i$. This becomes a constraint in the later dual form.

For ξ_i, we also have:

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = \frac{2}{\lambda_2 N} - \alpha_i - \beta_i = 0. \tag{12}$$

Since $\beta_i > 0$ always holds, we obtain another more tight constraint:

$$0 \le \alpha_i \le \frac{1}{\lambda_2 N}. \tag{13}$$

In the case of C, we replace it by

$$C = \sum_i \alpha_i \Phi(x_i) v_i^{\mathrm{T}} \tilde{\alpha} U^{-1}. \tag{14}$$

Therefore we can replace the C in the term $\Phi(x_i) - Cu_i$, and naturally obtain

$$\Phi(x_i) - Cu_i = \Phi(x_i) - \sum_i \alpha_i \Phi(x_i)\hat{\alpha}. \tag{15}$$

From Eq.(15), it is clear to recognize that the original selected center C becomes a linear combination of $\Phi(x_i)$, which means we can eliminate C from the original problem. After we replace R, C and ξ with new representations, we can reformulate the original problem with the dual form by maximizing:

$$\begin{aligned}&\hat{\mathcal{L}} = \max_{U,J,\alpha_i} \|J\|_* + \mathrm{tr}(Y^{\mathrm{T}}(U - J)) + \sum_{i,j} \alpha_i \alpha_j X_{i,j} \hat{\alpha}^2 \\ &\text{s.t. } 0 \le \alpha_i \le \tfrac{1}{\lambda_2 N},\ \hat{\alpha} = \tfrac{1}{\lambda_1 N},\end{aligned} \tag{16}$$

where $X_{i,j}$ is the linear relationship between x_i and x_j. Afterwards, we only have four parameters that need to be

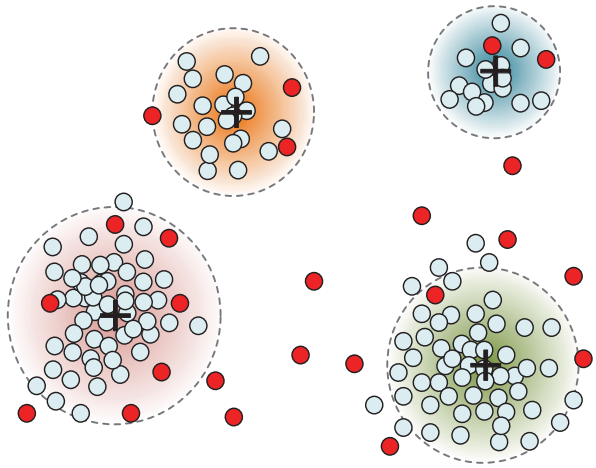

Fig. 2. Instead of finding the nonlinear solution globally, our proposed framework only finds the linear solution for each individual cluster. Light color sample points represent positive training samples. The red sample points represent the negative samples(outliers). The black "+" sign marks the center of each cluster.

optimized, namely, U, J, α_i, Y , each of which has its own solution. First, α_i can be tackled by following the routine solution of one-class SVM since we fix all the other variables. Second, parameter J can be solved through SVT [25] with the close form solution as shown in (5).

And U can also be solved with the close-form formulation:

$$\begin{aligned} U = \quad & (\tfrac{2}{\lambda_1 N} RR^{\mathrm{T}} + uI + CC^{\mathrm{T}} + 2RR^{\mathrm{T}})^{-1} \\ & (uJ - Y + 2\sum_i C^{\mathrm{T}} x_i v_i^{\mathrm{T}}). \end{aligned} \tag{17}$$

Finally, Y can be updated through the same method in Algorithm 1, $Y = Y + \mu(U - J)$.

