Machine Learning Methods for Attack Detection in the Smart Grid

Mete Ozay, *Member, IEEE*, Iñaki Esnaola, *Member, IEEE*, Fatos Tunay Yarman Vural, *Senior Member, IEEE*, Sanjeev R. Kulkarni, *Fellow, IEEE*, and H. Vincent Poor, *Fellow, IEEE*

***Abstract*— Attack detection problems in the smart grid are posed as statistical learning problems for different attack scenarios in which the measurements are observed in batch or online settings. In this approach, machine learning algorithms are used to classify measurements as being either secure or attacked. An attack detection framework is provided to exploit any available prior knowledge about the system and surmount constraints arising from the sparse structure of the problem in the proposed approach. Well-known batch and online learning algorithms (supervised and semisupervised) are employed with decision- and feature-level fusion to model the attack detection problem. The relationships between statistical and geometric properties of attack vectors employed in the attack scenarios and learning algorithms are analyzed to detect unobservable attacks using statistical learning methods. The proposed algorithms are examined on various IEEE test systems. Experimental analyses show that machine learning algorithms can detect attacks with performances higher than attack detection algorithms that employ state vector estimation methods in the proposed attack detection framework.**

***Index Terms*— Attack detection, classification, phase transition, smart grid security, sparse optimization.**

I. Introduction

MACHINE learning methods have been widely proposed in the smart grid literature for monitoring and control of power systems [1]–[4]. Rudin *et al.* [1] suggest an intelligent framework for the system design, in which machine learning algorithms are employed to predict the failures of the system components. Anderson *et al.* [2] employ machine learning algorithms for the energy management of loads and sources in smart grid networks. Malicious activity prediction and intrusion detection problems have been analyzed using machine learning techniques at the network layer of smart grid communication systems [3], [4].

In this paper, we focus on the false data injection attack detection problem in the smart grid at the physical layer. We use the distributed sparse attacks model proposed in [5], where the attacks are directed by injecting false data into the local measurements observed by either local network operators or smart phasor measurement units (PMUs) in a network with a hierarchical structure, i.e., the measurements are grouped into clusters. In addition, network operators who employ statistical learning algorithms for attack detection know the topology of the network, measurements observed in the clusters, and the measurement matrix [5].

In attack detection methods that employ state vector estimation (SVE), first, the state of the system is estimated from the observed measurements. Then the residual between the observed and the estimated measurements is computed. If the residual is greater than a given threshold, a data injection attack is declared [5]–[8]. However, an exact recovery of state vectors is a challenge for SVE-based methods in sparse networks [5], [9], [10], where the Jacobian measurement matrix is sparse. Sparse reconstruction methods can be employed to solve the problem, but the performance of this approach is limited by the sparsity of the state vectors [5], [11], [12]. In addition, if the false data injected vectors reside in the column space of the Jacobian measurement matrix and satisfy some sparsity conditions (e.g., the number of nonzero elements is at most κ^*, which is bounded by the size of the Jacobian matrix), then the false data injection attacks, called unobservable attacks, cannot be detected [7], [8].

The contributions of this paper are as follows.

1) We conduct a detailed analysis of the techniques proposed by Ozay *et al.* [13] who employed supervised learning algorithms to predict false data injection attacks. In addition, we discuss the validity of the fundamental assumptions of statistical learning theory in the smart grid. Then, we propose semi-supervised and online learning algorithms, and decision and feature level fusion algorithms in a generic attack construction framework, which can be employed in hierarchical and topological networks for different attack scenarios.
2) We analyze the geometric structure of the measurement space defined by measurement vectors, and the effect of false data injection attacks on the distance function of the vectors. This leads to algorithms for learning

Manuscript received February 9, 2014; revised October 30, 2014 and February 1, 2015; accepted February 4, 2015. Date of publication March 19, 2015; date of current version July 15, 2016. This research was supported in part by the U. S. National Science Foundation under Grant CMMI-1435778.

M. Ozay is with the School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: meteozay@gmail.com).

I. Esnaola is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA, and also with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield S10 2TN, U.K. (e-mail: esnaola@sheffield.ac.uk).

F. T. Yarman Vural is with the Department of Computer Engineering, Middle East Technical University, Ankara 06800, Turkey (e-mail: vural@ceng.metu.edu.tr).

S. R. Kulkarni and H. V. Poor are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: kulkarni@princeton.edu; poor@princeton.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2404803

the distance functions, detecting unobservable attacks, estimating the attack strategies, and predicting future attacks using a set of observations.

3) We empirically show that the statistical learning algorithms are capable of detecting both observable and unobservable attacks with performance better than attack detection algorithms that employ SVE methods. In addition, phase transitions can be observed in the performance of support vector machines (SVMs) at a value of κ^* [14].

In Section II, the attack detection problem is formulated as a statistical classification problem in a network according to the model proposed in [5]. In Section II, we establish the relationship between statistical learning methods and attack detection problems in the smart grid. Supervised, semi-supervised, and online learning algorithms, and decision and feature level fusion algorithms are used to solve the classification problem in Section III. In Section IV, our approach is numerically evaluated on IEEE test systems. Finally, the conclusion is drawn in Section V.

II. Problem Formulation

In this section, the attack detection problem is formalized as a machine learning problem.

A. False Data Injection Attacks

False data injection attacks are defined in the following model:

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^D$ contains the voltage phase angles at the buses, $\mathbf{z} \in \mathbb{R}^N$ is the vector of measurements, $\mathbf{H} \in \mathbb{R}^{N\times D}$ is the measurement Jacobian matrix, and $\mathbf{n} \in \mathbb{R}^N$ is the measurement noise, which is assumed to have independent components [7]. The attack detection problem is defined as that of deciding whether or not there is an attack on the measurements. If the noise is distributed normally with zero mean, then an SVE method can be employed by computing

$$\hat{\mathbf{x}} = (\mathbf{H}^T \Lambda \mathbf{H})^{-1} \mathbf{H}^T \Lambda \mathbf{z} \tag{2}$$

where Λ is a diagonal matrix whose diagonal elements are given by $\Lambda_{ii} = \nu_i^{-2}$ and ν_i^2 is the variance of n_i, $\forall i = 1, 2, \ldots, N$ [7], [13]. The goal of the attacker is to inject a false data vector $\mathbf{a} \in \mathbb{R}^N$ into the measurements without being detected by the operator. The resulting observation model is

$$\tilde{\mathbf{z}} = \mathbf{H}\mathbf{x} + \mathbf{a} + \mathbf{n}. \tag{3}$$

The false data injection vector, $\mathbf{a}$, is a nonzero vector, such that $\mathbf{a}_i \neq \mathbf{0}$, $\forall i \in \mathcal{A}$, where $\mathcal{A}$ is the set of indices of the measurement variables that will be attacked. The secure variables satisfy the constraint $\mathbf{a}_i = \mathbf{0}$, $\forall i \in \bar{\mathcal{A}}$, where $\bar{\mathcal{A}}$ is the set complement of $\mathcal{A}$ [13].

To detect an attack, the measurement residual [7], [13] is examined in ℓ_2-norm $\rho = \|\tilde{\mathbf{z}} - \mathbf{H}\hat{\mathbf{x}}\|_2^2$, where $\hat{\mathbf{x}} \in \mathbb{R}^D$ is the SVE. If $\rho > \tau$, where $\tau \in \mathbb{R}$ is an arbitrary threshold, which determines the tradeoff between the detection and the false alarm probabilities, then the network operator declares that the measurements are attacked.

One of the challenging problems of this approach is that the Jacobian measurement matrices of power systems in the smart grid are sparse under the DC power flow model [13], [15]. Therefore, the sparsity of the systems determines the performance of sparse SVE methods [11], [12]. In addition, unobservable attacks can be constructed even if the network operator can estimate the state vector correctly. For instance, if $\mathbf{a} = \mathbf{H}\mathbf{c}$, where $\mathbf{c} \in \mathbb{R}^D$ is an attack vector, then the attack is unobservable using the measurement residual ρ [7], [8]. In this paper, we show that statistical learning methods can be used to detect the unobservable attacks with performance higher than the attack detection algorithms that employ an SVE approach. Following the motivation mentioned above, a new approach is proposed using statistical learning methods.

B. Attack Detection Using Statistical Learning Methods

Given a set of samples $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^M$ and a set of labels $\mathcal{Y} = \{y_i\}_{i=1}^M$, where $(\mathbf{s}_i, y_i) \in \mathcal{S} \times \mathcal{Y}$ are independent and identically distributed (i.i.d.) with joint distribution P, the statistical learning problem can be defined as constructing a hypothesis function $f : \mathcal{S} \to \mathcal{Y}$, that captures the relationship between the samples and the labels [16]. Then, the attack detection problem is defined as a binary classification problem, where

$$y_i = \begin{cases} 1, & \text{if} \quad \mathbf{a}_i \neq \mathbf{0} \\ -1, & \text{if} \quad \mathbf{a}_i = \mathbf{0}. \end{cases} \tag{4}$$

In other words, $y_i = 1$, if the ith measurement is attacked, and $y_i = -1$ when there is no attack.

In this paper, the model proposed in [5] is employed for attack construction, where the measurements are observed in clusters in the network. Measurement matrices, and observation and attack vectors are partitioned into G blocks, denoted by $\mathcal{G}_g$ with $|\mathcal{G}_g| = N_g$ for $g = 1, 2, \ldots, G$. Therefore, the observation model is defined as

$$\begin{bmatrix} \tilde{\mathbf{z}}_1 \\ \vdots \\ \tilde{\mathbf{z}}_G \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_G \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_G \end{bmatrix} + \begin{bmatrix} \mathbf{n}_1 \\ \vdots \\ \mathbf{n}_G \end{bmatrix} \tag{5}$$

where $\tilde{\mathbf{z}}_g \in \mathbb{R}^{N_g}$ is the measurement observed in the gth cluster of nodes through measurement matrix $\mathbf{H_g} \in \mathbb{R}^{N_g \times D}$ and noise $\mathbf{n}_g \in \mathbb{R}^{N_g}$, and which is under attack $\mathbf{a}_g \in \mathbb{R}^{N_g}$ with $g = 1, 2, \ldots, G$ [5]. Within this framework, each observed measurement vector is considered as a sample, i.e., $\mathbf{s}_i \triangleq \tilde{\mathbf{z}}_g$, where $\tilde{\mathbf{z}}_g \in \mathbb{R}^{N_g}$.[1] Taking this into account, the measurements are classified in two groups, secure and attacked, by computing $f(\mathbf{s}_i), \forall i = 1, 2, \ldots, M$.

