IET Research Journals

Special Issue on Privacy and Security in Smart Grids

IET Journals
The Institution of Engineering and Technology

Detecting Load Redistribution Attacks via Support Vector Models

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Zhigang Chu[1*], *Oliver Kosut*[1], *Lalitha Sankar*[1]

[1] *School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, 85287, USA*
[*] *E-mail: zchu2@asu.edu*

Abstract: A machine learning-based detection framework is proposed to detect a class of cyber-attacks that redistribute loads by modifying measurements. The detection framework consists of a multi-output support vector regression (SVR) load predictor and a subsequent support vector machine (SVM) attack detector to determine the existence of load redistribution (LR) attacks utilizing loads predicted by the SVR predictor. Historical load data for training the SVR are obtained from the publicly available PJM zonal loads and are mapped to the IEEE 30-bus system. The features to predict loads are carefully extracted from the historical load data capturing both temporal and spatial correlations. The SVM attack detector is trained using normal data and randomly created LR attacks, so that it can maximally explore the attack space. An algorithm to create random LR attacks is introduced. The results show that the SVM detector trained merely using random attacks can effectively detect not only random attacks, but also intelligently designed attacks. Moreover, using the SVR predicted loads to re-dispatch generation when attacks are detected can significantly mitigate the attack consequences.

1 Introduction

Leveraging information technology, the operation of modern electric power grids largely rely on real-time sensing, monitoring, communication, and control. State estimation (SE) utilizes the power system measurements collected by the supervisory control and data acquisition (SCADA) system to estimate the operating states. These states are used by the energy management system (EMS) to allow for real-time control of power system. In the last decade, the cyber-security of SE has been studied with considerable attention. A class of false data injection (FDI) attacks that replace measurements with counterfeits have been shown to be able to easily spoof SE and the traditional bad data detector (BDD) [1]. This finding serves as the basis of a wide class of FDI attacks, called load redistribution (LR) attacks, which make it appear as if the loads are redistributed among load buses while the total load remain unchanged.

Worst-case consequences of LR attacks can be found using bi-level optimization problems. These attacks can be designed to cause physical or economic consequences. For physical consequences, [2] attempts to find an attack to mask the outage of a transmission line, and [3] designs attacks that can cause physical overflows. For economic consequences, [4] and [5] show that LR attacks can change locational marginal prices, and/or make profit for attackers. Therefore, it is crucial to develop techniques to detect and mitigate LR attacks.

Various attack detection techniques have been presented in the literature. In [6], the authors propose a multivariate Gaussian-based anomaly detector trained using correlation features of micro phasor measurement units (μPMUs), but this detector requires installation of μPMUs in the system. Liu *et al.* [7] detect and identify attacks using reactance perturbation, but this method only works when the attacker has limited resources. The authors of [8] attempt to mitigate LR attacks using a tri-level optimization approach, and the authors of [9] try to identify LR attacks by monitoring abnormal load deviations and suspicious branch flow changes. However, they both only focus on attacks that cause line overflows. In [10], a financially motivated FDI attack model is analyzed and a robust incentive-reduction strategy is proposed to deter such attacks by protecting a subset of meters. More generally, machine learning techniques are also deployed in detecting LR attacks. For example, [11] proposes supervised and semi-supervised machine learning algorithms to detect FDI attacks by exploiting the relationships between statistical and geometric properties of attack vectors employed in the attack scenarios. A deep reinforcement learning-based approach is proposed to detect LR attacks in [12]. In [13], three machine learning techniques are introduced for attack detection, namely nearest neighbor, semi-supervised one class SVM, and replicator neural network. These three algorithms compare estimated loads with historical loads and use thresholding to determine the existence of LR attacks.

Estimation-Detection Framework: In this paper, we introduce an LR attack detection framework based on support vector models by leveraging the historical load information commonly available to system operators. While there are existing approaches in the literature to prevent attacks by installing of new devices [6] or protecting specific measurements [10], guiding operators to utilize existing data available to design software-based solutions is complementary to those existing approaches. Our method determines the existence of LR attacks directly through the estimated loads, which can be conveniently applied in conjunction with the current EMS operations. When an LR attack occurs, the estimated loads obtained from the SE results are different from the true loads, but the net loads are the same. Thus, if accurate bus-level load predictions are available, the existence of LR attacks can be determined by comparing the predicted and estimated loads. Moreover, if an LR attack is detected, the predicted loads can be directly used to re-dispatch generation instead of using the estimated loads. By doing this, the attack consequences can be temporarily mitigated, giving operators time to perform other corrective actions.

Support Vector Models: In particular, we propose a support vector regression (SVR) [14] based load predictor to accurately predict loads, and a subsequent support vector machine (SVM) [15] based attack detector that compares the predicted and observed loads to detect LR attacks. Our choice of this modular design aims to separate the prediction and classification, so that each module can be independently enhanced (*e.g.,* using additional features) and also replaced by other methods, as seen fit. Support vector models are optimization-based machine learning approaches that can be used for both regression and classification purposes. There are many different machine learning methods, and we choose support vector models for the following reasons: (i) they are mature methods that have been proven to be effective for various regression/classification tasks in power systems, including transient stability assessment [16], component outage estimation [17], and state estimation [18]; (ii) they

are analytically developed models with fewer and easier to tune parameters compared to many other machine learning methods, *e.g.,* neural networks.

SVR has been widely used for load prediction in electric power systems. In [19], a short-term load forecasting algorithm is proposed combining SVR and particle swarm optimization. The authors of [20] proposes a SVR model that predicts very short term loads using weather data and day ahead predicted loads as features. Similar features along with additional time-related features are used to train a SVR model that predicts short term and mid term loads in [21]. In [22], Azad *et al,* predict the daily peak load using the historical peak load consumption and the corresponding temperature and relative humidity. Chong *et al,* propose a K-step ahead prediction using SVR in [23]. However, the existing work on load prediction is largely focused on predicting the net load, which is not helpful in detecting LR attacks because the net load remain unchanged under those attacks.

Proposed SVR Load Predictor: The aforementioned references focus on predicting the net load utilizing temporal correlation. To the best of our knowledge, we are one of the first to predict loads at each bus, leveraging both spatial and temporal correlations between all the loads in the system. Features selected for the SVR predictor include historical load values of all loads chosen at distinct time intervals prior to the target time (*e.g.*, one hour before, one day before, etc.) as well as the specific time information (*e.g.,* month, weekday/weekend). This choice allows for conveniently using the same features to predict loads at different buses as the temporal features for all loads implicitly capture the spatial correlations among them.

Proposed SVM Detector: SVM is a supervised learning approach to solve classification problems, based on learning separating hyperplanes. Our approach using SVR to detect attacks largely mirrors existing approaches; our key contribution is in how we generate the training data needed to learn the SVM model to classify accurately over a large class of attacks. We now describe the dataset and our approach to train and test the two models.

Dataset: We train and test our models using the publicly available PJM metered zonal load data [24]. We map each of the 20 zones of the PJM data to a load bus in the IEEE 30-bus system, leveraging the fact that there are 20 loads in this system.

Training and Testing: To apply SVM on attack detection, it is necessary to create training data in both classes, namely *normal* and *attacked* data. The SVR predicted loads and the true loads (assuming trustworthy historical data) naturally form the normal data. For the attacked data, we propose a novel approach that generates random LR attacks in order to maximally explore the attack space, and thereby enhance accuracy in detecting *any* LR attack. Each of these attacks alters a random number of loads, and a Gaussian distribution is used to generate the deviation of each load from its true value. The severity of the attacks is controlled by varying the maximum deviation percentage over all loads. Our approach also guarantees the net load change is 0 to satisfy the constraints of LR attacks. We use 80% of the data for training, and the remaining 20% for testing.

In addition to the random attacks, we also generate two types of intelligently designed LR attacks, namely cost maximization (CM) and line overflow (LO) attacks, to test the effectiveness of our SVM attack detector. CM attacks aim to maximize the operation cost [25]; and LO attacks attempt to overflow a target transmission line [3]. These two types of attacks are designed through optimizations to maximize their economic/physical consequences.

Our results show that the proposed attack estimation-detection framework can effectively predict and detect both random and intelligently designed LR attacks. Moreover, we illustrate that using the SVR predicted loads to re-dispatch when attacks are detected can significantly reduce the attack consequences.

Summary of Contributions: The key contributions of this paper are as follows:

1. We propose an LR attack detection framework consisting of an SVR load predictor and a subsequent SVM attack detector. This modular design enables separate enhancement of each block, and also provides sufficiently accurate predicted loads for attack mitigation purposes.

2. In the SVR predictor , the extracted features leverage both temporal and spatial correlations within the historical load data, which allow for prediction of bus-level loads. Through training and testing the proposed SVR predictor on the PJM metered load data [24], we show that it can predict every load in the system with high accuracy.

3. Utilizing the SVR predicted loads, we train the SVM detector using normal data and random LR attacks designed to maximally explore the attack space. An algorithm to create such random LR attacks is also proposed.

