Anomaly Detection via Online Oversampling Principal Component Analysis

Yuh-Jye Lee, Yi-Ren Yeh, and Yu-Chiang Frank Wang, *Member*, *IEEE*

Abstract—Anomaly detection has been an important research topic in data mining and machine learning. Many real-world applications such as intrusion or credit card fraud detection require an effective and efficient framework to identify deviated data instances. However, most anomaly detection methods are typically implemented in batch mode, and thus cannot be easily extended to large-scale problems without sacrificing computation and memory requirements. In this paper, we propose an online oversampling principal component analysis (osPCA) algorithm to address this problem, and we aim at detecting the presence of outliers from a large amount of data via an online updating technique. Unlike prior principal component analysis (PCA)-based approaches, we do not store the entire data matrix or covariance matrix, and thus our approach is especially of interest in online or large-scale problems. By oversampling the target instance and extracting the principal direction of the data, the proposed osPCA allows us to determine the anomaly of the target instance according to the variation of the resulting dominant eigenvector. Since our osPCA need not perform eigen analysis explicitly, the proposed framework is favored for online applications which have computation or memory limitations. Compared with the well-known power method for PCA and other popular anomaly detection algorithms, our experimental results verify the feasibility of our proposed method in terms of both accuracy and efficiency.

Index Terms—Anomaly detection, online updating, least squares, oversampling, principal component analysis

1 Introduction

Anomaly (or outlier) detection aims to identify a small group of instances which deviate remarkably from the existing data. A well-known definition of "outlier" is given in [1]: "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism," which gives the general idea of an outlier and motivates many anomaly detection methods [1], [2], [3], [4], [5], [6], [7]. Practically, anomaly detection can be found in applications such as homeland security, credit card fraud detection, intrusion and insider threat detection in cyber-security, fault detection, or malignant diagnosis [3], [4], [6], [8], [9]. However, since only a limited amount of labeled data are available in the above real-world applications, how to determine anomaly of unseen data (or events) draws attention from the researchers in data mining and machine learning communities [1], [2], [3], [4], [5], [6], [7].

Despite the rareness of the deviated data, its presence might enormously affect the solution model such as the distribution or principal directions of the data. For example, the calculation of data mean or the least squares solution of the associated linear regression model is both sensitive to outliers. As a result, anomaly detection needs to solve an unsupervised yet unbalanced data learning problem. Similarly, we observe that removing (or adding) an abnormal data instance will affect the principal direction of the resulting data than removing (or adding) a normal one does. Using the above "leave one out" (LOO) strategy, we can calculate the principal direction of the data set without the target instance present and that of the original data set. Thus, the outlierness (or anomaly) of the data instance can be determined by the variation of the resulting principal directions. More precisely, the difference between these two eigenvectors will indicate the anomaly of the target instance. By ranking the difference scores of all data points, one can identify the outlier data by a predefined threshold or a predetermined portion of the data.

We note that the above framework can be considered as a *decremental* PCA (dPCA)-based approach for anomaly detection. While it works well for applications with moderate data set size, the variation of principal directions might not be significant when the size of the data set is large. In real-world anomaly detection problems dealing with a large amount of data, adding or removing one target instance only produces negligible difference in the resulting eigenvectors, and one cannot simply apply the dPCA technique for anomaly detection. To address this practical problem, we advance the "oversampling" strategy to duplicate the target instance, and we perform an oversampling PCA (osPCA) on such an oversampled data set. It is obvious that the effect of an outlier instance will be amplified due to its duplicates present in the principal component analysis (PCA) formulation, and this makes the detection of outlier data easier. However, this LOO anomaly

- *Y.-J. Lee is with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10607, Taiwan. E-mail: yuh-jye@mail.ntust.edu.tw.*
- *Y.-R. Yeh is with the Intel-NTU Connected Context Computing Center, National Taiwan University, No. 1, Section 4, Roosevelt Road, Taipei 10617, Taiwan. E-mail: yryeh@citi.sinica.edu.tw.*
- *Y.-C.F. Wang is with the Research Center for Information Technology Innovation, Academia Sinica, No. 128 Academia Road, Section 2, Nankang, Taipei 11529, Taiwan. E-mail: ycwang@citi.sinica.edu.tw.*

Manuscript received 7 Apr. 2011; revised 31 Jan. 2012; accepted 26 Apr. 2012; published online 9 May 2012.
Recommended for acceptance by J. Yang.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-04-0186.
Digital Object Identifier no. 10.1109/TKDE.2012.99.

detection procedure with an oversampling strategy will markedly increase the computational load. For each target instance, one always needs to create a dense covariance matrix and solves the associated PCA problem. This will prohibit the use of our proposed framework for real-world large-scale applications. Although the well known power method is able to produce approximated PCA solutions, it requires the storage of the covariance matrix and cannot be easily extended to applications with streaming data or online settings. Therefore, we present an online updating technique for our osPCA. This updating technique allows us to efficiently calculate the approximated dominant eigenvector without performing eigen analysis or storing the data covariance matrix. Compared to the power method or other popular anomaly detection algorithms, the required computational costs and memory requirements are significantly reduced, and thus our method is especially preferable in online, streaming data, or large-scale problems. Detailed derivations and discussions of the osPCA with our proposed online updating technique will be presented in Section 4.

The remaining of this paper is organized as follows: Section 2 reviews prior anomaly detection methods. The osPCA for anomaly detection in Section 3. Section 4 details the proposed online updating technique for osPCA, and explains why this technique is computationally preferable to prior anomaly detection methods. Section 5 presents the experimental results, including comparisons with prior approaches. Finally, Section 6 concludes this paper.

2 Related Work

In the past, many outlier detection methods have been proposed [1], [2], [5], [10], [11], [12], [13], [14], [15]. Typically, these existing approaches can be divided into three categories: distribution (statistical), distance and density-based methods. Statistical approaches [1], [11] assume that the data follows some standard or predetermined distributions, and this type of approach aims to find the outliers which deviate form such distributions. However, most distribution models are assumed univariate, and thus the lack of robustness for multidimensional data is a concern. Moreover, since these methods are typically implemented in the original data space directly, their solution models might suffer from the noise present in the data. Nevertheless, the assumption or the prior knowledge of the data distribution is not easily determined for practical problems.

For distance-based methods [10], [13], [14], the distances between each data point of interest and its neighbors are calculated. If the result is above some predetermined threshold, the target instance will be considered as an outlier. While no prior knowledge on data distribution is needed, these approaches might encounter problems when the data distribution is complex (e.g., multi-clustered structure). In such cases, this type of approach will result in determining improper neighbors, and thus outliers cannot be correctly identified.

To alleviate the aforementioned problem, density-based methods are proposed [2], [12]. One of the representatives of this type of approach is to use a density-based local outlier factor (LOF) to measure the outlierness of each data instance [2]. Based on the local density of each data instance, the LOF determines the degree of outlierness, which provides suspicious ranking scores for all samples. The most important property of the LOF is the ability to estimate local data structure via density estimation. This allows users to identify outliers which are sheltered under a global data structure. However, it is worth noting that the estimation of local data density for each instance is very computationally expensive, especially when the size of the data set is large.

Besides the above work, some anomaly detection approaches are recently proposed [5], [15], [16]. Among them, the angle-based outlier detection (ABOD) method [5] is very unique. Simply speaking, ABOD calculates the variation of the angles between each target instance and the remaining data points, since it is observed that an outlier will produce a smaller angle variance than the normal ones do. It is not surprising that the major concern of ABOD is the computation complexity due a huge amount of instance pairs to be considered. Consequently, a fast ABOD algorithm is proposed to generate an approximation of the original ABOD solution. The difference between the standard and the fast ABOD approaches is that the latter only considers the variance of the angles between the target instance and its k nearest neighbors. However, the search of the nearest neighbors still prohibits its extension to large-scale problems (batch or online modes), since the user will need to keep all data instances to calculate the required angle information.

It is worth noting that the above methods are typically implemented in batch mode, and thus they cannot be easily extended to anomaly detection problems with streaming data or online settings. While some online or incremental-based anomaly detection methods have been recently proposed [17], [18], we found that their computational cost or memory requirements might not always satisfy online detection scenarios. For example, while the incremental LOF in [17] is able to update the LOFs when receiving a new target instance, this incremental method needs to maintain a preferred (or filtered) data subset. Thus, the memory requirement for the incremental LOF is $O(np)$ [17], [18], where n and p are the size and dimensionality of the data subset of interest, respectively. In [18], Ahmed proposed an online kernel density estimation for anomlay detection, but the proposed algorithm requires at least $O(np^2 + n^2)$ for computation complexity [18]. In online settings or large-scale data problems, the aforementioned methods might not meet the online requirement, in which both computation complexity and memory requirement are as low as possible. In this paper, the use of the osPCA with our proposed online updating technique is favored for such problems, since we only require $O(p)$ for both computation and memory costs (see Section 4 for detailed discussions).

