2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)

2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC) | 979-8-3503-0067-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/APSIPAASC58517.2023.10317364

Distributed on-line anomaly detection using kernel methods

Anthony Kuh* and Tyler Baguio†
* University of Hawaii, Honolulu, HI 96822
E-mail: kuh@hawaii.edu
† University of Hawaii, Honolulu, HI 96822
E-mail: tylerb8@hawaii.edu

***Abstract*—In this paper we develop an online distributed learning algorithm for unlabeled data. Here a goal is to detect anomalies from streaming data. The assumption is that data is drawn from a probability distribution, but that at some unknown time data could be drawn from a different distribution. We assume the data comes from distributed sources and develop an online Federated Learning (FL) algorithm based on online kernel methods using Random Fourier Features (RFF). This work combines previous work on online unsupervised kernel algorithms with online supervised kernel algorithms using FL and RFF. There are many applications of this work including detecting bad power grid data.**

I. Introduction

In this day and age we are seeing an explosion of data coming from a variety of sources and locations at different rates. Much of this data come from distributed sources such as sensors, IoT devices, mobile devices, cameras, individuals, groups, or organizations. There is a need to process, learn, and make inferences from the data. Here we consider data gathered by edge devices or clients with information passed to a central server or the cloud. In the past, data was sent from the clients to the cloud where processing, learning and making decisions would occur as shown in Fig. 1a) and labeled as the centralized learning model. However, edge devices now have much more computational power with faster processors (GPUs) allowing more computations and learning to occur at the edge. In addition there are many other concerns with sending data from edge devices to the cloud. There are security and privacy concerns of sending data and information to the cloud. There is also concerns about an excess amount of communications between the edge devices and the cloud. Federated Learning (FL) was developed by [1] to address these issues where processing and learning takes place at the edge device and not passed to the cloud. This is shown in Fig. 1b) where information, but not data is passed to the cloud. The cloud assists in the learning process by sending collective information back to each edge device so that collaborative learning can occur. The FL model is also able to handle heterogeneous data and heterogenous systems.

There are many applications for FL where there are concerns about data privacy, communication costs, and handling heterogeneous data and systems. These include mobile communications, autonomous systems with communications between autonomous cars for real-time information and decision making,

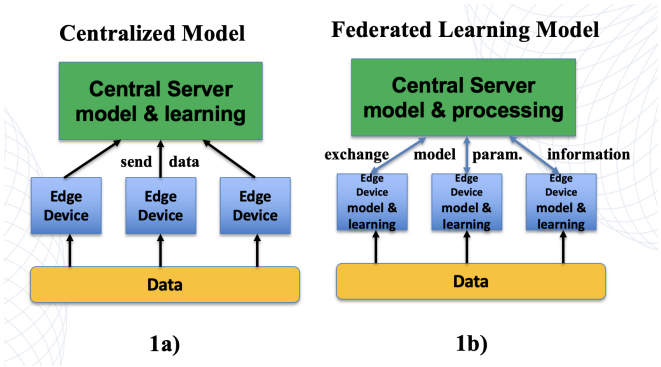

Fig. 1. Collaborative learning models

healthcare applications include sharing information between different hospitals, manufacturing for predictive maintenance, and smart grid applications for demand-side management, [2]. For demand-side management consider a group of homes in a community. Each home may be identical or similar in terms of the architecture, but energy usage in each home may be very different depending on the occupants, their energy preferences, and energy consuming devices in the home. This includes heating, air conditioning, hot water heaters, and appliances. Homes may also have rooftop solar panels, battery storage, and charging stations for Electric Vehicles (EV)s. This is a good application for FL as at each home gets data from many heterogeneous sources and there are privacy issues. This include environmental sensors monitoring local climate including solar irradiance, electrical grid data, data from sensors monitoring energy usage, and possibly data from sensors monitoring EV charging. From this data we can learn about household energy usage while preserving comfort, conserving energy and reducing costs.