4. The performance of the detection framework is tested on random attacks as well as two types of intelligently designed LR attacks. These attacks aim to cause economic/physical consequences. Our simulation results show that our detection framework can effectively detect both random and intelligently designed attacks, even though the detector is only trained using the former. Moreover, we show that using the SVR predicted loads to re-dispatch upon detection of LR attacks can significantly reduce the impact of LR attacks.

The rest of this paper is organized as follows. Section 2 introduces LR attacks and existing approaches to create intelligently designed LR attacks. Section 3 describes the structure of the proposed attack detection framework, the formulations of SVR and SVM, as well as random LR attack creation method for SVM training purpose. Section 4 illustrates the performance of the SVR load predictor and the SVM attack detector. Section 5 describes the attack mitigation strategy and demonstrates its performance. Concluding remarks are presented in Section 6.

2 Load Redistribution Attacks

2.1 Load Redistribution (LR) Attacks and Unobservable False Data Injection (FDI) Attacks

Definition 1: LR attacks are a class of cyber-attacks that redistribute loads among the buses, while keeping the net load unchanged. The false loads in an LR attack $\boldsymbol{P}_{\text{Atk}}$ satisfies

$$\boldsymbol{P}_{\text{Atk}} = \boldsymbol{P} + \Delta\boldsymbol{P}, \tag{1}$$

$$\sum_i \Delta P_i = 0, \tag{2}$$

where $\boldsymbol{P}$ is the true load vector, $\Delta\boldsymbol{P}$ is the load change caused by attack, and i is the load index.

Definition 2: The load shift τ is defined to be the largest load change in percentage of the true loads:

$$\tau = \max_i \left| \frac{\Delta P_i}{P_i} \right| \times 100\%. \tag{3}$$

We use τ as an intrinsic metric to characterize the detectability of LR attacks. We found that it is trivial to detect attacks with sufficiently large τ, because load deviations far from true values are suspicious. Thus, an attacker is likely to limit τ to avoid detection by this metric. In this paper, we only consider LR attacks with $\tau \leq 20\%$.

The most common way to generate LR attacks in the literature is through unobservable FDI attacks against power system state estimation (SE). FDI attacks are a class of cyber-attacks that involves an attacker maliciously replacing power system measurements with counterfeits. Under DC power flow assumption*, the true measurement vector $\mathbf{z}$, consisting of the line power flow and bus power injection measurements, is given by

$$\mathbf{z} = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{e}, \tag{4}$$

where $\boldsymbol{\theta}$ is the state vector (voltage angles), $\boldsymbol{H}$ is the dependency matrix between measurements and states, and $\boldsymbol{e}$ is the noise vector. A row of $\boldsymbol{H}$ corresponding to a power flow measurement has non-zero entries only at "from" and "to" buses, which are line admittance and

For simplicity, we focus on DC power flow settings, but our work can be generalized to AC cases as in [3].

its negative, respectively. A row of $\boldsymbol{H}$ corresponding to the power injection measurement at a bus contains a non-zero entry at that bus, which is the negative of the sum of all admittance on lines connecting that bus; as well as non-zero entries at all buses connecting to that bus, which are the admittances of corresponding transmission lines.

Definition 3: A false measurement vector $\bar{\mathbf{z}}$ created with state attack vector $\mathbf{c}$,

$$\bar{\mathbf{z}} = \boldsymbol{H}(\boldsymbol{\theta} + \mathbf{c}) + \boldsymbol{e}, \tag{5}$$

is *unobservable* to the conventional bad data detector (BDD) embedded with SE, because it is not distinguishable from the true measurements if the true states were $(\boldsymbol{\theta} + \mathbf{c})$.

Let $\boldsymbol{B}$ be the dependency matrix between bus power injections and states, and let $\boldsymbol{G}$ be a given generation vector, then the bus power injections without attack can be expressed as

$$\boldsymbol{G} - \boldsymbol{P} = \boldsymbol{B}\boldsymbol{\theta}. \tag{6}$$

With attack, the false injections are given by

$$\boldsymbol{G} - \boldsymbol{P}_{\text{Atk}} = \boldsymbol{B}(\boldsymbol{\theta} + \boldsymbol{c}). \tag{7}$$

Substituting (6) into (7) yields the load change vector

$$\Delta\boldsymbol{P} = \boldsymbol{P}_{\text{Atk}} - \boldsymbol{P} = -\boldsymbol{B}\boldsymbol{c}. \tag{8}$$

Note that since $\mathbf{1}^T\boldsymbol{B} = \mathbf{0}^T$, the net load change is $\sum_i \Delta P_i = -\mathbf{1}^T\boldsymbol{B}\boldsymbol{c} = 0$. Thus, given a generation dispatch, an unobservable FDI attack leads to an LR attack.

2.2 *Intelligently Designed LR Attacks*

Although an attacker can inject arbitrary $\boldsymbol{c}$ as long as it controls the measurements corresponding to all non-zero entries of $\boldsymbol{H}\boldsymbol{c}$, its goal will be to maliciously choose $\boldsymbol{c}$ so that the resulting false loads can mislead the system re-dispatch to cause physical and/or economical consequences. We define these attacks as *intelligent attacks*, whose consequences can be maximized by solving optimization problems. In this paper, we consider two specific intelligent attacks to test the robustness of our proposed detector, namely cost maximization (CM) attacks [25] and line overflow (LO) attacks [3].

CM attacks are a class of FDI attacks that aim to maximize the operation cost after re-dispatch. The attack vector $\boldsymbol{c}$ of CM attacks can be obtained through the following bi-level optimization problem:

$$\begin{aligned}
\underset{\boldsymbol{c}}{\text{maximize}} \quad & \boldsymbol{a}^T\boldsymbol{G}^* && \text{(9a)}\\
\text{subject to} \quad & -\tau\boldsymbol{P} \le \boldsymbol{B}\boldsymbol{c} \le \tau\boldsymbol{P} && \text{(9b)}\\
& \{\boldsymbol{G}^*, \boldsymbol{P}_{\boldsymbol{L}}^*\} = \arg\left\{\min_{\boldsymbol{G},\boldsymbol{P}_{\boldsymbol{L}}} \boldsymbol{a}^T\boldsymbol{G}\right\} && \text{(9c)}\\
& \text{subject to} \quad \sum\boldsymbol{G} = \sum\boldsymbol{P} && \text{(9d)}\\
& \qquad \boldsymbol{P}_{\boldsymbol{L}} = \boldsymbol{R}(\boldsymbol{G} - \boldsymbol{P} + \boldsymbol{B}\boldsymbol{c}) && \text{(9e)}\\
& \qquad -\boldsymbol{P}_{\boldsymbol{L}}^{\max} \le \boldsymbol{P}_{\boldsymbol{L}} \le \boldsymbol{P}_{\boldsymbol{L}}^{\max} && \text{(9f)}\\
& \qquad \boldsymbol{G}^{\min} \le \boldsymbol{G} \le \boldsymbol{G}^{\max} && \text{(9g)}
\end{aligned}$$

where $\boldsymbol{a}$ is the generation cost, $\boldsymbol{P}_{\boldsymbol{L}}$ is the cyber line power flows, $\boldsymbol{R}$ is the power transfer distribution factor (PTDF) matrix, $\boldsymbol{P}_{\boldsymbol{L}}^{\max}$ is the line power flow limits, and $\boldsymbol{G}^{\max}$ and $\boldsymbol{G}^{\min}$ are generation upper and lower limits, respectively. In the upper level, (9a) models the attacker's objective to maximize the operation cost, and (9b) models the load shift limit. The lower level problem (9c)-(9g) is the system DCOPF under attack. This bi-level optimization problem can be converted to a single level mixed-integer linear program (MILP) by replacing the lower level DCOPF with its Karush-Kuhn-Tucker (KKT) conditions [26], and then converting the complementary slackness conditions to mixed integer constraints. The optimal $\boldsymbol{c}$ is obtained by solving the MILP.

LO attacks attempt to maximize the physical power flow on a target line l after re-dispatch, and possibly cause overflows. Optimal $\boldsymbol{c}$ for LO attacks can be obtained by changing the objective function of (9) to maximizing physical power flow:

$$\begin{aligned}
\underset{\boldsymbol{c}}{\text{maximize}} \quad & \left|\boldsymbol{P}_{\boldsymbol{L}}^{l*} - \boldsymbol{R}_l \cdot \boldsymbol{B}\boldsymbol{c}\right| && \text{(10)}\\
\text{subject to} \quad & \text{(9b)} - \text{(9g)},
\end{aligned}$$

where $\boldsymbol{P}_{\boldsymbol{L}}^{l*}$ is the optimal cyber power flow on target line l, $\boldsymbol{R}_l$ is the l^{th} row of $\boldsymbol{R}$, and the second term in (10) characterizes the impact of false loads on the physical power flow of line l.