Note that when the training data set is in large-scale, it is highly expensive to find the nonlinear solution to the objective function. In some cases, it is impossible to get nonlinear solution due to the kernel in the objective function. Since the complex data set is divided into several clusters in our method, it is not necessary to find the nonlinear solution for clusters anymore. In this case, we can just simply employ the linear kernel to replace original nonlinear kernel to find linear solutions. Fig. 2 illustrates our idea of dividing sample set into multiple clusters and seeking optimal hyper-spheres for each of them.

For instance, let's consider the optimization over α_1 and α_2 with all other variables fixed. Since the first two terms are independent with α, we discard them in the optimization of α. Then, the Eq.(16) reduces to

$$\min_{\alpha_1,\alpha_2} \frac{1}{2} \sum_{i,j}^{2} \alpha_1 \alpha_2 X_{i,j} \hat{\alpha}^2 + \sum_i \alpha_i G_i + G, \tag{18}$$

where

$$\begin{aligned} & G_i = \sum_{j=3}^{N} \alpha_j X_{i,j} \hat{\alpha}^2, \\ & G = \sum_{i,j=3}^{N} \alpha_i \alpha_j X_{i,j} \hat{\alpha}^2, \\ & s.t.\ \ 0 \le \alpha_1,\ \alpha_2 \le \tfrac{1}{\lambda_2 N},\ \sum_{i=1}^{2} \alpha_i = \Delta, \end{aligned} \tag{19}$$

where $\Delta = 1 - \sum_{i=1}^{2} \alpha_i$. We discard G, which is independent of α_1 and α_2, and eliminate α_1 to obtain

$$\begin{aligned} \min_{\alpha_2} \tfrac{1}{2} \quad & (\Delta - \alpha_2)\alpha_2 X_{1,1} + (\Delta - \alpha_2)\alpha_2 X_{1,2} \\ & + \tfrac{1}{2}\alpha_2^2 X_{2,2} + (\Delta - \alpha_2) G_1 + \alpha_2 G_2, \end{aligned} \tag{20}$$

with the derivative respect to α_2

$$\begin{aligned} \tfrac{\partial \hat{\mathcal{L}}}{\partial \alpha_2} = \quad & -(\Delta - \alpha_2)X_{1,1} + (\Delta - \alpha_2)X_{1,2} \\ & + \alpha_2 X_{2,2} - G_1 + G_2. \end{aligned} \tag{21}$$

Setting this to zero and solving for α_2, we get

$$\alpha_2 = \frac{\Delta(X_{1,1} - X_{1,2}) + G_1 - G_2}{X_{1,1} + X_{2,2} - 2X_{1,2}}. \tag{22}$$

Once α_2 is found, α_1 can be revered from $\alpha_1 = \Delta - \alpha_2$. Algorithm 2 shows the details of our dual form solution. In the algorithm, we keep the SVT solution for J. U will be updated for each iteration after the α is changed.

Algorithm 2. *Dual form algorithm*

Input: data matrix X, parameter λ_1, λ_2, $Y = 0$, $J = 0$, $\mu = 10^{-6}$, $\mu_{\max} = 10^6$, $\rho = 1.1$, $\epsilon = 10^{-8}$

Output: U, α_1, α_2

1:**while** *not converged* **do**
2: update J using (5), given others fixed
3: update α_1, α_2 using (18)-(22), given others fixed
4: update U using (17), given others fixed
5: update the multipliers Y
$Y = Y + \mu(U - J)$
6: update the parameter μ by
$\mu = \min(\rho\mu, \mu_{\max})$
7: check the convergence conditions
$\|U - J\|_\infty < \epsilon$.
8: **end while**

D. Outlier Detection

For training purpose, the input samples only include normal data. After optimizing the objective functions, our method returns an indicator matrix, a center matrix, and the corresponding radius. With all these information, our detection strategy is very straightforward. For a given test sample, our method first finds the cluster with the closest center. After the closest center is selected, the distance between the sample and the center will be calculated. The sample is considered to be a member of that cluster only if it is within its radius. If it is outside the cluster boundary, then the the next closest cluster is considered. This process

