Online Unsupervised Kernel Affine Projection Algorithms

Muhammad Sharif Uddin and Anthony Kuh
University of Hawaii
Honolulu, HI 96822
E-mail: uddin@hawaii.edu, kuh@hawaii.edu

***Abstract*—In recent years, the application of unsupervised learning techniques has become of growing importance in a number of fields, e.g., feature selection, clustering etc. While an array of fast supervised learning algorithms have been developed over the years, the number of fast unsupervised learning algorithms is lacking. In this paper, we present a fast unsupervised kernel affine projection algorithm using a least-squares one-class support vector machine framework and coherence sparsification criterion. A kernel NLMS type algorithm is then developed as a special case. To validate the efficacy of the proposed algorithms, we then perform simulations to detect outliers in datasets.**

I. Introduction

With the advances in data acquisition technology, data is increasingly being gathered and stored from a multitude of sources, e.g., power grid, industrial systems, financial markets, computer networks, etc, for a variety of applications ranging from social networks to health care to environmental studies. Labeling all the collected data would require a considerable amount of time and resources making this a daunting proposition. Therefore, the data is often unlabeled. Extracting useful information from the unlabeled data require applications of unsupervised learning methods. Thus, unsupervised learning plays an important role in machine learning.

Unsupervised learning methods have been developed for many batch learning algorithms where learning takes place offline such as clustering or principal component analysis [1]. Unsupervised learning has also been successfully used to improve learning for deep neural networks as better representations are obtained in the hidden layers [2], [3]. In many cases however, there are preferences to learn in real-time using unsupervised online learning methods. Some key considerations in online learning are learning speed, performance of learning algorithm, and computational complexity. While there is a rich body of literature concerning online supervised learning (e.g., linear adaptive filtering with a desired target) [1], [4], there is much less work for online unsupervised learning algorithms. Our previous work with online density estimation performed well in outlier detection application, but the learning complexity was high [5]. We found a much lower complexity solution by using an online least-squares one-class support vector machine (SVM) [6], [7].

In this paper, we discuss implementation of online unsupervised learning algorithms for outlier detection using kernel methods, extending our work in [6], [7]. The proposed algorithms are based on least-squares one-class SVM classifiers. The one-class SVM is an unsupervised learning method, proposed in [8] to extract regions in the input space where most of the training objects lie. A least squares (LS) version of the one-class SVM was proposed by Choi in [9] such that the solution can be obtained by solving a linear system instead of a quadratic programming problem in the standard one-class SVM. However this advantage comes at the cost of loss of sparsity of the support vectors (SVs). Several approaches to sparsification of kernel-based solutions have been proposed in the literature [10]–[14]. In this paper we utilize the subspace method [14] to obtain a sparse representation of the decision hyperplane in least-squares one-class SVM. To further reduce the computational complexity, we then develop a kernel affine projection (KAP) [16] based solution. As a special case of the KAP algorithm, a kernel NLMS type is also proposed.

The rest of the paper is organized as follows: Section II discusses the least-squares one-class SVM and presents the kernel subspace method for inducing sparsity. Section III presents the proposed kernel affine projection based algorithms. The efficacy of the proposed algorithms in detecting outliers in the power grid is examined in Section IV. Finally, Section V provides the concluding remarks.

Notation: Upper and lower case letters denote random variables and their realizations, respectively; underlined letters stand for vectors; boldface upper case letters denote matrices, and $\mathbf{I}_n$ is the $n \times n$ identity matrix; $(\cdot)^T$ denotes matrix (vector) transpose; $|\cdot|$ denotes the absolute value and matrix determinant for scalars and matrices, respectively; $\|\cdot\|$ denotes the L^2 norm of vectors.

II. Sparse Online Least-squares One-class Support Vector Machine

The LS one-class SVM proposed by Choi in [9] gives a hyperplane that maximizes the distance from the origin and minimizes the distance to the training objects in least squares sense. The distance from this hyperplane may then be used as a measure to determine which objects resemble the training objects better than others. For training data $\underline{x}_1, \ldots, \underline{x}_n \in \mathbb{R}^d$, the optimization problem for LS one-class SVM is [9]

$$\min J = \frac{1}{2}\|\underline{w}\|^2 - \rho + \frac{C}{2}\|\underline{\xi}\|^2, \tag{1}$$

$$\text{subject to } \underline{w} = \mathbf{\Phi}\underline{\alpha} \tag{2}$$

$$\underline{\xi} = \underline{1}_n\rho - \mathbf{\Phi}^T\underline{w}, \tag{3}$$

where matrix $\mathbf{\Phi}=[\underline{\phi}(\underline{x}_1),\ldots,\underline{\phi}(\underline{x}_n)]$ and $\underline{1}_n$ is a $n\times 1$ vector of all ones. $\underline{\phi}(\cdot):\mathbb{R}^d\rightarrow\mathcal{H}$ is the mapping to a high dimensional feature space such that the dot product in $\mathcal{H}$ can be computed by evaluating a kernel $k(\underline{x},\underline{y})=\underline{\phi}(\underline{x})^T\underline{\phi}(\underline{y})$. The parameter C is predefined and $\underline{\xi}$ is a vector of slack variables.