3 Proposed Attack Detection Framework

Figure 1 illustrates the structure of our proposed LR attack detection framework. During the real-time operation, features are selected from the historical load data until the current time step to capture both spatial and temporal correlations. Loads at the next time step are then predicted by the SVR load predictor using these features. One SVR model is trained for each load using the same features. Subsequently, the SVM attack detector takes the predicted loads and loads estimated after SE to determine the existence of LR attacks.

For detecting attacks, it should suffice to skip the SVR load predictor and plug all SVR features into the SVM to perform classification. However, in this paper we include the SVR for the following two reasons. The first one is that we aim to not only find an attack detection technique, but also have a corrective mechanism when attacks are detected. Using the (accurate) predicted loads to perform control actions when attacks are flagged provides time to locate the attacked measurements without causing severe consequences. The second reason is for easier scaling of the proposed models to large-scale power systems. Without the SVR predictor, the number of features used in SVM classifier will be very large, making it difficult to train and perform real-time classifications. With the SVR predictor in place, the SVM detector only needs the predicted and observed load values as features, making it useful for large-scale systems.

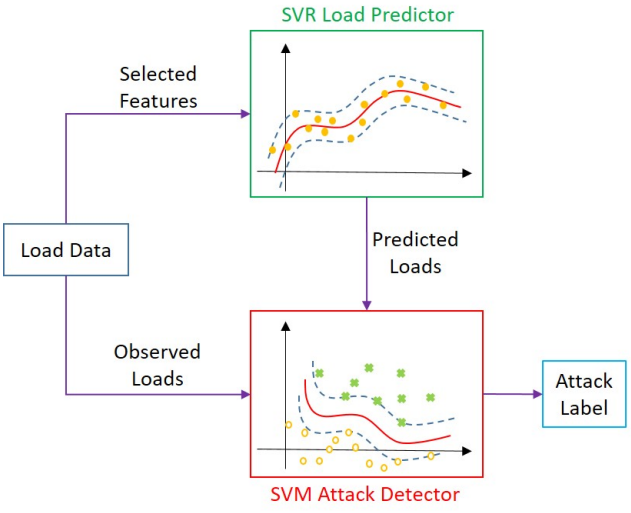

Fig. 1: Structure of the proposed LR attack detection framework.

3.1 *The SVR Load Predictor*

Given data samples $\boldsymbol{x}_j \in \mathbb{R}^p, j = 1, 2, 3, ..., m$ and target values $\boldsymbol{y} \in \mathbb{R}^m$, an SVR attempts to find the best parameters $\boldsymbol{w}_r$ and b_r to fit $|y_j - \boldsymbol{w}_r^T\phi(\boldsymbol{x}_j) - b_r| \le \varepsilon$ by solving the following optimization

problem [14]:

$$\underset{\boldsymbol{w}_r,b_r,\zeta_j,\zeta_j'}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{w}_r^T\boldsymbol{w}_r + M\sum_{j=1}^{n}(\zeta_j+\zeta_j') \tag{11a}$$

$$\text{subject to} \quad y_j - \boldsymbol{w}_r^T\phi(\boldsymbol{x}_j) - b_r \le \varepsilon + \zeta_j \quad (\alpha_j) \tag{11b}$$

$$\boldsymbol{w}_r^T\phi(\boldsymbol{x}_j) + b_r - y_j \le \varepsilon + \zeta_j' \quad (\alpha_j') \tag{11c}$$

$$\zeta_j, \zeta_j' \ge 0, \forall j, \tag{11d}$$

where ε is the regression tolerance, ζ_j, ζ_j' are slack variables to allow for outliers, M is the penalty factor for outliers, α_j, α_j' are dual variables, and $\phi(\cdot)$ is a function that implicitly maps the data samples to a higher dimensional space. The dual formulation has a smaller number of variables and allows for applying the kernel trick:

$$\underset{\boldsymbol{\alpha},\boldsymbol{\alpha'}}{\text{minimize}} \quad \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha'})^T\boldsymbol{Q}(\boldsymbol{\alpha}-\boldsymbol{\alpha'}) \tag{12a}$$

$$+\varepsilon\mathbf{1}^T(\boldsymbol{\alpha}+\boldsymbol{\alpha'}) - y^T(\boldsymbol{\alpha}-\boldsymbol{\alpha'})$$

$$\text{subject to} \quad \mathbf{1}^T(\boldsymbol{\alpha}-\boldsymbol{\alpha'}) = 0 \tag{12b}$$

$$0 \le \alpha_j, \alpha_j' \le M, \forall j \tag{12c}$$

where $\boldsymbol{Q}$ is a positive semi-definite matrix, and $Q_{ij} = Q(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ is the kernel. Once the optimal solutions $(\boldsymbol{\alpha}^*, \boldsymbol{\alpha'}^*)$ are obtained, the regression value y_{new} of a new data sample $\boldsymbol{x}_{\text{new}}$ can be computed as

$$y_{\text{new}} = \sum_{j=1}^{n}(\alpha_j^* - \alpha_j'^*)Q(\boldsymbol{x}_j, \boldsymbol{x}_{\text{new}}). \tag{13}$$

To accurately predict the load values, many different features can be used, including time, weather, temperature, location, and load type (residential/commercial/industrial). Intuitively, it would be the best if we use all the features to perform the prediction, but many of them are unavailable, and some of them may be redundant. The features used in the SVR load predictor also depend on the available dataset. For example, the time step of the prediction depends on how frequently the historical load data are recorded. For the specific dataset we use in this paper, we select time information and historical load values at several time points relative to the target time to capture the temporal correlation, and load values at the same time points for all loads to capture the spatial correlation.

The publicly available PJM hourly zonal metered data [24] is adopted. Our SVR load predictor aims to accurately predict the load values at hour $h+1$ when the current hour is h. The features we use include time information and historical load values up to hour h. We select month (mo), hour (hr), and weekday/weekend (wd) as the time information features, $\boldsymbol{t} = [mo, wd, hr]$. Note that hr here is the wall clock time, for example, $hr = 14$ for 2 PM, and is different than h, which is a unique point in time. Here we only distinguish between weekdays and weekends since loads tend to be similar during weekdays, *i.e.,* $wd = 1$ for weekdays and $wd = 2$ for weekends. The temporal correlation of each load is captured by including its historical values, at hour h and s previous hours; and at hour hr and $hr+1$ of d previous days, as features. For load i, the load value features $\boldsymbol{f}_i$ are given by

$$\boldsymbol{f}_i = [P_i^h, P_i^{h-1}, ..., P_i^{h-s}, P_i^{h-24d}, P_i^{h-24d+1}, ..., P_i^{h-24}, P_i^{h-23}]. \tag{14}$$

Thus, a training data sample $\boldsymbol{x}_j$ consists of $\boldsymbol{t}$ and $\boldsymbol{f}_i$ captures the temporal correlation of load i. To capture the spatial correlations, we concatenate the load value features of all the loads. Once the model is trained, we can extract the same feature at an arbitrary hour to form $\boldsymbol{x}_{\text{new}}$ and predict y_{new}, the loads at the next hour. Note that if bus-level historical data is available in a smaller time resolution, similar concept can be applied to predict bus-level loads at that time resolution.

3.2 The SVM Attack Detector

Given data samples $\boldsymbol{u}_j \in \mathbb{R}^q, j = 1, 2, 3, ...n$ and a vector of class labels $\boldsymbol{v} \in \{1, -1\}^n$, an SVM attempts to find the decision boundary with the maximal margin to best classify $\boldsymbol{u}_j$ by solving the following optimization problem [15]:

$$\underset{\boldsymbol{w}_m,b_m,\lambda_j}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{w}_m^T\boldsymbol{w}_m + C\sum_{j=1}^{n}\lambda_j \tag{15a}$$

$$\text{subject to} \quad v_j(\boldsymbol{w}_m^T\phi(\boldsymbol{u}_j) + b_m) \ge 1 - \lambda_j \quad (\beta_j) \tag{15b}$$

$$\lambda_j \ge 0, \forall j. \tag{15c}$$

Similar to the SVR formulation in (11), λ_j is a slack variable to allow for outliers, C is its penalty factor, and β_j is the dual variable. Again, applying the kernel trick, the dual formulation is used:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{Q}\boldsymbol{\beta} - \mathbf{1}^T\boldsymbol{\beta} \tag{16a}$$

$$\text{subject to} \quad \boldsymbol{v}^T\boldsymbol{\beta} = 0 \tag{16b}$$

$$0 \le \beta_j \le C, \forall j. \tag{16c}$$

Note that here $Q_{ij} = v_iv_jQ(\boldsymbol{u}_i, \boldsymbol{u}_j) = v_iv_j\phi(\boldsymbol{u}_i)^T\phi(\boldsymbol{u}_j)$. Once the optimal solution $\boldsymbol{\beta}$ is acquired, the label v_{new} for a new input data sample $\boldsymbol{u}_{\text{new}}$ can be obtained by

$$v_{\text{new}} = \text{sgn}(\sum_{j=1}^{n} v_j\beta_j^*Q(\boldsymbol{u}_j, \boldsymbol{u}_{\text{new}})) \tag{17}$$

where $\text{sgn}(\cdot)$ is the sign function.