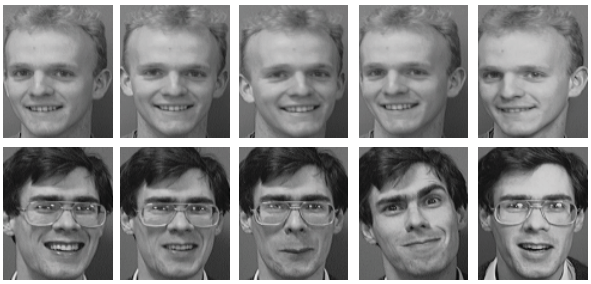

Fig. 3. Example images in ORL database.

continues until either all clusters are considered or the cluster containing the sample is determined. If the sample is not within any cluster, then it is labelled as an outlier.

Our method provides two important parameters which can be used for some interesting applications, such as active learning. The first parameter is the error ξ. This parameter indicates the extra distance of the sample from the closest cluster besides its radius. In some areas (e.g., astronomy) the ability to identify rare phenomena is of great importance. These rare observations can be identified by thresholding ξ. Secondly, the composition of each cluster can also provide valuable information. Depending on the situation some clusters can be removed from consideration. For example, different types of vehicles (e.g. SUVs, trucks, sedans) should belong to distinct clusters. If a user is only interested in SUVs and trucks, then when classifying new samples, the sedan cluster can safely be removed, making our system very flexible to adjust to different circumstances.

IV. Experiments

In this section, we evaluate the performance of our method by three groups of experiments on the ORL, Hong Kong PolyU NIR and CMU PIE databases. Our method is compared with three representative algorithms: the original one-class SVM [17], replicator neural networks (RNNs) [7], and manifold clustering [26]. Each model is trained on the normal data and tested on the data set with outliers. The following experiments will demonstrate that the proposed method is capable to to capture the properties of normal data. Since the three databases we used in this paper are face images with simple backgrounds, we only use the raw pixel values as the input features. Only histogram equalization is used for preprocessing.

We use two metrics to evaluate the performance of all compared methods:

- True positive rate (TPR) is defined as follows:

$$TPR = \frac{TP}{TP + FN}, \tag{23}$$

  where TP represents the true positives, and FN represents the false negatives.
- False positive rate (FPR) is defined as follows:

$$FPR = \frac{FP}{FP + TN}, \tag{24}$$

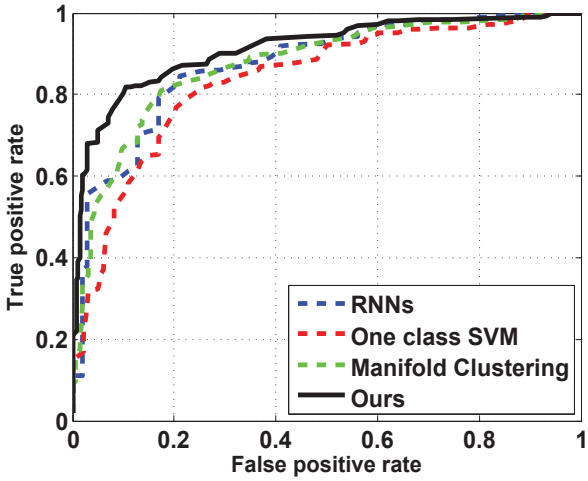

Fig. 4. Results of the experiments on ORL database displayed as ROC curves.

where TN represents the true negatives, and FP represents the false positives.

A. ORL Database

The ORL database is used in our first group of experiments. This database contains a total of 40 different subjects. For each subject, there are 10 different images, which were taken at different times, varying the lighting and facial expressions. Some example images in this database are shown in Fig. 3. By preprocessing, we identify and locate all the faces. The images were normalized in scale and orientation to make sure two eyes were aligned at the same position across different images. Then, we crop the facial areas from the original image as raw feature for matching. The size of each image in the experiments is 32 by 32 pixels, resulting in a 1024 dimensional feature vector. We randomly select 6 images from each subject to construct the training set. The remaining 4 images and other 4 images which are randomly selected from other subjects are used as the testing set. In this case, the images from other subjects are considered as outliers.

