Data Min Knowl Disc
DOI 10.1007/s10618-013-0334-x

Ensemble anomaly detection from multi-resolution trajectory features

Shin Ando · Theerasak Thanomphongphan · Yoichi Seki · Einoshin Suzuki

Received: 15 April 2012 / Accepted: 11 July 2013

Abstract The numerical, sequential observation of behaviors, such as trajectories, have become an important subject for data mining and knowledge discovery research. Processing the raw observation into representative features of the behaviors involves an implicit choice of time-scale and resolution, which critically affect the final output of the mining techniques. The choice is associated with the parameters of data-processing, e.g., smoothing and segmentation, which unintuitively yet strongly influence the intrinsic structure of the numerical data. Data mining techniques generally require users to provide an appropriately processed input, but selecting a resolution is an arduous task that may require an expensive, manual examination of outputs between different settings. In this paper, we propose a novel ensemble framework for aggregating outcomes

Responsible editor: M.J. Zaki.

Electronic supplementary material The online version of this article (doi:10.1007/s10618-013-0334-x) contains supplementary material, which is available to authorized users.

This paper extends work previously published as Ando et al. (2011).

S. Ando (✉) · Y. Seki
Division of Electronics and Informatics, Gunma University, 1-5-1 Tenjincho, Kiryu, Gunma, Japan
e-mail: shinando@gunma-u.ac.jp

Y. Seki
e-mail: seki@gunma-u.ac.jp

T. Thanomphongphan
Panasonic AVC Networks (Thailand) Co., Ltd., Bangsaothong, Samutprakarn, Thailand
e-mail: nices_jing@hotmail.com

E. Suzuki
Department of Informatics, ISEE, Kyushu University, Fukuoka, Japan
e-mail: suzuki@inf.kyushu-u.ac.jp

Published online: 01 August 2013

in different settings of scale and resolution parameters for an anomaly detection task. Such a task is difficult for existing ensemble approaches based on weighted combination because: (a) evaluating and weighing an output requires training samples of anomalies which are generally unavailable, (b) the detectability of anomalies can depend on the resolution, i.e., the distinction from normal instances may only be apparent within a small, selective range of parameters. In the proposed framework, predictions based on different resolutions are aggregated to construct *meta*-feature representations of the behavior instances. The *meta*-features provide the discriminative information for conducting a clustering-based anomaly detection. In the proposed framework, two interrelated tasks of the behavior analysis: processing the numerical data and discovering anomalous patterns, are addressed jointly, providing an intuitive alternative for a knowledge-intensive parameter selection. We also design an efficient clustering-based anomaly detection algorithm which reduces the computational burden of mining at multiple resolutions. We conduct an empirical study of the proposed framework using real-world trajectory data. It shows that the proposed framework achieves a significant improvement over the conventional ensemble approach.

Keywords Behavioral data mining · Trajectory data mining · Multi-resolution features · Ensemble anomaly detection

1 Introduction

In recent years, behavioral data have attracted interests in the data mining research as advances in surveillance and sensor technologies have come to provide the platform to track or monitor people and objects over a long duration. Due to the temporal and sequential nature of the data, many temporal data mining techniques are useful for discovering behavioral knowledge (Bonchi et al. 2009; Gaffney and Smyth 1999; Giannotti et al. 2007; Jiang et al. 2012; Pelekis et al. 2011; Wang et al. 2011) Various features based on trajectories have been proposed for the discriminative analyses of behaviors (Ando et al. 2011; Anjum and Cavallaro 2008; Khalid and Naftel 2005; Piciarelli and Foresti 2006, 2007; Piciarelli et al. 2008; Porikli and Haga 2004). Exploratory techniques such as clustering (Jain et al. 1999; Xu and Wunsch 2005) and anomaly detection (Chandola et al. 2009; Markou and Singh 2003a,b), which have various applications in time series and sequential data domains (Budhaditya et al. 2009; Liu et al. 2008; Xiong and Yeung 2002), have also been applied to trajectory data (Anjum and Cavallaro 2008; Johnson and Hogg 1995; Piciarelli and Foresti 2006; Piciarelli et al. 2008; Yankov et al. 2008).

When addressing temporal data, usually an implicit assumption is made regarding the time-scale and the resolution of the relevant events or patterns. For example, the moving average is often used to remove high frequency components from the time series and expose long-term patterns and behaviors (Roddick and Spiliopoulou 2002). The data are *smoothed* by replacing each value with the mean value over a local interval. The size of the interval naturally determines the resolution with which the processed data can be examined. The sliding window is another widely-used method for generating a vector representation from a time series, by extracting all subsequences

of a fixed size. Each subsequence is treated as a feature vector of an event or a behavior in that time window. The size of the window also determines the time-scale with which instances are represented and modeled or compared in subsequent techniques.

In unsupervised learning tasks such as anomaly detection, choosing an appropriate setting of the resolution with regards to the processing methods is difficult, since the patterns that correspond to interesting events and behaviors are unknown until the data have been processed and analyzed. Yet, these choices can affect the output of data mining significantly. In the following, we further elaborate on the motivation of our work using a concrete example.

In Kumar et al. (2010), a multi-agent task experiment is described where autonomous agents are implemented to explore an enclosed environment. The controller program was developed through trial and error, during which an unexpected behavior of an agent was noticed by the designer of the controller in the video log. At this point, we focus on the numerical representation of the behavior, and its detail is deferred to a later section. The velocity of the agents, defined as the displacement per unit time interval, is one of the essential information related to its behavior. The length of the time interval, which we denote by Δt, determines the resolution, i.e., the smoothness of the velocity sequence.

Figure 1 illustrates the time series of the Euclidean norm of the velocity during a normal behavior and the unexpected behavior. In each plot, x- and y-axes indicate the time and the norm of the velocity vector, respectively. The length of the interval is shown on the left of each row after Δt. The figures in the left column illustrate the subsequences of the velocity norm extracted from the same segment of the trajectory with different intervals. The diversity of the patterns indicate the impact of the parameter selection. Similarly, the right column shows the subsequence of velocity norm during an anomalous behavior with the same set of intervals. Based on the graphics in Fig. 1, the most preferable interval for detecting the anomalous pattern is that of the middle row, since it yields the most significant distinction between the two patterns. At a smaller interval shown on the top row, patterns are difficult to distinguish due to the jaggedness from the noise. Meanwhile, at a larger interval the patterns are smoother but less distinct as shown on the bottom row.

As illustrated in our example, the choice of the resolution such as Δt, strongly influences the intrinsic structure, and the outcome of the subsequent data mining. Yet, the choice is generally outside the methodology of data mining, and the user is

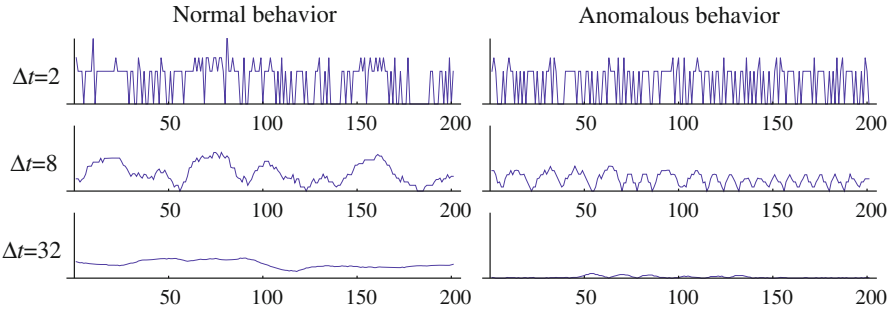


Fig. 1 Velocity norm sequences of normal and anomalous behaviors with different intervals

asked to provide appropriately processed data. The acceptable range of resolutions is naturally problem-dependent, and the parameter values should be chosen case-by-case. Adjusting the combination of values for multiple parameters by a graphical examination is a significant undertaking and also requires explicit examples of the anomalies.

In this paper, we address the above issue by mining from the behavioral features of multiple resolutions. For given observations of behaviors, we apply data-processing with a *grid* of parameter value combinations to generate representations in different resolutions. We perform anomaly detection using each representation, and the results are aggregated by an ensemble learning technique.