The outputs of the SVR load predictor $\hat{\boldsymbol{P}}$ are used as input features of the SVM attack detector, as well as the time information and the observed loads. Depending on the existence of attack, input data samples of the SVM are given by

$$\boldsymbol{u}_j = [mo, wd, hr, \hat{\boldsymbol{P}}, \boldsymbol{P}], \text{if } v_j = -1, \tag{18a}$$

$$\boldsymbol{u}_j = [mo, wd, hr, \hat{\boldsymbol{P}}, \boldsymbol{P}_{\text{Atk}}], \text{if } v_j = 1, \tag{18b}$$

where $v_j = -1$ indicates that there is no attack, and $v_j = 1$ otherwise. Feeding the training $\boldsymbol{u}_j$ along with their labels v_j into the SVM optimization dual problem (16) yields the optimal solution of $\boldsymbol{\beta}$. Using $\boldsymbol{\beta}$ and $\boldsymbol{u}_j$, we can determine the label v_{new} of any testing data $\boldsymbol{u}_{\text{new}}$ as in (17).

3.3 Generating Random LR Attacks to Train the SVM

We train the SVM detector using normal data and randomly designed LR attacks. The SVM detector trained using random attacks is expected to maximally explore the space of LR attacks, and hence, perform well in detecting any LR attacks. Given true loads $\boldsymbol{P}$, the false loads $\boldsymbol{P}_{\text{Atk}}$ in these random attacks are acquired using (1), $\boldsymbol{P}_{\text{Atk}} = \boldsymbol{P} + \Delta\boldsymbol{P}$. Thus, finding $\boldsymbol{P}_{\text{Atk}}$ is equivalent to finding $\Delta\boldsymbol{P}$. In each attack, we assume the attacker changes K loads at random, whose indices form a set $\mathcal{K}$, so that $P_{\mathcal{K}(k)}$ and $\Delta P_{\mathcal{K}(k)}$ indicates the load and load change of the k^{th} attacked load, respectively, $k = 1, 2, \ldots, K$. The load changes of these attacked loads, denoted $\boldsymbol{\gamma}$, can be arbitrary. However, according to the LR attack property (2), they must be constrained to have a 0 sum. Thus, we model $\boldsymbol{\gamma}$ with a joint Gaussian distribution with 0 mean and covariance matrix

$\boldsymbol{\Gamma}$:

$$\boldsymbol{\gamma} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma}) \tag{19}$$

$$\gamma_k = \Delta P_{\mathcal{K}(k)}. \tag{20}$$

Given a load shift τ, the diagonal entries of $\boldsymbol{\Gamma}$ must satisfy

$$\Gamma_{kk} = Var(\gamma_k) = (\frac{1}{2}\tau P_{\mathcal{K}(k)})^2, \forall k \tag{21}$$

to ensure that the probability of $|\gamma_k| \leq \tau P_{\mathcal{K}(k)}$ is 95%, because the probability of deviating beyond $2\times$standard deviation in a Gaussian distribution is 5%. Recall that the load changes caused by a valid LR attack must satisfy (2), which can be rewritten as

$$\sum_i \Delta P_i = \sum_k \Delta P_{\mathcal{K}(k)} = \mathbf{1}^T \boldsymbol{\gamma} = 0. \tag{22}$$

Eq. (22) is equivalent to

$$\begin{aligned} E[(\mathbf{1}^T\boldsymbol{\gamma})^2] &= E[\mathbf{1}^T\boldsymbol{\gamma}\boldsymbol{\gamma}^T\mathbf{1}] \\ &= \mathbf{1}^T\boldsymbol{\Gamma}\mathbf{1} \\ &= 0. \end{aligned} \tag{23}$$

Finding a valid $\boldsymbol{\gamma}$ is equivalent to finding a positive semidefinite matrix $\boldsymbol{\Gamma}$ that satisfies (21) and (23). Since $\boldsymbol{\Gamma}$ is a covariance matrix, it must be positive semidefinite:

$$\boldsymbol{\Gamma} \succeq 0. \tag{24}$$

Any $\boldsymbol{\Gamma}$ satisfying (21), (23) and (24) would suffice for our application. Finding $\boldsymbol{\Gamma}$ is equivalent to solving a semidefinite program with arbitrary objective, constrained by (21), (23) and (24). The procedure to acquire false loads $\boldsymbol{P}_{\text{Atk}}$ is summarized in Alg. 1. Varying the attack hour h, load shift τ, and number of attacked loads K, we can find feasible $\boldsymbol{\Gamma}$ to obtain $\boldsymbol{\gamma}$ using (19), and subsequently create an arbitrary number of false loads $\boldsymbol{P}_{\text{Atk}}$ using (1). Note that for specific combinations of h, τ, K, and $\mathcal{K}$, sometimes no feasible $\boldsymbol{\Gamma}$ can be found, but we can simply re-run Alg.1 with different inputs. Since (19) is drawing $\boldsymbol{\gamma}$ randomly from a Gaussian distribution, the resulting real load shift τ_r of $\boldsymbol{P}_{\text{Atk}}$ may be different than the input τ. We keep drawing $\boldsymbol{\gamma}$ until $\tau_r \leq \tau$. The false loads created are then used to generate data samples to train and test the SVM detector.

Algorithm 1 Generating random LR attack false loads

Input: h, K, τ
Output: $\boldsymbol{P}_{\text{Atk}}$

1. Obtain the true loads $\boldsymbol{P}$ at hour h.
2. Randomly select K loads to attack and let $\mathcal{K}$ denote the set of indices of the attacked loads.
3. Find a $\boldsymbol{\Gamma}$ satisfying (21), (23) and (24) with $\tau, K, \mathcal{K}$, and $\boldsymbol{P}$. This can be done by solving a semidefinite program with arbitrary objective, constrained by (21), (23) and (24). If no feasible $\boldsymbol{\Gamma}$ can be found, terminate.
4. Draw the non-zero load changes $\boldsymbol{\gamma}$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma})$ and obtain false loads $\boldsymbol{P}_{\text{Atk}}$ using (1).
5. Calculate the real load shift τ_r of $\boldsymbol{P}_{\text{Atk}}$ using (3). If $\tau_r > \tau$, go to step 4). Otherwise, terminate.

4 Numerical Results

We use the publicly available PJM zonal hourly metered load data [24] from 2015 through 2018 for 20 transmission zones as the historical data to train and test our LR attack detection framework. In order to conveniently create intelligently designed LR attacks as described in Section 2.2, we map each zone to a load bus in the IEEE 30-bus system, leveraging the fact that there are 20 loads in this system. The mapping relationship is adopted from [13], and all load values are multiplied by a scaling factor of 1.308×10^{-3} to obtain a system with moderate amount of congestion. Table 1 describes the mapping rules between load indices, PJM zones, and bus indices. The SVR and SVM models are implemented in Python using the Scikit-learn package [27]. The random, CM and LO attack creation are implemented in Matlab with solver Gurobi. All experiments are conducted on a 2.7 GHz CPU with 32 GB RAM.

Table 1 Mapping rules between load indices, PJM zones, and bus indices

Load	Zone	Bus	Load	Zone	Bus
1	DOM	2	11	PL	17
2	AE	3	12	PN	18
3	JC	4	13	PE	19
4	CE	7	14	RECO	20
5	AEP	8	15	ATSI	21
6	DPL	10	16	DUQ	23
7	PS	12	17	BC	24
8	DEOK	14	18	ME	26
9	PEP	15	19	EKPC	29
10	DAY	16	20	AP	30

4.1 The SVR Load Predictor Performance

The multi-output SVR load predictor is achieved by solving one SVR optimization problem (11) for each load. In our experiments, we trained three SVR models to justify the contribution of capturing spatial correlations, as well as to see the influence of different selected features. Model 1 predicts each load using only time information $\boldsymbol{t}$ and its own load value features. A data sample used in Model 1 to predict load i is given by

$$\boldsymbol{x}_{j,i} = [\boldsymbol{t}, \boldsymbol{f}_i] \forall i. \tag{25}$$

Model 2 and 3 use $\boldsymbol{t}$ and $\boldsymbol{f}_i, \forall i$, as features to predict all loads. A data sample in these two models is given by

$$\boldsymbol{x}_j = [\boldsymbol{t}, \boldsymbol{f}_1, \boldsymbol{f}_2, ...\boldsymbol{f}_{n_l}], \tag{26}$$

where n_l is the number of loads in the system. In Model 2, $s = 3$ and $d = 2$; and in Model 3, $s = 4$ and $d = 3$. The ground truth $y_{j,i} = P_i^{h+1}$ is the true load value at hour $h + 1$ for load i. Table 2 presents some properties of the three tested SVR models. Comparing Models 1 and 2, we can see the influence of considering spatial correlations in addition to temporal correlations, as these two models use the same temporal features, but Model 2 additionally uses the features of all the loads to capture spatial correlations.