A. Kernel Subspace Method

In an online learning scheme we sequentially process a stream of incoming data points. As more data become available, the memory and processing requirement increases. The subspace method [14] produces a sparse solution for the LS one-class SVM that allows the solution to be stored in a compact form and reduces the number of linear equations to be solved. At time step $n-1$, let $\mathcal{X}_{\mathcal{D}}^{(n-1)}=\{\underline{x}_{D,1},\ldots,\underline{x}_{D,m}\}$ be a subset of the training samples chosen as the support vectors for the LS one-class SVM, such that $m<n-1$. The corresponding feature matrix is denoted by $\mathbf{\Phi}_{D,(n-1)}=[\underline{\phi}(\underline{x}_{D,1}),\ldots,\underline{\phi}(\underline{x}_{D,m})]$.

Given the constraints on the training samples, we can now rewrite the optimization problem (1) as an online optimization problem:

$$\min_{\underline{\hat{w}}_n,\hat{\rho}_n,\underline{\xi}_n} J_n=\frac{1}{2}\|\underline{\hat{w}}_n\|^2-\hat{\rho}_n+\frac{C}{2}\|\underline{\xi}_n\|^2 \tag{4}$$

$$\text{subject to } \underline{\hat{w}}_n=\mathbf{\Phi}_{D,(n)}\underline{\hat{\alpha}}_n, \tag{5}$$

$$\underline{\xi}_n=\underline{1}_n\hat{\rho}_n-\mathbf{\Phi}_n^T\underline{\hat{w}}_n. \tag{6}$$

Let $\mathbf{K}_{D,(n)}=\mathbf{\Phi}_{D,(n)}^T\mathbf{\Phi}_{D,(n)}$ be the kernel matrix of the support vectors. Let $\mathbf{\Phi}_{\bar{D},(n)}$ denote the matrix containing the mapping of the sample points *not* in the support vector dictionary. Then we can rewrite $\mathbf{\Phi}_n=\begin{bmatrix}\mathbf{\Phi}_{D,(n)} & \mathbf{\Phi}_{\bar{D},(n)}\end{bmatrix}$ by rearranging the columns. Then Substituting $\underline{\hat{w}}_n$ and $\underline{\xi}_n$ into (4), and defining $\mathbf{\Phi}_n^T\mathbf{\Phi}_{D,(n)}=\mathbf{K}_{S,(n)}$, we have

$$J_n=\frac{1}{2}\underline{\hat{\alpha}}_n^T\mathbf{K}_{D,(n)}\underline{\hat{\alpha}}_n-\hat{\rho}_n+\frac{C}{2}\|\underline{1}_n\rho_n-\mathbf{K}_{S,(n)}\underline{\hat{\alpha}}_n\|^2. \tag{7}$$

Taking the derivatives of (7) with respect to $\underline{\hat{\alpha}}_n$ and $\hat{\rho_n}$, and setting equal to zero, we obtain the following set of linear equations:

$$\begin{bmatrix}\underline{1}_n^T\underline{1}_n & -\underline{1}_n^T\mathbf{K}_{S,(n)}\\ -\mathbf{K}_{S,(n)}^T\underline{1}_n & \mathbf{P}_n\end{bmatrix}\begin{bmatrix}\hat{\rho}_n\\ \underline{\hat{\alpha}}_n\end{bmatrix}=\begin{bmatrix}\frac{1}{C}\\ \underline{0}_m\end{bmatrix}, \tag{8}$$

where

$$\mathbf{P}_n\triangleq\mathbf{K}_{D,(n)}/C+\mathbf{K}_{S,(n)}^T\mathbf{K}_{S,(n)}. \tag{9}$$

Applying block matrix inversion lemma [15],

$$\hat{\rho}_n=\left(C\underline{1}_n^T\left(\mathbf{I}_n-\mathbf{K}_{S,(n)}\mathbf{P}_n^{-1}\mathbf{K}_{S,(n)}\right)\underline{1}_n\right)^{-1}, \tag{10}$$

$$\underline{\hat{\alpha}}_n=\mathbf{P}_n^{-1}\mathbf{K}_{S,(n)}^T\underline{1}_n\left(C\underline{1}_n^T\left(\mathbf{I}_n-\mathbf{K}_{S,(n)}\mathbf{P}_n^{-1}\mathbf{K}_{S,(n)}\right)\underline{1}_n\right)^{-1}. \tag{11}$$

B. Recursive Computations of $\underline{\hat{\alpha}}_n$ and ρ_n

In an online data stream application, ρ_n and $\underline{\hat{\alpha}}_n$ need to be updated at every time step when a new data point becomes available. Using (10) and (11) as update formulas requires computing the inverse of the matrix $\mathbf{P}_n$. This matrix inversion step has computational complexity $\mathcal{O}(m^3)$ which dominates the total complexity of the algorithm. To reduce the computational complexity of the algorithm we need to find recursive update formula with lower complexity for updating $\underline{\hat{\alpha}}_n$ and $\hat{\rho}_n$. The developed algorithm has similarities to the recursive least squares (RLS) algorithm developed for linear adaptive filters [4]. The main difference is that here we are working in the dual space where dimensionality of the space depends on the number of support vectors, m.