The ensemble learning have been studied extensively for supervised learning tasks (Dietterich 2000; Rokach 2010) and recently have drawn interest for clustering (Ailon et al. 2008; Gionis et al. 2007; Strehl and Ghosh 2003) and outlier detection (Lazarevic and Kumar 2005; Nguyen et al. 2010) as well. However, aggregating the results of anomaly detection from multiple resolutions is not a straightforward task with the conventional approaches. First, there is no means to evaluate and weigh the performances of anomaly detection at different resolutions because examples of anomalies are usually unavailable. Secondly, an anomaly detection method may be subject to a different bias in each resolution and inconsistent performances over multiple resolutions. For example, short-term anomalous behaviors may be overlooked at a low-resolution feature or a high-resolution feature may cause frequent false positive alarms for normal behaviors. It is also probable that the anomalous patterns are observable only in few specific resolutions, i.e., within a small range of parameters, thus some of or even many of the processing parameters may be chosen poorly. An ensemble method for anomaly detection is required to address such cases without the means to evaluate the performance.

Our intuition for constructing an ensemble of anomaly detection is to exploit the unevaluated predictions from different resolutions as the *meta*-features of the instances. We assume that the bias associated with each resolution, i.e., the influence on making certain distinctions easier or harder to recognize, is consistent. A series of predictions in different resolutions can then be seen as an orientation that characterizes an instance. Anomalous patterns associated with a small range of parameters exhibit different orientations in the *meta*-feature space, and are effectively partitioned by cluster analysis. We implement the above intuition as an ensemble framework called anomaly clustering ensemble (ACE). ACE takes a set of trajectories and combinations of data-processing parameters as its input. The output is the score for each temporal instance, and probable anomalous behaviors are indicated by their higher values.

The contribution of this paper is summarized as follows. First, we propose a framework of multi-resolution data mining to simultaneously address two interdependent tasks: processing the temporal data and discovering anomalous temporal patterns, and provide an intuitive alternative to knowledge-intensive parameter selection. Secondly, we present an effective *meta*-feature representation for ensemble anomaly detection. Finally, we present a clustering-based anomaly detection method which efficiently approximates the relative density between two datasets, which reduces the computational burden of performing data mining in multiple resolutions. We conduct an empirical study to evaluate the ACE framework using real-world trajectory data, for

which the appropriate settings of data processing were unknown. We also demonstrate the capability of ACE in the time series domain using a collection of standard benchmark data.

The rest of this paper is organized as follows. We review the related work in Section 2. Section 3 describes the formalization of the problem at hand. Section 4 introduces the overview of the proposed framework and describes its individual components. Section 5 presents the algorithmic implementation and the analysis of its complexity. Section 6 describes an offline parameter selection scheme. Section 7 describes the empirical and comparative evaluation of the proposed framework using real-world agent trajectories. In Sect. 8, we extend our empirical study using a collection of time series benchmark datasets. Section 9 discusses the results and the related issues. We present our conclusion in Sect. 10. Regarding the general notations, sets and functions are denoted by upper-case letters, while matrices are denoted by bold upper-case. The cardinality of a set is denoted as $\#(\cdot)$.

2 Related work

2.1 Trajectory and behavioral data analyses

The analysis of trajectories has been an important topic for robotics, computer vision, and pervasive computing (Kröger 2010; Morris and Trivedi 2008; Yang 2009; Zheng and Zhou 2011). Recently, the subject has attracted significant interest in data mining over a large variety of subjects and tasks. The examples of the subjects include: body gestures (Gaffney and Smyth 1999), local- and wide-scale movement of people (Bu et al. 2009; Rosswog and Ghose 2012) and vehicles (Bonchi et al. 2009; Han et al. 2008). The examples of the tasks are: clustering (Jiang et al. 2012), frequent patterns (Giannotti et al. 2007), event detection (Wang et al. 2011), and scalable application to large databases Pelekis et al. (2011).

Representation of the behaviors is one of the key techniques in the analysis and various features of the trajectories have been proposed. For example, in Khalid and Naftel (2005), feature vectors are extracted from the trajectory using Discrete Fourier Transform and polynomial function approximation. Spacial coordinates are directly used as features in Piciarelli and Foresti (2006), Piciarelli and Foresti (2007), Piciarelli et al. (2008). Subsequences of velocity norm are used in Ando et al. (2011), and the model parameters of Hidden Markov Model is used in Porikli and Haga (2004). In Anjum and Cavallaro (2008), the combination of the position, the velocity, the acceleration, the direction, the turn, and the principle component vectors is employed. The moving average and the sliding window are commonly used in feature extraction and recognized as a critical aspect in many studies (Roddick and Spiliopoulou 2002). Typically, their parameters are chosen empirically.

Clustering and anomaly detection are extensively studied applications of trajectory and behavior analyses. Cluster analysis can effectively uncover the intrinsic structure and generate a summary of the data (Morris and Trivedi 2008, 2009). Anomaly detection is the core task in many practical applications (Piciarelli and Foresti 2006; Piciarelli et al. 2008; Suzuki et al. 2007). Furthermore, two techniques are often used

in conjunction, as the clustering structure of the data is exploited to define anomalies. Khalid and Naftel (2006) first conducts a cluster analysis of the trajectory data and identifies *outliers* from each cluster using a user-defined threshold. In Anjum and Cavallaro (2008), members of small clusters are identified as anomalies.

Most anomaly detection and clustering algorithms based on numerical representations are directly applicable to temporal data and subsequences of time series. Methods of clustering are typically categorized into model-based or distance-based approaches (Cotofrei and Stoffel 2002; Dereszynski and Dietterich 2007; Keogh and Lin 2005; Piciarelli et al. 2008; Xiong and Yeung 2002; Yankov et al. 2008). The taxonomy of anomaly detection methods is discussed further in the next section. There are, however, caveats for clustering and anomaly detection regarding subsequences generated by the sliding window (Chiu et al. 2003; Keogh and Lin 2005; Keogh et al. 2005). which we discuss in Sect. 9.

2.2 Anomaly detection

Anomaly detection is part of a major topic of statistical learning that also includes outlier detection and novelty detection, and numerous practical applications are studied in this topic. Generally, a data with a small probability density value is identified as an anomaly (Chandola et al. 2009). The density can be measured relative to the complete dataset or an auxiliary set of normal samples. The specific definition of the anomaly depends on the application and the methodology.

The methods of anomaly detection can be categorized into three major approaches: (a) the model-based approach which employs stochastic generative models to estimate the probability density function, (b) the instance-based approach which exploits local approximation of the density function, and (c) the discriminative approach which directly estimates the decision function. In (a), the normal samples are typically modeled as a Gaussian mixture (Yamanishi et al. 2004), or learned using artificial neural networks (ANN) (Markou and Singh 2003b, 2006). Notable methods from (b) include local outlier factor (Breunig et al. 2000), distance-based outlier detection (Knorr et al. 2000; Angiulli et al. 2006; Ghoting et al. 2008), and discord discovery (Yankov et al. 2008). Examples of (c) are One-class SVM, which has been applied to trajectory data (Piciarelli and Foresti 2007) and direct density-ratio estimation (Hido et al. 2011).

Instance-based methods generally require less undertaking for understanding and modeling the generative process compared to the model-based methods. This is a practical advantage for real-world applications with numerous and complex influential factors. Meanwhile, computing the nearest-neighbors of each instance requires the time complexity of $O(n \log n)$ to $O(n^2)$ for n samples (Chandola et al. 2009), which is impractical in many cases. To address the issue of efficiency, sophisticated approximations have been proposed for specific applications such as streaming data (Angiulli and Fassetti 2010) and large databases (Ghoting et al. 2008; Kim et al. 2011).

While the problem setting of anomaly detection is typically that of an unsupervised learning, the problem has also been extensively studied in semi-supervised learning settings (Chandola et al. 2009). The motivation for implementing a semi-supervised approach comes from the availability of normal samples in practice, as they constitute the majority of observation. The examples of anomalies, on the other hand, are strongly

limited. In a typical semi-supervised setting, the examples of normal instances are given as the training set and the model learned from the training set is evaluated using a test set, which is possibly contaminated with anomalies (Noto et al. 2012). The normal samples for training may also be referred to as *inliers* (Angiulli and Fassetti 2010; Hido et al. 2011). In novelty detection, the semi-supervised approach is also called *inductive* novelty detection (Blanchard et al. 2010).

Technically, many semi-supervised techniques can function in an unsupervised setting, as the assumptions of the two settings are closely related (Chandola et al. 2009). In the unsupervised settings, it is implicitly assumed that the normal instances are far more frequent than the anomalies. Under such an assumption, the semi-supervised approach can learn a model from the unlabeled data that is robust to few anomalies in the training data. If the assumption does not hold, an unsupervised learning technique suffers from a high false alarm rate. Similarly, semi-supervised learning techniques are subject to low precision if the collection of normal instances are poor.