3 Anomaly Detection via PCA

We first briefly review the PCA algorithm in Section 3.1. Based on the LOO strategy, Section 3.2 presents our study on the effect of outliers on the derived principal directions.

3.1 Principal Component Analysis

PCA is a well known unsupervised dimension reduction method, which determines the principal directions of the data distribution. To obtain these principal directions, one needs to construct the data covariance matrix and calculate its dominant eigenvectors. These eigenvectors will be the *most informative* among the vectors in the original data space, and are thus considered as the principal directions. Let $\mathbf{A} = [\mathbf{x}_1^\top; \mathbf{x}_2^\top; \ldots; \mathbf{x}_n^\top] \in \mathbb{R}^{n \times p}$, where each row $\mathbf{x}_i$ represents a data instance in a p dimensional space, and n is the number of the instances. Typically, PCA is formulated as the following optimization problem

$$\max_{\mathbf{U} \in \mathbb{R}^{p \times k}, \|\mathbf{U}\| = \mathbf{I}} \sum_{i=1}^{n} \mathbf{U}^\top (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top \mathbf{U}, \tag{1}$$

where $\mathbf{U}$ is a matrix consisting of k dominant eigenvectors. From this formulation, one can see that the standard PCA can be viewed as a task of determining a subspace where the projected data has the largest variation.

Alternatively, one can approach the PCA problem as minimizing the data reconstruction error, i.e.

$$\min_{\mathbf{U} \in \mathbb{R}^{p \times k}, \|\mathbf{U}\| = \mathbf{I}} J(\mathbf{U}) = \sum_{i=1}^{n} \|(\mathbf{x}_i - \mu) - \mathbf{U}\mathbf{U}^\top (\mathbf{x}_i - \mu)\|^2, \tag{2}$$

where $\mathbf{U}^\top (\mathbf{x}_i - \mu)$ determines the optimal coefficients to weight each principal directions when reconstructing the approximated version of $(\mathbf{x}_i - \mu)$. Generally, the problem in either (1) or (2) can be solved by deriving an eigenvalue decomposition problem of the covariance data matrix, i.e.

$$\Sigma_{\mathbf{A}} \mathbf{U} = \mathbf{U} \Lambda, \tag{3}$$

where

$$\Sigma_{\mathbf{A}} = \frac{1}{n} \sum_{i=i}^{n} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top \tag{4}$$

is the covariance matrix, μ is the global mean. Each column of $\mathbf{U}$ represents an eigenvector of $\Sigma_{\mathbf{A}}$, and the corresponding diagonal entry in Λ is the associated eigenvalue. For the purpose of dimension reduction, the last few eigenvectors will be discarded due to their negligible contribution to the data distribution.

While PCA requires the calculation of global mean and data covariance matrix, we found that both of them are sensitive to the presence of outliers. As shown in [19], if there are outliers present in the data, dominant eigenvectors produced by PCA will be remarkably affected by them, and thus this will produce a significant variation of the resulting principal directions.

We will further discuss this issue in the following sections, and explain how we advance this property for anomaly detection.

3.2 The Use of PCA for Anomaly Detection

In this section, we study the variation of principal directions when we remove or add a data instance, and how we utilize this property to determine the outlierness of the target data point.

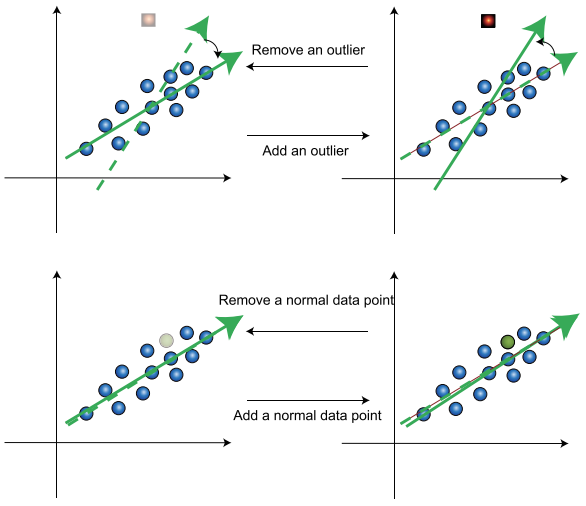


Fig. 1. The effects of adding/removing an outlier or a normal data instance on the principal directions.

We use Fig. 1 to illustrate the above observation. We note that the clustered blue circles in Fig. 1 represent normal data instances, the red square denotes an outlier, and the green arrow is the dominant principal direction. From Fig. 1, we see that the principal direction is deviated when an outlier instance is added. More specifically, the presence of such an outlier instance produces a large angle between the resulting and the original principal directions. On the other hand, this angle will be small when a normal data point is added. Therefore, we will use this property to determine the outlierness of the target data point using the LOO strategy.

We now present the idea of combining PCA and the LOO strategy for anomaly detection. Given a data set $\mathbf{A}$ with n data instances, we first extract the dominant principal direction $\mathbf{u}$ from it. If the target instance is $\mathbf{x}_t$, we next compute the leading principal direction $\tilde{\mathbf{u}}_t$ without $\mathbf{x}_t$ present. To identify the outliers in a data set, we simply repeat this procedure n times with the LOO strategy (one for each target instance)

$$\Sigma_{\tilde{\mathbf{A}}} \tilde{\mathbf{u}}_t = \lambda \tilde{\mathbf{u}}_t, \tag{5}$$

where $\tilde{\mathbf{A}} = \mathbf{A} \setminus \{\mathbf{x}_t\}$. We note that $\tilde{\mu}$ is the mean of $\tilde{\mathbf{A}}$, and thus

$$\Sigma_{\tilde{\mathbf{A}}} = \frac{1}{n-1} \sum_{\mathbf{x}_i \in \mathbf{A}/\{\mathbf{x}_t\}} (\mathbf{x}_i - \tilde{\mu})(\mathbf{x}_i - \tilde{\mu})^\top. \tag{6}$$

Once these eigenvectors $\tilde{\mathbf{u}}_t$ are obtained, we use the absolute value of cosine similarity to measure the variation of the principal directions, i.e.

$$s_t = 1 - \left| \frac{\langle \tilde{\mathbf{u}}_t, \mathbf{u} \rangle}{\|\tilde{\mathbf{u}}_t\| \|\mathbf{u}\|} \right|. \tag{7}$$

This s_t can be considered as a "score of outlierness," which indicates the anomaly of the target instance $\mathbf{x}_t$. We note that s_t can be also viewed as the influence of the target instance in the resulting principal direction, and a higher s_t score (closer to 1) means that the target instance is more likely to be an outlier. For a target instance, if its s_t is above some

threshold, we then identify this instance as an outlier. We refer to this process as a *decremental PCA* with LOO scheme for anomaly detection.

In contrast with decremental PCA with the LOO strategy, we also consider the use of adding/duplicating a data instance of interest when applying PCA for outlier detection. This setting is especially practical for streaming data anomaly detection problems. To be more precise, when receiving a new target instance $\mathbf{x}_t$, we solve the following PCA problem

$$\Sigma_{\tilde{\mathbf{A}}}\tilde{\mathbf{u}}_t = \lambda\tilde{\mathbf{u}}_t, \tag{8}$$

where $\tilde{\mathbf{A}} = \mathbf{A} \cup \{\mathbf{x}_t\}$. Again, $\tilde{\mu}$ is the mean of $\tilde{\mathbf{A}}$, and the covariance matrix can be calculated as

$$\begin{aligned}\Sigma_{\tilde{\mathbf{A}}} = &\frac{1}{n+1}\sum_{\mathbf{x}_i \in \mathbf{A}}(\mathbf{x}_i - \tilde{\boldsymbol{\mu}})(\mathbf{x}_i - \tilde{\boldsymbol{\mu}})^\top \\ &+ \frac{1}{n+1}(\mathbf{x}_t - \tilde{\boldsymbol{\mu}})(\mathbf{x}_t - \tilde{\boldsymbol{\mu}})^\top.\end{aligned} \tag{9}$$

After deriving the principal direction $\tilde{\mathbf{u}}_t$ of $\tilde{\mathbf{A}}$, we apply (7) and calculate the score s_t, and the outlierness of that target instance can be determined accordingly. This strategy is also preferable for online anomaly detection applications, in which we need to determine whether a newly received data instance (viewed as a target instance) is an outlier. If the recently received data points are normal ones, adding such instances will not significantly affect the principal directions (and vice versa). While one might argue that it might not be sufficient to simply use the variation of the principal direction to evaluate the anomaly of the data, we will explain in the next section why our oversampling PCA alleviates this problem and makes the online anomaly detection problem solvable.