Fig.4 compares the performance of our method with the other three algorithms. Our method provides the flexibility while performing slightly better than all the other algorithms. The limited increase in performance is most likely due to the small number of training samples is very sparse in the high-dimensional space. When the number of training sample is small, the difference between nonlinear and linear solution will be very closed to each other.

B. Hong Kong PolyU NIR Database

Hong Kong PolyU Near-infrared (NIR) database [27] contains 335 subjects and 100 images for each subject. PolyU NIR database is collected by a real time NIR face capture device. The related version of Hong Kong PolyU NIR database we used in our paper contains 55 subjects, each of which comprises six expressions, e.g., anger, disgust, fear, happiness, sadness and surprise, and different poses. Fig. 5

Fig. 5. Demo images in HK PolyU NIR database.

Fig. 7. Example images in PIE database.

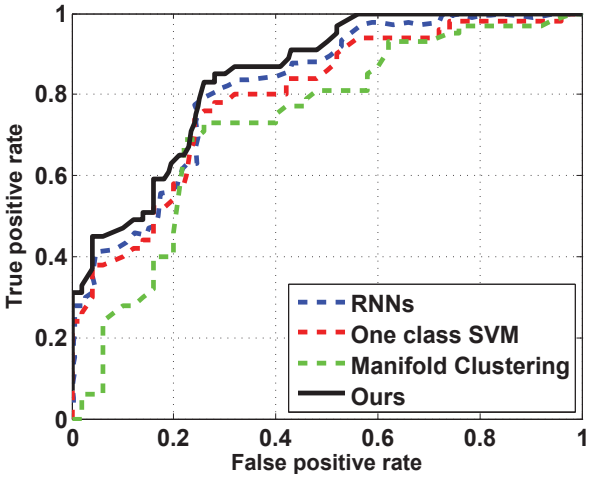

Fig. 6. Results of the experiments on Hong Kong PolyU NIR database displayed as ROC curves.

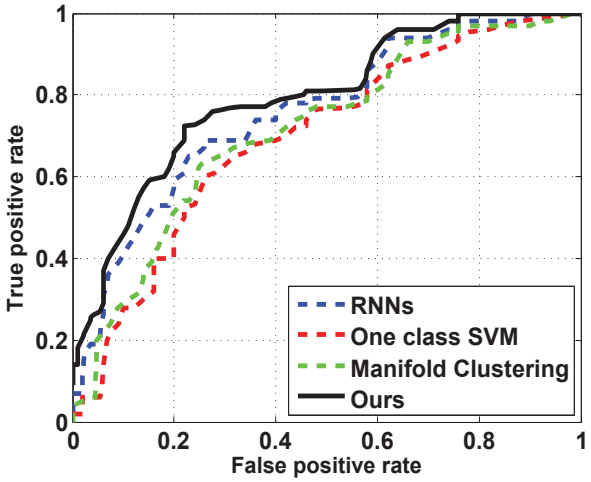

Fig. 8. Results of the experiments on PIE database displayed as ROC curves.

shows some example images in the HK PolyU NIR database. The size of each image is 32 $\times$ 32 pixels. Our training set contains 55 subjects with 80 images for each subject. For the testing purpose, we randomly select 40 images from the rest 54$\times$20 (exclude the image from itself) as our positive samples (outliers). The 20 negative samples (normal data) are selected from the rest of the images from the corresponding subject. Then we take average among all the different 55 subjects to get the final recognition accuracy.