In the proposed framework, we implement an efficient cluster-wise anomaly detection based on an approximation of relative densities between the target and the auxiliary data. It is implemented as a semi-supervised learning technique, which exploits the examples of normal instances as the *auxiliary* input and identifies anomalies in the *target* set. The anomalous instances are identified from the clusters with relatively low density based on the auxiliary data, i.e., those appearing to be rare in normal behaviors. Its output is a vector for each instance, which is the significant difference from the existing anomaly detection techniques. Summarizing the prediction of each instance into a scalar score is deferred to the higher-level component of the framework.

2.3 Ensemble methods for supervised and unsupervised learning

In machine learning, the means of combining outputs of different learning methods is studied under the topic of ensemble learning (Dietterich 2000; Rokach 2010), and sophisticated approaches have been developed for supervised learning tasks. Typically in classification, the aim is to combine individual models, or *weak learners*, in order to form a strong one. *Weak learners* are assumed to have different biases and to vary with regards to the examples they can and cannot classify. Larger weights are given to more accurate learners or those capable of classifying more *difficult* examples.

In recent studies, ensemble methods for cluster analysis have been explored in various disciplines, e.g., Consensus Clustering (Ailon et al. 2008), Clustering Ensemble (Fern and Brodley 2004; Strehl and Ghosh 2003), and Clustering Aggregation (Gionis et al. 2007). Motivation for generating a consensus from multiple clustering derives from the exploratory and subjective nature of the cluster analysis (Banerjee and Langford 2004; Jain et al. 1999). A set of objects can be partitioned in a variety of ways depending on the representation, the clustering method, and the criteria. In Yang and Chen (2011), ensemble clustering has been applied to temporal data, in which clustering based on various representations and criteria are weighted and combined to form an intuitive partitioning.

Learning an ensemble classifier or an ensemble clustering generally requires input data which are fully labeled with a pre-defined set of classes or clusters. The examples labeled as anomalies, however, are usually unavailable or expensive to prepare, which

poses a practical limitation for the ensemble approaches of anomaly detection with regards to training and evaluating the weak learners. In the existing ensemble methods for anomaly detection, the *weak* learners' scores are aggregated without weighting, with the goal of addressing high-dimensionality and random noise added to numerical features (Ho 1998; Lazarevic and Kumar 2005; Nguyen et al. 2010).

2.4 Multiscale/resolution representation

Variations of multiscale/resolution representation have been studied in different areas of signal processing, e.g., the pyramid representation for image processing. The examples of multiresolution analysis techniques in the time series domain are: wavelet transform (Daubechies 1992), iSAX representation (Castro and Azevedo 2010), and adaptive cellular bins (Wan et al. 2009; Wang et al. 2010). The wavelet transform is also known for its relation with the scale-space representation in image processing.

In existing multiresolution representations of time series, the resolution is generally assumed to be dependent on only one parameter. In contrast, the case we originally considered in Ando et al. (2011) includes multiple parameters that are related to controlling the resolution. The difficulty of choosing a resolution is significantly higher in the latter case, and prompts our motivation for an ensemble method that can aggregate the outcomes in different resolutions.

3 Problem description

The problem addressed in the ACE framework includes three sub-tasks: generating the behavior features, detecting anomalies from a single feature and constructing an ensemble of anomaly detection from multiple features. In the following sections, we formalize the respective tasks and specify their input/output.

3.1 Behavior feature representation

Finding a good numerical representation is a critical aspect of all data mining applications. In this paper, we consider the behaviors of an object that are relevant to its trajectory. As the object is tracked over a long duration, different behaviors occur over segments of the trajectory. Generating the numerical features of the behaviors from the trajectory naturally involves choosing a resolution with which the behaviors should be represented. In the following, we describe an example of the feature in which the resolution is dependent on multiple parameters.

The velocity is one of the basic characteristics of the object's behavior. To compute the velocity from the a series of coordinates, one needs to choose a time interval for measuring the displacement. Furthermore, to represent the behavior over a finite length of time, it is also necessary to divide the sequence of feature values into segments of a specified size. These choices directly affect the resolution, with which the behaviors are examined in subsequent procedures.

Let $X = \{\mathbf{x}(t)\}_{t=1}^{T}$ denote the input trajectory consisting of T coordinates, where $\mathbf{x}(t)$ denotes the coordinate at time t. We denote by δ and λ the parameters for smoothing and segmentation, respectively. The velocity $\mathbf{v}(t)$ is defined as

$$\mathbf{v}(t) = \frac{\Delta \mathbf{x}(t)}{\delta}, \tag{1}$$

where δ denotes the interval and the displacement $\Delta \mathbf{x}(t)$ is defined as follows.

$$\Delta \mathbf{x}(t) = \left(\frac{1}{\delta + 1} \sum_{i=0}^{\delta} \mathbf{x}(t+i) \right) - \left(\frac{1}{\delta + 1} \sum_{i=0}^{\delta} \mathbf{x}(t-i) \right) \tag{2}$$

In the general case, moving average can be used for smoothing instead of (2) and the window size can be substituted for λ in the proposed framework.

Let $s_{\delta,\lambda}(t)$ denote the subsequence of the Euclidean norm of the velocity

$$s_{\delta,\lambda}(t) = (\|\mathbf{v}(t-\lambda)\|_2, \|\mathbf{v}(t-\lambda+1)\|_2, \ldots, \|\mathbf{v}(t+\lambda)\|_2) \tag{3}$$

which is used to represent the behavior of the object at t. We employ the sliding window to extract the instances of behaviors exhaustively from the time series of the Euclidean norms. Each instance is represented by a subsequence in a set

$$S_{\delta,\lambda} = \left\{ s_{\delta,\lambda}(t') \right\}_{t'=1}^{\eta}, \tag{4}$$

where η denotes the number of subsequences. In relation to t and the number of time series T, $t' = t - \lambda$ and $\eta = T - 2\lambda$. Since the behaviors are primarily represented as subsequences, we use t in place of t' in the rest of the formalization for simplicity.

Given the combination of values (δ, λ), $S_{\delta,\lambda}$ is uniquely determined from the time series. In the following, the combination is referred to as a *resolution* and denoted as $\theta = (\delta, \lambda)$. Note that by taking the Euclidean norm of the velocity, the feature is invariant to the absolute coordinates and the orientation of the axes. The feature therefore represents the behaviors of a single object, independently of a specific coordinate system.

3.2 Single-feature anomaly detection

The goal of the single-feature anomaly detection is to predict the labels of the target dataset, whether each instance is anomalous, within a single *resolution*. In addition to the target dataset, an auxiliary set of subsequences is included in the input. The auxiliary set consists exclusively of the normal instances. It is assumed that the normal instances in the two datasets are sampled from an identical source distribution.

Let $S_\theta = \{s_\theta(t)\}_{t=1}^{\eta}$ and $S'_\theta = \{s_\theta(t)\}_{t=1}^{\eta'}$ denote the target and the auxiliary sets of subsequences, both generated with the *resolution* θ. The predicted labels are denoted by $\{y(t;\theta)\}_{t=1}^{\eta}$. Note that a label is not limited to a scalar value. As described in Sect. 4, y corresponds to a vector of binary values in the proposed framework.

3.3 Ensemble anomaly detection

The ensemble anomaly detection takes the predicted labels of the behavior instances in multiple *resolutions* and computes their scores. The probable anomalies are indicated by their higher scores.

Let $\boldsymbol{\Theta} = \{\theta_i\}_{i=1}^{q}$ denote the set of all *resolutions* that merit consideration for the problem at hand. Let

$$\mathbf{y}(t) = (y(t; \theta_1), \ldots, y(t; \theta_q))$$

denote the collective predictions for t, which forms a *meta*-feature representation of the behavior instance at time t. The set of *meta*-features is written as $\{\mathbf{y}(t)\}_{t=1}^{\eta}$, and the output set of score as $\{A(t)\}_{t=1}^{\eta}$.

4 Proposed framework

The implementation of the ACE framework consists of the following components: (1) the multi-*resolution* feature generation, (2) the single-feature anomaly clustering (SAC), (3) the *meta*-feature clustering, and (4) the cluster-based anomaly score. The multi-*resolution* feature generation addresses the behavior feature generation task with regards to a grid of parameter values, i.e., all candidate *resolutions*. The SAC addresses the single-feature anomaly detection tasks based on a cluster analysis and comparisons of approximate density. The *meta*-feature clustering partitions the behavior instances based on the *meta*-features constructed from the collection of labels predicted by the SAC. The cluster-based anomaly score is computed for each instance based on the cluster analysis in the *meta*-feature space. The *meta*-feature clustering and the cluster-based anomaly score components collectively address the ensemble anomaly detection problem. Figure 2 illustrates an overview of the framework. In the following sections, the implementations of the components are described individually in detail.