It is worth noting that if an outlier instance is far away from the data cloud (of normal data instances) but along the direction of its dominant eigenvector, our method will not be able to identify such anomaly. It is worth pointing out that, such an outlier actually indicates the anomaly in most (if not all) of the feature attributes. This means that, most of the feature attributes of this instance are way beyond the normal range/distribution (in the same scale) of each feature variable. As a result, the anomaly of such a data input can be easily detected by simple outlier detection methods such as single feature variable thresholding. For example, one can calculate the mean and standard deviation of the normal data instances projected onto the dominant eigenvector. For an input data point, if its projected coefficient onto this eigenvector is beyond two or three times of the standard deviation (i.e., away from 95.45 or 99.73 percent of normal data instances), it will be flagged as an outlier.

We would also like to point out that, such an outlier instance might not be presented in practical outlier detection scenarios due to the violation of system limitations. Taking network traffic/behavior anomaly detection for example, one might consider power, bandwidth, capacity (data rates), and other parameters of a router/switch as the features to be observed. If a data instance is far away from the normal data cloud but along its principal direction, we will have most of these router parameters simultaneously above their normal ranges, while some of them might even exceed their physical limitations. Therefore, the anomaly of this input will be easily detected by system designs and does not require a more advanced outlier detection method like ours.

4 Oversampling PCA for Anomaly Detection

For practical anomaly detection problems, the size of the data set is typically large, and thus it might not be easy to observe the variation of principal directions caused by the presence of a single outlier. Furthermore, in the above PCA framework for anomaly detection, we need to perform n PCA analysis for a data set with n data instances in a p-dimensional space, which is not computationally feasible for large-scale and online problems. Our proposed oversampling PCA (osPCA) together with an online updating strategy will address the above issues, as we now discuss.

In Section 4.1, we introduce our osPCA, and discuss how and why we are able to detect the presence of abnormal data instances according to the associated principal directions, even when the size of data is large. In Section 4.2, the well-known power method [20] is applied to determine the principal direction without the need to solve each eigenvalue decomposition problem. While this power method alleviates the computation cost in determining the principal direction as verified in our previous work in [19], we will discuss its limitations and explain why the use of power method is not practical in online settings. In Section 4.3, we present a least squares approximation of our osPCA, followed by the proposed online updating algorithm which is able to solve the online osPCA efficiently.

4.1 Oversampling Principal Components Analysis (osPCA)

As mentioned earlier, when the size of the data set is large, adding (or removing) a single outlier instance will not significantly affect the resulting principal direction of the data. Therefore, we advance the oversampling strategy and present an oversampling PCA (osPCA) algorithm for large-scale anomaly detection problems.

The proposed osPCA scheme will duplicate the target instance multiple times, and the idea is to amplify the effect of outlier rather than that of normal data. While it might not be sufficient to perform anomaly detection simply based on the most dominant eigenvector and ignore the remaining ones, our online osPCA method aims to efficiently determine the anomaly of each target instance without sacrificing computation and memory efficiency. More specifically, if the target instance is an outlier, this oversampling scheme allows us to overemphasize its effect on the most dominant eigenvector, and thus we can focus on extracting and approximating the dominant principal direction in an online fashion, instead of calculating multiple eigenvectors carefully.

We now give the detailed formulation of the osPCA. Suppose that we oversample the target instance $\tilde{n}$ times, the associated PCA can be formulated as follows

$$\Sigma_{\tilde{\mathbf{A}}}\tilde{\mathbf{u}}_t = \lambda\tilde{\mathbf{u}}_t, \tag{10}$$

where $\tilde{\mathbf{A}} = \mathbf{A} \cup \{\mathbf{x}_t, \ldots, \mathbf{x}_t\} \in \mathbb{R}^{(n+\tilde{n})\times p}$. The mean of $\tilde{\mathbf{A}}$ is $\tilde{\boldsymbol{\mu}}$, and thus

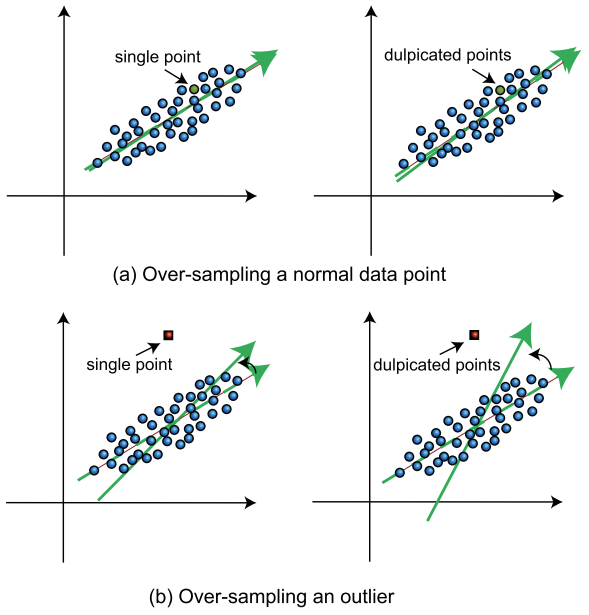


Fig. 2. The effect of an oversampled normal data or outlier instance on the principal direction.

$$\begin{aligned}\Sigma_{\tilde{\mathbf{A}}} &= \frac{1}{n+\tilde{n}}\sum_{\mathbf{x}_i\in\mathbf{A}}\mathbf{x}_i\mathbf{x}_i^\top + \frac{1}{n+\tilde{n}}\sum_{i=1}^{\tilde{n}}\mathbf{x}_t\mathbf{x}_t^\top - \tilde{\boldsymbol{\mu}}\tilde{\boldsymbol{\mu}}^\top \\ &= \frac{1}{1+r}\frac{\mathbf{A}\mathbf{A}^\top}{n} + \frac{r}{1+r}\mathbf{x}_t\mathbf{x}_t^\top - \tilde{\boldsymbol{\mu}}\tilde{\boldsymbol{\mu}}^\top.\end{aligned} \tag{11}$$

In this osPCA framework, we will duplicate the target instance $\tilde{n}$ times (e.g., 10 percent of the size of the original data set), and we will compute the score of outlierness s_t of that target instance, as defined in (7). If this score is above some predetermined threshold, we will consider this instance as an outlier. With this oversampling strategy, if the target instance is a normal data (see Fig. 2 a for example), we will observe negligible changes in the principal directions and the mean of the data. The case of oversampling an abnormal instance is shown in Fig. 2b. It is worth noting that the use of osPCA not only determines outliers from the existing data, it can be applied to anomaly detection problems with streaming data or those with online requirements, as we discuss later.

Clearly, the major concern is the computation cost of calculating or updating the principal directions in large-scale problems. We will discuss this issue and propose our solutions in the following sections.

4.2 Effects of the Oversampling Ratio on osPCA

Using the proposed osPCA for anomaly detection, the oversampling ratio r in (11) will be the parameter for the user to be determined. We note that, since there is no training or validation data for practical anomaly detection problems, one cannot perform cross-validation or similar strategies to determine this parameter in advance.