There has been considerable research and interest in FL algorithms with the following papers giving a good discussion of FL research [3], [4]. Most of the learning algorithm for FL are supervised learning algorithms. In previous work we developed online kernel regression algorithms that used principles of FL, [5], [6]. The algorithms developed in [6] are based on edge devices (clients) learning data received using online kernel methods implemented using Random Fourier Features (RFF),

Authorized licensed use limited to: UNIV OF HAWAII LIBRARY. Downloaded on September 08,2024 at 20:58:10 UTC from IEEE Xplore. Restrictions apply.

[7]. The clients send weight information to a server where the server coordinates learning by aggregating information from the clients and then sending this information back to the clients. Communication savings are achieved as we only sample a small number of clients at each update and the client only sends a small fraction of RFF coefficients when sampled. This is called Partial Sharing-Based Online Federated Learning (PSO-FED). PSO-FED is shown to converge and through simulations have good learning performance (even when a small fraction of RFF are transmitted). The algorithm is also shown to have some robustness against data poisoning (Byzantine attacks).

In this paper we extend our learning algorithms to unsupervised learning. Although the vast majority of machine learning deals with supervised learning algorithms where a target output is associated with each input data a considerable amount of data is unlabeled. Many learning algorithms have been developed for unsupervised learning ranging from clustering algorithms where similar input data is grouped together to Principal Component Analysis (PCA) which is a statistical method for reducing the dimensionality of a dataset by linear transforming the dataset to find the coordinates with the largest variance. Here we will focus on anomaly detection with a survey of machine learning methods presented in [8]. Here we will examine where data is drawn from a probability distribution in an online manner. At some point data is drawn from an anomalous distribution and the goal is to detect this change. We first formulate an online kernel algorithm following [9] and then use RFF to approximate the kernels to detect the change. We then adapt this algorithm based on distributed learning using principles of FL.

Let us briefly discuss two applications for anomaly detection. Consider a network of environmental sensors monitoring water quality. The sensors could be placed under water in a lake. Each sensor would periodically sample environmental conditions such as temperature, pH, turbidity, and perhaps other chemicals. PAD-OFL could learn the support of the distribution under normal conditions. Anomalies could occur due to natural conditions such as unusual algae growth or due to human factors such as the release of pathogens and/or pollutants into the lake. In a second application consider monitoring a power grid for bad data. Sensor could be distributed on the grid to monitor voltage, current, and other electrical measurements. Bad data could be the result of natural circumstances such as a downed power line due to fallen trees or a malicious man-made attack where bad data is injected. A more detailed discussion presenting simulation results is presented for detecting bad power grid data in the applications section.

The paper is organized as follows. In Section 2 we first discuss using a one class least squares kernel algorithm for modeling a probability distribution. We then approximate the kernels using RFF, and develop an online learning algorithm for anomaly detection. In Section 3 we assume the data comes from distributed sources (clients) and information is processed by a central server. We then develop a FL model for anomaly detection. Section 4 presents a simple example and an application detecting bad data for the electric power grid where algorithms of Section 3 are used. Section 5 summarizes the paper and discusses further directions.

II. Modeling Distributions and Anomaly Detection

A. Using Kernel Methods

In a key paper by Schölkopf, [10] a one-class support vector is used to model the support of a probability distribution. Here we will assume inputs $x(i), 1 \leq i \leq m$ are drawn from a sample of a random vector $X \in \mathcal{R}^n$. This can be represented by

$$\min \frac{1}{2}||w||^2 - \rho + \gamma \sum_i \xi_i \tag{1}$$

$$\text{subject to } w^T\phi(x(i)) \geq \rho - \xi_i. \text{ and } \xi_i \geq 0,$$

where $\phi(\cdot) : \mathcal{R}^n \rightarrow \mathcal{Z}$ is the mapping to a high dimensional feature space such that the dot product of ϕ is usually computed by evaluating a kernel $k(x, y) = \phi(x)^T\phi(y)$. The parameter $\gamma > 0$ is a regularization parameter and controls the fraction of possible outliers [10] and ξ_i are non-zero slack variables. The goal here in feature space is to have as many data points (in feature space) $\phi(x(i))$ as possible to lie on one side of a hyperplane (described by $f(x) = w^T\phi(x) - \rho = 0$) away from the origin such that the distance to the origin, ρ/w is as large as possible. This is know as the One-Class Support Vector Machine (SVM). The hyperplane solution can be found by solving the dual optimization problem [10]