4.1 Multi-*resolution* feature generation

Let $\{\delta_i\}_{i=1}^{\pi}$ and $\{\lambda_j\}_{j=1}^{\pi'}$ denote the sets of candidate parameter values for smoothing and the sliding window. Let $\boldsymbol{\Theta}$ denote the following set of combinations.

$$\boldsymbol{\Theta} = \left\{(\delta_i, \lambda_j)\right\}_{i \in \{1,\ldots,\pi\}, j \in \{1,\ldots,\pi'\}}$$

We rewrite the elements of $\boldsymbol{\Theta}$ to denote each *resolution* as $\theta \in \boldsymbol{\Theta}$, i.e.,

$$\boldsymbol{\Theta} = \{\theta_i\}_{i=1}^{q}, \tag{5}$$

where q denotes the number of all combinations. Let

$$\mathbf{S}_{\boldsymbol{\Theta}} = \{S_\theta\}_{\theta \in \boldsymbol{\Theta}}$$

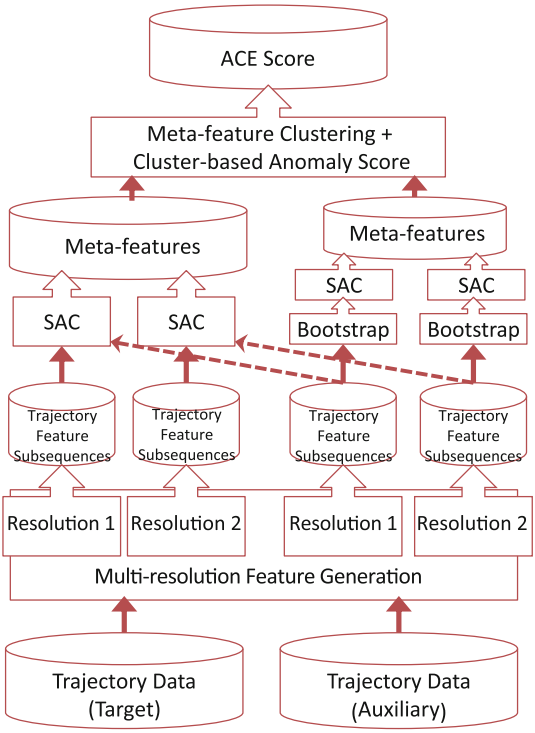


Fig. 2 Illustrated summary of the ACE framework

denote the sets of subsequences computed from the target trajectory data X using (1)–(3). Similarly for the auxiliary trajectory data X', the multi-*resolution* features $\mathbf{S}'_{\boldsymbol{\Theta}}$ is defined as follows.

$$\mathbf{S}'_{\boldsymbol{\Theta}} = \{S'_\theta\}_{\theta \in \boldsymbol{\Theta}}$$

4.2 Single-feature anomaly clustering

The single-feature anomaly clustering (SAC) identifies the anomalies in a target dataset based on a cluster-wise comparison of approximate density functions with an auxiliary dataset. The intuition underlying the cluster-wise approach is that between the two datasets that exclude and contain anomalies, one should be able to observe the largest discrepancies in density near the centers of the *anomalous* clusters. In density-based anomaly detection, the probability density at each instance is approximated by the distances to its nearest-neighbors (Angiulli et al. 2006). Data exhibiting lower densities, i.e., those relatively farther from their neighbors, are considered anomalies. We extend this approach to a cluster-wise anomaly detection by comparing the densities between two datasets at the centroids of obtained clusters. It is assumed here that there exists a distance function feasible for clustering and density approximation. Without the loss of generality, we consider the distance function to be the Euclidean distance in the rest of this paper.

Let $S_\theta = \{s_\theta(t)\}_{t=1}^{\eta}$, $S'_\theta = \{s'_\theta(t)\}_{t=1}^{\eta'}$ denote the sets of subsequences generated from the target and the auxiliary time-series by a resolution θ, respectively. For brevity, we omit θ in the subscript when there is only one relevant *resolution* in context.

First, we apply clustering to S to obtain k cluster centroids $\{\mu_i\}_{i=1}^{k}$, k clusters $\{C_i | C_i \subset S\}_{i=1}^{k}$, and a mutual distance matrix $\mathbf{D} = \left[d_{i,t}\right]_{k\times\eta}$. Given a distance function $D(\cdot,\cdot)$, $d_{i,t} = D(\mu_i, s(t))$.

Let $N_\nu(\mu_i) \subset S$ denote ν elements of C_i nearest to μ_i, i.e.,

$$N_\nu(\mu_i) = \underset{\{s_j\}_{j=1}^{\nu}\subset C_i}{\arg\min} \sum_{j=1}^{\nu} D(s_j, \mu_i) \tag{6}$$

and

$$d_\nu(\mu_i) = \max_{s\in N_\nu(\mu_i)} D(s, \mu_i)$$

the distance to the νth nearest element.

We partition the auxiliary data into k subsets $\{C'_i\}_{i=1}^{k} : C'_i \subset S'$, such that the nearest centroid of its elements is unique for each partition. Formally, the following should be satisfied for all C'_i.

$$\forall s' \in C'_i : \mu_i = \underset{\mu\in\{\mu_j\}_{j=1}^{k}}{\arg\min}\ D(\mu, s') \tag{7}$$

We define a set of auxiliary data that are nearest neighbors and within $d_\nu(\mu_i)$ radius of the centroid μ_i, denoted by $N'(\mu_i)$.

$$N'(\mu_i) = \left\{s' | s' \in C'_i, D(s', \mu_i) \le d_\nu(\mu_i)\right\} \tag{8}$$

The insight underlying the distance-based approach is that the distance to the nearest-neighbors reflects the density of a point in the feature space (Chandola et al. 2009). With regards to μ_i, $N_\nu(\mu_i)$ and $N'(\mu_i)$ provide the approximation of densities over the target and the auxiliary data, respectively. If the former is significantly larger than the latter, it suggests that the cluster C_i represents an anomalous behavior.

We test the significance of the difference between the means of the two sets $N_\nu(\mu_i)$ and $N'(\mu_i)$, by Mann–Whitney U test, or rank-sum test (Lehmann 2006). The test exploits the U-statistic, i.e., the sum of ranks, which follows a normal distribution for a sufficiently large sample set and is known to be robust against extreme values. We perform a one-sided test with a null hypothesis that the mean distance to $N_\nu(\mu_i)$ is equivalent to or greater than that of $N'(\mu_i)$. The rejection of the null-hypothesis indicates that the cluster likely includes anomalies. Based on the p value of the test, we assign each μ_i a label a_i. Each label takes a value from $\{0, 1\}$, such that

$$a_i = \begin{cases} 1 \text{ if } \ \alpha > p \text{ value} \\ 0 \text{ otherwise,} \end{cases} \tag{9}$$

where α is the confidence level. If $a_i = 1$, it indicates that the ith cluster is predicted anomalous. Note that the additional computation required for the rank-sum test is only to sort the within-cluster distances of each cluster, since the distance matrix $\mathbf{D}$ has been computed in clustering. The number of neighbors used for locally approximating the density is a common parameter in the density-based approaches. In the proposed method, the value of ν can be chosen based on the the cluster size, e.g., from 50 to 75 % of the cluster size. However, for the power of the rank-sum test, it is preferable to have a sample size larger than 20–30 at the least.

We assign the predicted label $\mathbf{b}_\theta(t)$ to each instance $s(t)$ to represent the results of clustering and the test of means. Let $\mathbf{B}_\theta$ denote a $k \times \eta$ binary matrix

$$\mathbf{B}_\theta = \begin{bmatrix} b(1,1) & \dots & b(1,\eta) \\ \vdots & \ddots & \vdots \\ b(k,1) & \dots & b(k,\eta) \end{bmatrix}, \tag{10}$$

where $b(i,t)$ is given as follows.

$$b(i,t) = \begin{cases} a_i \text{if } s_\theta(t) \in C_i \\ 0 \text{ otherwise} \end{cases} \tag{11}$$

Each column in $\mathbf{B}_\theta$ is a vector of binary values indicating the cluster to which the subsequence $s_\theta(t)$ is assigned and whether the clusters is labeled as anomalous. We rewrite $\mathbf{B}_\theta$ as

$$[\mathbf{b}_\theta(1) \dots \mathbf{b}_\theta(\eta)] = \mathbf{B}_\theta \tag{12}$$

and denote by $\mathbf{b}_\theta(t)$ the column vector corresponding to $s_\theta(t)$. In the case that $s_\theta(t)$ is assigned to a normal cluster, $\mathbf{b}_\theta(t) = \mathbf{0}$. The SAC yields $\{\mathbf{b}_\theta(t)\}_{t=1}^{\eta}$ as the output to the single-feature anomaly detection problem.