When applying our osPCA to detect the presence of outliers, calculating the principal direction of the updated data matrix (with oversampled data introduced) can be considered as the task of eigenvalue decomposition of the perturbed covariance matrix. Theoretically, the degree of perturbation is dependent on the oversampling ratio r, and the sensitivity of deriving the associated dominant eigenvector can be studied as follows:

To discuss such perturbation effects, let $\mathbf{A} = [\mathbf{x}_1^\top; \mathbf{x}_2^\top; \ldots; \mathbf{x}_n^\top] \in \mathbb{R}^{n\times p}$ as the data matrix, where each row represents a data instance in a p dimensional space, and n is the number of the instances. For at target instance $\mathbf{x}_t$ oversampled $\tilde{n}$ times, we can derive the resulting covariance matrix. Let $\epsilon = \frac{\tilde{n}}{n+\tilde{n}}$, we calculate the perturbed data covariance matrix Σ_ϵ as

$$\begin{aligned}\Sigma_\epsilon &= \frac{1}{n+\tilde{n}}\left\{\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu}_\epsilon)(\mathbf{x}_i-\boldsymbol{\mu}_\epsilon)^\top \right. \\ &\quad \left. + \sum_{i=1}^{\tilde{n}}(\mathbf{x}_t-\boldsymbol{\mu}_\epsilon)(\mathbf{x}_t-\boldsymbol{\mu}_\epsilon)^\top\right\} \\ &= \frac{1}{n+\tilde{n}}\left\{\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^\top \right. \\ &\quad \left. + \sum_{i=1}^{\tilde{n}}(\mathbf{x}_t-\boldsymbol{\mu})(\mathbf{x}_t-\boldsymbol{\mu})^\top\right\} + O(\epsilon^2) \\ &= (1-\epsilon)\Sigma + \epsilon\Sigma_{\mathbf{x}_t} + O(\epsilon^2).\end{aligned} \tag{12}$$

Note that $\|\boldsymbol{\mu}_\epsilon - \boldsymbol{\mu}\| = \epsilon\|\mathbf{x}_t - \boldsymbol{\mu}\| = O(\epsilon)$ and $\|(\boldsymbol{\mu}_\epsilon - \boldsymbol{\mu})(\boldsymbol{\mu}_\epsilon - \boldsymbol{\mu})^\top\| = O(\epsilon^2)$. Based on (12), we can observe that a normal data instance (i.e., close to μ) would make $\epsilon \to 0$ and $\|\Sigma_{\mathbf{x}_t}\| \to 0$, and thus the perturbed covariance matrix Σ_ϵ will not be remarkably different from the original one Σ. On the other hand, if an outlier instance (i.e., far away from μ) is a target input to be oversampled, Σ_ϵ will be significantly affected by $\Sigma_{\mathbf{x}_t}$ (due to a larger ϵ), and thus the derived principal direction will also be remarkably different from the one without noteworthy perturbation. More details of this study, which focuses on the effects of the perturbed data on the resulting covariance matrix, can be found in [21] (see Lemma 2.1 in [21]).

The above theoretical analysis supports our use of the variation of the dominant eigenvector for anomaly detection. Using (12), while we can theoretically estimate the perturbed eigenvector u_ϵ with a residual for an oversampled target instance, such an estimation is associated with the residual term $O(\epsilon^2)$, and ϵ is a function of $\tilde{n}$ (and thus a function of the oversampling ratio r). Based on (12), while a larger r values will more significantly affect the resulting principal direction, the presence of the residual term prevents us from performing further theoretical evaluation or comparisons (e.g., threshold determination). Nevertheless, one can expect to detect an outlier instance using the above strategy. No matter how larger the oversampling ratio r is, the presence of outlier data will affect more on the dominant eigenvector than a normal instance does. In practice, we also find that our anomaly detection performance is *not* sensitive to the choice of the oversampling ratio r (see Section 5.3).

4.3 The Power Method for osPCA

Typically, the solution to PCA is determined by solving an eigenvalue decomposition problem. In the LOO scenario, one will need to solve the PCA and to calculate the principal directions n times for a data set with n instances. This is very computationally expensive, and prohibits the practical use of such a framework for anomaly detection.

It can be observed that, in the PCA formulation with the LOO setting, it is not necessary to recompute the covariance matrices for each PCA. This is because when we duplicate a data point of interest, the difference between the updated covariance matrix and the original one can be easily determined. Let $\mathbf{Q} = \frac{\mathbf{A}\mathbf{A}^\top}{n}$ be the outer product matrix and $\mathbf{x}_t$ be the target instance (to be oversampled), we use the following technique to update the mean $\tilde{\boldsymbol{\mu}}$ and the covariance matrix $\Sigma_{\tilde{\mathbf{A}}}$

$$\tilde{\boldsymbol{\mu}} = \frac{\boldsymbol{\mu} + r \cdot \mathbf{x}_t}{1 + r}, \tag{13}$$

and

$$\boldsymbol{\Sigma}_{\tilde{\mathbf{A}}} = \frac{1}{1+r}\mathbf{Q} + \frac{r}{1+r}\mathbf{x}_t\mathbf{x}_t^\top - \tilde{\boldsymbol{\mu}}\tilde{\boldsymbol{\mu}}^\top, \tag{14}$$

where $r < 1$ is the parameter controlling the size when oversampling $\mathbf{x}_t$. From (14), we can see that one only needs to keep the matrix $\mathbf{Q}$ when calculating $\Sigma_{\tilde{\mathbf{A}}}$, and there is no need to re-compute the entire covariance matrix in this LOO framework.

Once the update covariance matrix $\Sigma_{\tilde{A}}$ is obtained, the principal directions can be obtained by solving the eigenvalue decomposition problem of each of the matrices $\Sigma_{\tilde{A}}$. To alleviate this computation load, we apply the well-known *power method* [20], which is a simple iterative algorithm and does not compute matrix decomposition. This method starts with an initial normalized vector $\mathbf{u}_0$, which could be an approximation of the dominant eigenvector or a nonzero random vector. Next, the new $\mathbf{u}_{k+1}$ (a better approximated version of the dominant eigenvector) is updated by

$$\mathbf{u}_{k+1} = \frac{\boldsymbol{\Sigma}_{\tilde{\mathbf{A}}}\mathbf{u}_k}{\|\boldsymbol{\Sigma}_{\tilde{\mathbf{A}}}\mathbf{u}_k\|}. \tag{15}$$

The sequence $\{\mathbf{u}_k\}$ converges under the assumption that the dominant eigenvalue of $\Sigma_{\tilde{\mathbf{A}}}$ is markedly larger than others. From (15), it is clear that the power method only requires matrix multiplications, not decompositions; therefore, the use of the power method can alleviate the computation cost in calculating the dominant principal direction.

We note that, to avoid keeping the data covariance matrix $\Sigma_{\tilde{\mathbf{A}}} \in \mathbb{R}^{p\times p}$ during the entire updating process, we can first compute $\mathbf{y} = \mathbf{A}\mathbf{u}_{k-1}$ and then calculate $\mathbf{u}_k = \mathbf{y}^\top\mathbf{A}$. As a result, when applying this technique for the power method, we do not need to compute and store the covariance matrix. However, as can be seen from the above process, we still need to keep the data matrix $\mathbf{A}$ (with the memory cost $O(n \times p)$) for the matrix-vector multiplication. Moreover, this multiplication needs to be performed for each iteration of the power method.

In our anomaly detection framework, we only consider the first principal component and evaluate its variation in computing the score of outlierness of each sample. One could use the deflation process [20] if other principal directions besides the dominant one need to be determined.

4.4 Least Squares Approximation and Online Updating for osPCA

In the previous section, we apply a matrix update technique in (14) and the power method to solve our oversampling PCA for outlier detection. However, the major concern of the power method is that it does not guarantee a fast convergence, even if we use prior principal directions as its initial solutions. Moreover, the use of power method still requires the user to keep the entire covariance matrix, which prohibits the problems with high-dimensional data or with limited memory resources. Inspired by [22], [23], we propose an online updating algorithm to calculate the dominant eigenvector when oversampling a target instance. We now discuss the details of our proposed algorithm.