We start by defining

$$\underline{q}_n\triangleq\mathbf{K}_{S,(n)}^T\underline{1}_n, \tag{12}$$

$$\underline{r}_n\triangleq\mathbf{P}_n^{-1}\mathbf{K}_{S,(n)}^T\underline{1}_n. \tag{13}$$

Then (10) and (11) become

$$\hat{\rho}_n=\frac{1}{C}(n-\underline{q}_n^T\underline{r}_n)^{-1}, \tag{14}$$

$$\underline{\hat{\alpha}}_n=\underline{r}_n\hat{\rho}_n. \tag{15}$$

Once the vectors $\underline{q}_n$ and $\underline{r}_n$ are known, $\hat{\rho}_n$ and $\underline{\hat{\alpha}}_n$ can be computed in $\mathcal{O}(m)$ time. Hence, we wish to obtain a recursive formulas for computing $\underline{q}_n$ and $\underline{r}_n$.

At time step n, when the new data point $\underline{x}_n$ becomes available, based on the support vector selection criterion, one of the following two cases can happen:

Case 1: Data not added as SV: When the new data is not added as a support vector, the matrix $\mathbf{K}_D$ remains unchanged. The matrix $\mathbf{K}_{S,(n)}$ is updated. Then using (9) and applying the Sherman-Morrison formula [15] we can compute $\mathbf{P}_n^{-1}$ recursively as

$$\mathbf{P}_n^{-1}=\mathbf{P}_{n-1}^{-1}-\underline{s}_n\underline{k}_n^T\mathbf{P}_{n-1}^{-1}, \tag{16}$$

where

$$\underline{s}_n=\frac{\mathbf{P}_{n-1}^{-1}\underline{k}_n}{1+\underline{k}_n^T\mathbf{P}_{n-1}^{-1}\underline{k}_n}. \tag{17}$$

Then $\underline{q}_n$ and $\underline{r}_n$ can be recursively computed as

$$\underline{q}_n=\underline{q}_{n-1}+\underline{k}_n, \tag{18}$$

$$\underline{r}_n=\underline{r}_{n-1}+\underline{s}_n\left(1-\underline{k}_n^T\underline{r}_{n-1}\right). \tag{19}$$

Case 2: Data added as SV: In this case we have $\mathcal{X}_{\mathcal{D}}^{(n)}=\mathcal{X}_{\mathcal{D}}^{(n-1)}\cup\{\underline{x}_n\}$ and $m=m+1$. Accordingly, $\mathbf{K}_D$ is updated as follows:

$$\mathbf{K}_{D,(n)}=\begin{bmatrix}\mathbf{K}_{D,(n-1)} & \underline{k}_n\\ \underline{k}_n^T & k_{nn}\end{bmatrix}, \tag{20}$$

where $k_{nn}=k(\underline{x}_n,\underline{x}_n)=\underline{\phi}(\underline{x}_n)^T\underline{\phi}(\underline{x}_n)$. Then from (9), the recursive formulas for updating $\mathbf{P}_n^{-1}$ can be obtained by applying Sherman-Morrison formula and the block matrix inversion lemma [15]:

$$\mathbf{P}_n^{-1}=\begin{bmatrix}\mathbf{T}_n & \underline{0}_{m-1}\\ \underline{0}_{m-1}^T & 0\end{bmatrix}+\frac{\underline{v}_n\underline{v}_n^T}{\beta}, \tag{21}$$

where

$$\mathbf{T}_n = \left(\mathbf{I}_{m-1} - \underline{s}_n \underline{k}_n^T\right) \mathbf{P}_{n-1}^{-1}, \tag{22}$$

$$\underline{v}_n = \begin{bmatrix} \mathbf{T}_n \underline{u}_n \\ -1 \end{bmatrix}, \tag{23}$$

$$\beta = \frac{k_{nn}}{C} + \underline{k}_n^T \underline{k}_n + k_{nn}^2 - \underline{u}_n^T \mathbf{T}_n \underline{u}_n, \tag{24}$$

$$\underline{s}_n = \frac{\mathbf{P}_{n-1}^{-1} \underline{k}_n}{1 + \underline{k}_n^T \mathbf{P}_{n-1}^{-1} \underline{k}_n}, \tag{25}$$

$$\underline{u}_n = \underline{k}_n (1/C + k_{nn}) + \mathbf{K}_{D,(n-1)} \underline{k}_n. \tag{26}$$