Due to its use of cluster analysis, the SAC is inclined to detect anomalies with clustering tendencies, rather than those in isolation. We discuss the motivation of this implementation in Sect. 9. Because of its semi-supervised and density-based approach, it is also assumed that the density can be approximated by an arbitrary distance function and the normality is well-represented by the auxiliary data. If the auxiliary data is scarce over a region of the data, $N'_\nu(\mu_i)$ in that region can be sparse. In such a case, the difference in means is likely to be significant, i.e., the null hypothesis is easily rejected. For the extreme case that $N'_\nu(\mu_i)$ is empty, and the test is incomputable, we suggest to reject the null hypothesis as a natural extension of the sparse case. In this procedure, C_i may be falsely identified as anomalous if $N'_\nu(\mu_i)$ is sparse from poorly collected auxiliary data. The SAC is therefore subject to the same limitations of the general unsupervised and semi-supervised anomaly detection techniques as discussed in Sect. 2.2.

With regards to the cluster analysis, it is necessary that the algorithm identifies central points of high density. For practical applications, its output should also be stable, i.e., algorithms such as k-means with a known, strong dependence on the

initial values, are not preferred. It is also preferable if k can be chosen beforehand as the complexity of the framework depends on k as discussed in Sect. 5. The algorithm used for our empirical evaluation is described in Sect. 5.1.

Both ν and k are important parameters in the SAC. In the general case, they can be determined using an independent dataset by cross-validation. For the SAC, since it operates in a semi-supervised setting, it is beneficial to exploit the auxiliary data for determining the parameters. We discuss such means in Sect. 6.

4.3 *Meta*-feature clustering

In the *meta*-feature clustering, behavior instances are partitioned based on their predicted labels over multiple *resolutions*. Let $\{\mathbf{B}_{\theta_i}\}_{i=1}^{q}$ denote the output of the SAC over candidate *resolutions* Θ, a set of q matrices of the size $k \times \eta$. Note that the dimension of the input of the SAC is determined by the *resolution* θ, while the dimension of its output depends on its parameter k.

We define a binary matrix $\mathbf{B}_\Theta$ by joining the matrices vertically as follows.

$$\mathbf{B}_\Theta = \begin{bmatrix} \mathbf{B}_{\theta_1} \\ \vdots \\ \mathbf{B}_{\theta_q} \end{bmatrix} \tag{13}$$

A *meta*-feature vector $\mathbf{m}(t)$ is then defined as the tth column of $\mathbf{B}_{\boldsymbol{\Theta}}$.

$$[\mathbf{m}(1) \ldots \mathbf{m}(\eta)]_{kq \times \eta} = \mathbf{B}_\Theta \tag{14}$$

Each variable of $\mathbf{m}(t)$ corresponds to a cluster found in the SAC. A *meta*-feature vector is formed by concatenating the predicted labels of the clusters from different *resolutions*, thereby mapping a behavior instance to a kq-dimensional *meta*-feature space.

The *meta*-feature vectors $M = \{\mathbf{m}(t)\}_{t=1}^{\eta}$ are partitioned into κ clusters by a clustering algorithm using the Euclidean distance. We denote the centroid of the ith cluster by $\mathbf{c}_i$ and the closest centroid from the tth instance by $\mathbf{c}(t)$, which takes a value from $\{\mathbf{c}_i\}_{i=1}^{\kappa}$.

4.4 Cluster-based anomaly score

The goal of the cluster-based anomaly score is to generate scalar scores that aggregates the results of anomaly detection in different *resolutions*. We measure the distance between the *meta*-features of the normal patterns and the *meta*-cluster centroids to use as such scores. The normal *meta*-feature patterns are generated separately from the auxiliary data as follows. Let R_θ and R'_θ denote two sets of ρ bootstrapped samples of the auxiliary set S'_θ (Steland 1998). We compute the binary matrix $\mathbf{R}_\theta$ from (R_θ, R'_θ) by the SAC. Furthermore, we compute the *meta*-feature vectors from $\{\mathbf{R}_\theta\}_{\theta \in \Theta}$, in a similar manner as (13) and (14), as follows.

$$\left[\mathbf{r}_1, \ldots, \mathbf{r}_\rho\right] = \mathbf{R}_\Theta$$

One may use two exclusive sets of samples as R_θ and R'_θ provided that the numbers of samples are sufficient for performing the test of means.

We refer to $Q = \{\mathbf{r}_j\}_{j=1}^{\rho}$ as the set of *meta*-reference vectors. Although the *meta*-reference vectors are projections of the normal behaviors to the *meta*-feature space, they are unlikely to be $\mathbf{0}$, as normal instances can be predicted anomalous due to the noise in the observed data or the bias in a particular *resolution*.

A *meta*-feature vector, e.g., a *meta*-feature cluster centroid $\mathbf{c}$, consists of q sequences of k variables. Each sequence of k variables reflects the assignment of a behavior instance to an *anomalous* cluster with respect to the clustering at a particular *resolution*. The summation over each sequence of variables gives a summarized view of the instance, i.e., whether it is likely to be an anomaly at that *resolution*.

We define the *summarizing* operation using a $kq \times q$ binary matrix $\mathbf{U}$. $\mathbf{U}$ is written as

$$\mathbf{U} = \begin{bmatrix} \mathbf{u} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{u} \end{bmatrix}, \tag{15}$$

where $\mathbf{u}$ denotes a column vector of q 1's.

Let $\hat{\mathbf{c}}(t)$ denote the summarized vector of $\mathbf{c}(t)$. $\hat{\mathbf{c}}(t)$ is given by $\mathbf{U}$ and $\{\mathbf{c}(t) : t \in \{1, \ldots, \eta\}\}$ as follows.

$$\left[\hat{\mathbf{c}}(1) \ldots \hat{\mathbf{c}}(\eta)\right]_{q \times \eta} = \mathbf{U}\left[\mathbf{c}(1) \ldots \mathbf{c}(\eta)\right] \tag{16}$$

Each *summarized meta*-feature vector consists of q variables. Its ith variable represents the prediction at the resolution θ_i. Collectively, they reflect an orientation of how anomalous an instance appears to be, with regards to different *resolutions*.

The summarizing operation can be applied to the *meta*-reference vectors in the same manner as follows.

$$\left[\hat{\mathbf{r}}_1 \ldots \hat{\mathbf{r}}_\rho\right]_{q \times \rho} = \mathbf{U}\left[\mathbf{r}_1 \ldots \mathbf{r}_\rho\right]_{kq \times \rho} \tag{17}$$

The *summarized meta*-feature and *meta*-reference vectors are of the same length and their corresponding variables represent the predictions at the same resolutions. The distances from $\hat{\mathbf{c}}_i$ to $\{\hat{\mathbf{r}}_j\}$ quantifies the deviation of the centroid $\mathbf{c}_i$ from the normal behaviors as the latter summarizes the regions of the *meta*-feature space to which normal behaviors are projected.

For each centroid $\mathbf{c}_i$, we define its anomaly score $A(\mathbf{c}_i)$ based on its *summarized* vector $\hat{\mathbf{c}}_i$. The score is defined as the average distance to the neighboring *meta*-reference vectors N,

$$A(\mathbf{c}_i) = \min_{N \subset \{\hat{\mathbf{r}}\}} \frac{1}{\#(N)} \sum_{\mathbf{r} \in N} D(\hat{\mathbf{c}}_i, \hat{\mathbf{r}}), \tag{18}$$

where $D(\cdot, \cdot)$ denotes the Euclidean distance. We define the score of the tth instance, $A(t)$, as follows.

$$A(t) = A(\mathbf{c}(t)) \tag{19}$$

Note that prior to the *summarization*, a *meta*-feature vector and a *meta*-reference vector are of the same length, but the distance between them is not a meaningful value, since their corresponding variables represent the assignments for two independent clustering of different datasets, respectively. In order to exploit as much discriminative information as possible, the clustering is first performed in the *meta*-feature space, then the cluster-based anomaly score is computed from the *summarized* features.

5 Algorithm

This section describes the algorithmic implementation of ACE and its computational complexity.

5.1 Implementation

The pseudocode of the ACE framework is shown in Algorithm 1. It includes the *meta*-feature generation (MFG), the SAC, and the cluster-based anomaly score (CAS) as functions. In addition, the clustering algorithm k-means++ (Arthur and Vassilvitskii 2007), an extension of k-means with an enhanced seeding, is used in the SAC and the *meta*-feature clustering.