Recall that, in Section 3, PCA can be considered as a problem to minimize the reconstruction error

$$\min_{\mathbf{U}\in\mathbb{R}^{p\times k},\mathbf{U}^\top\mathbf{U}=\mathbf{I}} J(\mathbf{U}) = \sum_{i=1}^{n} \|\bar{\mathbf{x}}_i - \mathbf{U}\mathbf{U}^\top\bar{\mathbf{x}}_i\|^2, \tag{16}$$

where $\bar{\mathbf{x}}_i$ is $(\mathbf{x}_i - \mu)$, $\mathbf{U}$ is the matrix consisting of k dominant eigenvectors, and $\mathbf{U}\mathbf{U}^\top\bar{\mathbf{x}}_i$ is the reconstructed version of $\bar{\mathbf{x}}_i$ using the eigenvectors in $\mathbf{U}$. The above reconstruction error function can be further approximated by a least squares form [24]

$$\begin{aligned}\min_{\mathbf{U}\in\mathbb{R}^{p\times k},\mathbf{U}^\top\mathbf{U}=\mathbf{I}} J_{ls}(\mathbf{U}) &= \sum_{i=1}^{n} \|\bar{\mathbf{x}}_i - \mathbf{U}\mathbf{U}'^\top\bar{\mathbf{x}}_i\|^2 \\ &= \sum_{i=1}^{n} \|\bar{\mathbf{x}}_i - \mathbf{U}\mathbf{y}_i\|^2,\end{aligned} \tag{17}$$

where $\mathbf{U}'$ is the approximation of $\mathbf{U}$, and thus $\mathbf{y}_i = \mathbf{U}'^\top\bar{\mathbf{x}}_i \in \mathbb{R}^k$ is the approximation of the projected data $\mathbf{U}^\top\bar{\mathbf{x}}_i$ in the lower k dimensional space. Based on this formulation, the reconstruction error has a quadratic form and is a function of $\mathbf{U}$, which can be computed by solving a least squares problem. The trick for this least squares problem is the approximation of $\mathbf{U}^\top\bar{\mathbf{x}}_i$ by $\mathbf{y}_i = \mathbf{U}'^\top\bar{\mathbf{x}}_i$. In an online setting, we approximate each $\mathbf{U}_i^\top\bar{\mathbf{x}}_i$ by its previous solution $\mathbf{U}_{i-1}^\top\bar{\mathbf{x}}_i$ as follows

$$\min_{\mathbf{U}_t\in\mathbb{R}^{p\times k},\mathbf{U}^\top\mathbf{U}=\mathbf{I}} J_{ls}(\mathbf{U}_t) = \sum_{i=1}^{t} \|\bar{\mathbf{x}}_i - \mathbf{U}_i\mathbf{y}_i\|^2, \tag{18}$$

where $\mathbf{y}_i = \mathbf{U}_{i-1}^\top\bar{\mathbf{x}}_i$. This projection approximation provides a fast calculation of principle directions in our oversampling PCA. Linking this least squares form to our online oversampling strategy, we have

$$\min_{\tilde{\mathbf{U}}\in\mathbb{R}^{p\times k},\mathbf{U}^\top\mathbf{U}=\mathbf{I}} J_{ls}(\tilde{\mathbf{U}}) \approx \sum_{i=1}^{n} \|\bar{\mathbf{x}}_i - \tilde{\mathbf{U}}\mathbf{y}_i\|^2 + \|\bar{\mathbf{x}}_t - \tilde{\mathbf{U}}\mathbf{y}_t\|^2. \tag{19}$$

In (19), $\mathbf{y}_i$ and $\mathbf{y}_t$ are approximated by $\mathbf{U}^\top\bar{\mathbf{x}}_i$ and $\mathbf{U}^\top\bar{\mathbf{x}}_t$, respectively, where $\mathbf{U}$ is the solution of the original PCA (which can be calculated in advance), and $\bar{\mathbf{x}}_t$ is the target instance. When oversampling the target instance $\tilde{n}$ times, we approximate the solution $\tilde{\mathbf{U}}$ by solving the following optimization problem

$$\min_{\tilde{\mathbf{U}}\in\mathbb{R}^{p\times k},\mathbf{U}^\top\mathbf{U}=\mathbf{I}} J_{ls}(\tilde{\mathbf{U}}) \approx \sum_{i=1}^{n} \|\bar{\mathbf{x}}_i - \tilde{\mathbf{U}}\mathbf{y}_i\|^2 + \tilde{n}\|\bar{\mathbf{x}}_t - \tilde{\mathbf{U}}\mathbf{y}_t\|^2. \tag{20}$$

Equivalently, we convert the above problem into the following form

TABLE 1
Comparisons of Our Proposed osPCA (with Power Method and the Proposed Online Updating Technique), Fast ABOD, and LOF for Online Anomaly Detection in Terms of Computational Complexity and Memory Requirements

	osPCA [19] (power method)	Online osPCA	Fast ABOD [5]	LOF [2]
Computation complexity	$O(mp^2)$(or $O(mnp)$)	$O(p)$	$O(n^2p + k^2p)$	$O(n^2p + k)$
Memory requirement	$O(p^2)$(or $O(np)$)	$O(p)$	$O(np)$	$O(np)$

Note that n and p are the size and dimensionality of data, respectively. The power method requires the number of iterations m, and the number of nearest neighbors k is used in both ABOD and LOF.

$$\min_{\tilde{\mathbf{U}}\in\mathbb{R}^{p\times k},\mathbf{U}^\top\mathbf{U}=\mathbf{I}} J_{ls}(\tilde{\mathbf{U}}) \approx \beta\left(\sum_{i=1}^{n}\|\bar{\mathbf{x}}_i - \tilde{\mathbf{U}}\mathbf{y}_i\|^2\right) + \|\bar{\mathbf{x}}_t - \tilde{\mathbf{U}}\mathbf{y}_t\|^2, \tag{21}$$

where β can be regarded as a weighting factor to suppress the information from existing data. Note that the relation between β and the oversampled number $\tilde{n}$ is $\beta = \frac{1}{\tilde{n}} = \frac{1}{nr}$, where r is the ratio of the oversampled number over the size of the original data set. To improve the convergence rate, we use the solution of the original PCA (without oversampling data) as the initial projection matrix in (21). If only the dominant principal direction (i.e., $\mathrm{k} = 1$) is of concern, we calculate the solution of $\tilde{\mathbf{u}}$ by taking the derivative of (21) with respect to $\tilde{\mathbf{u}}$, and thus we have

$$\tilde{\mathbf{u}} = \frac{\beta(\sum_{i=1}^{n} y_i\bar{\mathbf{x}}_i) + y_t\bar{\mathbf{x}}_t}{\beta(\sum_{i=1}^{n} y_i^2) + y_t^2}. \tag{22}$$

Compared with (10) and (15), (22) provides an effective and efficient updating technique for osPCA, which allows us to determine the principal direction of the data. This updating process makes anomaly detection in online or streaming data settings feasible. More importantly, since we only need to calculate the solution of the original PCA offline, we do *not* need to keep the entire covariance or outer matrix in the entire updating process. Once the final principal direction is determined, we use the cosine similarity to determine the difference between the current solution and the original one (without oversampling), and thus the score of outlierness for the target instance can be determined accordingly (as discussed in Section 3.2). The pseudocode of our *online osPCA* with the LOO strategy for outlier detection is described in Algorithm 1. It is worth noting that we only need to compute $\mathbf{x}_{proj}$ and y once in Algorithm 1, and thus we can further reduce the computation time when calculating $\tilde{\mathbf{u}}$.

Algorithm 1. Anomaly Detection via Online Oversampling PCA

Require: The data matrix $\mathbf{A} = [\mathbf{x}_1^\top; \mathbf{x}_2^\top; \ldots; \mathbf{x}_n^\top]$ and the weight β.

Ensure: Score of outlierness $\mathbf{s} = [s_1 s_2 \ldots s_n]$. If s_i is higher than a threshold, $\mathbf{x}_i$ is an outlier.

Compute first principal direction $\mathbf{u}$ by using (18);
Keep $\bar{\mathbf{x}}_{proj} = \sum_{j=1}^{n} y_j\bar{\mathbf{x}}_j$ and $y = \sum_{j=1}^{n} y_j^2$ in (22);
for $i = 1$ **to** n **do**
  $\tilde{\mathbf{u}} \leftarrow \frac{\beta\bar{\mathbf{x}}_{proj} + y_i\bar{\mathbf{x}}_i}{\beta y + y_i^2}$ by (18)
  $s_i \leftarrow 1 - |\frac{\langle\tilde{\mathbf{w}},\mathbf{w}\rangle}{\|\tilde{\mathbf{u}}\|\|\mathbf{u}\|}|$ by (7)
end for

Table 1 compares computation complexity and memory requirements of several anomaly detection methods, including fast ABOD [5], LOF [2], our previous osPCA using power method [19], and the proposed online osPCA. In this table, we list computation and memory costs of each method when determining the outlierness of a newly received data instance (i.e., in a streaming data fashion). For ABOD and LOF, the memory requirements are both $O(np)$ since they need to store the entire data matrix for the k nearest neighbor search (recall that n and p are the size and dimensionality of the data, respectively). The time complexities for ABOD and LOF are $O(n^2p + k^2p)$ and $O(n^2p + k)$, in which $O(n^2p)$ is required for finding k nearest neighbors and thus is the bottleneck of the computation complexity.