Using (20)–(25), we obtain the recursive formulas for $\underline{q}_n$ and $\underline{r}_n$ as follows:

$$\underline{q}_n = \begin{bmatrix} \underline{q}_{n-1} + \underline{k}_n \\ \underline{k}_n^T \underline{1}_{m-1} + k_{nn} \end{bmatrix}, \tag{27}$$

$$\underline{r}_n = \begin{bmatrix} \underline{r}_{n-1} + \underline{s}_n \left(1 - \underline{k}_n^T \underline{r}_{n-1}\right) \\ 0 \end{bmatrix} + \frac{\underline{v}_n^T \underline{q}_n}{\lambda} \underline{v}_n. \tag{28}$$

Finally, ρ_n and $\hat{\underline{\alpha}}_n$ are updated using (14)–(15).

The overall computational complexity of the update equations is $\mathcal{O}(m^2)$. The recursive solutions $\hat{\underline{\alpha}}_n$ and ρ_n for optimization problem (4) subject to error constraints on all n observed data points. The recursive solution with further reduced complexity may be obtained by limiting the error constraint to most recent p observations. Similar algorithms have been proposed in linear adaptive filter theory [16], [17] and supervised learning methods [12]. These algorithms are classified as Affine Projection algorithms [17]. We develop a similar algorithm next.

III. Kernel Affine Projection Solution

We can formulate the affine projection problem as an optimization that minimizes the squared L^2 norms of the change in the weight vector and the bias term, subject to constraint on most recent p data points. The minimization problem at time step n is given by

$$\min_{\underline{\alpha}, \rho} \|\underline{\alpha} - \underline{\alpha}_{n-1}\|^2 + (\rho - \rho_{n-1})^2 \tag{29}$$
$$\text{subject to } \underline{1}_p \rho = \mathbf{K}_{p,(n)}^T \underline{\alpha},$$

where $\mathbf{K}_{p,(n)} = [\underline{k}_n, \underline{k}_{n-1}, \ldots, \underline{k}_{n-p+1}]$. At time step n, when new data point $\underline{x}_n$ becomes available, we have two cases:

Case 1: Data not an SV: Problem (29) can be solved by minimizing the Lagrangian function,

$$L(\underline{\alpha}, \rho, \underline{\lambda}) = \|\underline{\alpha} - \underline{\alpha}_{n-1}\|^2 + (\rho - \rho_{n-1})^2 + \underline{\lambda}^T \left(\underline{1}_p \rho - \mathbf{K}_{p,(n)}^T \underline{\alpha}\right), \tag{30}$$

where $\underline{\lambda}$ is the Lagrange multiplier. Solving the Lagrangian, we obtain the update equations for $\underline{\alpha}_n$ and ρ_n as

$$\underline{\alpha}_n = \underline{\alpha}_{n-1} + \mu \mathbf{K}_{p,(n)} \left(\mathbf{K}_{p,(n)}^T \mathbf{K}_{p,(n)} + \underline{1}_p \underline{1}_p^T + \epsilon \mathbf{I}_p\right)^{-1} \times \times \left(\underline{1}_p \rho_{n-1} - \mathbf{K}_{p,(n)}^T \underline{\alpha}_{n-1}\right), \tag{31}$$

$$\rho_n = \rho_{n-1} - \mu \underline{1}_p^T \left(\mathbf{K}_{p,(n)}^T \mathbf{K}_{p,(n)} + \underline{1}_p \underline{1}_p^T + \epsilon \mathbf{I}_p\right)^{-1} \times \times \left(\underline{1}_p \rho_{n-1} - \mathbf{K}_{p,(n)}^T \underline{\alpha}_{n-1}\right), \tag{32}$$

where μ is the step-size control parameter and ϵ is the regularization parameter.

Case 2: Data added as SV: In this case the new data is added to the support vector dictionary. To accommodate the new support vector, we modify the optimization problem (29) as

$$\min \|\underline{\alpha} - \begin{bmatrix} \underline{\alpha}_{n-1} \\ 0 \end{bmatrix}\|^2 + (\rho - \rho_{n-1})^2 \tag{33}$$
$$\text{subject to } \underline{1}_p \rho = \mathbf{K}_{p,(n)}^T \underline{\alpha}. \tag{34}$$

The Lagrangian function is given by

$$L(\underline{\alpha}, \rho, \underline{\lambda}) = \left\| \underline{\alpha} - \begin{bmatrix} \underline{\alpha}_{n-1} \\ 0 \end{bmatrix} \right\|^2 + (\rho - \rho_{n-1})^2 + + \underline{\lambda}^T \left(\underline{1}_p \rho - \mathbf{K}_{p,(n)}^T \underline{\alpha}\right). \tag{35}$$