Table 2 Statistics of SVR models

Model	s	d	m	p	Training time (h)
1	3	2	35011	11	1.927
2	3	2	35011	163	4.234
3	4	3	34987	223	33.324

The dimension of the data matrix $\boldsymbol{X}, m \times p$, and target value matrix $\boldsymbol{Y}, m \times n_l$, depend on the values of s and d. Derivation of m and p are described in the Appendix. For each model, the training

data matrix $\boldsymbol{X}_{\text{train}}$ contains all data from 2015 - 2017, and data in 2018 are used as $\boldsymbol{X}_{\text{test}}$. Each column of $\boldsymbol{X}_{\text{train}}$ is scaled to zero mean and unit variance, and each column of $\boldsymbol{X}_{\text{test}}$ is scaled using the mean and variance of the corresponding column in $\boldsymbol{X}_{\text{train}}$. The same split and scaling are performed on $\boldsymbol{Y}$ to obtain $\boldsymbol{Y}_{\text{train}}$ and $\boldsymbol{Y}_{\text{test}}$ as well. The parameters in training the SVR models are chosen as $\varepsilon = 10^{-2}$ and $M = 100$. The radial basis function (RBF) kernel

$$Q(\boldsymbol{x}_i, \boldsymbol{x}_j) = -\sigma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \tag{27}$$

is used with $\sigma = 10^{-2}$. Applying the trained SVR predictor on $\boldsymbol{X}_{\text{train}}$ and $\boldsymbol{X}_{\text{test}}$ yields the predicted loads $\hat{\boldsymbol{Y}}_{\text{train}}$ and $\hat{\boldsymbol{Y}}_{\text{test}}$, respectively.

Two metrics are used to evaluate the performance of the SVR load predictor, namely root mean square error (RMSE) and mean absolute percentage error (MAPE). RMSE measures the square root of the average squared error for each load, and hence the unit is MW. MAPE measures on average how much the predicted loads deviate from their true values in percentage. These metrics for each load i are calculated as

$$\text{RMSE}_{\text{train},i} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(\boldsymbol{Y}_{\text{train},i,j} - \hat{\boldsymbol{Y}}_{\text{train},i,j})^2} \tag{28}$$

$$\text{MAPE}_{\text{train},i} = \frac{1}{m}\sum_{j=1}^{m}\left|\frac{\boldsymbol{Y}_{\text{train},i,j} - \hat{\boldsymbol{Y}}_{\text{train},i,j}}{\boldsymbol{Y}_{\text{train},i,j}}\right| \tag{29}$$

where $\boldsymbol{Y}_{\text{train},i}$ is the i^{th} column of $\boldsymbol{Y}_{\text{train}}$, and $\bar{\boldsymbol{Y}}_{\text{train},i}$ is its mean. These metrics are similarly applied on $\boldsymbol{Y}_{\text{test}}$ to evaluate the performance of the SVR load predictor on testing data.

Figures 2 illustrates the RMSE and MAPE for the SVR models. RMSE values largely depend on the load values itself, for example, load 5 has the largest RMSE value because it is the biggest load in the system. From Figure 2(b) we can see that the MAPE for most loads are around 1%, and MAPE for load 19, the most difficult load to predict, is around 2%. Comparing these quantities for Models 1 and 2, we can see that they are both smaller for Model 2. Recall that the difference between Models 1 and 2 is that Model 2 considers all prior loads, while Model 1 only includes the prior data at the load of interest. This result indicates that considering spatial correlations does improve the performance of the SVR load predictor. Comparing Models 2 and 3, it can be concluded that including too much historical data as features decreases the accuracy of the SVR load predictor. Besides, it can be seen from Table 2 that using too many features makes it extremely slow in training the SVR model. Thus, in the following sections, Model 2 is adopted to generate predicted loads used by the SVM attack detector.

In addition, we benchmark the performance of our SVR predictor against three commonly used regression techniques, namely least-squares (LS), ridge regression, and LASSO, in terms of RMSE and MAPE. Least-squares is pure linear regression. Ridge regression is least-squares linear regression with regularization on the l_2-norm of the coefficients, while LASSO regularizes on the l_1-norm. Least-squares attempts to solve

$$\underset{\boldsymbol{w},b}{\text{minimize}} \quad \sum_j (y_j - \boldsymbol{w}^T\boldsymbol{x}_j - b)^2. \tag{30}$$

With regularization, ridge regression aims to find the optimal solution to the following optimization problem

$$\underset{\boldsymbol{w},b}{\text{minimize}} \quad \sum_j (y_j - \boldsymbol{w}^T\boldsymbol{x}_j - b)^2 + \rho_r\|\boldsymbol{w}\|_2^2, \tag{31}$$

while LASSO solves

$$\underset{\boldsymbol{w},b}{\text{minimize}} \quad \sum_j (y_j - \boldsymbol{w}^T\boldsymbol{x}_j - b)^2 + \rho_l\|\boldsymbol{w}\|_1, \tag{32}$$

where ρ_r and ρ_l are regularization parameters for ridge regression and LASSO, respectively.

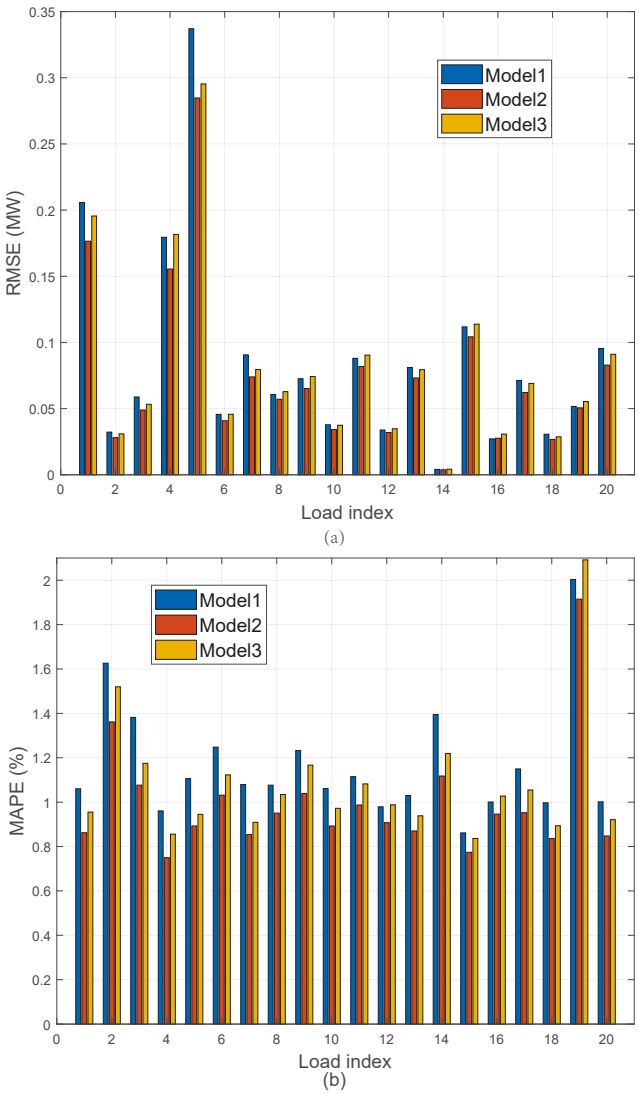

Fig. 2: Performance of the SVR models under two metrics: (a) RMSE, and (b) MAPE. Model 1 does not capture spatial correlations. Model 2 uses temporal features of 3 previous hours and 2 previous days. Model 3 uses temporal features of 4 previous hours and 3 previous days. Both Models 2 and 3 capture spatial correlation.

Figures 3(a) and 3(b) illustrate the RMSE and MAPE, respectively, on testing data $\boldsymbol{X}_{\text{test}}$ of our SVR predictor (Model 2), least-squares, ridge regression, and LASSO. All models are trained in Scikit-learn using the same training data $\boldsymbol{X}_{\text{train}}$. Ridge regression and LASSO regularization parameters are $\rho_r = \rho_l = 1$, which are the default settings as in the SVR case. It can be seen that the SVR model outperforms the other three regression approaches.

4.2 The SVM Attack Detector Performance on Random Attacks

The predicted loads $\hat{\boldsymbol{P}}$ of $m = 35011$ hours, along with their ground truth values $\boldsymbol{P}$ and time information, yield 35011 *normal* data samples for the SVM detector in the form of (18a). The length of each data sample $q = 3 + 20 \times 2 = 43$. The *normal* data matrix $\boldsymbol{U}_{\text{normal}}$ is of size 35011×43. We randomly select 80% of these vectors for training and the remaining 20% for testing. We create 10^5 *attacked* data samples in the form of (18b) using Alg. 1, resulting in $\boldsymbol{U}_{\text{attack}}$ of size $10^5 \times 43$ with real load shift τ_r ranging from 1% to 20%. From now on, we omit the subscript in τ_r for easier presentation.