As for the power method, it needs to perform (15) iteratively with m times, its time complexity in the online detection procedure for outlier detection is $O(np^2 + mp^2)$ (we have $O(np^2)$ for deriving the updated covariance matrix, and $O(mp^2)$ for the implementation of the power method). Practically, we reduce the above complexity to $O(mp^2)$ by applying the covariance update trick in (14). As discussed in Section 4.3, the time complexity might be $O(mnp)$ if we choose to store the data matrix instead of keeping the covariance matrix during the updating process. As a result, the associated memory requirement will be reduced from $O(p^2)$ to $O(np)$. Finally, when using our online updating technique for osPCA, we simply update the principal direction by (22) and result in $O(p)$ for both computation complexity and memory requirement, respectively.

5 Experimental Results

5.1 Anomaly Detection on Synthetic and Real-World Data

5.1.1 Two-Dimensional Synthetic Data Set

To verify the feasibility of our proposed algorithm, we conduct experiments on both synthetic and real data sets. We first generate a two-dimensional synthetic data, which consists of 190 normal instances (shown in blue dots in Fig. 3a) and 10 deviated instances (red stars in Fig. 3a). The normal data points are sampled from the following multivariate normal distribution

$$\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{23}$$

where

$$[\boldsymbol{\mu}] = \begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix} \text{ and } \boldsymbol{\Sigma} = \begin{bmatrix}5 & 2\\ 2 & 1\end{bmatrix}.$$

We note that each deviated data point is sampled from a different multivariate normal distribution $N(\boldsymbol{\mu}_d, \Sigma)$, where $\boldsymbol{\mu}_d = [\mu_{d1}, \mu_{d2}]^\top$, μ_{d1} and μ_{d2} are randomly sampled from the

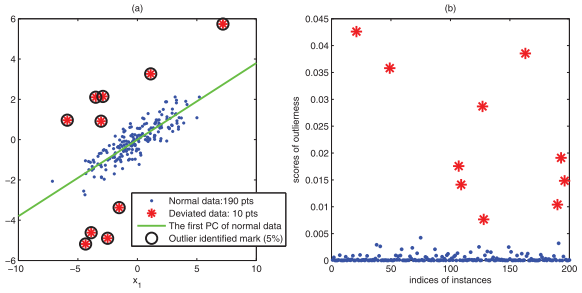


Fig. 3. Outlier detection results with the two-dimensional synthetic data.

range $[-5, 5]$. We apply our online osPCA algorithm on the entire data set, and rank the score of outlierness (i.e., s_t in Section 3.2) accordingly. We aim to identify the top 5 percent of the data as deviated data, since this number is consistent with the number of outliers we generated. The scores of outlierness of all 200 data points are shown in Fig. 3b. It is clear that the scores of the deviated data (shown in red) are clearly different from those of normal data, and thus all outliers are detected by setting a proper threshold to filter the top 5 percent of the data. Note that we mark the filtered data points with black circles in Fig. 3a. This initial result on a simple synthetic data set shows the effectiveness of our proposed algorithm.

5.1.2 Real-World Data Sets

Next, we evaluate the proposed method on six real data sets. The detailed information for each data set is presented in Table 2. The `pendigits` , `pima`, and `adult` data sets are from the UCI repository of machine learning data archive [25]. The `splice` and `cod-rna` are available at http://www.csie.ntu.edu.tw/cjlin/libsvmtools/data sets/, and the KDD intrusion detection data set is available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. To compare the anomaly detection results of our proposed method with other methods, we implement decremental PCA with the LOO strategy, osPCA with power method [19], LOF [2] and fast ABOD [5]. We use the area under the ROC curve (AUC) [26] to evaluate the detection performance (i.e., the larger the AUC value, the better the detection performance). For the `pendigits` data set, we consider the digit "0" as the normal data instances (a total of 780 instances) and use other digits "1" to "9" (20 data samples randomly chosen for each category) as the outliers to be detected. For other data sets for binary classification in Table 2, we consider the data from the majority class as normal data and randomly select 1 percent data instances from the minority class as outlier samples. In all our experiments, we repeat the procedure with 5 random trials. We present the average AUC score and runtime estimates for each data set, as shown in Tables 3, 4, and 5. We note that, for osPCA with power method or our online updating technique, we vary the oversampling ratio r for the target instance from 0.1 to 0.2 and report the best performance; for LOF and fast ABOD, we choose the parameter k (number of nearest neighbors) which produces the best performances for fair comparisons.

From these three tables, we observe that our proposed online osPCA consistently achieved better or comparable results, while ours is the most computationally efficient one among the methods considered. By comparing the first and the second (or third) columns in Tables 3 and 4, it is interesting to note that the AUC score of osPCA is

TABLE 2
Description of Data Sets

Data set	Size	Attributes	Classes
`pima`	768	8	2
`splice`	1000	60	2
`pnedigits`	7494	16	10
`adult`	48842	123	2
`cod-rna`	59535	8	2
`kdd_tcp`	190065	38	5

TABLE 3
AUC Scores of Decremental PCA (dPCA), Oversampling PCA (osPCA) with Power Method, Our osPCA with Online Updating Algorithm, Fast ABOD, and LOF on the `pendigits` data set

Scenario	dPCA	osPCA [19]	Online osPCA	Fast ABOD [5]	LOF [2]
`0 vs. 1`	0.9145 ± 0.0385	0.9965 ± 0.0004	0.9869 ± 0.0104	0.9519 ± 0.0287	0.9943 ± 0.0007
`0 vs. 2`	0.9573 ± 0.0317	0.9959 ± 0.0003	0.9879 ± 0.0225	0.9214 ± 0.0279	0.9966 ± 0.0002
`0 vs. 3`	0.4570 ± 0.0554	0.9987 ± 0.0003	0.9199 ± 0.0453	0.9342 ± 0.0157	0.9970 ± 0.0002
`0 vs. 4`	0.7392 ± 0.0686	0.9897 ± 0.0016	0.8442 ± 0.0582	0.9737 ± 0.0069	0.9859 ± 0.0017
`0 vs. 5`	0.8126 ± 0.0485	0.9961 ± 0.0005	0.9623 ± 0.0260	0.9721 ± 0.0086	0.9980 ± 0.0003
`0 vs. 6`	0.9773 ± 0.0077	0.9793 ± 0.0015	0.9851 ± 0.0176	0.9447 ± 0.0196	0.9741 ± 0.0028
`0 vs. 7`	0.8387 ± 0.0439	0.9968 ± 0.0003	0.9800 ± 0.0305	0.9642 ± 0.0087	0.9968 ± 0.0004
`0 vs. 8`	0.8519 ± 0.0476	0.9816 ± 0.0172	0.9245 ± 0.0395	0.9913 ± 0.0019	0.9939 ± 0.0016
`0 vs. 9`	0.6914 ± 0.0635	0.9968 ± 0.0008	0.9776 ± 0.0290	0.9901 ± 0.0025	0.9945 ± 0.0006
`0 vs. {1-9}`	0.7847 ± 0.0498	0.9933 ± 0.0016	0.9731 ± 0.0189	0.9953 ± 0.0005	0.9947 ± 0.0021

TABLE 4
AUC Scores of dPCA, osPCA with Power Method, Our osPCA with Online Updating Algorithm, Fast ABOD, and LOF on `pima`, `splice`, `adult`, and `cod-rna` Data Sets

Data set	dPCA	osPCA [19]	Online osPCA	Fast ABOD [5]	LOF [2]
`pima`	0.7024 ± 0.1553	0.7090 ± 0.1373	0.7086 ± 0.1429	0.4140 ± 0.1613	0.6487 ± 0.1665
`splice`	0.4671 ± 0.1376	0.5196 ± 0.1427	0.7319 ± 0.0961	0.5743 ± 0.0802	0.4374 ± 0.0795
`adult`	0.6391 ± 0.0164	0.6652 ± 0.0212	0.6867 ± 0.0195	0.3121 ± 0.0154	0.5833 ± 0.0114
`cod-rna`	0.8300 ± 0.0137	0.8312 ± 0.0116	0.8309 ± 0.0118	0.3447 ± 0.0298	0.8498 ± 0.0104

TABLE 5
Average CPU Time (in seconds) for Anomaly Detection Using dPCA, osPCA with Power Method, Our osPCA with Online Updating Algorithm, Fast ABOD, and LOF