The crucial part of the traditional attack detection algorithm, which we call SVE, is the estimation of $\hat{\mathbf{x}}$. If the attack vectors, $\mathbf{a}$, are constructed in the column space of $\mathbf{H}$, then they are annihilated in the computation of the residual [7]. Therefore, SVE cannot detect the attacks and these attacks

[1] For simplicity of notation, we use i as the index of measurements $\tilde{\mathbf{z}}_i$, $\mathbf{z}_i$, and attack vectors $\mathbf{a}_i$, $\forall i = 1, 2, \ldots, M$.

are called unobservable. On the other hand, we observe that the distance between the attacked and the secure measurement vectors is defined by the attack vector in $\mathcal{S}$. If the attacks are unobservable, i.e., $\mathbf{a}_i = \mathbf{H}\mathbf{c}_i$ and $\mathbf{a}_j = \mathbf{H}\mathbf{c}_j$, where $\mathbf{c}_i \in \mathbb{R}^D$ and $\mathbf{c}_j \in \mathbb{R}^D$ are the attack vectors, then the distance between $\tilde{\mathbf{z}}_i = \mathbf{z}_i + \mathbf{a}_i$ and $\tilde{\mathbf{z}}_j = \mathbf{z}_j + \mathbf{a}_j$ is computed as

$$\|\tilde{\mathbf{z}}_i - \tilde{\mathbf{z}}_j\|_2 = \begin{cases} \|\mathbf{z}_i - \mathbf{z}_j\|_2 + \|\mathbf{a}_i - \mathbf{a}_j\|_2, & \text{if} \quad i, j \in \mathcal{A} \\ \|\mathbf{z}_i - \mathbf{z}_j\|_2 + \|\mathbf{a}_i\|_2, & \text{if} \quad i \in \mathcal{A},\ j \in \bar{\mathcal{A}} \\ \|\mathbf{z}_i - \mathbf{z}_j\|_2, & \text{if} \quad i, j \in \bar{\mathcal{A}} \end{cases} \tag{6}$$

where $\tilde{\mathbf{z}}_i \in \mathcal{S}$ and $\tilde{\mathbf{z}}_j \in \mathcal{S}$. In (6), we can extract information on the attack vectors by observing the measurements. Since the distances between secure and attacked measurements are discriminated by the attack vectors, the attacks can be recognized by the learning algorithms, which use the information of these distances, even if the attacks are unobservable.

Two main assumptions from statistical learning theory need to be considered to classify measurements, which satisfy (6).

1) We assume that $(\mathbf{s}_i, y_i) \in \mathcal{S} \times \mathcal{Y}$ are distributed according to a joint distribution P [17]. In a smart grid setting, this distribution assumption is satisfied for the attack models, in which the measurements $\tilde{\mathbf{z}}$ are functions of $\mathbf{a}$, and we can extract statistical information about both the attacked and the secure measurements from the observations.
2) We assume that $(\mathbf{s}_i, y_i)$, $\forall i$, are sampled from P, independently and identically. This assumption is also satisfied in the smart grid if the entries of $\mathbf{n}$ and $\mathbf{a}$ are i.i.d. random variables [16].

To explain the significance of the above assumptions in the smart grid, we consider the following example. Assume that measurements $1, 2 \in \mathcal{A}$ and $3, 4 \in \bar{\mathcal{A}}$, are given, such that $y_1, y_2 = 1$ and $y_3, y_4 = -1$. Furthermore, assume that $\mathbf{z}_1 = 3 \cdot \mathbf{I}$, $\mathbf{z}_2 = 5 \cdot \mathbf{I}$, $\mathbf{z}_3 = 2 \cdot \mathbf{I}$, and $\mathbf{z}_4 = 4 \cdot \mathbf{I}$, where $\mathbf{I} = (1, 1)^T$. If the attack vectors are identical but not independent, then the attack vectors can be constructed as $\mathbf{a}_1 = \mathbf{a}_2 = -1 \cdot \mathbf{I}$. As a result, we observe that $\tilde{\mathbf{z}}_1 = \tilde{\mathbf{z}}_3 = 2 \cdot \mathbf{I}$ and $\tilde{\mathbf{z}}_2 = \tilde{\mathbf{z}}_4 = 4 \cdot \mathbf{I}$. Therefore, our assumption about the existence of a joint distribution P is not satisfied and we cannot classify the measurements with the aforementioned approach.

III. Attack Detection Using Machine Learning Methods

In this section, the attack detection problem is modeled by statistical classification of measurements using machine learning methods.

A. Supervised Learning Methods

In the following, the classification function f is computed in a supervised learning framework by a network operator using a set of training data $\mathsf{Tr} = \{(\mathbf{s}_i, y_i)\}_{i=1}^{M^{\mathsf{Tr}}}$. The class label, y_i', of a new observation, $\mathbf{s}_i'$, is predicted using $y_i' = f(\mathbf{s}_i')$. We employ four learning algorithms for attack detection.

1) Perceptron: Given a sample $\mathbf{s}_i$, a perceptron predicts y_i using the classification function $f(\mathbf{s}_i) = \mathsf{sign}(\mathbf{w} \cdot \mathbf{s}_i)$, where $\mathbf{w} \in \mathbb{R}^{N_i}$ is a weight vector and $\mathsf{sign}(\mathbf{w} \cdot \mathbf{s}_i)$ is defined as [17]

$$\mathsf{sign}(\mathbf{w} \cdot \mathbf{s}_i) = \begin{cases} -1, & \text{if} \quad \mathbf{w} \cdot \mathbf{s}_i < 0 \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

in the training phase, the weights are adjusted at each iteration $t = 1, 2, \ldots, T$ of the algorithm for each training sample using

$$\mathbf{w}(t+1) := \mathbf{w}(t) + \Delta\mathbf{w} \tag{8}$$

where $\Delta\mathbf{w} = \gamma(y_i - f(\mathbf{s}_i))\mathbf{s}_i$ and γ is the learning rate. The algorithm is iterated until a stopping criterion, such as the number of algorithm steps, or an error threshold, is achieved. In the testing phase, the label of a new test sample is predicted by $f(\mathbf{s}'_i) = \mathsf{sign}(\mathbf{w}(T) \cdot \mathbf{s}'_i)$.

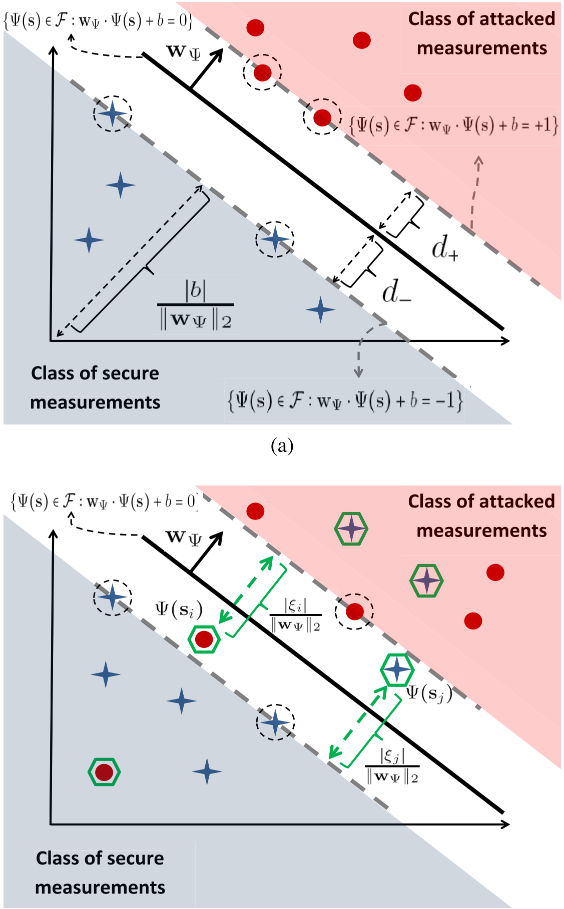
Despite its success in various machine learning applications, the convergence of the algorithm is assured only when the samples are linearly separable [17]. For that reason, the perceptron can be successfully used for the detection of the attacks only if the measurements can be separated by a hyperplane. In Section III.B and III.C, we give examples of classification algorithms that overcome this limitation by employing nonlinear classification rules or feature extraction methods.

2) k-Nearest Neighbor: This algorithm labels an unlabeled sample $\mathbf{s}_i'$ according to the labels of its k-nearest neighborhood in the feature space [17]. In particular, the observed measurements $\mathbf{s}_i \in \mathcal{S}$, $\forall i = 1, 2, \ldots, M$ are taken as feature vectors. The set of k-nearest neighbors (k-NNs) of $\mathbf{s}_i'$, $\aleph(\mathbf{s}_i') = \{\mathbf{s}_{i(1)}, \mathbf{s}_{i(2)}, \ldots, \mathbf{s}_{i(k)}\}$, is constructed by computing the Euclidean distances between the samples [18], where $i(1), i(2), \ldots, i(M)$ are defined as

$$\|\mathbf{s}'_i - \mathbf{s}_{i(1)}\|_2 \leq \|\mathbf{s}'_i - \mathbf{s}_{i(2)}\|_2 \leq \cdots \leq \|\mathbf{s}'_i - \mathbf{s}_{i(M)}\|_2. \tag{9}$$

Then, the most frequently observed class label is computed using majority voting among the class labels of the samples in the neighborhood, and assigned as the class label of $\mathbf{s}_i'$ [19]. One of the challenges of k-NN is the curse of dimensionality, which is the difficulty of the learning problem when the sample size is small compared with the dimension of the feature vector [17], [19], [20]. In attack detection, this problem can be handled using the following approaches.

1) Feature selection algorithms can be used to reduce the dimension of the feature vectors [19], [20]. Development of feature selection algorithms may be a promising direction for smart grid security, and is an interesting topic for future work.
2) Kernel machines, such as SVMs, can be used to map the feature vectors in $\mathcal{S}$ to Hilbert spaces, where the feature vectors are processed implicitly in the mappings and the computation of the learning models. We give the details of the kernel machines and SVMs in Section III-A3, III-B and III-C.
3) The samples can be processed in small sizes, e.g., by selecting a single measurement vector as a sample, which leads to one-dimensional samples. We employ this approach in Section IV. If the sample size is large,


{Ψ(s) ∈ 𝓕 : w_Ψ · Ψ(s) + b = 0}
w_Ψ
Class of attacked measurements
{Ψ(s) ∈ 𝓕 : w_Ψ · Ψ(s) + b = +1}
d_+
d_−
|b| / ‖w_Ψ‖_2
Class of secure measurements
{Ψ(s) ∈ 𝓕 : w_Ψ · Ψ(s) + b = −1}
{Ψ(s) ∈ 𝓕 : w_Ψ · Ψ(s) + b = 0}
w_Ψ
Class of attacked measurements
Ψ(s_i)
|ξ_i| / ‖w_Ψ‖_2
Ψ(s_j)
|ξ_j| / ‖w_Ψ‖_2
Class of secure measurements

(a)

(b)

Fig. 1. Classification using SVM. Positive and negative samples, which belong to the class of attacked and secure measurements, are depicted by disk and star markers, respectively. Support vectors and misclassified samples are depicted by dashed circles and hexagonal markers, respectively. (a) Attack detection using the linearly separable data set. (b) Attack detection using the linearly nonseparable data set.

distributed learning and optimization methods can be used [5], [15].