The MFG function generates the multi-*resolution* features $\mathbf{S}_{\boldsymbol{\Theta}}$ and $\mathbf{S}'_{\boldsymbol{\Theta}}$ from the target and auxiliary datasets, respectively (line 9). Then, for each resolution $\theta \in \Theta$, the SAC computes the cluster-assignment matrices $\mathbf{B}_\theta$ and $\mathbf{R}_\theta$ (lines 11–12). In *meta*-feature clustering, the *meta*-feature vectors $\{\mathbf{m}(t)\}_{t=1}^{\eta}$ generated from $\mathbf{B}_\Theta$ are partitioned using the clustering algorithm, returning the centroids $\{\mathbf{c}_j\}_{j=1}^{\kappa}$ and the cluster labels of all instances $\{\mathbf{c}(t)\}_{t=1}^{\eta}$ (line 16). The CAS function computes the anomaly score $A(t)$ from the centroids, the cluster labels, and the *meta*-reference vectors $\{\mathbf{r}(t)\}_{i=1}^{\rho}$ (line 17).

Algorithm 2 shows the pseudocode of the SAC. It includes a clustering algorithm and test of means as internal functions. The clustering algorithm outputs the centroids and the partitioning of the target data (line 7). For each centroid, bootstrap samples of the nearest target data and the nearest auxiliary data are prepared (lines 10–11) and their means are compared by rank-sum test (Steland 1998) (line 12).

Algorithm 3 shows the pseudocode of the CAS function. The function computes the dissimilarity between the summarized features of the centroids and the summarized reference vectors (line 4). The output score is computed by propagating the scores of the centroids to the assigned instances (lines 7–9).

Algorithm 1 Anomaly Clustering Ensemble (ACE)

INPUT: Target trajectory data X, auxiliary trajectory data X', set of resolutions $\boldsymbol{\Theta} = \{\theta_i\}_{i=1}^{q}$, # of *meta*-clusters κ, bootstrap sampling size ρ
OUTPUT: *meta*-cluster label $\{\mathbf{c}(t)\}_{t=1}^{\eta}$, Anomaly Score $\{A(t)\}_{t=1}^{\eta}$
function MFG : (trajectory/candidate resolutions) → (multi-*resolution* subsequences)
function SAC : (target/auxiliary subsequences) → (anomalous cluster-assignment matrix)
function Bootstrap: (set/sampling size) → (sample set with replacement)
function Clustering: (feature vector set) → (centroids/cluster labels)
function CAS: (centroids/cluster labels/*meta*-reference vectors)→(anomaly score)
METHOD
$\mathbf{S}_{\boldsymbol{\Theta}} \leftarrow \mathrm{MFG(X}, \boldsymbol{\Theta}), \mathbf{S}'_{\boldsymbol{\Theta}} \leftarrow \mathrm{MFG(X'}, \boldsymbol{\Theta})$
for all $\theta \in \boldsymbol{\Theta}$ **do**
  $\mathbf{B}_\theta \leftarrow \mathrm{SAC(S}_\theta, \mathrm{S}'_\theta)$
  $\mathbf{R}_\theta \leftarrow \mathrm{SAC}\left(\mathrm{Bootstrap(S}'_\theta, \rho), \mathrm{Bootstrap(S}'_\theta, \rho)\right)$
end for
Compute $\{\mathbf{m}(t)\}$ by substituting $\{\mathbf{B}_\theta\}_{\theta\in\boldsymbol{\Theta}}$ to (13) and (14)
Compute $\{\mathbf{r}_i\}$ by substituting $\{\mathbf{R}_\theta\}_{\theta\in\boldsymbol{\Theta}}$ to (13) and (14)
$\left(\{\mathbf{c}_j\}_{j=1}^{\kappa}, \{\mathbf{c}(t)\}\right) \leftarrow \mathrm{Clustering}\,(\{\mathbf{m(t)}\}, \kappa)$
$\{A(t)\} \leftarrow \mathrm{CAS}(\{\mathbf{c_j}\}, \{\mathbf{c(t)}\}, \{\mathbf{r_i}\})$
return $\{\mathbf{c}(t)\}_{t=1}^{\eta}, \{A(t)\}_{i=1}^{\eta}$

Algorithm 2 Single-feature Anomaly Clustering (SAC)

INPUT: Target and auxiliary subsequence sets $S_\theta = \{s_i\}_{i=1}^{\eta}$, $S'_\theta = \{s'_i\}_{i=1}^{\eta'}$, # of clusters k, bootstrap sampling size ρ, confidence level α
OUTPUT: Anomalous cluster-assignment matrix $\mathbf{B}_\theta$
function Clustering (subsequence set) → (centroids/partitioning)
function Bootstrap: (set/sampling size)→(bootstrap sample set)
function MeanTest (Rank-sum Test)→(p-value)
METHOD:
$\left(\{\mu_i\}_{i=1}^{k}, \{C_i\}_{i=1}^{k}\right) \leftarrow \mathrm{Cluster(S}_\theta, \mathrm{k})$
Partition auxiliary data into subsets $\{C'_i\}_{i=1}^{k}$ following (7)
for $i = 1, \ldots, k$ **do**
  $\nu \leftarrow \frac{\#(C_i)}{2}$
  Select $N_\nu(\mu_i)$, $N'_\nu(\mu_i)$ following (6), (8)
  Compute a_i by substituting α and MeanTest $\left(\mathrm{Bootstrap(N}_\nu(\mu_\mathrm{i}), \rho), \mathrm{Bootstrap(N}'_\nu(\mu_\mathrm{i}), \rho)\right)$ to (9)
end for
Compute $\mathbf{B}_\theta$ by substituting $\{a_i\}, \{C_i\}$ to (10) and (11)
return $\mathbf{B}_\theta$

5.2 Computational complexity

In this section, we discuss the complexity of the SAC and the ACE framework in comparison to conventional instance-based anomaly detection methods. Computing and storing the mutual distance matrix for η subsequences of length w require $O(\eta^2 w)$. The time complexity for sorting the scores and selecting the top ν anomalous instances is $O(\eta \log \eta)$.

In comparison, the SAC computes the distances from the centroids to the subsequences in each iterative step of k-means++ which requires $O(k\eta w)$. Assigning auxiliary data to the nearest centroid is done in the same order. The iteration generally

Algorithm 3 Cluster-based Anomaly Score (CAS)

INPUT: *meta*-cluster centroids $\{\mathbf{c}_i\}_{i=1}^{\kappa}$, *meta*-cluster labels $\{\mathbf{c}(t)\}$, *meta*-reference vectors $R = \{\mathbf{r}_j\}$
OUTPUT: Anomaly score $\{A(t)\}$
METHOD
Compute $\{\hat{\mathbf{c}}_i\}$ and $\{\hat{\mathbf{r}}_j\}$ by substituting $\{\mathbf{c}_i\}$ and $\{\mathbf{r}_j\}$ to (16) and (17), respectively
for $i = 1, \ldots, \kappa$ **do**
  Compute $A(\mathbf{c}_i)$ by substituting $\hat{\mathbf{c}}_i$, $\{\hat{\mathbf{r}}_j\}$ to (18)
  for all t **do**
    Compute $A(t)$ by substituting $\mathbf{c}(t)$ to (19)
  end for
end for
return $\{A(t)\}$

converges with a small number of repetitions, which can be considered a constant r. The complexity of the SAC is thus $O(rk\eta w)$, where generally $rk \ll \eta$ in real-world data. With regards to the space complexity, the SAC requires $O(k\eta)$ for the distance matrix while $O(\eta^2)$ is required for the mutual distance matrix in the instance-based approach in general.

Detecting anomalous centroids by rank-sum test requires sorting the within-cluster distances for each cluster. Let $\#(C_j)$ denote the cardinality of the jth cluster. The time complexity for sorting in each cluster is $O\left(\sum_{j=1}^{k} \#(C_j) \log \#(C_j)\right)$, which is less than $O(\eta \log \eta)$ since $\eta = \sum_j \#\left(C_{j=1}^{k}\right)$. From above discussions, the SAC achieves better efficiency than conventional instance-based anomaly detection methods.

When considering q-candidate *resolutions*, the ACE framework requires $O(qkr\eta w)$ for the SAC, and $O(\hat{k}\eta q)$ for the *meta*-clustering with $\hat{k}$ clusters. The number of iterations can be considered a constant. Computing the anomaly score requires $O(\hat{k}q\rho)$. The dominant complexity among the components is therefore that of the SAC. Since the conventional instance-based methods require $O(q\eta^2 w)$ in all and $kr \ll \eta$, the ACE framework achieves better efficiency in comparison to the conventional approach.