Solving the Lagrangian, we obtain the update equations for $\underline{\alpha}_n$ and ρ_n as:

$$\underline{\alpha}_n = \begin{bmatrix} \underline{\alpha}_{n-1} \\ 0 \end{bmatrix} + \mu \mathbf{K}_{p,(n)} \left(\mathbf{K}_{p,(n)}^T \mathbf{K}_{p,(n)} + \underline{1}_p \underline{1}_p^T + \epsilon \mathbf{I}_p\right)^{-1} \times \times \left(\underline{1}_p \rho_{n-1} - \mathbf{K}_{p,(n)}^T \begin{bmatrix} \underline{\alpha}_{n-1} \\ 0 \end{bmatrix}\right), \tag{36}$$

$$\rho_n = \rho_{n-1} - \mu \underline{1}_p^T \left(\mathbf{K}_{p,(n)}^T \mathbf{K}_{p,(n)} + \underline{1}_p \underline{1}_p^T + \epsilon \mathbf{I}_p\right)^{-1} \times \times \left(\underline{1}_p \rho_{n-1} - \mathbf{K}_{p,(n)}^T \begin{bmatrix} \underline{\alpha}_{n-1} \\ 0 \end{bmatrix}\right). \tag{37}$$

The complexity of the affine projection update is $\mathcal{O}(p^2 m)$.

A. Special Case: Projection Order $p = 1$

Now consider a special case when the projection order p is 1. In this case, the minimization problem (29) is constrained by only the most recent observed data point $\underline{x}_n$. Then the recursive update equations (31)-(32) and (36)-(37) become

1) Data not a support vector:

$$\underline{\alpha}_n = \underline{\alpha}_{n-1} + \mu \frac{\left(\rho_{n-1} - \underline{k}_n^T \underline{\alpha}_{n-1}\right)}{1 + \underline{k}_n^T \underline{k}_n} \underline{k}_n, \tag{38}$$

$$\rho_n = \rho_{n-1} - \mu \frac{\left(\rho_{n-1} \underline{k}_n^T \underline{\alpha}_{n-1}\right)}{1 + \underline{k}_n^T \underline{k}_n}, \tag{39}$$

where μ is a step-size control parameter.

2) Data added as a support vector:

$$\underline{\alpha}_n = \begin{bmatrix} \underline{\alpha}_{n-1} \\ 0 \end{bmatrix} + \mu \frac{\left(\rho_{n-1} - \underline{k}_n^T \underline{\alpha}_{n-1}\right)}{1 + \underline{k}_n^T \underline{k}_n + k_{nn}^2} \begin{bmatrix} \underline{k}_n \\ k_{nn} \end{bmatrix}, \tag{40}$$

$$\rho_n = \rho_{n-1} - \mu \frac{\left(\rho_{n-1} - \underline{k}_n^T \underline{\alpha}_{n-1}\right)}{1 + \underline{k}_n^T \underline{k}_n + k_{nn}^2}, \tag{41}$$

where μ is the step-size control parameter.

IV. Experiments

To study the feasibility of the proposed algorithms, we perform simulations on the IEEE 14 bus test system [18] in an outlier detection setting and compare the performance of the proposed methods with the normalized residual (r^N_{max}) test based on traditional AC state estimation in the power grid [21]. The least-squares one-class SVM find the hyperplane that minimizes the squared distances to the training points in the feature space. Therefore, the distance from the hyperplane can be used as a measure of resemblance between a data point and the training set. For new data $\underline{x}_n$, we perform the following threshold test:

$$d(\underline{x}_n) = \frac{|\hat{\underline{\alpha}}^T_{n-1}\underline{k}_n - \rho_n|}{\sqrt{\hat{\underline{\alpha}}^T_{n-1}\mathbf{K}_{D,(n-1)}\hat{\underline{\alpha}}_{n-1}}} \geq d_{th} \implies \text{Outlier}.$$

In the proposed sparse online LS one-class SVM our goal is to build a diverse support vector dictionary to approximate the input space while inducing sparsity. In this paper, for the subspace method we utilize the coherence criterion $\delta_n = \max(\underline{k}_n) \leq \text{threshold}, \gamma$ proposed in [12] for adding support vectors to the dictionary. To moderate the addition of support vectors into the dictionary, we introduce another threshold $\gamma' < \gamma$. After comparing the distance threshold, before a data point is added as a support vector we have three cases:

- $\delta_n > \gamma$: very similar to dictionary; only hyperplane updated.
- $\gamma \geq \delta_n \geq \gamma'$: somewhat similar; both dictionary and hyperplane updated.
- $\delta_n < \gamma'$: very dissimilar; discard without updating either dictionary or hyperplane.

A similar approach has also been used in [13].