$$\min \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j k(x(i), x(j)) \tag{2}$$

$$\text{subject to } 0 \leq \alpha_i \leq \gamma \text{ and } \sum_i \alpha_i = 1,$$

where α_i are the Lagrange multipliers. The solution for the weight vector w is given by

$$w = \sum_i \alpha_i \phi(x(i)).$$

We would like to consider online learning algorithms for this problem and can use a different squared error cost function which leads to a simpler online learning algorithms based on linear adaptive filters. This was implemented by Choi in [11] using a squared error loss function and is referred to as the Least Squares OC-SVM (LS-OC-SVM). For this optimization problem the goal is to perform linear regression in feature space with the hyperplane again as far away from the origin as possible. In the input space the manifold formed by the equation $f(x) = w^T\phi(x) - \rho = 0$ describes a surface where most of the support of the probability distribution lie near to f(x). The LS-OC-SVM optimization problem can be formulated as

$$\min J(w, \rho) = \min \frac{1}{2}||w||^2 - \rho + \frac{\gamma}{2}||\xi||^2 \tag{3}$$

$$\text{subject to } w = \mathbf{\Phi}\alpha \text{ and } \xi_i = \mathbf{1}\rho - \mathbf{\Phi}^T w.$$

Authorized licensed use limited to: UNIV OF HAWAII LIBRARY. Downloaded on September 08,2024 at 20:58:10 UTC from IEEE Xplore. Restrictions apply.

where the feature input matrix $\mathbf{\Phi} = [\phi(x_1), \ldots, \phi(x_m)]$, $\alpha = [\alpha_1, \ldots, \alpha_m]^T$, and $\mathbf{1}$ represents a column vector of 1s. Note hear that for the LS-OC-SVM we have constraints that are equality constraints as opposed to the OC-SVM where we have constraints that are inequality constraints. Defining the kernel matrix $\mathbf{K} = \mathbf{\Phi}^T\mathbf{\Phi}$ we can then make substitutions in equation (3) to get the following

$$J(\alpha, \rho) = \frac{1}{2}\alpha^T\mathbf{K}\alpha + \frac{\gamma}{2}||\mathbf{1}\rho - \mathbf{K}\alpha||^2 \tag{4}$$

This is a quadratic function of the variables α and ρ with the solution found by solving a system of linear equations given by

$$\begin{bmatrix} \mathbf{1}^T\mathbf{1} & -\mathbf{1}^T\mathbf{K} \\ -\mathbf{1} & \mathbf{K} + \mathbf{I}/\gamma \end{bmatrix} \begin{bmatrix} \rho \\ \alpha \end{bmatrix} = \begin{bmatrix} 1/\gamma \\ \mathbf{0} \end{bmatrix} \tag{5}$$

where $\mathbf{0}$ is a column vector of zeros and $\mathbf{I}$ is an identity matrix.

Here all the input data are support vectors and the hyperplane in the dual space is given by $f(x) = \alpha^T\mathbf{k}(x) - \rho = 0$ where $\mathbf{k}(x) = \mathbf{\Phi}^T\phi(x)$. A problem with solving the set of linear equations, from equation (5) is that the matrix $\mathbf{K}$ grows as the number of input data grow, m as all input data are support vectors for the LS-OC-SVM. For m data the computational complexity for the solution grows as m^3 and alternative subspace methods are needed when m grows large. Subspace methods require that a dictionary of support vectors are chosen. For online learning the dictionary is grown online as data is processed. In [9] an online dictionary is grown using an approximate linear dependence (ALD) criterion, [12]. Alternatively, in [13] a dictionary is grown using a simpler coherence criterion. A problem with these methods is that they add complexity to the learning problem and we propose an alternate solution in the next subsection.