3) Support Vector Machines: We seek a hyperplane that linearly separates attacked and secure measurements into two half-spaces using hyperplanes in a D'-dimensional feature space, $\mathcal{F}$, which is constructed by a nonlinear mapping $\Psi : \mathcal{S} \to \mathcal{F}$ [13], [21]. A hyperplane is represented by a weight vector $\mathbf{w}_\Psi \in \mathbb{R}^{D'}$ and a bias variable $b \in \mathbb{R}$, which results in

$$\mathbf{w}_\Psi \cdot \Psi(\mathbf{s}) + b = 0 \tag{10}$$

where $\Psi(\mathbf{s})$ is the feature vector of the sample that lies on the hyperplane in $\mathcal{F}$, as shown in Fig. 1. We choose the hyperplane that is at the largest distance from the closest positive and negative samples. This constraint can be formulated as

$$y_i(\mathbf{w}_\Psi \cdot \Psi(\mathbf{s}) + b) - 1 \geq 0, \quad \forall i = 1, 2, \ldots, M^{\mathrm{Tr}}. \tag{11}$$

Since $d_+ = d_- = 1/\|\mathbf{w}_\Psi\|_2$, where d_+ and d_- are the shortest distances from the hyperplane to the closest positive and negative samples, respectively, a maximum margin hyperplane can be computed by minimizing $\|\mathbf{w}_\Psi\|_2$.

If the training examples in the transformed space are not linearly separable [Fig. 1(b)], then the optimization problem can be modified by introducing slack variables $\xi_i \geq 0$, $\forall i = 1, 2, \ldots, M^{\mathrm{Tr}}$, in (11), which yields

$$y_i(\mathbf{w}_\Psi \cdot \Psi(\mathbf{s}_i) + b) - 1 + \xi_i \geq 0 \quad \forall i = 1, 2, \ldots, M^{\mathrm{Tr}}. \tag{12}$$

The hyperplane $\mathbf{w}_\Psi$ is computed by solving the following optimization problem in primal or dual form [21]–[23]:

$$\begin{aligned} &\min \ \|\mathbf{w}_\Psi\|_2^2 + C \sum_{i=1}^{M^{\mathrm{Tr}}} \xi_i \\ &\text{s.t.} \ y_i(\mathbf{w}_\Psi \cdot \Psi(\mathbf{s}_i) + b) - 1 + \xi_i \geq 0 \\ &\qquad\qquad \xi_i \geq 0 \quad \forall i = 1, 2, \ldots, M^{\mathrm{Tr}} \end{aligned} \tag{13}$$

where C is a constant that penalizes (an upper bound on) the training error of the soft margin SVM.

4) Sparse Logistic Regression: In utilizing this approach for attack detection, we solve the classification problem using the alternating direction method of multipliers (ADMM) [24] considering the sparse SVE approach in [5]. Note that the hyperplanes defined in (10) can be computed by employing the generalized logistic regression models presented in [19], which provide the distributions

$$P(y_i|\mathbf{s}_i) = \frac{1}{1 + \exp(-y_i(\mathbf{w} \cdot \mathbf{s}_i + b))} \tag{14}$$

$$P(y_i|\Psi(\mathbf{s}_i)) = \frac{1}{1 + \exp(-y_i(\mathbf{w}_\Psi \cdot \Psi(\mathbf{s}_i) + b))} \tag{15}$$

in $\mathcal{S}$ and $\mathcal{F}$, respectively. For this purpose, we minimize the logistic loss functions

$$\mathcal{L}(\mathbf{s}_i, y_i) = \log\left(1 + \exp\left(-y_i(\mathbf{w} \cdot \mathbf{s}_i + b)\right)\right) \tag{16}$$

$$\mathcal{L}(\Psi(\mathbf{s}_i), y_i) = \log(1 + \exp(-y_i(\mathbf{w}_\Psi \cdot \Psi(\mathbf{s}_i) + b))). \tag{17}$$

Defining a feature matrix $\mathbf{S} = (\mathbf{s}_1^T, \mathbf{s}_2^T, \ldots, \mathbf{s}_{M_{tr}}^T)^T$ and a label vector $\mathbf{Y} = (y_1, y_2, \ldots, y_{M_{tr}})^T$, the ADMM optimization problem [24] is constructed as

$$\begin{aligned} &\min \ \ \mathcal{L}(\mathbf{S}, \mathbf{Y}) + \mu(\mathbf{r}) \\ &\text{s.t.} \ \ \ \mathbf{w} - \mathbf{r} = \mathbf{0} \end{aligned} \tag{18}$$

where $\mathbf{w}$ is a weight vector, $\mathbf{r}$ is a vector of optimization variables, $\mu(\mathbf{r}) = \lambda\|\mathbf{r}\|_1$ is a regularization function, and λ is a regularization parameter, which is introduced to control the sparsity of the solution [24].

B. Semisupervised Learning Methods

In semisupervised learning methods, the information obtained from the unlabeled test samples is used during the computation of the learning models [25].

In this section, a semisupervised SVM (S3VM) algorithm, called S3VM [26], [27], is employed to establish the analytical relationship between supervised and semisupervised learning algorithms. In this setting, the unlabeled samples are incorporated into cost function of the optimization problem (13) as

$$\min \ \|\mathbf{w}\|_2^2 + C_1 \sum_{i=1}^{M^{\mathrm{Tr}}} L^{\mathrm{Tr}}(\mathbf{s}_i, y_i) + C_2 \sum_{i=1}^{M^{\mathrm{Te}}} L^{\mathrm{Te}}(\mathbf{s}'_i) \tag{19}$$

where C_1 and C_2 are confidence parameters, and $L^{\mathrm{Tr}}(\mathbf{s}_i, y_i) = \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{s}_i + b))$ and $L^{\mathrm{Te}}(\mathbf{s}'_i) = \max(0, 1 - \|\mathbf{s}'_i\|_1)$ are the loss functions of the training and test samples, respectively.

The main assumption of the S3VM is that the samples in the same cluster have the same labels and the number of subclusters is not large [27]. In other words, attacked and secure measurement vectors should be clustered in distinct regions in the feature spaces. Moreover, the difference between the number of attacked and the secure measurements should not be large in order to avoid the formation of subclusters.

This requirement can be validated by analyzing the feature space. Following (6), if $\|\mathbf{z}_i - \mathbf{z}_j\|_2 + \|\mathbf{a}_i - \mathbf{a}_j\|_2 \leq \|\mathbf{a}_i\|_2 + \|\mathbf{a}_j\|_2$, and $\|\mathbf{z}_k - \mathbf{z}_l\|_2 \leq \|\mathbf{a}_i\|_2 + \|\mathbf{a}_j\|_2$, $\forall i, j \in \mathcal{A}$ and $\forall k, l \in \bar{\mathcal{A}}$, then the samples belonging to different classes are well separated in different classes. Moreover, this requirement is satisfied in (19) by adjusting C_2 [27]. A survey of the methods that are used to provide optimal C_2 and solve (19) is given in [27].

C. Decision- and Feature-Level Fusion Methods

One of the challenges of statistical learning theory is to find a classification rule that performs better than a set of rules of individual classifiers, or to find a feature set that represents the samples better than a set of individual features. One approach to solve this problem is to combine a collection of classifiers or a set of features to boost the performance of the individual classifiers. The former approach is called decision-level fusion or ensemble learning, and the latter approach is called feature-level fusion. In this section, we consider Adaboost [28] and multiple kernel learning (MKL) [29] for ensemble learning and feature-level fusion.

1) Ensemble Learning for Decision-Level Fusion: Various methods, such as bagging, boosting, and stacking, have been developed to combine classifiers in ensemble learning situations [17], [30]. In the following, Adaboost is explained as an ensemble learning approach, in which a collection of weak classifiers are generated and combined using a combination rule to construct a stronger classifier, which performs better than the weak classifiers [17], [28], [31].

At each iteration $t = 1, 2, \ldots, T$ of the algorithm, a decision or hypothesis $f_t(\cdot)$ of the weak classifier is computed with respect to the distribution on the training samples $D_t(\cdot)$ at t by minimizing the weighted error $\epsilon_t = \sum_{i=1}^{M^{\mathrm{Tr}}} D_t(i) I(f_t(\mathbf{s}_i) \neq y_i)$, where $I(\cdot)$ is the indicator function. The distribution is initialized uniformly $D_1(i) = 1/M^{\mathrm{Tr}}$ at $t = 1$, and is updated by a parameter $\alpha_t = 1/2 \log(1 - \epsilon_t/\epsilon_t)$ as follows [31]:

$$D_{t+1}(i) = \frac{D_t(i) \exp^{-\alpha_t y_i f_t(\mathbf{s}_i)}}{Z_t} \tag{20}$$

where Z_t is a normalization parameter, called the partition function. At the output of the algorithm, a strong classifier $H(\cdot)$ is constructed for a sample $\mathbf{s}'$ using $H(\mathbf{s}') = \mathsf{sign}(\sum_{t=1}^{T} \alpha_t f_t(\mathbf{s}'))$.

2) Multiple Kernel Learning for Feature-Level Fusion: Feature-level fusion methods combine the feature spaces instead of the decisions of the classifiers. One of the feature-level fusion methods is MKL, in which different feature mappings are represented by kernels that are combined to produce a new kernel, which represents the samples better than the other kernels [29]. Therefore, MKL provides an approach to solve the feature mapping selection problem of SVM. To see this relationship, we first give the dual form of (13)

$$\begin{aligned} \max \quad & \sum_{i=1}^{M^{\mathrm{Tr}}} \beta_i - \frac{1}{2} \sum_{i=1}^{M^{\mathrm{Tr}}} \sum_{j=1}^{M^{\mathrm{Tr}}} \beta_i \beta_j y_i y_j k(\mathbf{s}_i, \mathbf{s}_j) \\ \text{subject to} & \sum_{i=1}^{M^{\mathrm{Tr}}} \beta_i y_i = 0, \quad 0 \leq \beta_i \leq C, \quad \forall i = 1, 2, \ldots, M^{\mathrm{Tr}} \end{aligned} \tag{21}$$

where β_i is the dual variable and $k(\mathbf{s}_i, \mathbf{s}_j) = \Psi(\mathbf{s}_i) \cdot \Psi(\mathbf{s}_j)$ is the kernel function. Therefore, (21) is a single kernel learning algorithm, which employs a single kernel matrix $K \in \mathbb{R}^{M^{\mathrm{Tr}} \times M^{\mathrm{Tr}}}$ with elements $K(i, j) = k(\mathbf{s}_i, \mathbf{s}_j)$. If we define the weighted combination of U kernels as $K = \sum_{u=1}^{U} d_u K_u$, where $d_u \geq 0$ are the normalized weights, such that $\sum_{u=1}^{U} d_u = 1$, then we obtain the following optimization problem of the MKL [32]:

$$\begin{aligned} \max \quad & \sum_{i=1}^{M^{\mathrm{Tr}}} \beta_i - \frac{1}{2} \sum_{i=1}^{M^{\mathrm{Tr}}} \sum_{j=1}^{M^{\mathrm{Tr}}} \beta_i \beta_j y_i y_j \sum_{u=1}^{U} d_u K_u(\mathbf{s}_i, \mathbf{s}_j) \\ \text{subject to} & \sum_{i=1}^{M^{\mathrm{Tr}}} \beta_i y_i = 0, \quad 0 \leq \beta_i \leq C, \quad \forall i = 1, 2, \ldots, M^{\mathrm{Tr}}. \end{aligned} \tag{22}$$

In (22), the kernels with $d_u = 0$ are eliminated, and, therefore, MKL can be considered as a kernel selection method. In the experiments, SVM algorithms are implemented with different kernels and these kernels are combined under MKL.