Time (sec.)	dPCA	osPCA [19]	Online osPCA	Fast ABOD [5]	LOF [2]
pima	0.027	0.131	0.017	1.237	0.076
splice	0.414	0.185	0.020	120.4	0.085
pendigit	0.059	0.089	0.012	120.3	0.079
adult	16.72	56.47	1.548	$>$ 1.0E+4	679.6
cod-rna	1.204	14.82	1.336	$>$ 1.0E+3	21.32

TABLE 6
Outlier Detection Results on the KDD Intrusion Detection Data Set

Types & sizes of outliers	osPCA [19]		Online osPCA		LOF [2]	
	AUC	Time (sec.)	AUC	Time (sec.)	AUC	Time (sec.)
dos (50)	0.9859 $\pm$ 0.0023	33.82	0.9858 $\pm$ 0.0036	2.698	0.9954 $\pm$ 0.0003	90.03
probe (50)	0.9954 $\pm$ 0.0003	27.64	0.9987 $\pm$ 0.0007	2.697	0.9886 $\pm$ 0.0016	90.98
r2l (50)	0.9581 $\pm$ 0.0090	33.70	0.9417 $\pm$ 0.0060	2.695	0.8990 $\pm$ 0.0131	89.28
u2r (49)	0.9842	33.67	0.9712	2.695	0.9914	91.70
all attacks (60)	0.9799 $\pm$ 0.0032	33.77	0.9584 $\pm$ 0.0071	2.734	0.9765 $\pm$ 0.0060	98.96

Note that we did not report the standard deviation of AUC score for the u2r case because the total number of instances in u2r attacks is 49, which is not enough to perform five random trials.

significantly better than that of dPCA (without oversampling strategy). This confirms that the oversampling strategy indeed increases the outlierness of rare but abnormal data, and thus this strategy makes anomaly detection in large-scale problems easier. Comparing the second and the third columns, we note that the performance of our proposed online osPCA is comparable to that of osPCA with power method. This observation is consistent with our discussion in Section 4 that using the proposed online approximation technique, our online osPCA is able to produce the approximated version of the principal direction without sacrificing computation and memory requirements.

For the KDD intrusion detection data set, there are four categories of attacks to be considered as outliers:

- DOS: denial-of-service.
- R2L: unauthorized access from a remote machine.
- U2R: unauthorized access to local superuser (root) privileges.
- Probe: surveillance and other probing.

We use binary and continuous features (38 features) and focus on the 10 percent training subset under the tcp protocol. The size of normal data is 76,813. In this experiment, data points from four different attacks are considered as outliers. Table 6 shows detection performance (in terms of AUC) and the numbers of test samples of each attack category. Only LOF is used for comparison, since it is shown to outperform the ABOD method in Tables 3 and 4. From this table, we see that our online osPCA again achieved comparable performance with LOF, while the LOF required significant longer computation time. Nevertheless, the effectiveness of our online osPCA is verified by the experiments conducted in this section, and it is clear that our approach is the most computationally efficient one among the methods we considered for comparison.

5.2 Online Anomaly Detection For Practical Scenario

For online anomaly detection applications such as spam mail filtering, one typically designs an initial classifier using the training normal data, and this classifier is updated by the newly received normal or outlier data accordingly. However, in practical scenarios, even the training normal data collected in advance can be contaminated by noise or incorrect data labeling. To construct a simple yet effective model for online detection, one should disregard these potentially deviated data instances from the training set of normal data (it is not practical to collect training outlier data anyway). The flowchart of our online detection procedure is shown in Fig. 4. As can be seen in Fig. 4, there are two phases required in this framework: Data cleaning and online detection. In the data cleaning phase, our goal is to filter out the most deviated data using our osPCA before performing online anomaly detection. This data cleaning phase is done offline, and the percentage of the training normal data to be disregarded can be determined by the user. In our implementation, we choose to disregard 5 percent of the training normal data after this data cleaning process, and we use the smallest score of outlierness (i.e., s_t) of the remaining training data instances as the threshold for outlier detection. More specifically, in the second phase of online detection, we use this threshold to determine the anomaly of each received data point. If s_t of a newly received data instance is above the threshold, it will be identified as an outlier; otherwise, it will be considered as a normal data point, and we will update our osPCA model accordingly.

In the online detection phase, we use the dominant principal direction of the filtered training normal data

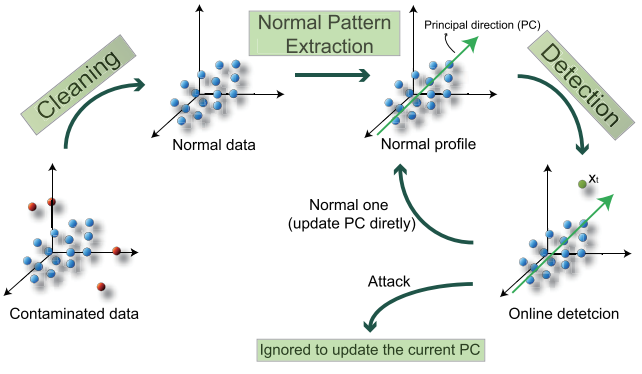


Fig. 4. The framework of our online anomaly detection.

TABLE 7
Online Anomaly Detection Results on the KDD Intrusion Detection Data Set

	TP Rate	FP Rate	Time (sec.)
osPCA [19]	0.9183± 0.0223	0.0427 ± 0.0054	≈1.0E-1
Online osPCA	0.9133± 0.0327	0.0697 ± 0.0188	<1.0E-4

Note that TP and FP indicate true and false positive rates, respectively. The runtime estimate reports the testing time in determining the anomaly of a newly received target instance.

extracted in the data cleaning phase to detect each arriving target instance. We note that, during the entire online detection phase, we only need to keep this p-dimensional eigenvector, and thus the memory requirement is only $O(p)$. To determine the outlierness of a newly received instance, we apply the osPCA with the proposed online updating technique to evaluate the variation of the updated principal direction (as discussed in Section 4.3). If the resulting s_t in (7) is above the threshold determined previously, the target instance will be detected as an outlier; otherwise, we will consider this input as a normal data instance and update the principal direction accordingly (also in the same online updating fashion).

We now our proposed osPCA for online anomaly detection using the KDD data set. We first extract 2,000 normal instances points from the data set for training. In the data cleaning phase, we filter the top 5 percent (100 points) to avoid noisy training data or those with incorrect class labels. Next, we extract the dominant principal direction using our online osPCA, and we use this principal direction to calculate the score of outlierness of each receiving test input. In this online testing phase, the numbers of normal data and attacks from each category are 2,000 and 25 (i.e., 100 attacks in total), respectively. Since the threshold for anomaly detection is determined by the data cleaning phase, we use threshold and report the true and false positive rates on the receiving test data instances.

To verify the effectiveness of online osPCA with the proposed online updating technique in such online detection problems, we compared our online osPCA with osPCA with power method (i.e., the osPCA which receives a target instance will recalculate the principal direction, and use the above threshold to perform anomaly detection). We note that, we do expect that the latter case will provide marginally better detection results, since it will store the entire data and update the mean accordingly. However, the latter would require higher computational and memory costs, and the online version of the proposed osPCA is much more efficient in terms of both requirements.

Table 7 lists the performance of online anomaly detection using our previous osPCA with power method [19] and the proposed online osPCA. We see that at (about) the same true positive rate at 91.3 percent, our online osPCA achieved a slightly larger false positive rate at 6.9 percent (versus 4.2 percent); moreover, the proposed online osPCA performs at least 1,000 times faster than that using power method. This again confirms that our online osPCA approximates the solution of osPCA well while significantly reducing the computation time. Therefore, our proposed method is preferable to online anomaly detection problems. It is worth noting that we do not explicitly compare our online osPCA to the incremental version of LOF [17], since Section 5.2 already verifies that the osPCA indeed achieves

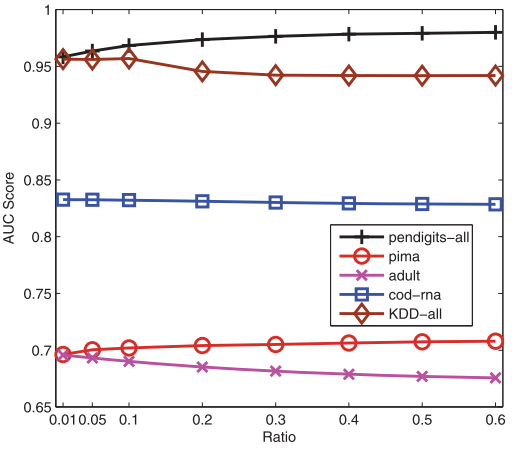


Fig. 5. Effects of the oversampling ratio r on the AUC performance for different data sets.

better or comparable performance with significantly less computation time than methods like standard LOF and fast ABOD. While the incremental LOF can at most produce comparable performances as the standard LOF does, LOF-based methods have been shown to be less efficient than osPCA in prior discussions. Finally, we conclude that the above observations not only confirm the effectiveness and efficiency of our online osPCA, they also imply that our approach outperforms incremental versions of LOF or ABOD methods in terms of both detection performance and computation time.