Fig.6 shows the performance of all compared methods on PolyU NIR database. From our observation, we conclude that our proposed method not only outperforms the original one-class SVM but also performs better than RNNs. From the perspective of manifold, these face images should lie in the image space with several clusters, since there are different expressions related with these images. The proposed method will first partition these clusters and learn the center and radius linearly. In this database, since there are no illumination changes included, the centers of different clusters are not very far away from each other. In this case, the improvement of our method is not very significant. In some cases our method is just slightly better than others.

C. PIE Database

The CMU Pose, Illumination, and Expression (PIE) database [28] is used for further evaluating the performance of all compared methods. PIE database contains 68 subjects, with 41,368 face images. The face images were captured by 13 synchronized cameras and 21 ashes, under varying pose, illumination, and expression. In our experiments, 11,560 different faces from 68 different people are used. Fig. 7 shows some example images in the PIE database. Compared with Hong Kong PolyU NIR data set, significant changes of the pose, illumination and expression conditions are incorporated in these experiments. Another reason makes CMU PIE distinct is that it provides sparse image sequence rather than a dense one sampled from a video. The continuity of data set has been ruined by distinct illumination, pose, or expression, which is not a common assumption in previous work. Rather it could be a scenario in real world applications. Therefore, we test all the algorithms in this database and compare their performance in a more general case.

Similar to the previous experiments setup, we randomly select 80 images from each individual as our training samples. Then we build the test sample set by random selecting 40 images from the rest images of each subjects, as our positive samples (outliers). And we also select 20 negative samples from the rest of images, and add them into the test set. Then the final results are computed from the average of these 68 classes. The image size is fixed at 32$\times$32.

Fig.8 shows the different results of different methods for PIE database. Compared with the previous databases, PIE database not only has different expressions, but also has illumination changes within each subject. The face images

TABLE I

AUC OF ALL COMPARED METHODS ON THREE DATABASES.

Databases	RNNs [7]	One-class SVM [17]	Manifold Clustering [26]	Ours
ORL	0.8761	0.8462	0.8700	**0.9128**
HK PolyU NIR	0.7530	0.6893	0.7066	**0.7898**
PIE	0.8140	0.7880	0.7290	**0.8420**

for each training sample should be divided into more clusters than the previous databases, i.e., similar face expression with different poses. It is harder to obtain a global solution for the training samples with one-class SVM. However, in our method, we partition these samples into several clusters and find the optimal for each cluster. In this case, our method simplifies the distribution which makes the solution easier to obtain. From Fig.8, we conclude that the difference between two methods are larger than the previous case. Recall the experimental results in two previous databases, we notice that our proposed method performs better when the training sample distribution is more complex and denser. Therefore, we can conclude that the linear solution for each cluster is better than the nonlinear global solutions when the distribution of training samples is more complex.

Table 1 reports the AUC (area under ROC curve) scores of all compared methods on three databases. It shows that our method consistently outperforms related methods.

V. CONCLUSIONS

In this paper, we introduce a novel outlier detection framework by extending the one-class SVM approach. The original one-class SVM tries to find a nonlinear global solution to include most of the training samples with a tight hyper-sphere through the use of kernel trick. Since the computation for nonlinear kernel are really expensive when the data set is large, we design a novel framework to tackle this problem linearly. Compared with one-class SVM, we first partition the data set into several simpler clusters, then try to find the linear solution for each individual cluster separately. We build up our global solution by combining these local solutions together. When a test sample is input to a trained model and it is determined that it does not belong to any sub clusters, then it can be labeled as an outlier. The experimental results on three databases demonstrate the effectiveness of our approach.

ACKNOWLEDGEMENTS

This research is supported in part by the NSF CNS award 1314484, Office of Naval Research award N00014-12-1-1028, Air Force Office of Scientific Research award FA9550-12-1-0201, and U.S. Army Research Office grant W911NF-13-1-0160.

REFERENCES

[1] G. J. Williams, R. A. Baxter, H. He, S. Hawkins, and L. Gu, "A comparative study of rnn for outlier detection in data mining," in *ICDM*, 2002, pp. 709–712.