D. Online Learning Methods for Real-Time Attack Detection

In the smart grid, the measurements are observed in real time, where the samples are collected sequentially in time. In this scenario, we relax the distribution assumption in Section II-B, since the samples are observed in an arbitrary sequence [33]. Moreover, smart PMUs, which employ learning algorithms, may be required to detect the attacks when the measurements are observed without processing the whole set of training samples. To solve these challenging problems, we may use online versions of the learning algorithms given in Section III-A, III-B and III-C.

In a general online learning setting, a sequence of training samples (or a single sample) is given to the learning algorithm at each observation or algorithm processing time. Then, the algorithm computes the learning model using only the given samples and predicts the labels. The learning model is updated with respect to the error of the algorithm, which is computed using a loss function on the given samples. Therefore, the perceptron and Adaboost are convenient for online learning in this setting. For instance, an online perceptron (OP) is implemented by predicting the label y_i of a single sample $\mathbf{s}_i$ at each time t, and updating the weight vector $\mathbf{w}$ using $\Delta\mathbf{w}$ for the misclassified samples with $y_i \neq \mathsf{sign}(f(\mathbf{s}_i))$ [34]. This simple approach is applied for the development of online MKL [34] and regression algorithms [35].

TABLE I
DEFINITIONS OF PERFORMANCE MEASURES

	Attacked	Secure
Classified as Attacked	*tp*	*fp*
Classified as Secure	*fn*	*tn*

E. Performance Analysis

In smart grid networks, the major concern is not just the detection of attacked variables but also that of the secure variables with high performance. In other words, we require the algorithms to predict the samples with high precision and recall performance in order to avoid false alarms. Therefore, we measure the true positives (tp), the true negatives (tn), the false positives (fp), and the false negatives (fn), which are defined in Table I.

In addition, the learning abilities and memorization properties of the algorithms are measured by Precision (Prec), Recall (Rec), and Accuracy (Acc) values, which are defined as [13]

$$\text{Prec} = \frac{\text{tp}}{\text{tp}+\text{fp}}, \quad \text{Rec} = \frac{\text{tp}}{\text{tp}+\text{fn}}, \quad \text{Acc} = \frac{\text{tp}+\text{tn}}{\text{tp}+\text{tn}+\text{fp}+\text{fn}}. \tag{23}$$

Precision values give information about the prediction performance of the algorithms. On the other hand, Recall values measure the degree of attack retrieval. Finally, the total classification performance of the algorithms is measured by Accuracy. For instance, if Prec $= 1$, then none of the secure measurements are misclassified as attacked. If Rec $= 1$, then none of the attacked measurements are misclassified as secure. If Acc $= 1$, then each measurement classified as attacked is actually exposed to an attack, and each measurement classified as secure is actually a secure measurement.

IV. EXPERIMENTS

The classification algorithms are analyzed in the IEEE 9-bus, 57-bus, and 118-bus test systems in the experiments. The measurement matrices $\mathbf{H}$ of the systems are obtained from the MATPOWER toolbox [36]. The operating points of the test systems provided in the MATPOWER case files are used in generating $\mathbf{z}$. Training and test data are generated by repeating this process 50 times for each simulated point and data set. In the experiments, we assume that the attacker has access to κ measurements, which are randomly chosen to generate a κ-sparse attack vector $\mathbf{a}$ with Gaussian distributed nonzero elements with the same mean and variance as the entries of $\mathbf{z}$ [5], [13], [15]. We assume that concept drift [37] and data set shift [38] do not occur. Therefore, we use $G = N$ in the simulations following the results in [5].

We analyze the behavior of each algorithm on each system for both observable and unobservable attacks by generating attack vectors with different values of $\kappa/N \in [0, 1]$. More precisely, if $\kappa \geq N - D + 1$, then attack vectors that are not observable by SVE, i.e., unobservable attacks, are generated [5]. Otherwise, the generated attacks are observable.

The LIBSVM [39] implementation is used for the SVM, and the ADMM [24] implementation is used for sparse logistic regression (SLR). k values of the k-NN algorithm are optimized by searching $k \in \{1, 2, \ldots, \sqrt{M^{\text{Tr}}}\}$ using leave-one-out cross validation, where M^{Tr} is the number of training samples. Both the linear and the Gaussian kernels are used for the implementation of SVM. A grid search method [39]–[41] is employed to search the parameters of the SVM in an interval $\mathcal{I} = [\mathcal{I}_{\min}, \mathcal{I}_{\max}]$, where $\mathcal{I}_{\min}$ and $\mathcal{I}_{\max}$ are user-defined values. In order to follow linear paths in the search space, log values of parameters are considered in the grid search method [41]. Keerthi and Lin [41] analyzed the asymptotic properties of the SVM for $\mathcal{I} = [0, \infty)$. In the experiments, $\mathcal{I}_{\min} = -10$ is chosen to compute a lower limit 2^{-10} of the parameter values following the theoretical results given in [39] and [41]. Since the classification performance of the SVM does not change for parameter values that are greater than a threshold [41], we used $\mathcal{I}_{\max} = 10$ as employed in the experimental analyses in [41]. Therefore, the kernel width parameter σ of a Gaussian kernel is searched for in the interval $\log(\sigma) \in [-10, 10]$, and the cost penalization parameter C of the SVM is searched for in the interval $\log(C) \in [-10, 10]$. The regularization parameter of the SLR is computed as

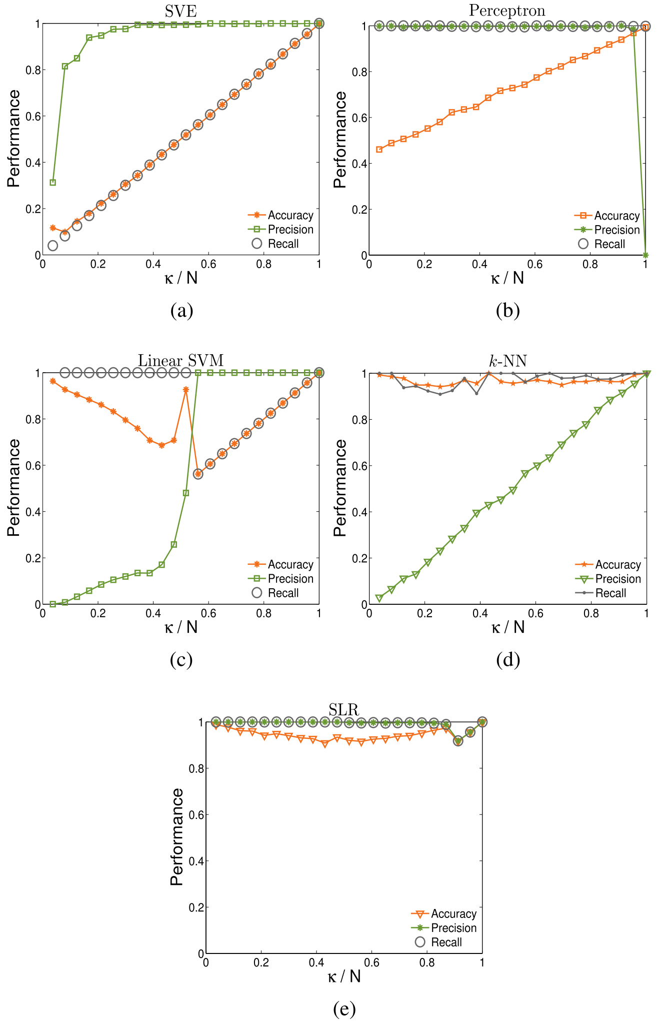
$$\lambda = \Omega \lambda_{\max} \tag{24}$$

where $\lambda_{\max} = \|\mathbf{H}\tilde{\mathbf{z}}\|_{\infty}$ determines the critical value of λ above which the solution of the ADMM problem is $\mathbf{0}$ and Ω is searched for in the interval $\Omega \in [10^{-3}, 1]$ [5], [24], [42]. An optimal $\hat{\lambda}$ is computed by analyzing the solution (or regularization) path of the LASSO-type optimization algorithms using a given training data set. As the sparsity of the systems that generate data sets increases, lower values are calculated for Ω [5], [24], [42]. The absolute and relative tolerances, which determine the values of upper bounds on the Euclidean norms of primal and dual residuals, are chosen as 10^{-4} and 10^{-2}, respectively [24]. The penalty parameter is initially selected as 1 and dynamically updated at each iteration of the algorithm [24]. The maximum number of iterations is chosen as 10^4 [5], [24].

In the experiments, we observe that the selection of tolerance parameters does not affect the convergence rates if their relative values do not change. In addition, selection of the initial value of the penalty parameter also does not affect the convergence rate if the relative values of tolerance parameters are fixed [24]. For instance, similar convergence rates are observed when we chose 10^{-4} and 10^{-2}, or 10^{-6} and 10^{-4}, as tolerance parameters. $\|\tilde{\mathbf{z}} - \mathbf{H}\hat{\mathbf{x}}_b\| \leq \tau$ is computed in order to decide whether there is an attack using the SVE and assuming a chi-square test with 95% confidence in the computation of τ [6], [13].

A. Results for Supervised Learning Algorithms

The performance of different algorithms is compared for the IEEE 57-bus system in Fig. 2. Accuracy values of the SVE and perceptron increase as κ/N increases in Fig. 2(a) and (b). In addition, Recall values of the SVE increase linearly as κ/N increases. Precision values of the



(a) (b) (c) (d) (e)

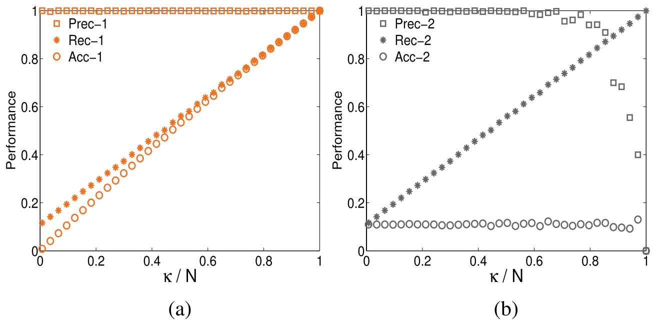
Fig. 2. Results for the IEEE 57-bus system. Accuracy values of the SVE and perceptron increase, while Precision values of the k-NN and SLR increase as κ/N increases. Both Accuracy and Precision values of the SVM increase and phase transitions occur. (a) SVE. (b) Perceptron. (c) SVM with linear kernel. (d) k-NN. (e) SLR.

perceptron are high and do not decrease, and Accuracy and Recall values increase, since fn values decrease and tn values increase. In Fig. 2(c), a phase transition around $\kappa^* = N - D + 1$, is observed for the performance of the SVM. Since the distance between measurement vectors of attacked and secure variables increases as κ/N increases following (6), we observe that the Accuracy, Precision, and Recall values of the k-NN increase in Fig. 2(d). Accuracy and Recall values of the k-NN and SLR are >0.9 and do not change as κ/N increases in Fig. 2(e).