B. Approximating Kernels with Random Fourier Features

In a landmark paper by Rahimi and Recht, [7] they showed that if the kernel function can be expressed in terms of one variable, that is $k(u, v) = k(u - v)$, then we can have an alternative approach to construct a set of basis functions. Here assume that $k(u, u) = 1$. From Bochner's theorem we can then represent the kernel function in terms of its inverse n-dimentsional Fourier transform by

$$\begin{aligned} k(u-v) &= \left(\frac{1}{2\pi}\right)^n \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} p_W(\omega) \exp(j\omega^T(u-v)) d\omega_1 \ldots d\omega_n \\ &= \mathbf{E}_W[\psi_\omega(u)\psi_\omega(v)^*] \end{aligned} \tag{6}$$

where $\psi_\omega(u) = \exp(j\omega^T u)$ and $*$ represents complex conjugate. From the righthand side of the equation we have $\psi_\omega(u)\psi_\omega(v)^*$ is an unbiased estimate of $k(u-v)$ as $p_W(\omega)$ is an n-dimensional probability density function. Note that

$$k(u - v) = \mathbf{E}_W \cos(\omega^T(u - v))$$

since $p_W(\omega)$ is real and even function of its n components (as $k(u, v)$ is real, even, and positive semi-definite). Let $c_\omega(u) = \sqrt{2}\cos(\omega^T u + \theta)$ where ω is drawn from $p_W(\omega)$ and θ is drawn from a uniform $[0, 2\pi]$ random variable. We then have

$$\begin{aligned} \mathbf{E}_{W,\theta}[c_\omega(u)c_\omega(v)] &= \mathbf{E}_{W,\theta}[\cos(\omega^T(u-v)) + \cos(\omega^T(u+v) + 2\theta)] \\ &= k(u - v) \end{aligned}$$

where the last equality holds as the expectation of the second term is 0.

Let $(\omega_j, \theta_j), 1 \leq j \leq D$ be D samples drawn from W and θ. We then have

$$k(u, v) = \langle \phi(u), \phi(v) \rangle \approx \frac{1}{D} \sum_{j=1}^{D} c_{\omega_j}(u) c_{\omega_j}(v) = c^T(u)c(v) \tag{7}$$

where $c(u) = \frac{1}{\sqrt{D}}[c_{\omega_1}(u), \ldots, c_{\omega_D}(u)]^T$. Here we approximate the kernel by drawing D samples and averaging. As an example of W let the kernel function be

$$k(u, v) = \exp(\sigma^2||u - v||^2/2)$$

then the kernel function is a Gaussian kernel and the inverse Fourier transform gives a Gaussian random vector with each component being iid with zero mean and variance σ^2. From the kernel representation result we then have the estimate of $f(\cdot)$ given by

$$\begin{aligned} f(x) = \alpha^T k(x) - \rho &= \sum_{i=1}^{m} \alpha_i k(x(i), x) - \rho \\ &\approx \sum_{i=1}^{m} \alpha_i c(x(i))^T c(x) - \rho. \end{aligned} \tag{8}$$

If we let $\beta = \sum_{i=1}^{m} \alpha_i c(x(i))$, then we have that

$$f(x) \approx \beta^T c(x) - \rho. \tag{9}$$

This reduces the problem to a linear regression problem in D-dimensional space generated by the RFF with the goal of finding the parameters β and b. Let $\mathbf{C} = [c(x(1)), \ldots, c(x(m))]$. Then with regularization we are trying to find β and ρ to minimize

$$J(\beta, \rho) = \frac{1}{2}||\beta||^2 + \frac{\gamma}{2}||e||^2 - \rho$$

where $e = \mathbf{C}^T\beta - \rho\mathbf{1}$. Solving this optimization problem is found by taking gradients and partial derivatives and setting to 0 resulting in

$$\nabla_\beta J(\beta, \rho) = (\mathbf{I} + \gamma\mathbf{C}\mathbf{C}^T)\beta - \gamma\mathbf{C}\mathbf{1}\rho = 0 \tag{10}$$

and

$$\frac{\partial J(\beta, \rho)}{\partial \rho} = \gamma m\rho - \gamma\mathbf{C}^T\beta - 1 = 0 \tag{11}$$

resulting in the following set of linear equations,

$$\begin{bmatrix} 1 & -{m_c}^T \\ -m_c & \mathbf{R}_c + \mathbf{I}/(m\gamma) \end{bmatrix} \begin{bmatrix} \rho \\ \beta \end{bmatrix} = \begin{bmatrix} 1/(m\gamma) \\ 0 \end{bmatrix} \tag{12}$$

where $m_c = \frac{1}{m}\mathbf{C}\mathbf{1}$ and $\mathbf{R}_c = \frac{1}{m}\mathbf{C}\mathbf{C}^T$. The solution is found by solving a system of $D + 1$ linear equations.