6 Parameter selection scheme

Choosing the number of clusters is a fundamental problem in most clustering algorithms. For those that do not require a pre-specified number of clusters, there usually exist other parameters that affect the final number of clusters, such as $minPts$ and ϵ in DBSCAN (Ester et al. 1996). The problem is referred to as the model selection, and has a rich history of literature that provides techniques for selecting the number of clusters (Fraley and Raftery 1998; Jain et al. 1999; Xu and Wunsch 2005). With sufficient knowledge of these techniques and the time to carefully examine the data, one may adequately select a fitting model.

In this section, we discuss a parameter selection scheme for the clustering algorithm used internally by the SAC as discussed in Sect. 4.2. We note that it is not within the scope of this paper to improve upon the previous literature on model selection. Our goal rather is to provide an adequate and practical means of parameter selection specifically for the applications of ACE.

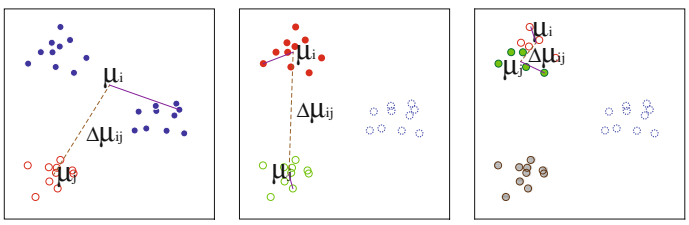
There are three aspects of the ACE framework that we consider to be critical in the selection of the number of clusters k. First, with regards to the process of the SAC, under-partitioning can induce costly true negative errors when anomalous instances are clustered with a large number of normal instances. Meanwhile, over-partitioning of the normal behaviors is less likely to induce false positive errors. As such, it is important that the clusters are compact and sufficiently distanced from each other.

Secondly, while most existing model selection techniques are based on probabilistic modeling assumptions, it would be an additional burden to find a generative model for the application of ACE , since it already requires a choice of an appropriate distance function. Finally, the number of clusters k can be selected based on the empirical evaluation over an independent dataset in general. In practice, however, it is probable that preparing an independent dataset with a sufficient number of anomalies or expending significant computation after receiving the target data is costly. In the problem at hand, auxiliary data are available off-line for an extensive analysis.

Based on the above considerations, we propose a parameter selection scheme for k based on a distance-based criteria which can be evaluated over the auxiliary data. Intra-cluster and inter-cluster distances are essential measures of the quality of clustering. Individually, both measures generally decrease as k becomes larger. However, the ratio of the intra-cluster distance to the inter-cluster distance can give an insight as to how well the clusters are interspersed, e.g., a larger ratio suggests more congested and overlapping clusters.

Considering the risk of under-partitioning, our aim is to keep the ratio no larger than that of a *natural* clustering. The ratio becomes larger than that of a *natural* clustering when: (i) a singular group of instances are divided into multiple clusters, or (ii) multiple groups are combined into one cluster. Figure 3 illustrates these cases in a 2-dimensional example. A set of points are partitioned into clusters using k-means algorithm with $k = 2$–4. The points in a cluster are represented by the same markers: filled, dotted, solid, or textured circles. μ_i and μ_j denote two of the centroids and inter-cluster distance $\Delta\mu_{ij}$ is represented by a dotted line. A typical intra-cluster distance is shown by the solid line.

In Fig. 3, (a) and (c) respectively correspond to cases (i) and (ii) mentioned above. In (a), the cluster of solid circles subsumes two groups. As a result, the intra-cluster distance from the centroid μ_i to its members, is similar to the inter-cluster distance. In (c), one group of instances is represented by two centroids, μ_i and μ_j, therefore the



(a) $k = 2$ **(b)** $k = 3$ **(c)** $k = 4$

Fig. 3 Intra-/inter-cluster distances

inter-cluster distance is similar to the intra-cluster distance. In (b), where the *natural* partitioning is attained, the intra-cluster distance is smaller relative to the inter-cluster distance.

To implement the above intuition, we define the mean ratio of large intra-cluster distance $\bar{r}$. Let $\{\mu_i\}_{i=1}^k$ denote the cluster centroids and $\Delta\mu_{ij} = D(\mu_i, \mu_j)$. We denote by $\{\bar{d}_i\}_{i=1}^k$ the average distance from the members of ith cluster to the centroid μ_i. The ratio among the cluster centroids r_{ij} is defined as $r_{ij} = \frac{\Delta\mu_{ij}}{\bar{d}_i}$.

We denote the top five percent of the ratios as

$$M' = \left\{ r_{ij} : r_{ij} \geq Q_{95}(M) \right\},$$

where $M = \{r_{ij}\}_{i,j\in\{1,\ldots,k\}}$ and $Q_{95}(M)$ denotes the 95-percentile of M.

Taking the mean of ratios over M', $\bar{r}$ is defined as follows.

$$\bar{r} = \frac{1}{\#(M')} \sum_{r\in M'} r$$

Let k_x denote the number of clusters for clustering the auxiliary data, with which the largest $\bar{r}$ is attained. For clustering the target data, we suggest using k_g with $\alpha\%$ augmentation,

$$k_\mathrm{g} = \left(1 + \frac{\alpha}{100}\right) k_\mathrm{x}$$

since the target data may additionally include anomalous clusters.

7 Empirical evaluation

This section describes an empirical study for evaluating the ACE framework. Since most published datasets are pre-processed or come with a suggested setting for processing, we conducted the study using several raw trajectory datasets for which the appropriate settings had been unknown, in order to avoid human bias for selecting the candidate *resolutions*.

7.1 Data description

The trajectory data were taken from the video log of a multi-agent task experiments performed by autonomous robots (Kumar et al. 2010). The agent trajectories make a preferable benchmark for behavioral data mining as the controller code can be helpful for the semantic interpretation of the behaviors. For practical purposes, detecting when the agents digress from the designed task is an important task in developing real-world, autonomous agents. The developmental process involves repetitive testing to find and eliminate problematic behaviors, which are often not identified from the internal states. Many unexpected behaviors and anomalies occur even in a simple task due to their

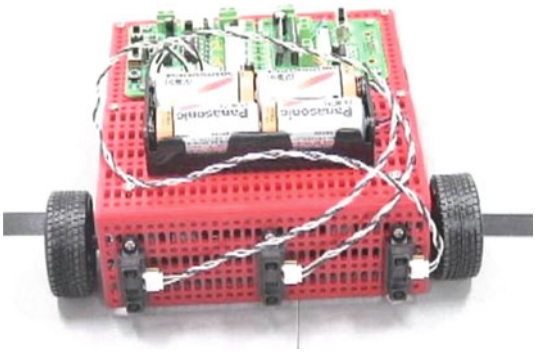


Fig. 4 Exterior view of the robot

Algorithm 4 Pseudocode of the Robot Controller

```
INPUT: sensor threshold value: l_threshold
sensor values: (l_center, l_left, l_right)
loop
  if l_center ≤ l_threshold then
    move forward
  else if l_left > l_right then
    turn right
  else
    turn left
  end if
end loop
```

complex interactions with the environment. The cost of manually reviewing the test log therefore can be significant.

The basic hardware used in Kumar et al. (2010) is RoboDesigner of Japan ROBOTECH[1] It consists of a chassis, a controller board, two motors, three wheels, a battery, and three infra-red sensors. Figure 4 shows the exterior view of the robot. The controller program is designed to let the robot roam the field without colliding with other obstacles. In short, the robot advances forward if no obstacle is detected, and otherwise makes a turn, to the left or the right depending on the sensor values. The program is described as a pseudocode in Algorithm 4.

The testing environment is a square field surrounded by blocks and walls, and each edge is 192 cm in length. Four robots using the same controller travel in the field from respective initial positions. The robots' behavior is recorded with a USB camera in a 320 × 240 pixel frame at a frame rate of 20 fps. A single run includes approximately 4,000 frames. The pixels of the robot is extracted from the video frames by edge detection, and its coordinate is given by the mean position of the pixels.

[1] http://www.japan-robotech.com.

(1) (2) (3) (4) (5)

Fig. 5 Image snapshots during the *Pivoting* behavior

7.2 Description of the anomalous behavior

The designer of the controller program examined the video logs of the experiment and found in three different runs an unexpected behavior. The robot showing this behavior alternates between turning left and right over 20 s facing the corner of the field. We refer to this behavior as *pivoting*. Figure 5 shows the image frames taken during the behavior occurred. The robot indicated by the dotted circle in the left bottom corner of the image exhibits the *pivoting* behavior. The robot is turning to its left between frames (1)–(3), and then to its right in frames (3)–(5). As the *pivoting* behavior was unexpected to the designer and disrupts the robot from traveling in the field, it is considered an anomaly and used as one of the benchmarks for anomaly detection.