We obtain different SVM models to compare their performances by varying the penalty factor C and $\tau_{\min}$ (the minimal τ used in the

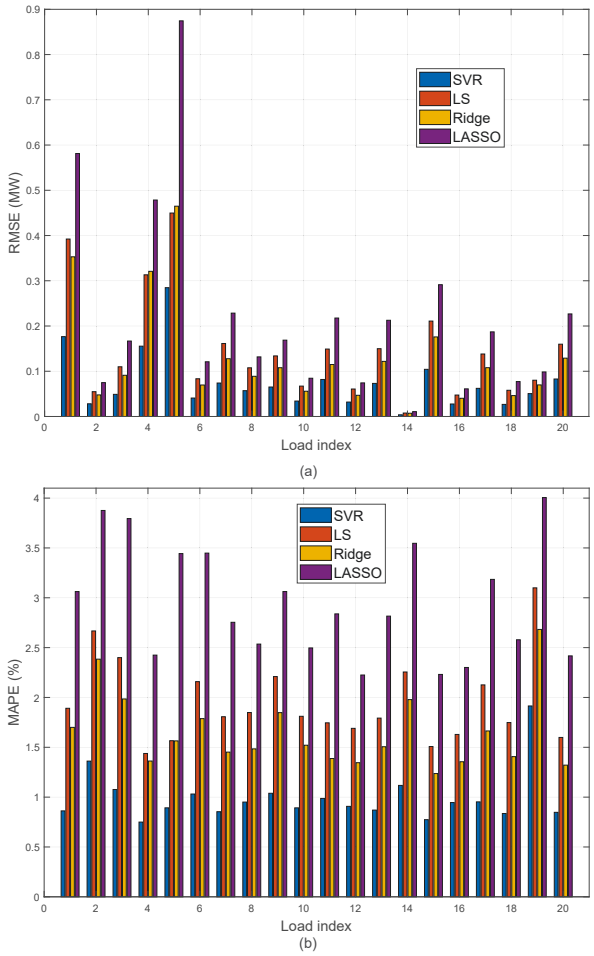

Fig. 3: Performance comparison of SVR against least-squares, ridge regression, and LASSO, in terms of (a) RMSE, and (b) MAPE.

training data). The normal data in the training data matrix $\boldsymbol{U}_{\text{train}}$ are the same for all models, *i.e.,* the same 80% of $\boldsymbol{U}_{\text{normal}}$. The attacked data in $\boldsymbol{U}_{\text{train}}$ include 80% of attacked data samples with $\tau \geq \tau_{\min}$. The testing data $\boldsymbol{U}_{\text{test}}$ consists of the remaining 20% of attacked data that are not used in training with all load shifts, and are the same for all models. For each model, every column of training data matrix $\boldsymbol{U}_{\text{train}}$ is scaled to zero mean and unit variance, and the same scaling is performed to the testing data. The kernel function used in the SVM detector is also the RBF kernel in the form of (27), but this time σ is calculated as $\sigma = 1/q$ (this is the "*scale*" option in Scikit-learn).

Figure 4 illustrates the effect of $\tau_{\min}$ on missed detection rate and false alarm rate. The false alarm rate is calculated by applying the detector on all $m = 35011$ normal data samples, including both training and testing. The parameter C is fixed at 1000. $\tau_{\min}$ controls the amount of attacked training data. For instance, if $\tau_{\min} = 3\%$, $\boldsymbol{U}_{\text{train}}$ contains 80% of attacks with $\tau \geq 3\%$, but does not contain any attack with $\tau < 3\%$. Intuitively, attacks with higher τ are further away from the normal data than those with lower τ. Thus, a detector trained with a low $\tau_{\min}$ will have a high false alarm rate, as the SVM is trying to find a decision boundary between normal data and attacks with small load shift. However, it should perform better in detecting attacks with small τ than detectors trained with large $\tau_{\min}$. If system operators can tolerate the consequences of small attacks, they can increase $\tau_{\min}$ to achieve smaller false alarm, and vice versa. In Figure 4, the blue lines indicate the missed detection rate of attacks with certain load shift τ, and the red line shows the false alarm rate. It can be seen that as $\tau_{\min}$ increases, the false alarm rate decreases, but the missed detection rate increases for attacks with small load shifts. This observation justifies the intuition discussed above, indicating that $\tau_{\min}$ is indeed a trade-off between false alarm rate and detection probability for small attacks. Note that for attacks with large τ, the effect of $\tau_{\min}$ is insignificant. For testing attacks with extremely small τ, the missed detection rates are very high even with small $\tau_{\min}$, because these attacks are in principle very difficult to detect. However, these attacks are also unlikely to cause severe consequences. From Figure 4, we can see that $\tau_{\min} = 3\%$ is a good choice for our dataset.

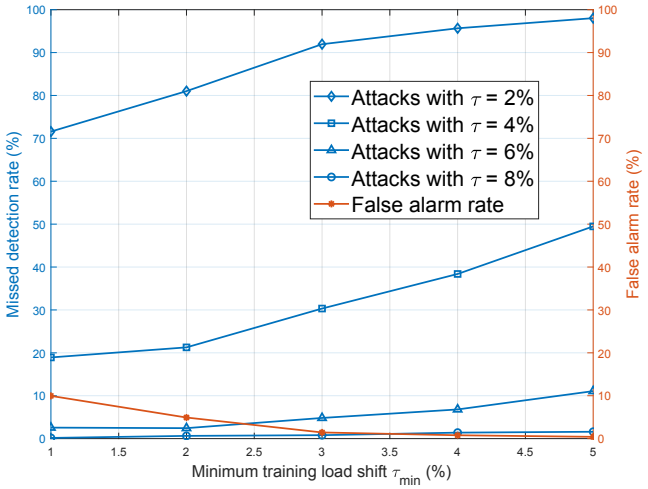

Fig. 4: Effect of minimum training load shift $\tau_{\min}$. False alarm rate and missed detection rate when testing random attacks are each plotted as a function of $\tau_{\min}$. Data is shown for $C = 1000$.

The parameter C trades off misclassification of training examples against simplicity of the decision boundary. A small C makes the decision boundary smooth, while a large C aims at classifying all training samples correctly. Therefore, detector with large C is expected to have a better performance. However, a large C allows for fewer outliers, making it harder to solve the SVM optimization problem (15), so the training time increases. System operators can determine the value of C according to the computational capability of their hardware. Figure 5 shows the performance of models trained with different C on testing random attacks while fixing $\tau_{\min} = 3\%$. The larger C is, the higher detection probability we can achieve. This model performs well on attacks with large τ, and the detection probability almost achieves 100% starting at $\tau = 7\%$. System operators can similarly vary $\tau_{\min}$ and C to obtain SVM model with satisfactory performance, in terms of false alarm rate and missed detection rate.

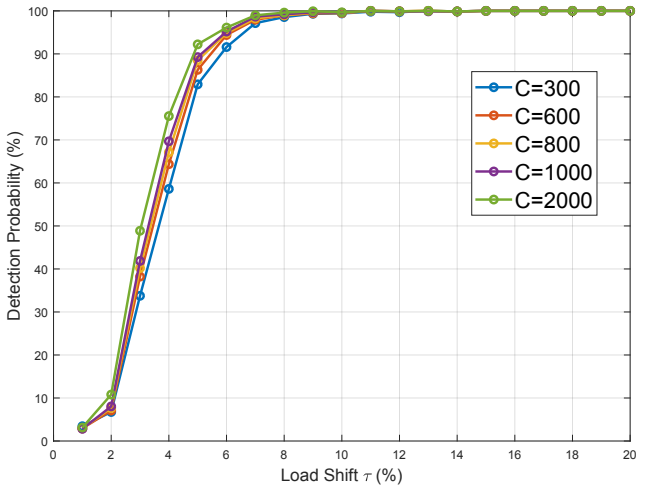

Fig. 5: Effect of outlier penalty factor C on testing random attack detection probability. Data is shown for $\tau_{\min} = 3\%$.

4.3 The SVM Attack Detector Performance on Intelligently Designed LR Attacks

In this section, we evaluate the performance of the trained SVM detector on cost maximization (CM) and line overflow (LO) attacks. According to the previous section, here we choose SVM parameters $C = 2000$ and $\tau_{\min} = 3\%$ to balance false alarm rate and missed detection. The procedures to generate these attacks are described as follows. On the IEEE 30-bus system, we first perform base case DCOPF for each hour in year 2015 through 2018 using the true loads. At hour h, if there are at least 2 lines whose power flows are greater than 80% of their ratings, we say those lines are *critical lines*, and h is a *critical hour*. The total number of critical hours is found to be 8038. We focus on critical hours because the false loads are likely to cause congestions at those times, which in turn change the generation dispatch to have consequences. For each critical hour, we solve optimization problem (9) 20 times to obtain attack vector $\boldsymbol{c}$ fo CM attacks with $\tau = 1\%, 2\%, \ldots, 20\%$. For each critical line, we solve (10) 20 times to obtain $\boldsymbol{c}$ for LO attacks, also with $\tau = 1\%, 2\%, \ldots, 20\%$. Every non-zero $\boldsymbol{c}$ is used to construct false load vector $\boldsymbol{P}_{\text{Atk}}$ as in (8). If a $\boldsymbol{P}_{\text{Atk}}$ makes the DCOPF infeasible, it is discarded. The total number of false loads for CM attacks and LO attacks are 113031 and 343135, respectively.