The class-based performance values of the algorithms are measured using classwise performance indices, where Class-1 and Class-2 denote the class of attacked and secure variables, respectively. The classwise performance indices are defined as follows:

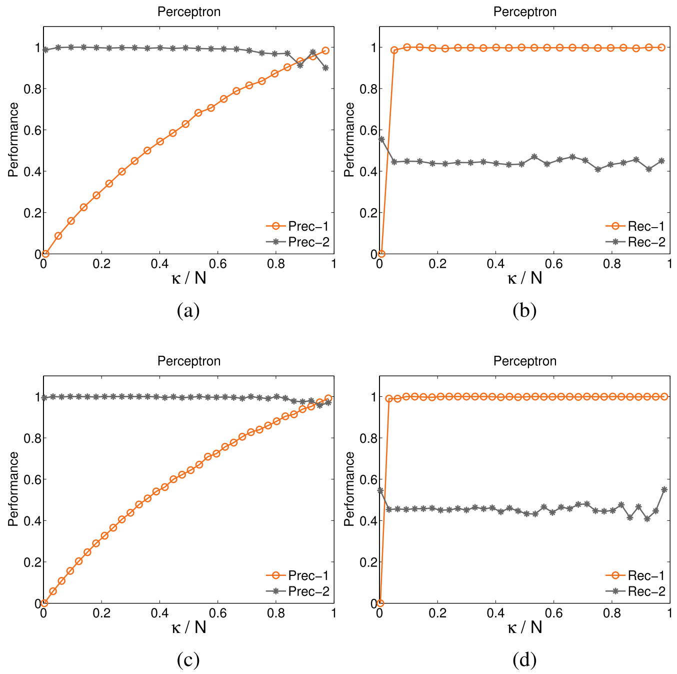
$$\text{Class}-1\text{: Prec}-1 = \frac{\text{tp}}{\text{tp}+\text{fp}}, \quad \text{Rec}-1 = \frac{\text{tp}}{\text{tp}+\text{fn}} \tag{25}$$

$$\text{Class}-2\text{: Prec}-2 = \frac{\text{tn}}{\text{tn}+\text{fn}}, \quad \text{Rec}-2 = \frac{\text{tn}}{\text{fp}+\text{tn}}. \tag{26}$$



(a) (b)

Fig. 3. Experiments using the SVE for the IEEE 57-bus test system. Note that fp values increase as κ/N increases. (a) Performance values for Class-1. (b) Performance values for Class-2.

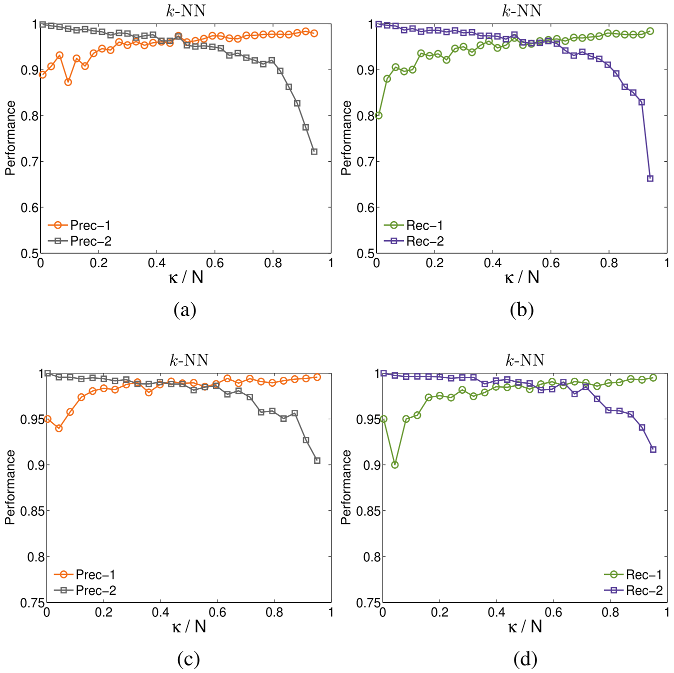


(a) (b) (c) (d)

Fig. 4. Performance analysis of the perceptron. (a) Results for the IEEE 57-bus. (b) Results for the IEEE 57-bus. (c) Results for the IEEE 118-bus. (d) Results for the IEEE 118-bus.

In Fig. 3(a), we observe that the Precision, Recall, and Accuracy values of the SVE increase as κ/N increases for Class-1. Note that the first value of Acc-1 is observed at 0.008. In Fig. 3(b), Precision values for Class-2 decrease with the percentage of attacked variables, i.e., the number of secure variables that are incorrectly classified by the SVE increases as the number of attacked variables increases. Although the SVE may correctly detect the attacked variables as κ/N increases, the secure variables are incorrectly labeled as attacked variables, and therefore, the SVE gives more false alarms than the other algorithms.

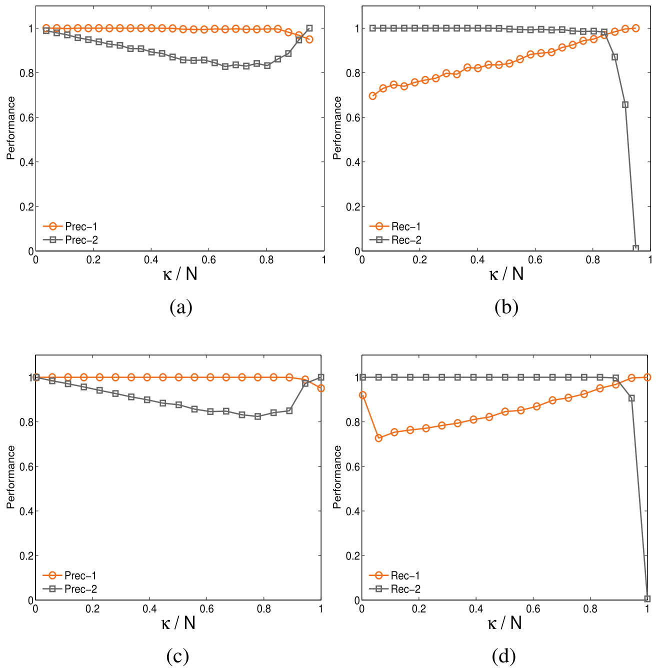
Performance values for the perceptron are given in Fig. 4. We observe that Precision values for Class-1 increase and Recall values do not change drastically for both of the classes as κ/N increases. Moreover, we do not observe any performance increase for the Recall values of the secure class in the perceptron.



(a) (b)

(c) (d)

Fig. 5. Since the k-NN is sensitive to class balance and sparsity of the data, the performance values for Class-1 increase and the values for Class-2 decrease as κ/N increases. Note that the performance curves intersect at the critical values κ^*. (a) Precision values for the IEEE 57-bus. (b) Recall values for the IEEE 57-bus. (c) Precision values for the IEEE 118-bus. (d) Recall values for the IEEE 118-bus.



(a) (b)

(c) (d)

Fig. 6. Experiments using the SLR. Note that the SLR can handle the variety in the sparsity of the data as κ/N changes. (a) Results for the IEEE 57-bus. (b) Results for the IEEE 57-bus. (c) Results for the IEEE 118-bus. (d) Results for the IEEE 118-bus.

In Fig. 5, the results for k-NN are shown. We observe that performance values for Class-1 increase and the values for Class-2 decrease as κ/N increases since k-NN is sensitive to class-balance and sparsity of the data [43]. In addition, classification hypotheses are computed by forming neighborhoods in Euclidean spaces, and the ℓ_2 norm of vectors of attacked measurements increases as κ/N increases in (6); therefore, decision boundaries of the hypotheses are biased toward Class-1.

Fig. 6 shows the results for the SLR, where the performance values for Class-2 (secure variables) increase as the system size increases. Moreover, we observe that the performance values for Class-2 do not decrease rapidly as κ/N increases, compared with the other supervised algorithms. In addition, the performance values for Class-1 are higher than the values of the other algorithms, especially for lower κ/N values. The reason is that the SLR can handle the variety in the sparsity of the data as κ/N changes. This task is accomplished by controlling and learning the sparsity of the solution in (18) using the training data in order to learn the sparse structure of the measurements defined in the observation model (5).

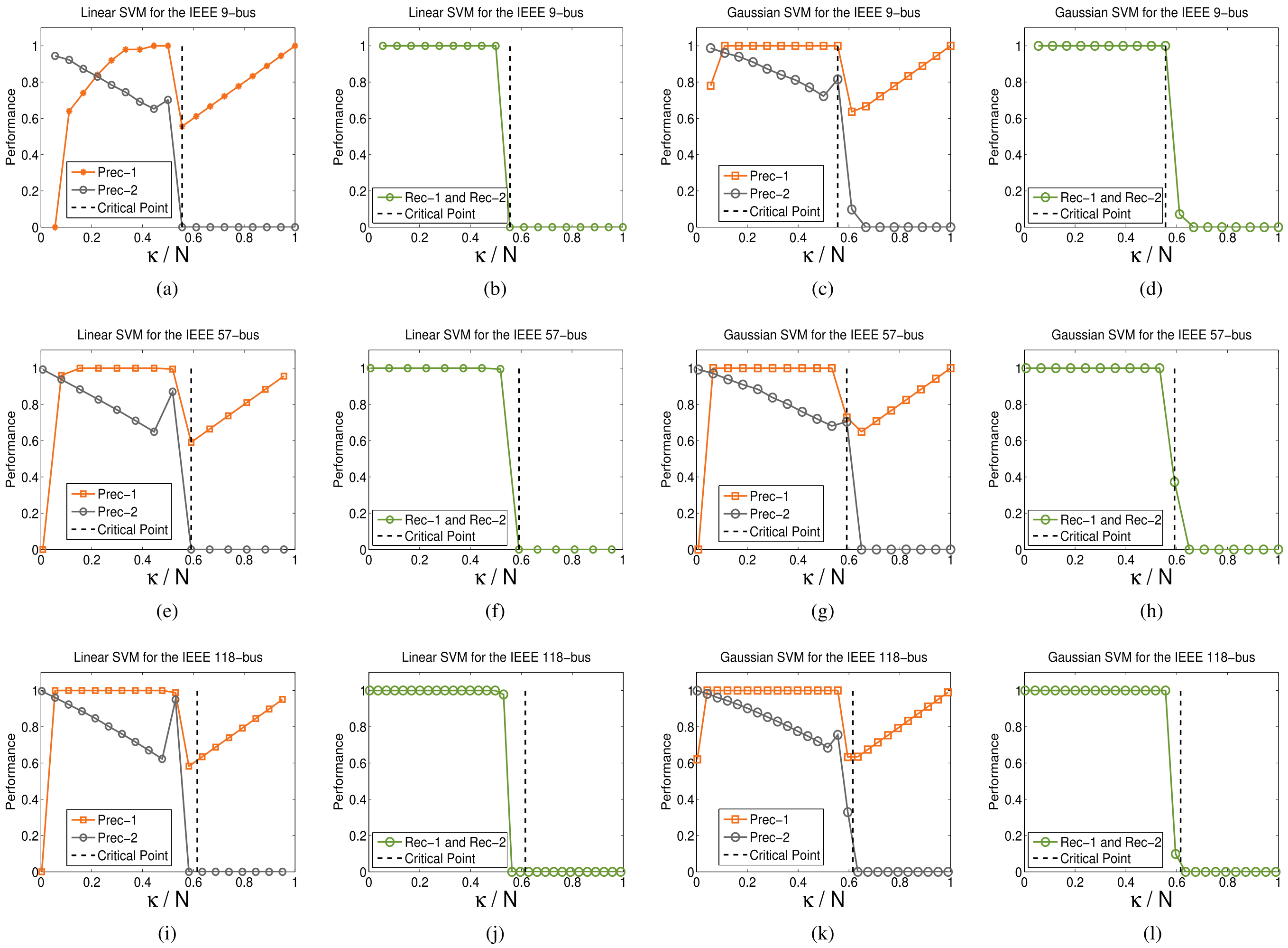
The results of the experiments for the SVM are shown in Fig. 7, where a phase transition for the performance values is observed. It is worth noting that the values of κ at which the phase transition occurs correspond to the minimum number of measurement variables, κ^*, that the attacker needs to compromise in order to construct unobservable attacks [7]. κ^* is depicted as a vertical dotted line in Fig. 7. For instance, $\kappa^* = 10$ and $\kappa^*/N = 0.56$ for the IEEE 9-bus test system. The transitions are observed before the critical points when the linear kernel SVM is employed in the experiments for the IEEE 57-bus and 118-bus systems. In addition, the phase transitions of performance values occur at the critical points when Gaussian kernels are used.