Authorized licensed use limited to: UNIV OF HAWAII LIBRARY. Downloaded on September 08,2024 at 20:58:10 UTC from IEEE Xplore. Restrictions apply.

C. Anomaly Detection and Online Learning

Solving for β and ρ gives us a manifold described by $g(x) = \beta^T c(x) - \rho = 0$. Given we use enough training data m and enough RFF, D the support of the random variable X lies close to this manifold with the distance of a point x from the manifold given by

$$d(x) = \frac{|g(x)|}{\sqrt{\beta^T c(x)^T c(x)\beta}} \tag{13}$$

Following [9], if the distance of a data point is far from the manifold we classify this as an anomalous data point. Here we define a threshold value $\delta > 0$ such that if $d(x) > \delta$ then the data point is an anomaly. The setting of δ determines the fraction of points that are classified as anomaly data.

Here we assume that the first m data points are drawn from the random variable X. This can be considered the baseline data. The LS-OC-SVM can be trained to find the manifold using batch methods solving equation (12) or by online methods. Online methods include using the Recursive Least Square (RLS) algorithm or the Least Mean Square (LMS) algorithm described in [14]. The nice thing is that instead of working with kernel online learning algorithms we work with simpler linear adaptive filter algorithms.

After seeing the m data points we are presented with more data and need to determine if the data is anomalous data. Here we set the parameter δ, initialize $\beta(m)$ and $\rho(m)$, and let $z(k) = c(x(k))$ by solving equation (12), and then use the LMS learning algorithm to update the parameters. The learning algorithm is given below.

==========

Centralized Anomaly Detection with Online Learning using Kernels and RFF (CAD-OL)

1) **Initialization: Learn** $\beta(m)$ **and** $\rho(m)$**, set** $\delta > 0$**, set step sizes** μ_1, μ_2**, and** $k = m$**.**
2) **Get data point,** $x(k+1)$
3) **If** $x(k+1)$ **is anomaly data that is (**$d(x(k+1)) > \delta$**), label point** $A(k+1) = 1$ **and**

$$\beta(k+1) = \beta(k), \;\; \rho(k+1) = \rho(k)$$

4) **otherwise update parameters using LMS algorithm estimating gradients and derivatives from equations (10) and (11)**

$$e(k+1) = z(k+1)^T \beta(k) - \rho(k),$$

$$\beta(k+1) = (1 - \frac{\mu_1}{\gamma m})\beta(k) - \mu_1 e(k+1) z(k+1),$$

$$\rho(k+1) = \rho(k) + \mu_2(\frac{1}{\gamma m} + e(k+1)),$$

and $A(k+1) = 0$

5) $k \leftarrow k+1$**, go to 2)**

==========

Note that CAD-OL can detect anomalies if δ is set correctly with the vector of anomalies recorded in A. It also tracks slowly varying changes in the distribution as the manifold parameters slowly change as new data is presented.

III. Online Unsupervised Federated Learning using PSO-FED

As mentioned in the introduction many sources of data come from distributed sources and this Section develops the models and Federated Learning (FL) algorithms for this scenario. Here we assume we have L clients and also assume that clients receive data synchronously at each update, but this can be easily modified to asynchronous updates [5], [15]. At time k client l receives input data $x(k,l)$ with $z(k,l) = c(x(k,l))$. There is also a server that communicates with all the clients and helps to coordinate learning. One approach as mentioned earlier is a centralized learning approach where the clients send all data to the server. Then the server can just use CAD-OL discussed in the previous section. As mentioned earlier this results in high communication costs and loss of data privacy. This motivates the need to use FL algorithms.