In power grid, the measurement vector $\underline{x}_n$ consists of real and reactive power measurements. In the traditional state estimation, these measurements are collected from different parts of the grid and processed in a centralized manner to estimate the system states and detect outliers using the r^N_{max} test [21]. At time step n, the AC state estimator employs the nonlinear measurement model given by [21]

$$\underline{x}_n = \underline{h}(\underline{v}_n) + \underline{e}_n, \tag{42}$$

where $\underline{v}_n$ is the state vector, $\underline{h}(\cdot)$ is a nonlinear vector function relating measurements to states and $\underline{e}_n$ is the iid measurement error with mean $\underline{0}$ and covariance $\mathbf{R}$. The vector $\underline{v}_n$ usually consists of the steady state bus voltage magnitudes and phase angles [21]. The weighted least squares (WLS) estimate $\hat{\underline{v}}_n$ is obtained using the Gauss-Newton method by the following iterative procedure [21]:

$$\mathbf{G}\left(\underline{v}^\ell_n\right)\Delta\underline{v}^{\ell+1}_n = \mathbf{H}\left(\underline{v}^\ell_n\right)^T\mathbf{R}^{-1}\left[\underline{x}_n - \underline{h}\left(\underline{v}^\ell_n\right)\right], \tag{43}$$

$$\underline{v}^{\ell+1}_n = \underline{v}^\ell_n + \Delta\underline{v}^{\ell+1}_n, \tag{44}$$

where $\mathbf{G}\left(\underline{v}^\ell_n\right) = \left(\mathbf{H}\left(\underline{v}^\ell_n\right)^T\mathbf{R}^{-1}\mathbf{H}\left(\underline{v}^\ell_n\right)\right)$ is the gain matrix; $\mathbf{H}\left(\underline{v}^\ell_n\right) = \left[\frac{\partial\underline{h}(\underline{v}_n)}{\partial\underline{v}_n}\right]_{\underline{v}_n=\underline{v}^\ell_n}$ is the Jacobian matrix; ℓ is the iteration number. The covariance matrix of the estimate of the measurement, $\hat{\underline{x}} = \underline{h}(\hat{\underline{v}})$ is given by

$$\mathbf{T} = \mathbf{H}\mathbf{G}^{-1}\mathbf{H}^T, \tag{45}$$

where $\mathbf{H} = \left[\frac{\partial\underline{h}(\underline{v})}{\partial\underline{v}}\right]_{\underline{v}=\hat{\underline{v}}}$.

The measurement residual is calculated as

$$\underline{r} = \underline{x} - \hat{\underline{x}}, \tag{46}$$

which is a white Gaussian process of zero mean and has a covariance matrix given by the difference between the measurement error covariance and the measurement estimate covariance [21], i.e., $\mathbf{\Omega} = \mathbf{R} - \mathbf{T}$.

In the r^N_{max} test, $\underline{r}$ is normalized and the largest normalized residual is compared with a threshold:

$$\max\frac{|r_i|}{\sqrt{\mathbf{\Omega}[i,i]}} \leq \text{threshold}. \tag{47}$$

If the i-th measurement violates the threshold, then it is suspected as bad data, removed from the measurement set and the state estimation process is repeated with the reduced set.

A shortcoming of the residual test is the detection of bad data in multiple interacting measurements [21]. In addition, data injection attacks can be designed that are undetectable by the r^N_{max} test [19], [22]. For instance, if an attacker designed an attack vector as $\underline{a} = \underline{h}(\hat{\underline{v}} + \underline{c}) - \underline{h}(\hat{\underline{v}})$ and changed the measurement to $\underline{x}_{bad} = \underline{v} + \underline{a}$, then $\underline{a}$ can pass the bad data detection test [19]. The AC state estimator will yield an erroneous state $\underline{v}_{bad} = \hat{\underline{v}} + \underline{c}$.

On the other hand, the proposed algorithm does not require any information about the system states and depends entirely upon the historical observations. Thus, outlier detection can be performed *before* the data is sent to a central station. By dividing the system into smaller subsystems, we can perform outlier detection in each subsystem, thus reducing communication overhead. This decentralized method of outlier detection is demonstrated next.

A. Bad data detection

We first implemented the proposed algorithm for bad data detection in the IEEE 14 bus test system [18]. Fig. 1 shows the network diagram of the test system. For distributed bad data detection using least-squares one-class SVM, we divide the 14 bus system into two subsystems and shown in Fig. 1. For our simulations, we only consider bad data detection in the region enclosed in blue lines.

In the simulations, we assume that the system is operating in a quasi steady state. We created a data stream of 2000 samples by adding 2% measurement noise to the true measurements. For outlier detection using least-squares one-class SVM, we consider the measurement vector $\underline{X} = [P_1,\ P_2,\ P_3,\ P_{1-2},\ P_{1-5},\ P_{4-7},\ P_{4-2},\ P_{4-5},\ P_{4-9},\ P_{2-5},\ P_{5-6}]^T$. To study the efficacy of the proposed methods in detecting bad data, we randomly injected 200 bad data in the interacting pair P_1 and P_{1-2} with gross errors of magnitudes $10 - 30\%$ of the true measurement.