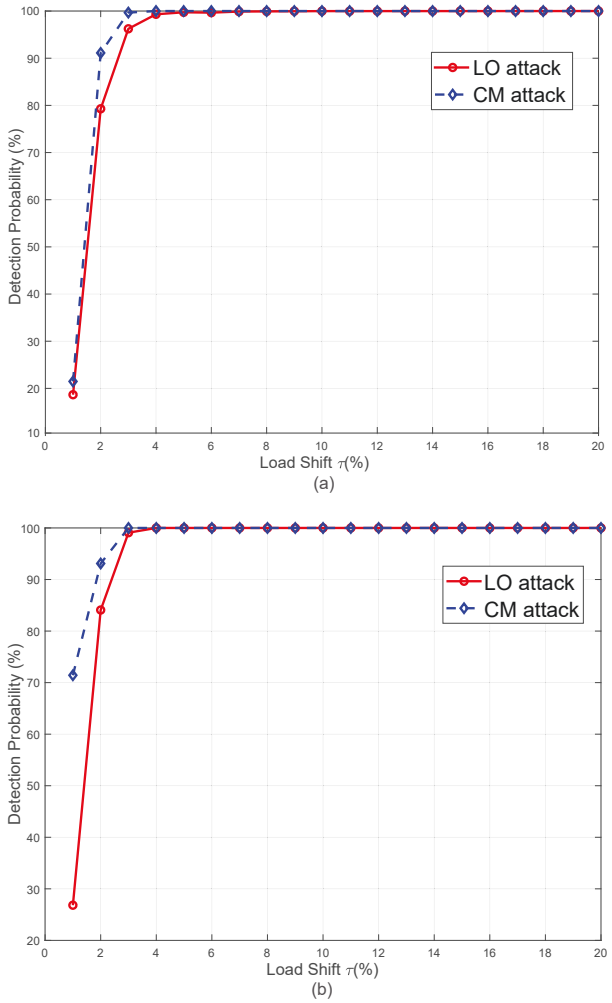

Fig. 6: Detection probability on CM and LO attacks as a function of load shift τ. Subplot (a) is for all attacks, and subplot (b) is only for attacks with consequences. Data is shown for $\tau_{\min} = 3\%$ and $C = 2000$.

Figure 6(a) illustrates the detection probability versus the load shift τ on CM and LO attacks. For both attacks, the detection probabilities almost achieve 100% when $\tau \geq 4\%$. For attacks with $\tau = 3\%$, the detector performance drops to 97% for LO attacks, but it is still perfect in detecting CM attacks. Comparing with the performance on random attacks as shown in Figure 5, it can be seen that intelligently designed attacks are easier to detect than random attacks.

Figure 6(b) illustrates the detection probability versus load shift τ on CM and LO attacks with consequences. CM attacks with consequences are those that increase the operating cost by more than 1%. LO attacks with consequences are those result in physical overflows. Comparing Figures 6(a) and 6(b), it can be seen that the detector performs even better on attacks with consequences.

5 Attack Mitigation

If LR attack is flagged by our detection framework, the simplest way to mitigate the attacks is to temporarily use the loads output by the SVR load predictor for re-dispatching purposes. To test the mitigation performance using this method, we compare the worst consequences of intelligently designed attacks with and without our detection framework.

In order to obtain the consequences, we run DCOPF three times using different loads. Under normal operation, running DCOPF with true loads $\boldsymbol{P}_{\text{normal}}$ yields the attack-free generation dispatch $\boldsymbol{G}_{\text{normal}}$. Using attacked loads $\boldsymbol{P}_{\text{Atk}}$ to run DCOPF gives attacked dispatch $\boldsymbol{G}_{\text{Atk}}$. Applying $\boldsymbol{G}_{\text{Atk}}$ on true loads $\boldsymbol{P}_{\text{normal}}$ yields attacked line flows $\boldsymbol{P}_{\boldsymbol{L},\text{Atk}} = \boldsymbol{R}(\boldsymbol{G}_{\text{Atk}} - \boldsymbol{P}_{\text{normal}})$. When an attack is detected, the system runs DCOPF using the SVR predicted loads $\boldsymbol{P}_{\text{SVR}}$ and the resulting dispatch is $\boldsymbol{G}_{\text{SVR}}$. The corresponding line flows are given by $\boldsymbol{P}_{\boldsymbol{L},\text{SVR}} = \boldsymbol{R}(\boldsymbol{G}_{\text{SVR}} - \boldsymbol{P}_{\text{normal}})$.

Figure 7(a) illustrates the mitigation results for CM attacks. The word "maximum" on the y-axis indicates the worst consequence among all attacks with each load shift τ. The red line indicates the maximum cost increase without using our proposed detection framework, calculated as $\boldsymbol{a}^T(\boldsymbol{G}_{\text{Atk}} - \boldsymbol{G}_{\text{normal}})$ (recall that $\boldsymbol{a}$ is the generation cost vector). When an attack is detected, the resulting cost increase is obtained by $\boldsymbol{a}^T(\boldsymbol{G}_{\text{SVR}} - \boldsymbol{G}_{\text{normal}})$. When the detector fails to detect an attack, the cost increase is the attack consequence $\boldsymbol{a}^T(\boldsymbol{G}_{\text{Atk}} - \boldsymbol{G}_{\text{normal}})$. Thus, for each load shift, if all attacks are detected, the data point on the blue line is given by $\boldsymbol{a}^T(\boldsymbol{G}_{\text{SVR}} - \boldsymbol{G}_{\text{normal}})$. Otherwise, it is $\max[\boldsymbol{a}^T(\boldsymbol{G}_{\text{Atk}} - \boldsymbol{G}_{\text{normal}}), \boldsymbol{a}^T(\boldsymbol{G}_{\text{SVR}} - \boldsymbol{G}_{\text{normal}})]$. Similar procedure is performed to create Figure 7(b) for LO attacks. The red line is obtained by taking the maximum $\boldsymbol{P}^l_{\boldsymbol{L},\text{Atk}}$ for each load shift (line l is the target line). The blue line is obtained by $\boldsymbol{P}^l_{\boldsymbol{L},\text{SVR}}$ if all attacks are detected, and $\max[\boldsymbol{P}^l_{\boldsymbol{L},\text{Atk}}, \boldsymbol{P}^l_{\boldsymbol{L},\text{SVR}}]$ otherwise.

From Figures 7(a), we can see that for load shift $\tau \geq 3\%$, the increases in operation cost are significantly reduced by using SVR predicted loads when an attack is flagged. For LO attacks, the overflows are significantly reduced for load shift $\tau \geq 4\%$. The largest cost increase caused by CM attacks that are not detected is 8.17% (at $\tau = 2\%$), and the largest overflow caused by LO attacks that are not detected is 3.96% (at $\tau = 3\%$). Thus, even though our detector fails to detect a small number of attacks, their consequences are minor. Note that at $\tau = 1\%$, using the SVR predicted loads leads to larger overflow due to inaccurate predictions, but the overflow is still very small. Therefore, the consequences of LR attacks can be successfully mitigated using the SVR predicted loads, which gives operators time to take other corrective actions.

6 Concluding Remarks

A machine learning based load redistribution (LR) attack detection framework is proposed. This detection framework consists of a support vector regression (SVR)-based load predictor and a support vector machine (SVM)-based attack detector. The SVR load predictor is trained using features selected from historical load data to capture both spatial and temporal correlations. The prediction results indicate that the SVR load predictor can accurately predict loads at