Here we combine the unsupervised anomaly detection learning algorithm discussed in the previous section, CAD-OL with PAO-FED discussed in [6]. At each update the server chooses some random set of $P << L$ clients who each communicate with the server about their parameter information. When a server chooses a client, that client gets the server model of the parameters and adaptively updates their parameter based on its current input, $x(k,l)$ (CAD-OL step 4) unless that data point is an anomaly. The server then takes the sum of all the P clients updates as its new parameter estimate. The process then repeats as the server chooses a new set of P clients at the next update. In [6] this is referred to as Online-Fed learning.

To implement PAO-FED, [6] we start at each update by choosing $R << D$ parameters of the vector β for each client. Each client can have the same components chosen (coorinated) or choose different components (uncoordinated). Initially, these can be chosen randomly or sequentially. To denote which components of β are chosen we will define a selection matrix $S(k,l)$ at time k for client l which is a diagonal matrix of dimension D with 1s on the diagonal element where there will be an update on that component of β and 0s otherwise. We also let $\mathcal{P}(k)$ denote the set of clients chosen at time k. The algorithm for each client is similar to CAD-OL for edge devices with a server aggregating the selected clients and sending this information to each of the chosen clients. At each update the clients must also rotate which components of β are updated to iterate for the next update. We initially assume that we learn the support of the distribution.

==========

Partial Sharing Anomaly Detection with Online FED Learning using Kernels and RFF (PAD-OFL)
(learning support of distribution)

1) **Initialization: Set** $\beta(0)$**,** $\rho(0)$ **,** $\beta(0,l), \rho(0,l), 1 \le l \le L$**, set step sizes** μ_1, μ_2**, determine initial chosen components,** $S(0,l),\ 1 \le l \le L$**, and** $k = 0$**.**
2) **Get data points,** $x(k,l)$**, choose** P **clients,** $\mathcal{P}(k)$**, and communicate** $S(k,l)\beta(k)$ **to selected clients.**

Authorized licensed use limited to: UNIV OF HAWAII LIBRARY. Downloaded on September 08,2024 at 20:58:10 UTC from IEEE Xplore. Restrictions apply.

3) **Client Update: If** $j \in \mathcal{P}(k)$

$$\beta'(k,j) = S(k,j)\beta(k) + (\mathbf{I}_D - S(k,j))\beta(k,j)$$

$$e(k+1,j) = z(k+1,j)^T\beta'(k.j) - \rho(k,j),$$

$$\beta(k+1,j) = (1-\frac{\mu_1}{\gamma m})\beta'(k,j) - \mu_1 e(k+1,j)z(k+1,j),$$

Else

$$e(k+1,j) = z(k+1,j)^T\beta(k.j) - \rho(k,j),$$

$$\beta(k+1,j) = (1-\frac{\mu_1}{\gamma m})\beta(k,j) - \mu_1 e(k+1,j)z(k+1,j),$$

Endif

$$\rho(k+1,j) = \rho(k,j) + \mu_2(\frac{1}{\gamma m} + e(k+1,j),$$

4) **Selection matrix update:** $S(k,j+1) = \mathbf{circshift}(\mathbf{diag}(S(k,j)),r)$**. Shift of** r**.**
5) **Server update: average client update parameters to get global update**

$$\beta(k+1) = \frac{1}{P}\sum_{j\in\mathcal{P}(k)}\beta(k+1,j)$$

$$= \frac{1}{P}\sum_{j\in S_k} S(k+1,j)\beta(k+1,j) + (\mathbf{I}_D - S(k+1,j))\beta(k)$$

$$\rho(k+1) = \frac{1}{P}\sum_{j\in\mathcal{P}(k)}\rho(k+1,j)$$

6) $k \leftarrow k+1$**, go to 2)**

==========

The second part of PAD-OFL is to detect anomalies. Here the algorithm is the same as for learning support of distribution except that we have a check if data point $x(k,l)$ is an anomaly by checking if $d(x(k,l)) >$ after step 2). The procedure is the same as CAD-OL. For PAD-OFL note that at each update the client l only communicates R components of $\beta(k,l)$ to the server and the server only communicates R components of $\beta(k)$ to each client.