5.3 Sensitivity Tests of the Oversampling Ratio

In the above experiments, we varied the oversampling ratio r between 0.1 and 0.2 for the target instance, and the best results with the smallest r were presented. To verify that this choice will not significantly affect the performance of anomaly detection, we perform additional experiments with different r values in the range between 0.01 and 0.6 for each data set. We present the AUC values with different ratios for all data sets in Fig. 5 for comparisons. From this figure, we see that the AUC results were comparable for different oversampling ratios for most cases. From these observations, we do not find that the choice of r will remarkably affect the use of osPCA for anomaly detection. Therefore, a smaller r (e.g., r between 0.1 and 0.2 as suggested) is typically preferable due to its computational efficiency.

6 Conclusion

In this paper, we proposed an online anomaly detection method based on oversample PCA. We showed that the osPCA with LOO strategy will amplify the effect of outliers, and thus we can successfully use the variation of the dominant principal direction to identify the presence of rare but abnormal data. When oversampling a data instance, our proposed online updating technique enables the osPCA to efficiently update the principal direction without solving eigenvalue decomposition problems. Furthermore, our method does not need to keep the entire covariance or data matrices during the online detection process. Therefore, compared with other anomaly detection methods, our approach is able to achieve satisfactory results while significantly reducing computational costs and memory

requirements. Thus, our online osPCA is preferable for online large-scale or streaming data problems.

Future research will be directed to the following anomaly detection scenarios: normal data with multiclustering structure, and data in a extremely high dimensional space. For the former case, it is typically not easy to use linear models such as PCA to estimate the data distribution if there exists multiple data clusters. Moreover, many learning algorithms encounter the "curse of dimensionality" problem in a extremely high-dimensional space. In our proposed method, although we are able to handle high-dimensional data since we do not need to compute or to keep the covariance matrix, PCA might not be preferable in estimating the principal directions for such kind of data. Therefore, we will pursue the study of these issues in our future work.

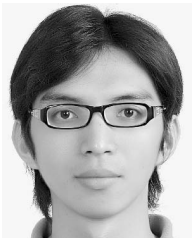
ACKNOWLEDGMENTS

This work is supported in part by National Science Council of Taiwan via NSC 100-2218-E-011-007, NSC 100-2218-E-011-008, and NSC 100-2221-E-001-018-MY2.

REFERENCES

[1] D.M. Hawkins, *Identification of Outliers.* Chapman and Hall, 1980.
[2] M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 2000.
[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys,* vol. 41, no. 3, pp. 15:1-15:58, 2009.
[4] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A.D. Joseph, and N. Taft, "In-Network Pca and Anomaly Detection," *Proc. Advances in Neural Information Processing Systems 19,* 2007.
[5] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-Based Outlier Detection in High-Dimensional Data," *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and data Mining,* 2008.
[6] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," *Proc. Third SIAM Int'l Conf. Data Mining,* 2003.
[7] X. Song, M. Wu, and C.J., and S. Ranka, "Conditional Anomaly Detection," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 5, pp. 631-645, May 2007.
[8] S. Rawat, A.K. Pujari, and V.P. Gulati, "On the Use of Singular Value Decomposition for a Fast Intrusion Detection System," *Electronic Notes in Theoretical Computer Science,* vol. 142, no. 3, pp. 215-228, 2006.
[9] W. Wang, X. Guan, and X. Zhang, "A Novel Intrusion Detection Method Based on Principal Component Analysis in Computer Security," *Proc. Int'l Symp. Neural Networks,* 2004.
[10] F. Angiulli, S. Basta, and C. Pizzuti, "Distance-Based Detection and Prediction of Outliers," *IEEE Trans. Knowledge and Data Eng.,* vol. 18, no. 2, pp. 145-160, 2006.
[11] V. Barnett and T. Lewis, *Outliers in Statistical Data.* John Wiley&Sons, 1994.
[12] W. Jin, A.K.H. Tung, J. Han, and W. Wang, "Ranking Outliers Using Symmetric Neighborhood Relationship," *Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining,* 2006.
[13] N.L.D. Khoa and S. Chawla, "Robust Outlier Detection Using Commute Time and Eigenspace Embedding," *Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining,* 2010.
[14] E.M. Knox and R.T. Ng, "Algorithms for Mining Distance-Based Outliers in Large Data Sets," *Proc. Int'l Conf. Very Large Data Bases,* 1998.
[15] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data," *Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining,* 2009.
[16] C.C. Aggarwal and P.S. Yu, "Outlier Detection for High Dimensional Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 2001.
[17] D. Pokrajac, A. Lazarevic, and L. Latecki, "Incremental Local Outlier Detection for Data Streams," *Proc. IEEE Symp. Computational Intelligence and Data Mining,* 2007.
[18] T. Ahmed, "Online Anomaly Detection using KDE," *Proc. IEEE Conf. Global Telecomm.,* 2009.
[19] Y.-R. Yeh, Z.-Y. Lee, and Y.-J. Lee, "Anomaly Detection via Oversampling Principal Component Analysis," *Proc. First KES Int'l Symp. Intelligent Decision Technologies,* pp. 449-458, 2009,
[20] G.H. Golub and C.F.V. Loan, *Matrix Computations.* Johns Hopkins Univ. Press, 1983.
[21] R. Sibson, "Studies in the Robustness of Multidimensional Scaling: Perturbational Analysis of Classical Scaling," *J. Royal Statistical Soc. B,* vol. 41, pp. 217-229, 1979.
[22] B. Yang, "Projection Approximation Subspace Tracking," *IEEE Trans. Signal Processing,* vol. 43, no. 1, pp. 95-107, Jan. 1995.
[23] S. Papadimitriou, J. Sun, and C. Faloutsos, "Streaming Pattern Discovery in Multiple Time-Series," *Proc. 31st Int'l Conf. Very Large Data Bases,* 2005.
[24] S. Haykin, *Adaptive Filter Theory.* Prentice Hall, 1991.
[25] A. Asuncion and D. Newman, "UCI Repository of Machine Learning Databases," http://www.ics.uci.edu/mlearn/mlrepository.html, 2007.
[26] A.P. Bradley, "The Use of the Area under the Roc Curve in the Evaluation of Machine Learning Algorithms," *Pattern Recognition,* vol. 30, pp. 1145-1159, 1997.

Yuh-Jye Lee received the master's degree in applied mathematics from the National Tsing Hua University, Taiwan, in 1992, and the PhD degree in computer sciences from the University of Wisconsin-Madison in 2001. He is an associate professor in the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology. His research interests include machine learning, data mining, optimization, information security, and operations research. He developed new algorithms for large data mining problems such as classification and regression problem, abnormal detection, and dimension reduction. Using the methodologies such as support vector machines, reduced kernel method, chunking and smoothing techniques allows us to get a very robust solution (prediction) for a large data set. These methods have been applied to solve many real-world problems such as intrusion detection system, face detection, microarray gene expression analysis, and breast cancer diagnosis and prognosis.



Yi-Ren Yeh received the MS and PhD degrees from the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taiwan, in 2006 and 2010, respectively. From August 2008 to May 2009, he was a visiting scholar in CyLab, Carnegie Mellon University, Pittsburgh. His research interests include machine learning, data mining, optimization, numerical methods, and pattern recognition. He is currently a postdoctoral research fellow in the Research Center for Information Technology Innovation at Academia Sinica, Taipei, Taiwan.

Yu-Chiang Frank Wang received the BS degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2001 and the MS and PhD degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, in 2004 and 2009, respectively. In 2009, he joined the Research Center for Information Technology Innovation (CITI) at Academia Sinica, Taiwan, where he holds the position of a tenure-track assistant research fellow. He leads the Multimedia and Machine Learning Lab at CITI, and works in the fields of pattern recognition, computer vision, and machine learning. From July to December 2010, he was a visiting scholar in the Department of Computer Science and Information Engineering at National Taiwan University Science and Technology, Taiwan. He is a member of the IEEE.