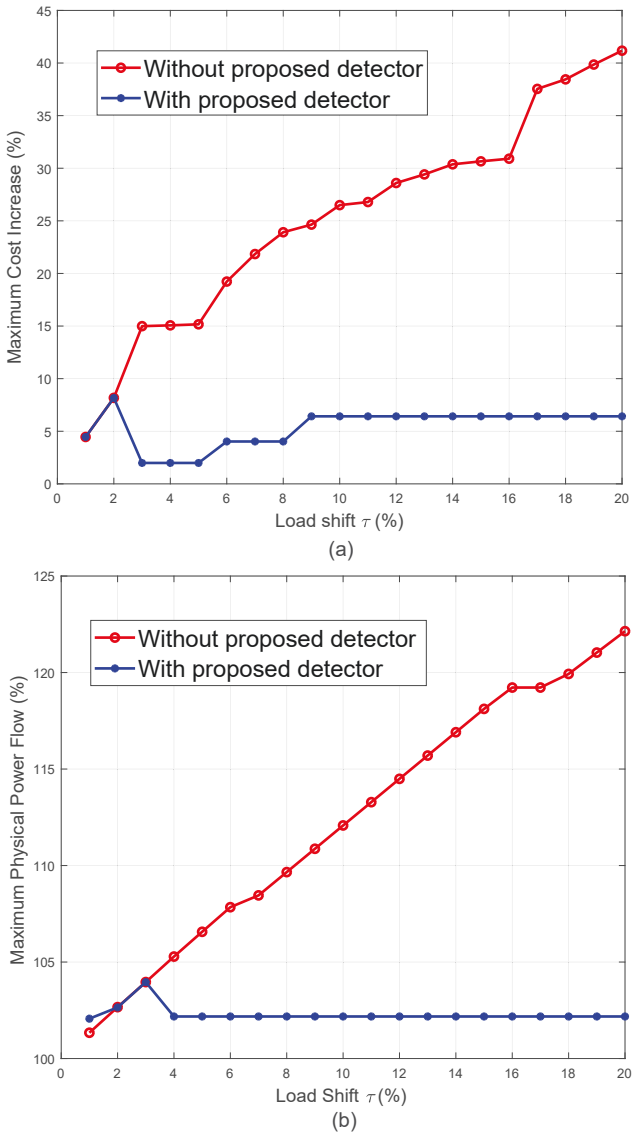

Fig. 7: Mitigation results of (a) CM attacks and (b) LO attacks. For each load shift, the points on the red lines indicate the worst consequence as a result of attack, and the points on the blue lines indicate the worst consequence with our attack detection framework. Points on the blue line are obtained by taking the maximum of two quantities: (i) resulting worst consequence if re-dispatch using SVR predicted loads when attack is flagged; and (ii) the worst attack consequence when the detector fails.

all buses. The SVM attack detector is trained using randomly generated LR attacks, and is shown to be effective in detecting both randomly generated and intelligently designed attacks, especially those with consequences. Using the proposed attack detection framework, system operators can make control decisions based on the predicted loads when attack is flagged to mitigate the consequence of the attacks. It also gives operators time to find the source of the attacks. Future work will include finding attack localization techniques that help system operators identify the loads and/or meters that are modified by the attacker.

Acknowledgment

This material is based on work supported by the National Science Foundation (NSF) under grant number CNS-1449080, and two grants from the Power System Engineering Research Center (PSERC) S-72 and S-74.

7 References

1 Liu, Y., Ning, P., Reiter, M.K. 'False data injection attacks against state estimation in electric power grids'. In: 16th ACM Conference on Computer and Communications Security. CCS '09, Chicago, Illinois, USA, 2009. pp. 21–32
2 Zhang, J., Sankar, L.: 'Physical system consequences of unobservable state-and-topology cyber-physical attacks', *IEEE Transactions on Smart Grid*, 2016, **7**, (4), pp. 2016–2025
3 Liang, J., Sankar, L., Kosut, O.: 'Vulnerability analysis and consequences of false data injection attack on power system state estimation', *IEEE Transactions on Power Systems*, 2016, **31**, (5), pp. 3864–3872
4 Moslemi, R., Mesbahi, A., Velni, J.M.: 'Design of robust profitable false data injection attacks in multi-settlement electricity markets', *IET Generation, Transmission Distribution*, 2018, **12**, (6), pp. 1263–1270
5 Jia, L., Kim, J., Thomas, R.J., Tong, L.: 'Impact of data quality on real-time locational marginal price', *IEEE Trans Power Systems*, 2014, **29**, (2), pp. 627–636
6 An, Y., Liu, D.: 'Multivariate gaussian-based false data detection against cyber-attacks', *IEEE Access*, 2019, **7**, pp. 119804–119812
7 Liu, C., Wu, J., Long, C., Kundur, D.: 'Reactance perturbation for detecting and identifying FDI attacks in power system state estimation', *IEEE Journal of Selected Topics in Signal Processing*, 2018, **12**, (4), pp. 763–776
8 Che, L., Liu, X., Li, Z.: 'Mitigating false data attacks induced overloads using a corrective dispatch scheme', *IEEE Transactions on Smart Grid*, 2019, **10**, (3), pp. 3081–3091
9 Li, X., Hedman, K.W.: 'Enhancing power system cyber-security with systematic two-stage detection strategy', *IEEE Transactions on Power Systems*, 2019, pp. 1–1
10 Liu, C., Zhou, M., Wu, J., Long, C., Kundur, D.: 'Financially motivated fdi on sced in real-time electricity markets: Attacks and mitigation', *IEEE Transactions on Smart Grid*, 2019, **10**, (2), pp. 1949–1959
11 Ozay, M., Esnaola, I., Yarman Vural, F.T., Kulkarni, S.R., Poor, H.V.: 'Machine learning methods for attack detection in the smart grid', *IEEE Transactions on Neural Networks and Learning Systems*, 2016, **27**, (8), pp. 1773–1786
12 An, D., Yang, Q., Liu, W., Zhang, Y.: 'Defending against data integrity attacks in smart grid: A deep reinforcement learning-based approach', *IEEE Access*, 2019, **7**, pp. 110835–110845
13 Pinceti, A., Sankar, L., Kosut, O. 'Load redistribution attack detection using machine learning: A data-driven approach'. In: 2018 IEEE Power Energy Society General Meeting (PESGM), 2018. pp. 1–5
14 Smola, A.J., SchÃűlkopf, B.: 'A tutorial on support vector regression', *Statistics and Computing*, 2004,
15 Cortes, C., Vapnik, V.: 'Support-vector networks', *Machine Learning*, 1995, **20**, (3), pp. 273–297
16 Yuanhang, D., Lei, C., Weiling, Z., Yong, M. 'Multi-support vector machine power system transient stability assessment based on relief algorithm'. In: 2015 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), 2015. pp. 1–5
17 Eskandarpour, R., Khodaei, A. 'Component outage estimation based on support vector machine'. In: 2017 IEEE Power Energy Society General Meeting, 2017. pp. 1–5
18 Kirincic, V., Ceperic, E., Vlahinic, S., Lerga, J.: 'Support vector machine state estimation', *Applied Artificial Intelligence*, 2019, **33**, (6), pp. 517–530
19 Qiang, S., Pu, Y.: 'Short-term power load forecasting based on support vector machine and particle swarm optimization', *Journal of Algorithms & Computational Technology*, 2019, **13**
20 Capuno, M., Kim, J.S., Song, H.: 'Very short-term load forecasting using hybrid algebraic prediction and support vector regression', *Mathematical Problems in Engineering*, 2017,
21 Su, F., Xu, Y., Tang, X. 'Short-and mid-term load forecasting using machine learning models'. In: 2017 China International Electrical and Energy Conference (CIEEC), 2017. pp. 406–411
22 Azad, M.K., Uddin, S., Takruri, M. 'Support vector regression based electricity peak load forecasting'. In: 2018 11th International Symposium on Mechatronics and its Applications (ISMA), 2018. pp. 1–5
23 Chong, L.W., Rengasamy, D., Wong, Y.W., Rajkumar, R.K. 'Load prediction using support vector regression'. In: TENCON 2017 - 2017 IEEE Region 10 Conference, 2017. pp. 1069–1074
24 PJM. 'PJM metered hourly zonal load data'. (PJM), 2019. PJM Data Miner 2 https://dataminer2.pjm.com/feed/hrl_load_metered/definition
25 Yuan, Y., Li, Z., Ren, K.: 'Modeling load redistribution attacks in power systems', *Smart Grid, IEEE Transactions on*, 2011, **2**, (2), pp. 382–390
26 Boyd, S., Vandenberghe, L.: 'Convex Optimization'. (Cambridge University Press, 2004)
27 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al.: 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research*, 2011, **12**, pp. 2825–2830

Appendix

The parameters s and d in (14) determines the dimension of SVR input data matrix $\boldsymbol{X}$, $m \times p$. For example, for Model 2, $s = 3$ and $d = 2$, the length of $\boldsymbol{f}_i$ is given by

$$n_f = s + 1 + 2d = 8. \tag{33}$$

The resulting data sample length $p = 3 + 20 \times n_f = 163$. Since we use load values of previous $d = 2$ days as features, the start hour of our data is 01/03/2015, 0 AM. The end hour is 12/31/2018, 10 PM because for 12/31/2018, 11 PM, we do not have ground truth values of its next hour. In each of the four years, the hour when daylight saving time ends has two load values with identical time stamps, and we approximate the load value at this hour by taking the average of those two values. As a result, the number of data samples for the SVR load predictor is

$$m = (365 \times 3 + 366 - d) * 24 - 1 - 4 = 35011. \quad (34)$$

The target values for hour h are the metered loads of the 20 zones at hour $h + 1$. Thus, for each data sample of length $p = 163$, the SVR outputs a vector of length 20 as prediction. We use the first 26253 data samples in year 2015 through 2017 to train the SVR load predictor and use the remaining 8758 data samples in 2018 to test its performance. The resulting training data matrix $\boldsymbol{X}_{\text{train}}$ is of size 26253×163, training target value matrix $\boldsymbol{Y}_{\text{train}}$ is of size 26253×20, testing data matrix $\boldsymbol{X}_{\text{test}}$ is of size 8758×163, and the testing target value matrix $\boldsymbol{Y}_{\text{test}}$ is of size 8758×20. The dimensions of these matrices for other models can be similarly determined.