B. Results for Semisupervised Learning Algorithms

We use the S3VM with default parameters as suggested in [44]. The results of the S3VM are shown in Fig. 8. We do not observe sharp phase transitions in the S3VM unlike the supervised SVM, since the information obtained from unlabeled data contributes to the performance values in the computation of the learning models. For instance, Precision values of Class-2 decrease sharply near the critical point for the supervised SVM in Fig. 7. However, the S3VM employs the unlabeled samples during the computation of the learning model in (19), and partially solves this problem.

C. Results for Decision- and Feature-Level Fusion Algorithms

In this section, we analyze Adaboost and MKL. Decision stumps are used as weak classifiers in Adaboost [31]. Each decision stump is a single-level two-leaf binary decision tree that is used to construct a set of dichotomies consisting of binary labelings of samples [31]. The number of weak classifiers is selected using leave-one-out cross validation in the training set. We use MKL with a linear and a Gaussian kernel with the default parameters suggested in the Simple MKL implementation [32]. The results given in Fig. 9 show that Recall values of MKL for Class-1 are less than the values



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Fig. 7. Experiments using the SVM with linear and Gaussian kernels. Phase transitions of performance values occur at the critical point κ^*. See the text for more detailed explanation. (a) Linear SVM. (b) Linear SVM. (c) Gaussian SVM. (d) Gaussian SVM. (e) Linear SVM. (f) Linear SVM. (g) Gaussian SVM. (h) Gaussian SVM. (i) Linear SVM. (j) Linear SVM. (k) Gaussian SVM. (l) Gaussian SVM.

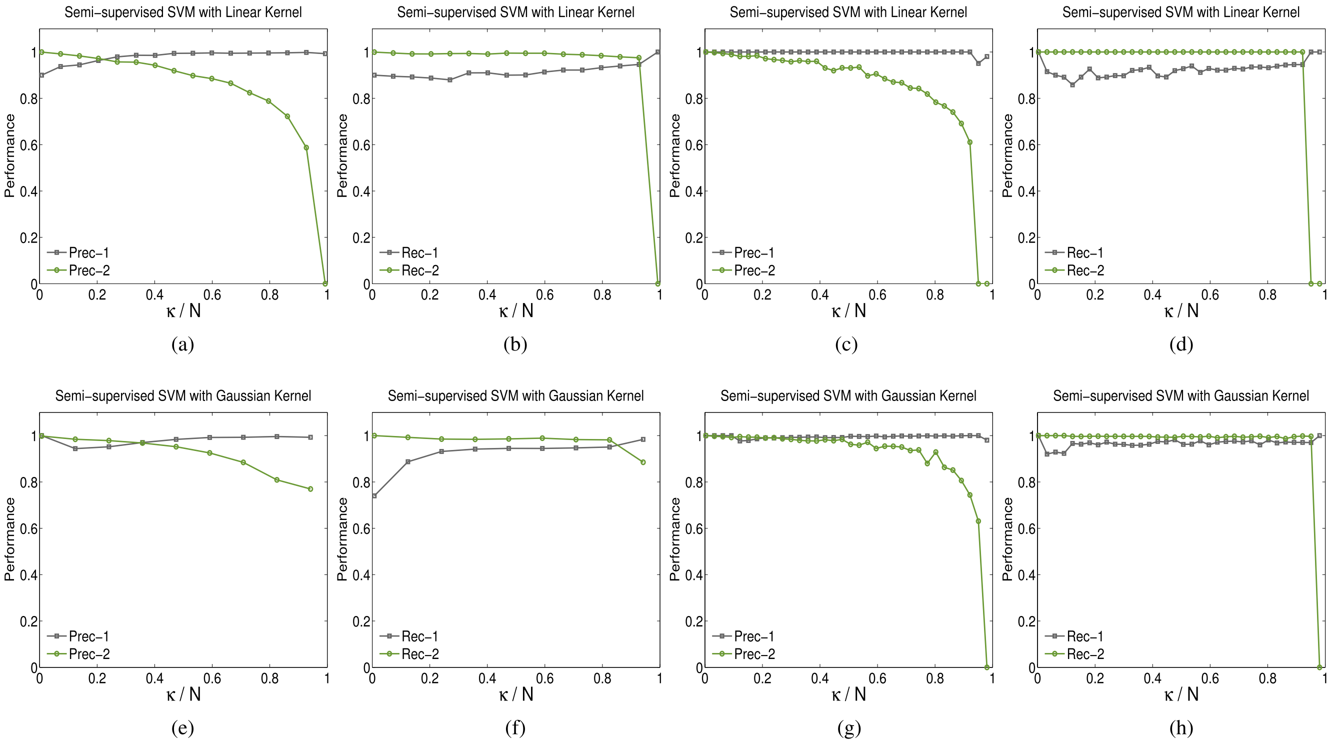
of Adaboost. In addition, Precision values of MKL decrease faster than the values of Adaboost as κ/N increases for Class-2. Therefore, the fn values of MKL are greater than the values of Adaboost, or in other words, the number of attacked measurements misclassified as secure by MKL is greater than that of Adaboost. This phenomenon is observed in the results for S3VM and supervised SVM given in Section III. However, there are no phase transitions of the performance values of MKL compared with the supervised SVM.

D. Results for Online Learning Algorithms

We consider four online learning algorithms, namely, OP, OP with weighted models (OPWM), Online SVM, and Online SLR. Note that these algorithms are the online versions of the batch learning algorithms given in Section III-A and developed considering the online algorithm design approach given in Section III-D. The details of the implementations of the OP, OPWM, Online SVM, and SLR are given in [34], [35], and [45].

When the OP is used, only the model $\mathbf{w}(t)$ computed using the last observed measurement at time t is considered for the classification of the test samples. On the other hand, we consider an average of the models $\mathbf{w}_{\text{ave}}(t) = 1/T \sum_{t=1}^{T} \mathbf{w}(t)$ that is computed by minimizing margin errors in the OPWM. Results are given for the OP in Fig. 10. In the weighted models, we observe phase transitions of the performance values for Class-2 in Fig. 10(e)–(h). However, the phase transitions occur before the critical values, and the values of the phase transition points decrease as the system size increases. In addition, we do not observe sharp phase transitions in the OP.

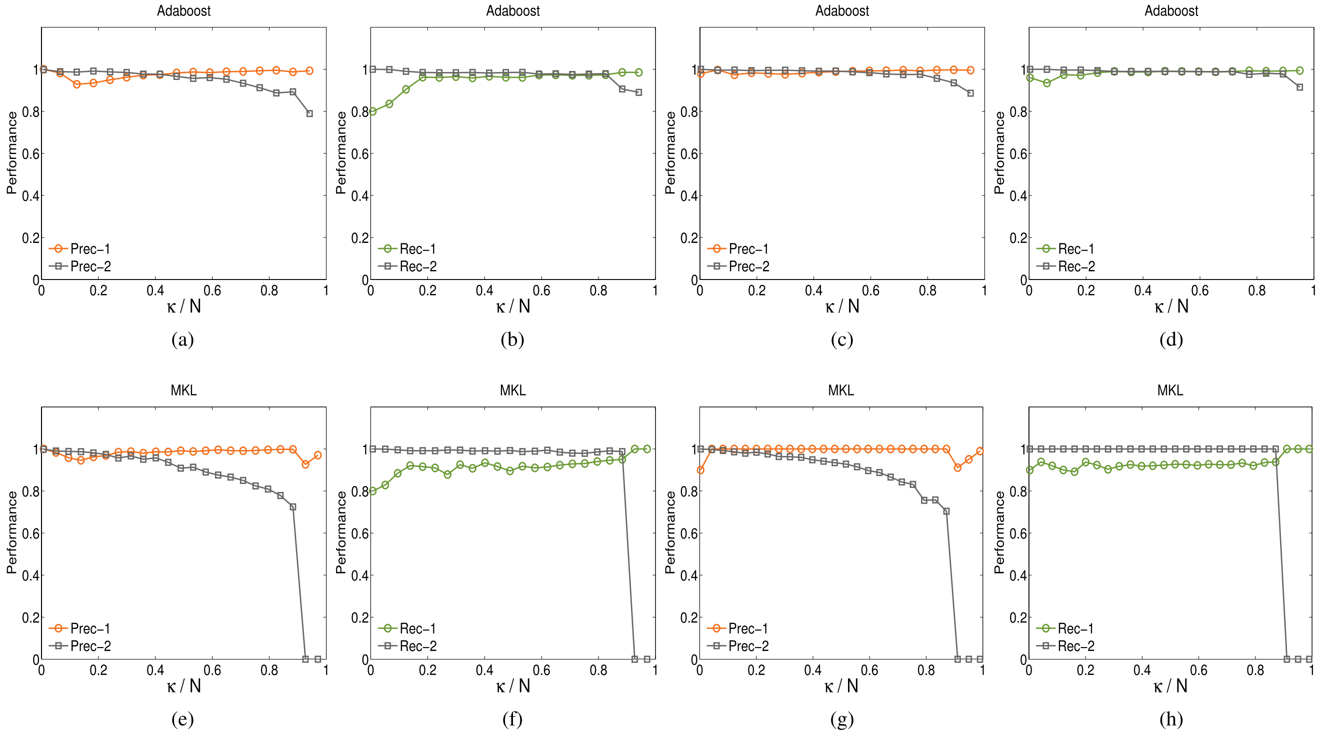
In the OP, if the label of a measurement $\mathbf{s}$ is not correctly labeled, then the measurement vector is added to a set of supporting measurements $\mathbb{S}$ that are used to update the hypotheses in the training process. However, the hypotheses are updated in the OPWM if a measurement $\mathbf{s}'$ is not correctly labeled, and the vectors of $\mathbf{s}'$ and $\mathbf{s} \in \mathbb{S}$ are linearly independent. Since the smallest number of linearly dependent measurements increases as κ/N increases [5], [46], the size of $\mathcal{S}$ decreases and the bias is decreased toward Class-1.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 8. Sharp phase transitions are not observed in the S3VM unlike the supervised SVM, since the information obtained from unlabeled data contributes to the performance values in the computation of the learning models. (a) Results for the IEEE 57-bus. (b) Results for the IEEE 57-bus. (c) Results for the IEEE 118-bus. (d) Results for the IEEE 118-bus. (e) Results for the IEEE 57-bus. (f) Results for the IEEE 57-bus. (g) Results for the IEEE 118-bus. (h) Results for the IEEE 118-bus.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 9. Experiments using Adaboost and MKL. Note that the fn values of MKL are greater than the values of Adaboost, and there are no phase transitions of the performance values of MKL compared with the supervised SVM. (a) Adaboost for the IEEE 57-bus. (b) Adaboost for the IEEE 57-bus. (c) Adaboost for the IEEE 118-bus. (d) Adaboost for the IEEE 118-bus. (e) MKL for the IEEE 57-bus. (f) MKL for the IEEE 57-bus. (g) MKL for the IEEE 118-bus. (h) MKL for the IEEE 118-bus.