IV. Application of Algorithm

Let us first examine a simple example that was presented in [9]. This is the synthetic square-noise dataset with normal points labeled in blue and noisy points labeled as red. This is shown in Fig. 2. If we assume the normal points are drawn from a probability distribution we can get the distribution support lies within two square boxes that contain the blue points. It is not difficult to simulate the LS-OC-SVM to construct the square boundary boxes approximately and then being able to detect the red point anomalies. The online CAD-OL has also been implemented and can get a good approximation of the two square boundary boxes and detect anomalies using Gaussian kernels and RFF. Similarly, PAD-OFL can be implemented with a number of clients to get the two square boundaries and detect the red point anomalies.

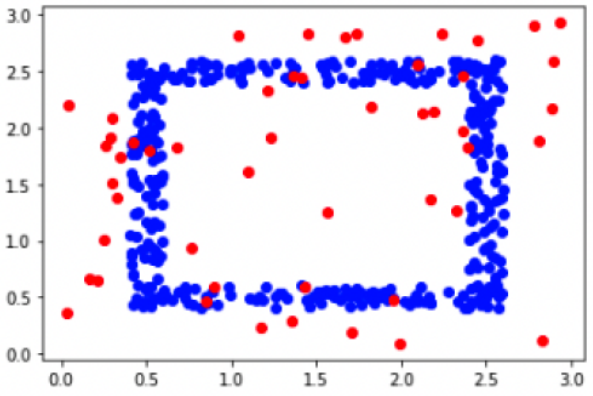

Fig. 2. Synthetic square-noise data set

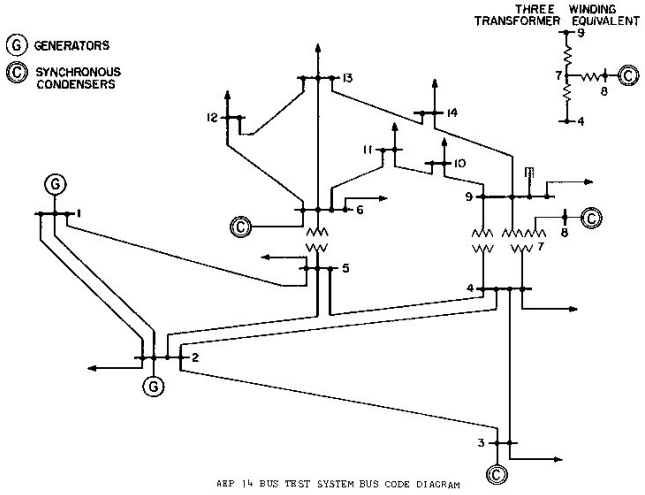

Fig. 3. IEEE 14-bus test system

For real applications, the probability distributions could be in high dimensions with complicated support. A goal is to implement the algorithms developed here on some of these applications. Here we briefly discuss one example of detecting bad electric power grid data.

Power injection	At buses 1, 2, 3, 8
Power flow	At branches 1-2, 1-5, 2-4, 2-5, 4-5, 4-7, 4-9, 5-6, 6-11, 6-14, 9-10, 9-14, 12-13
Voltage magnitude	At buses 3, 6, 8, 10, 14

Table 1. Measurement configuration for IEEE 14-bus system

In [9], detection of bad power grid data was considered. The IEEE 14-bus system [16] shown in Fig. 3 was used to test for bad grid data. A standard residual test for non-critical branch measurements was able to detect bad grid data, however branch P4-7 is a critical branch that is not observable and the standard residual test does not work. If we use an online LS-OC-SVM

Authorized licensed use limited to: UNIV OF HAWAII LIBRARY. Downloaded on September 08,2024 at 20:58:10 UTC from IEEE Xplore. Restrictions apply.

using a kernel RLS algorithm we can detect bad grid data for branch P4-7. The experiment was done by considering a vector $x = [P_{4-7}, P_{4-2}, P_{4_5}, P_{4-9}]^T$ which contains the critical measurement along with measurements from neighboring buses. The true value for this flow is $P_{4-7} = 0.2285 p.u.$ We created a stream of 1000 samples by adding zero mean Gaussian noise with variance .0004 to the true value. The data was trained on a LS-OC-SVM to produce a manifold where the support of the distribution lies close to the manifold. Gaussian kernels were used and first trained on 700 training data. Then 300 test data were used with 5 of the data points injected with bad data. The LS0-OC-SVM was able to detect all five bad data points.