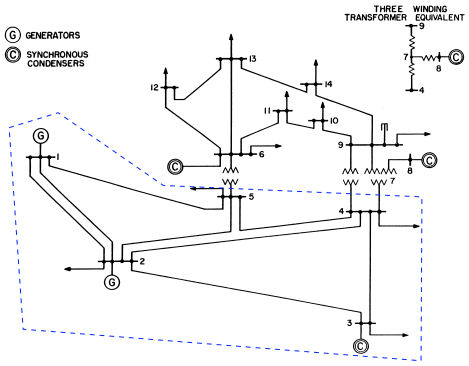

Fig. 1. IEEE 14 bus test system [18]. The blue line shows the area under consideration.

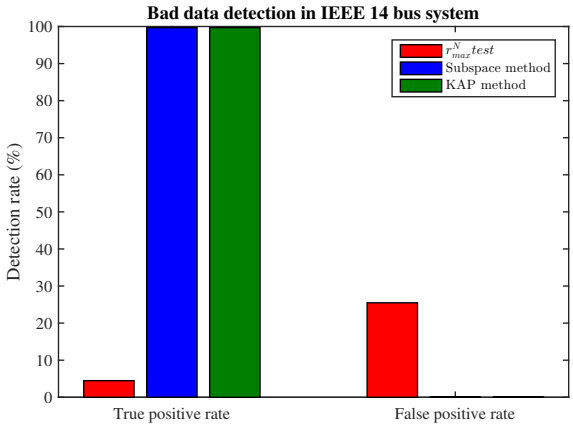

Fig. 2. Comparison of bad data detection rates in IEEE 14 bus test system. The projection order for KAP algorithm is set to 1.

Fig. 2 shows the comparison of the detection rates of r^N_{max} test, least-squares one-class SVM subspace method and the kernel affine projection method, averaged over 100 simulations.For the affine projection method we only consider projection order 1. We observe that r^N_{max} test has low positive detection rate for interacting bad data. The added measurement noise also causes high false positive detection rate in the r^N_{max} test. In contrast, the both of the proposed online algorithms have high positive detection rates in interacting bad data while achieving a low false detection rate.

B. False data attack detection

In AC state estimation, an attack vector $\underline{a}$ can pass the traditional r^N_{max} test, if it satisfies $\underline{a} = \underline{h}(\hat{\underline{x}} + \underline{c}) - \underline{h}(\hat{\underline{x}})$, where $\underline{c}$ is the change in the estimated state vector [19]. Two types of data attacks are usually studied in the literature: attack on state variable(s) and attack on certain measurements [19], [20].

To successfully alter any of the state variables without being detected, the attack must change all the measurements that depend upon that state variable [19], [20]. Once it has been determined which measurements need to be altered, the real and reactive power flows from bus i to j can be calculated from [21]

$$P_{ij} = V_i^2(g_{si} + g_{ij}) - V_iV_jg_{ij}\cos(\theta_i - \theta_j) - V_iV_jb_{ij}\sin(\theta_i - \theta_j), \quad (48)$$

$$Q_{ij} = -V_i^2(b_{si} + b_{ij}) - V_iV_jg_{ij}\sin(\theta_i - \theta_j) + V_iV_jb_{ij}\cos(\theta_i - \theta_j), \quad (49)$$

where g_{si}, b_{si}, g_{ij} and b_{ij} are network parameters. The power injected at bus i is then

$$P_i = \textstyle\sum_j P_{ij} \quad \text{and} \quad Q_i = \textstyle\sum_j Q_{ij}. \quad (50)$$

An injected false data vector will contain altered power measurements that differ from the other data in current operating conditions, hence it will be an outlier. In the traditional r^N_{max} test, if the attack vector is $\underline{a} = \underline{h}(\underline{x}+\underline{c}) - \underline{h}(\underline{x})$, then the attack becomes unobservable [19]. On the other hand, this alteration in the power measurements is detectable in our data driven approach. By choosing a proper mapping function $\underline{\phi}(\cdot)$, we can obtain $\|\underline{\phi}(\underline{x}+\underline{a}) - \underline{\phi}(\underline{x})\| \geq$ threshold. Thus, we can use a similarity measure in the feature space to detect false data injection attacks.

To study the feasibility of the proposed LS one-class SVM methods for detecting malicious data attacks, we performed simulations on the IEEE 14 bus system [18], using a Gaussian kernel $k(\underline{x}_1, \underline{x}_2) = \exp(-\|\underline{x}_1 - \underline{x}_2\|/\sigma^2)$, and compared performance with the r^N_{max} test. We used the state estimator in the MATPOWER toolbox [23] for the r^N_{max} test. For distributed bad data detection using least-squares one-class SVM, we divide the 14 bus system into two subsystems as shown in Fig. 1. For our simulations, we only consider attack detection in the region enclosed in blue lines with the measurement vector $\underline{X} = [P_1,\ P_2,\ P_3,\ P_{1-2},\ P_{1-5},\ P_{4-7},\ P_{4-2},\ P_{4-5},\ P_{4-9},\ P_{2-5},\ P_{5-6}]^T$.