Therefore, fn values decrease and fp values increase [47]. As a result, we observe that the Recall values of the OP are less than that of the OPWM for Class-1. The results of the Online SVM and Online SLR are provided in Fig. 10 for different IEEE test systems. We observe phase transitions of performance values in the Online SVM similar to the batch supervised SVM.

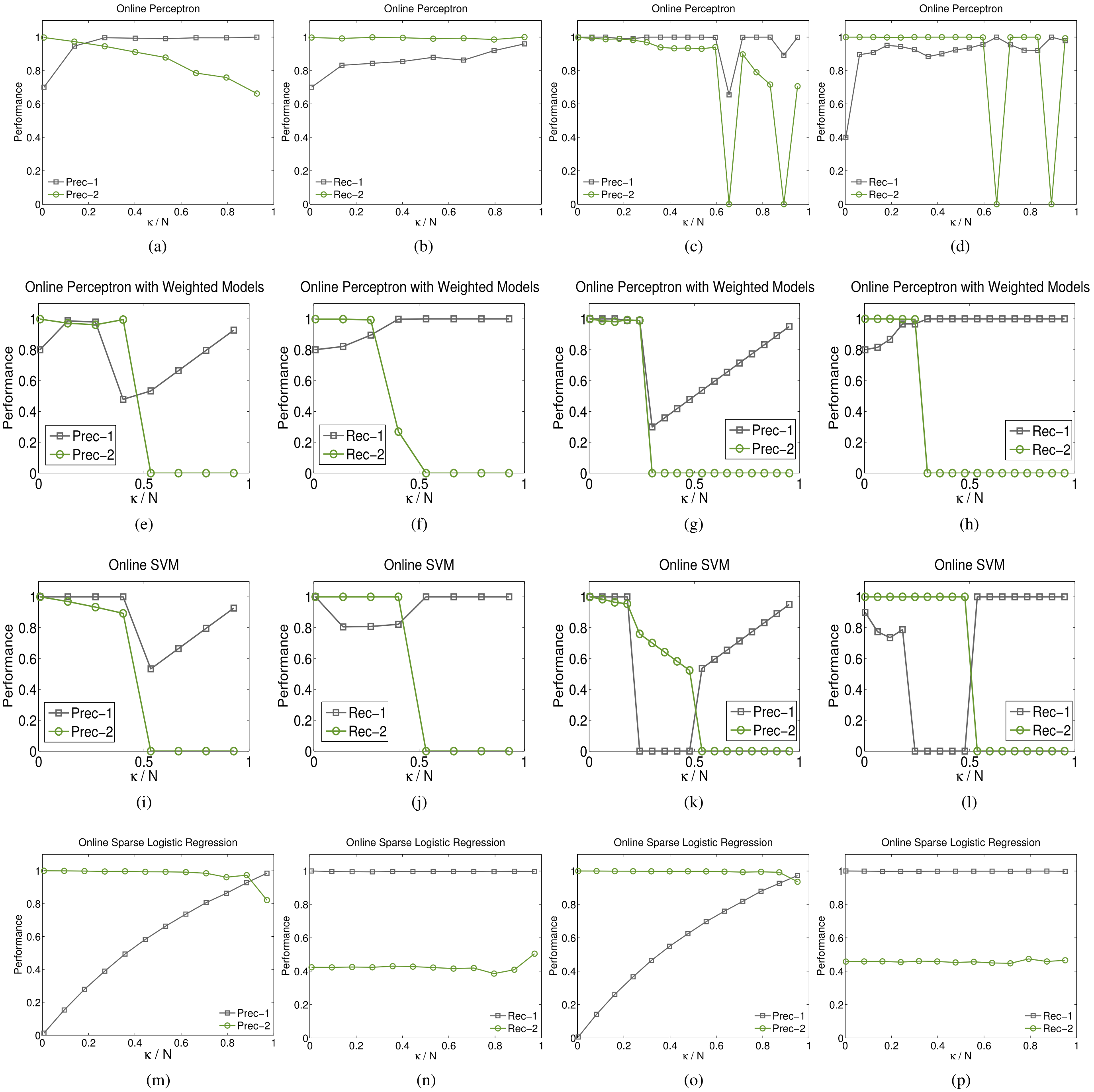
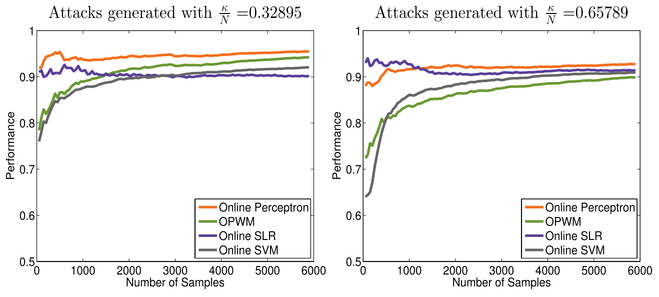


Fig. 10. Experiments using the OP, OPWMs, Online SVM, and SLR. Recall values of the OP are less than that of the OPWM for Class-1. Multiple phase transitions of performance values of the Online SVM are observed in the IEEE 118-bus system. (a) OP for the IEEE 57-bus. (b) OP for the IEEE 57-bus. (c) OP for the IEEE 118-bus. (d) OP for the IEEE 118-bus. (e) OPWM for the IEEE 57-bus. (f) OPWM for the IEEE 57-bus. (g) OPWM for the IEEE 118-bus. (h) OPWM for the IEEE 118-bus. (i) Online SVM for the IEEE 57-bus. (j) Online SVM for the IEEE 57-bus. (k) Online SVM for the IEEE 118-bus. (l) Online SVM for the IEEE 118-bus. (m) Online SLR for the IEEE 57-bus. (n) Online SLR for the IEEE 57-bus. (o) Online SLR for the IEEE 118-bus. (p) Online SLR for the IEEE 118-bus.

Learning curves of online learning algorithms are given in Fig. 11 for both observable attacks generated with $\kappa/N = 0.33$ and unobservable attacks generated with $\kappa/N = 0.66$. Since the cost function of each online learning algorithm is different, the learning performance is measured and depicted using Accuracy (Acc) defined in (23). In the results, the performance values of the Online SVM and OPWM increase as the number of samples increases, since the algorithms employ margin learning approaches that provide better learning rates as the number of training samples increases [34], [45].

Briefly, we suggest using Online SLR for the scenarios in which the precision of the classification of secure variables is important to avoid false alarms. On the other hand, if the



(a)

(b)

Fig. 11. Learning curves of online learning algorithms. (a) Observable attacks. (b) Unobservable attacks.

classification of attacked variables with high Precision and Recall values is an important task, we suggest using the OP.

V. Conclusion

The attack detection problem has been reformulated as a machine learning problem and the performance of supervised, semisupervised, classifier and feature space fusion, and online learning algorithms have been analyzed for different attack scenarios.

In a supervised binary classification problem, the attacked and secure measurements are labeled in two separate classes. In the experiments, we have observed that the state-of-the-art machine learning algorithms perform better than the well-known attack detection algorithms that employ an SVE approach for the detection of both observable and unobservable attacks.

We have observed that the perceptron is less sensitive and the k-NN is more sensitive to the system size than the other algorithms. In addition, the imbalanced data problem affects the performance of the k-NN. Therefore, k-NN may perform better in small-sized systems and worse in large-sized systems when compared to other algorithms. The SVM performs better than the other algorithms in large-scale systems. In the performance tests of the SVM, we observe a phase transition at κ^*, which is the minimum number of measurements that are required to be accessible by the attackers in order to construct unobservable attacks. Moreover, a large value of κ does not necessarily imply high impact of data injection attacks. For example, if the attack vector $\mathbf{a}$ has small values in all elements, then the impact of $\mathbf{a}$ may still be limited. More important, if $\mathbf{a}$ is a vector with small values compared with the noise, then even machine learning-based approaches may fail.

We have observed two challenges of SVMs in their application to attack detection problems in smart grid. First, the performance of the SVM is affected by the selection of kernel types. For instance, we have observed that the linear and Gaussian kernel SVM perform similarly in the IEEE 9-bus system. However, for the IEEE 57-bus system, the Gaussian kernel SVM outperforms its linear counterparts. Moreover, the values of the phase transition points of the performance of the Gaussian kernel SVM coincide with the theoretically computed κ^* values. This implies that the feature vectors in $\mathcal{F}$, which are computed using Gaussian kernels, are linearly separable for higher values of κ. Interestingly, the transition points miss κ^* in the IEEE 118-bus system, which means that alternative kernels, are needed for this system. Second, the SVM is sensitive to the sparsity of the systems. To solve this problem, sparse SVM [48] and kernel machines [49] can be employed. In this paper, we approached this problem using the SLR. However, obtaining an optimal regularization parameter, $\hat{\lambda}$, is computationally challenging [24].

To use information extracted from test data in the computation of the learning models, semisupervised methods have been employed in the proposed approach. In semisupervised learning algorithms, we have used test data together with training data in an optimization algorithm used to compute the learning model. The numerical results show that the semisupervised learning methods are more robust to the degree of sparsity of the data than the supervised learning methods.

We have employed Adaboost and MKL as decision- and feature-level fusion algorithms. Experimental results show that fusion methods provide learning models that are more robust to changes in the system size and data sparsity than the other methods. On the other hand, computational complexities of most of the classifier and feature fusion methods are higher than those of the single classifier and feature extraction methods.

Finally, we have analyzed online learning methods for real-time attack detection problems. Since a sequence of training samples or just a single sample is processed at each time, the computational complexity of most of the online algorithms is less than the batch learning algorithms. In the experiments, we have observed that the classification performance of online learning algorithms is comparable with that of the batch algorithms.