[2] V. Barnett and T. Lewis, *Outliers in statistical data*. Wiley, 1994.

[3] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. Wiley, 1987.

[4] D. M. Hawkins, *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1982.

[5] X. Zhou, C. Yang, and W. Yu, "Automatic mitral leaflet tracking in echocardiography by outlier detection in the low-rank representation," in *CVPR*, 2012, pp. 972–979.

[6] S. D. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule," in *KDD*, 2003, pp. 29–38.

[7] S. Hawkins, H. He, G. J. Williams, and R. A. Baxter, "Outlier detection using replicator neural networks," in *DaWaK*, 2002, pp. 170–180.

[8] W. Jin, A. K. H. Tung, and J. Han, "Mining top-n local outliers in large databases," in *KDD*, 2001, pp. 293–298.

[9] S. Li and I. W. Tsang, "Maximum margin/volume outlier detection," in *ICTAI*, 2011, pp. 385–392.

[10] D. Barbará and P. Chen, "Using the fractal dimension to cluster datasets," in *KDD*, 2000, pp. 260–264.

[11] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang, "Efficient clustering-based outlier detection algorithm for dynamic data stream," in *FSKD (5)*, 2008, pp. 298–304.

[12] A. Koufakou and M. Georgiopoulos, "A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes," *Data Min. Knowl. Discov.*, vol. 20, no. 2, pp. 259–289, 2010.

[13] S. Shekhar, C. T. Lu, and P. Zhang, "Detecting graph-based spatial outliers: algorithms and applications (a summary of results)," in *KDD*, 2001, pp. 371–376.

[14] Y. Xiao, B. Liu, L. Cao, X. Wu, C. Zhang, Z. Hao, F. Yang, and J. Cao, "Multi-sphere support vector data description for outliers detection on multi-distribution data," in *ICDM Workshops*, 2009, pp. 82–87.

[15] T. Le, D. Tran, W. Ma, and D. Sharma, "A theoretical framework for multi-sphere support vector data description," in *ICONIP (2)*, 2010, pp. 132–142.

[16] T. Le, D. Tran, P. Nguyen, W. Ma, and D. Sharma, "Proximity multi-sphere support vector clustering," *Neural Computing and Applications*, vol. 22, no. 7-8, pp. 1309–1319, 2013.

[17] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[18] K. A. Heller, K. M. Svore, A. D. Keromytis, and S. J. Stolfo, "One class support vector machines for detecting anomalous windows registry accesses," in *The Workshop on Data Mining for Computer Security*, 2003.

[19] H. M. Lukashevich, S. Nowak, and P. Dunker, "Using one-class svm outliers detection for verification of collaboratively tagged image training sets," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2009, pp. 682–685.

[20] E. J. Pauwels and O. Ambekar, "One class classification for anomaly detection: Support vector data description revisited," in *ICDM*, 2011, pp. 25–39.

[21] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1982.

[22] Z. C. Lin, M. M. Chen, L. Q. Wu, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *Technique Report, UIUC*, 2009.

[23] S. Li and Y. Fu, "Low-rank coding with b-matching constraint for semi-supervised classification," in *IJCAI*, 2013, pp. 1472–1478.

[24] L. Li, S. Li, and Y. Fu, "Discriminative dictionary learning with low-rank regularization for face recognition," in *FG*, 2013, pp. 1–6.

[25] J. F. Cai, E. J. Candes, and Z. W. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[26] M. Breitenbach and G. Z. Grudic, "Clustering through ranking on manifolds," in *ICML*, 2005, pp. 73–80.

[27] B. Zhang, L. Zhang, D. Zhang, and L. Shen, "Directional binary code with application to polyu near-infrared face database," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2337–2344, 2010.

[28] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression database," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1615–1618, 2003.