We then used CAD-OL using RFF using the LMS learning algorithm with suitably adjusted hyperparameters and small learning step sizes. This algorithm like the LSO-OC-SVM was also able to detect all five bad data points. Note that much power grid data is measured from distributed sources and we then applied the PAD-OFL to attempt to detect the bad grid data. We used 10 clients where data was equally and randomly distributed among these clients and then used the LMS algorithm where at each update we chose 3 of the clients to update. A total of 30 RFF were used and at each update we chose 5 of these RFF to update. The savings in communication cost over using all RFF is a factor of six. PAD-OFL using these parameters was also able to detect all five bad data points. PAD-OFL achieved communication savings and also reduced computations compared with centralized approaches and using LSO-OC-SVM.

We have discussed a power grid application where we could implement online FL algorithms such as PAD-OFL. There are many other applications where online anomaly detection for distributed sources using PAD-OFL could be used. These include personalized healthcare, networked autonomous vehicles, and precision agriculture with some of these applications are discussed in [2].

V. Summary and Further Direction

This paper discussed implementing online unsupervised learning algorithms for anomaly detection. We consider situations where data is coming from distributed sources and use principles of Federated Learning. We also use kernel methods and represent kernels using random Fourier Features (RFF). The paper combines implementation of LS-OC-SVM [9], with RFF [7], and using the PAO-FED algorithm [6]. The algorithm developed is called Partial Sharing Anomaly Detection with Online FED Learning using Kernels and RFF (PAD-OFL). We simulated results detecting bad power grid data showing the efficacy of both CAD-OL and PAD-OFL.

There is much further work to be considered including using PAD-OFL on numerous other applications and understanding the analytical behavior of PAD-OFL using tools from signal processing and information theory.

Acknowledgements

This work was supported in part by the United States National Science Foundation (NSF) grant ECCS-2142987.

References

[1] J. Konečný, B. McMahan, and D. Ramage. Federated optimization:distributed optimization beyond the datacenter, 2015.

[2] V. C. Gogineni, S. Werner, F. Gauthier, Y.-F. Huang, and A. Kuh. Personalized online federated learning for IOT/CPS: challenges and future directions. *IEEE Internet of Things Magazine*, 5:78–84, 2022.

[3] P. Kairouz and et. al. Advances and open problems in federated learning, 2021.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Proc. Magazine*, 37(3):50–60, May 2020.

[5] A. Kuh. Real time kernel learning for sensor networks using principles of federated learning. In *Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 2089 – 2093, Tokyo, Japan, Dec. 2021.

[6] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh. Communication-efficient online federated learning strategies for kernel regression. *IEEE Internet of Things Journal*, 10(5):4531–4544, Nov. 2022.

[7] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.

[8] L. Ruff, J. R. Kauffmann, R. A Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K. R. Müller. A unifying review of deep and shallow anomaly detection. *IEEE Proceedings*, 109(5):756–795, May 2021.

[9] M. Uddin and A. Kuh. Online least-squares one-class support vector machine for outlier detection in power grid data. In *2016 IEEE ICASSP*, Shanghai, China, Mar. 2016.

[10] 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*, 13(7):1443–1471, 2001.

[11] Y.-S. Choi. Least squares one-class support vector machine. *Pattern Recognition Letters*, 30(13):1236–1240, 2009.

[12] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Trans. on Signal Proc.*, 52(3):2275–2285, Aug. 2004.

[13] C. Richard, J.-C. Bermudez, and P. Honeine. Online prediction of time series data with kernels. *IEEE Trans. on Signal Proc.*, 57(3):1058–1067, Mar. 2009.

[14] S. Haykin. *Adaptive Filter Theory, 5th Ed.* Pearson, 2014.

[15] F. Gauthier, V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh. Asynchronous online federated learning with reduced communication requirements. Technical report, arXiv preprint arXiv:2303.15226, 2023.

[16] Power system test case archive. Technical report, Univ. of Washington, http://www.ee.washington.edu/research/pstca/.

Authorized licensed use limited to: UNIV OF HAWAII LIBRARY. Downloaded on September 08,2024 at 20:58:10 UTC from IEEE Xplore. Restrictions apply.