We investigate attacks targeting one state variable from the set $\mathcal{A} = \{V_2,\ V_3,\ V_4,\ V_5,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5\}$. To simulate attack on a state variable in $\mathcal{A}$, we generate a data stream of 2000 observations where the last 200 observations are malicious data containing measurements to alter the state variable by $5-10\%$ of its true value. We repeat this procedure for each of the state variables in $\mathcal{A}$. The parameters used for the kernel RLS solution to online LS one-class SVM are $C = 2$, $\sigma^2 = 0.5$, $\gamma = 0.8$, $\gamma' = 0.1$. For the kernel affine projection algorithm, the parameters are $\sigma^2 = 0.5$, , $p = 1$, $\gamma = 0.8$, $\gamma' = 0.1$, $\mu = 0.01$. Using the aforementioned parameters, we performed 100 simulations for each of the variables in $\mathcal{A}$ and compared the average detection rates of the proposed methods with the detection rates of the r^N_{max} test. As expected from the attack design, the r^N_{max} test fails to detect any of the attacks. On the other hand, our algorithms achieved a 100% success rate in detection of the false data injection attacks. Since *all* the power measurements related to a state is altered, this translates to a large deviation from the learned hyperplane. Thus, our methods are able to achieve high true positive detection rates while minimizing false positive detection rates.

V. Conclusion

This paper presents kernel subspace and kernel affine projection methods for online unsupervised learning in a least-squares one-class SVM framework. In both methods we used the coherence criterion for choosing support vectors. Simulation experiments show that the proposed methods work well in detecting bad data in sensor measurements and malicious data injection attacks in the power grid. In our future work, we will investigate the performance of the kernel subspace and kernel affine projection methods on other datasets. There are many other possible directions for potential future research. One possible direction is considering changes in the data distribution. It is possible that the underlying distribution of the collected data may slowly change over time. In that case a low complexity method needs to be developed to discard old uninformative data. Another possible direction is to consider multiple kernels for improving learning rate.

VI. Acknowledgment

This work was supported in part by NSF grant ECCS-1310634 and the University of Hawaii REIS project.

References

[1] C.M. Bishop, *Pattern Recognition and Machine Learning*, Information Science and Statistics. Springer, 2006.

[2] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," Jan. 2018.

[3] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1734–1747, Sept. 2016.

[4] S. O. Haykin, *Adaptive Filter Theory*, Pearson Education, 2013.

[5] M. S. Uddin, A. Kuh, Y. Weng, and M. D. Ilić, "Online bad data detection using kernel density estimation," in *PES General Meeting, 2015 IEEE*, Denver, CO, July 2015.

[6] M. S. Uddin and A. Kuh, "Online least-squares one-class support vector machine for outlier detection in power grid data," in *2016 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 2628–2632.

[7] A. Kuh, M. S. Uddin, and P. Ng, "Online unsupervised kernel learning algorithms," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Dec 2017, pp. 1019–1025.

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

[9] Young-Sik Choi, "Least squares one-class support vector machine," *Pattern Recognition Letters*, vol. 30, no. 13, pp. 1236 – 1240, 2009.

[10] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, River Edge, NJ, 2002.

[11] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug 2004.

[12] C. Richard, J.C.M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2009.

[13] W. Liu, I. Park, and J.C. Principe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Networks*, vol. 20, no. 12, pp. 1950–1961, Dec 2009.

[14] Anthony Kuh, Chaopin Zhu, and Danilo Mandic, "Sensor network localization using least squares kernel regression," in *Knowledge-Based Intelligent Information and Engineering Systems*, Bogdan Gabrys, Robert J. Howlett, and Lakhmi C. Jain, Eds., Berlin, Heidelberg, 2006, pp. 1280–1287, Springer Berlin Heidelberg.

[15] F. Zhang, Ed., *The Schur Complement and Its Application*, vol. 4 of *Numerical Methods and Algorithms*, Springer US, 2005.

[16] Kazuhiko Ozeki and Tetsuo Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electronics and Communications in Japan (Part I: Communications)*, vol. 67, no. 5, pp. 19–27, 1984.

[17] K. Ozeki, *Theory of Affine Projection Algorithms for Adaptive Filtering*, Mathematics for Industry. Springer Japan, 2015.

[18] "Power system test case archive," Available at http://www.ee.washington.edu/research/pstca/.

[19] G. Hug and J. A. Giampapa, "Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1362–1370, Sept 2012.

[20] G. Chaojun, P. Jirutitijaroen, and M. Motani, "Detecting false data injection attacks in AC state estimation," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2476–2483, Sept 2015.

[21] Ali Abur and Antonio Gómez Expósito, *Power System State Estimation: Theory and Implementation*, Marcel Dekker, Inc, New York, NY, 2004.

[22] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," in *Proc. 16th ACM Conf. Computer and Communications Security*, 2009, pp. 21–32.

[23] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.