In future work, we plan to first apply the proposed approach and the methods to an attack classification problem for deciding which of several possible attack types have occurred given that an attack has been detected. Then, we plan to consider the relationship between measurement noise and bias-variance properties of learning models for the development of attack detection and classification algorithms. In addition, we plan to expand our analyses for varying numbers of clusters G and cluster sizes N_g, $\forall g = 1, 2, \ldots, G$, by relaxing the assumptions made in this paper for attack detection in smart grid systems, e.g., when the samples are not i.i.d. but rather are obtained from nonstationary distributions, such as when, concept drift [37] and data set shift [38] occur.

References

[1] C. Rudin *et al.*, "Machine learning for the New York City power grid," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 328–345, Feb. 2012.

[2] R. N. Anderson, A. Boulanger, W. B. Powell, and W. Scott, "Adaptive stochastic control for the smart grid," *Proc. IEEE*, vol. 99, no. 6, pp. 1098–1115, Jun. 2011.

[3] Z. M. Fadlullah, M. M. Fouda, N. Kato, X. Shen, and Y. Nozaki, "An early warning system against malicious activities for smart grid communications," *IEEE Netw.*, vol. 25, no. 5, pp. 50–55, Sep./Oct. 2011.

[4] Y. Zhang, L. Wang, W. Sun, R. C. Green, and M. Alam, "Distributed intrusion detection system in a multi-layer network architecture of smart grids," *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 796–808, Dec. 2011.

[5] M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni, and H. V. Poor, "Sparse attack construction and state estimation in the smart grid: Centralized and distributed models," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 7, pp. 1306–1318, Jul. 2013.

[6] A. Abur and A. G. Expósito, *Power System State Estimation: Theory and Implementation*. New York, NY, USA: Marcel Dekker, 2004.

[7] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, Chicago, IL, USA, Nov. 2009, pp. 21–32.

[8] O. Kosut, L. Jia, R. J. Thomas, and L. Tong, "Malicious data attacks on the smart grid," *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 645–658, Dec. 2011.

[9] E. Cotilla-Sanchez, P. D. H. Hines, C. Barrows, and S. Blumsack, "Comparing the topological and electrical structure of the North American electric power infrastructure," *IEEE Syst. J.*, vol. 6, no. 4, pp. 616–626, Dec. 2012.

[10] T. T. Kim and H. V. Poor, "Strategic protection against data injection attacks on power grids," *IEEE Trans. Smart Grid*, vol. 2, no. 2, pp. 326–333, Jun. 2011.

[11] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[12] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[13] M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni, and H. V. Poor, "Smarter security in the smart grid," in *Proc. 3rd IEEE Int. Conf. Smart Grid Commun.*, Tainan, Taiwan, Nov. 2012, pp. 312–317.

[14] L. Saitta, A. Giordana, and A. Cornuéjols, *Phase Transitions in Machine Learning*. New York, NY, USA: Cambridge Univ. Press, 2011.

[15] M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni, and H. V. Poor, "Distributed models for sparse attack construction and state vector estimation in the smart grid," in *Proc. 3rd IEEE Int. Conf. Smart Grid Commun.*, Tainan, Taiwan, Nov. 2012, pp. 306–311.

[16] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Advanced Lectures on Machine Learning*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds. Berlin, Germany: Springer-Verlag, 2004.

[17] S. Kulkarni and G. Harman, *An Elementary Introduction to Statistical Learning Theory*. Hoboken, NJ, USA: Wiley, 2011.

[18] Q. Wang, S. R. Kulkarni, and S. Verdú, "Divergence estimation for multidimensional densities via k-nearest-neighbor distances," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2392–2405, May 2009.

[19] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Orlando, FL, USA: Academic, 2006.

[20] R. D. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York, NY, USA: Wiley, 2001.

[21] I. Steinwart and A. Christmann, *Support Vector Machines*. New York, NY, USA: Springer-Verlag, 2008.

[22] S. R. Kulkarni and G. Harman, "Statistical learning theory: A tutorial," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 3, no. 6, pp. 543–556, 2011.

[23] O. Chapelle, "Training a support vector machine in the primal," *Neural Comput.*, vol. 19, no. 5, pp. 1155–1178, 2007.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[25] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[26] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th Int. Conf. Mach. Learn.*, Bled, Slovenia, Jun. 1999, pp. 200–209.

[27] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *J. Mach. Learn. Res.*, vol. 9, no. 6, pp. 203–233, 2008.

[28] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[29] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, AB, Canada, Jul. 2004, pp. 6–13.

[30] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Hoboken, NJ, USA: Wiley, 2004.

[31] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. Cambridge, MA, USA: MIT Press, 2012.

[32] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, Nov. 2008.

[33] S. Kakade and A. T. Kalai, "From batch to transductive online learning," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA, USA: MIT Press, 2005, pp. 611–618.

[34] F. Orabona, J. Keshet, and B. Caputo, "Bounded kernel-based online learning," *J. Mach. Learn. Res.*, vol. 10, pp. 2643–2666, Dec. 2009.

[35] P. Carbonetto, M. Schmidt, and N. de Freitas, "An interior-point stochastic approximation method and an L1-regularized delta rule," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Red Hook, NY, USA: Curran Associates, 2008, pp. 233–240.

[36] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.

[37] P. L. Bartlett, S. Ben-David, and S. R. Kulkarni, "Learning changing concepts by exploiting the structure of change," *Mach. Learn.*, vol. 41, no. 2, pp. 153–174, 2000.

[38] J. Quiñero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, Eds., *Dataset Shift in Machine Learning*. Cambridge, MA, USA: MIT Press, 2009.

[39] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, 2011, Art. ID 27.

[40] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

[41] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with Gaussian kernel," *Neural Comput.*, vol. 15, no. 7, pp. 1667–1689, 2003.

[42] K. Koh, S.-J. Kim, and S. Boyd, "An interior-point method for large-scale ℓ_1-regularized logistic regression," *J. Mach. Learn. Res.*, vol. 8, pp. 1519–1555, Jul. 2007.

[43] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. New York, NY, USA: Springer-Verlag, 1996.

[44] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 169–184.

[45] F. Orabona. (2009). *DOGMA: A MATLAB Toolbox for Online Learning*. [Online]. Available: http://dogma.sourceforge.net

[46] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization," *Proc. Nat. Acad. Sci. United States Amer.*, vol. 100, no. 5, pp. 2197–2202, 2003.

[47] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[48] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *J. Mach. Learn. Res.*, vol. 3, pp. 1229–1243, Mar. 2003.

[49] M. Wu, B. Schölkopf, and G. Bakir, "A direct method for building sparse kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 603–624, Apr. 2006.

Mete Ozay (M'09) received the Ph.D. degree in computer engineering from Middle East Technical University, Ankara, Turkey, in 2013.

He was a Visiting Ph.D. Student and Research Collaborator with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, from 2011 to 2012. He is currently with the School of Computer Science, University of Birmingham, Birmingham, U.K. His current research interests include pure and applied mathematics, mathematical physics, information theory, and theoretical computer science.

Iñaki Esnaola (S'08–M'11) received the M.S. degree in electrical engineering from the University of Navarra, Pamplona, Spain, in 2006, and the Ph.D. degree in electrical engineering from the University of Delaware, Newark, DE, USA, in 2011.

He was a Research Intern with Bell Laboratories, Alcatel-Lucent, Holmdel, NJ, USA, from 2010 to 2011, and a Post-Doctoral Research Associate with Princeton University from 2011 to 2013. He is currently a Lecturer with the Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, U.K., and a Visiting Research Collaborator with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA. His current research interests include information theory and communication theory with an emphasis on the application to electricity grid problems.

Fatos Tunay Yarman Vural (SM'98) received the B.S. (Hons.) degree in electrical engineering from the Technical University of Istanbul, Istanbul, Turkey, in 1973, the M.S. degree in electrical engineering from Bogazici University, Istanbul, in 1975, and the Ph.D. degree in electrical engineering and computer science from Princeton University, Princeton, NJ, USA, in 1981.

She was a Research Scientist with the Marmara Research Institute, Istanbul, from 1981 to 1983. From 1983 to 1985, she was a Visiting Professor with Drexel University, Philadelphia, PA, USA. From 1985 to 1992, she was a Technical and Deputy Manager with Yapitel Inc., San Diego, CA, USA. She was the Chair of the Department of Computer Engineering with Middle East Technical University (METU), Ankara, Turkey, from 1996 to 2000. Since 1992, she has been a Professor with the Department of Computer Engineering, METU. She has been involved in teaching, consulting, and organizing conferences in her research areas. Her current research interests include computer vision, image processing, pattern recognition, and artificial intelligence.

Prof. Vural is a member of the Turkish Informatics Foundation and the Turkish Information Association, and Chamber of Electrical Engineers in Turkey.

Sanjeev R. Kulkarni (M'91–SM'96–F'04) received the B.S. degree in mathematics and B.S. degree in electrical engineering from Clarkson University, Potsdam, NY, USA, the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, and the Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1991.

He was a member of the Technical Staff with the MIT Lincoln Laboratory, Lexington, MA, USA, from 1985 to 1991. He was a Research Fellow with Australian National University, Canberra, ACT, Australia, in 1996, the Susquehanna International Group, Bala Cynwyd, PA, USA, in 1998, and Flarion Technologies, Bridgewater, NJ, USA, in 2001. Since 1991, he has been with Princeton University, Princeton, NJ, USA, where he is currently a Professor of Electrical Engineering and the Dean of the Graduate School. He is an Affiliated Faculty Member with the Department of Operations Research and Financial Engineering and the Department of Philosophy. His current research interests include statistical pattern recognition, nonparametric estimation, learning and adaptive systems, information theory, wireless networks, and image/video processing.

Prof. Kulkarni was a recipient of the ARO Young Investigator Award in 1992, the NSF Young Investigator Award in 1994, and a number of teaching awards at Princeton University. He has served as an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY.

H. Vincent Poor (S'72–M'77–SM'82–F'87) received the Ph.D. degree in electrical engineering and computer science from Princeton University, Princeton, NJ, USA, in 1977.

He was on the faculty of the University of Illinois at Urbana-Champaign, Champaign, IL, USA, from 1977 to 1990. Since 1990, he has been on the faculty of Princeton University, where he is currently the Michael Henry Strater University Professor of Electrical Engineering and the Dean of the School of Engineering and Applied Science. His current research interests include wireless networks and related fields. Among his publication in these areas is the recent book *Mechanisms and Games for Dynamic Spectrum Allocation* (Cambridge University Press, 2014).

Dr. Poor is a member of the National Academy of Engineering and the National Academy of Sciences, and a Foreign Member of the Academia Europaea and the Royal Society. He is a fellow of the American Academy of Arts and Sciences, the Royal Academy of Engineering (U.K.), and the Royal Society of Edinburgh. In 1990, he served as the President of the IEEE Information Theory Society, and from 2004 to 2007, he served as the Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY. Recent recognition of his work includes the 2014 URSI Booker Gold Medal, and honorary doctorates from several universities in Asia and Europe.