7.3 Setup

Our first experiment is conducted using three trajectories from the multi-agent robot experiment. The trajectories, referred to as T1, T2, and T3, consist of 4122, 4677, and 5380 coordinates, respectively. T1 is used for setting the parameters of the proposed and baseline methods, and the rest is used for evaluation. For performance evaluation, each frame of the video is assigned a label, whether the robot is in the *pivoting* behavior, by manual inspection. The number of frames exhibiting the *pivoting* behavior were 1281, 580, and 313, in the respective datasets.[2]

The first two-thirds of the evaluation data sets were used as the target data and the rest as the auxiliary data. Figure 6 illustrates the trajectory of $T1$. The coordinates were rescaled to a 256×256 pixel space. The target and the auxiliary trajectories are shown by a blue line and an orange line, respectively.

The behavior of the robot is represented by the velocity norm feature as described in Sect. 3.1. Since the experiment setting does not suggest any dependence on the absolute coordinates or the axes orientation, it is natural to represent the behavior independently of the coordinate system. Figure 7 illustrates the time series of the velocity norm computed from T1-3 with $\delta = 2$. The blue and the orange lines correspond to the target and the auxiliary data, respectively.Note that due to the asymmetric shape of the robot, $|\mathbf{v}| > 0$ when the robot is turning.

The *resolution* of the velocity norm subsequence representation is specified by the combination of the number of frames per time interval δ and window size $w = 2\lambda + 1$.

[2] The dataset is a property of JST-ANR project (http://www.i.kyushu-u.ac.jp/~suzuki/jstanr.htm) and available from the project web page.

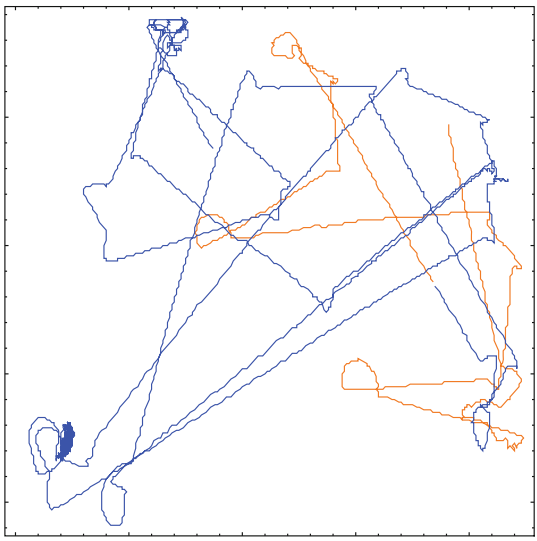


Fig. 6 Trajectories of T1 dataset

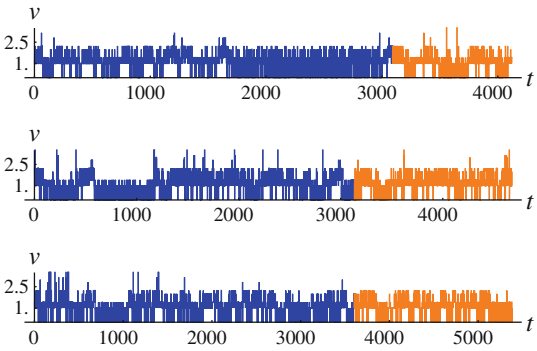


Fig. 7 Velocity norm time series of T1-3 ($\delta = 2$)

From a brief review of the auxiliary data, we defined the *resolutions* $\boldsymbol{\Theta}$ as a set of combinations from $\delta \in \{1, 3, 9, 27\}$ and $\lambda \in \{2, 6, 12, 18\}$. The same number of sequences are generated in all candidate *resolutions*. The number of subsequences in the target/auxiliary sets generated from T1, T2, and T3 are: 3074/1012, 3116/1525, 3587/1757, respectively.

Based on a preparatory experiment using T1, the numbers of clusters for the SAC and for *meta*-feature clustering were set to $k = 10$ and $\hat{k} = 16$, respectively. The size of the bootstrap samples for the rank-sum test $n_b = 100$, the confidence level for the rank-sum test $\alpha = 1.0 \times 10^{-5}$, and the size of the *meta*-reference vectors $\rho = 200$, are chosen for statistical robustness.

7.4 Validation data description

For the purpose of validation, we apply ACE to independent sets of trajectories with the same setup. We introduce two more trajectories taken from another multi-agent

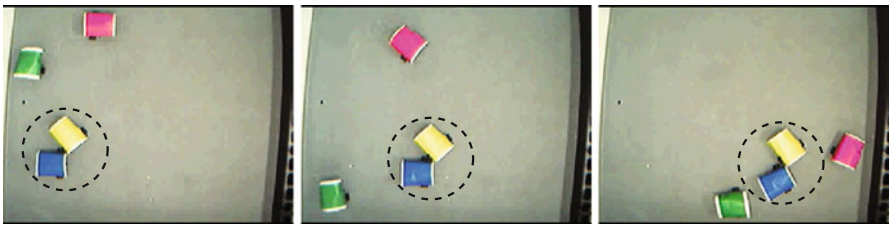


Fig. 8 Snapshots during a *Push-Slide* behavior

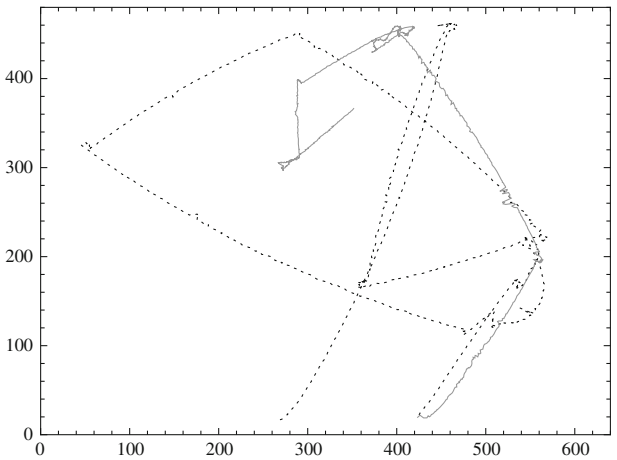


Fig. 9 Trajectories of Y1 dataset

experiment. In the second experiment, a different type of unexpected behavior was observed when two robots run into each other and proceed to push against each other continuously while moving in a diagonal direction or in a curve. We refer to the behavior as the *push-slide*. Figure 8 shows a sequence of images during this behavior. Two robots in the dotted circle are pushing against each other and moving toward the bottom-left corner.

Two segments of a robot's trajectory, i.e., a target segment containing the *push-slide* and an auxiliary segment containing only normal behaviors, were extracted from the logs of two trials. The coordinates of the robot in each snapshot is given by the mean position of the same-colored pixels. We refer to the two pairs of trajectories as Y1 and Y2. The target/auxiliary segments of Y1 and Y2 consist of 2,500/1,200 and 1,100/500 points, respectively. Among the target data there are 600 and 430 instances of the *push-slide* behavior. Figure 9 illustrates the trajectories of Y1 in a 2D-plot. The target and auxiliary segments are indicated by dotted and solid lines respectively. The velocity norm features are computed from the trajectory data using the same resolutions as in the previous experiment. Figure 10 illustrates the velocity norm sequences for $\delta = 2$.

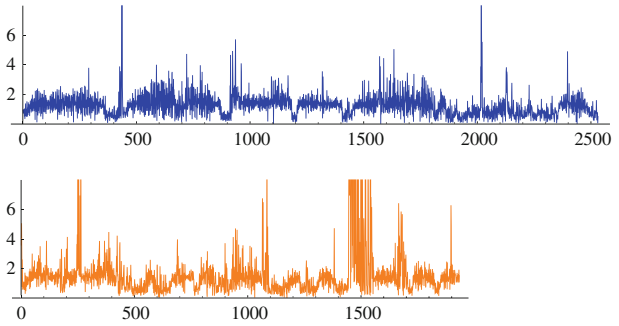


Fig. 10 Sequences of velocity norm from Y1

7.5 Evaluation

Controlling the rate of false alarm, also referred to as the type I error, is one of the key properties of the detection technique. Receiver operating characteristic (ROC) is used to measure the capability of the detection techniques to control the type I and type II errors with its parameters (Webb and Copsey 2011).

The ROC curve depicts the trade-off between the true positive rate (TPR) and the false positive rate (FPR) by plotting all possible instances of confusion matrices respectively as a coordinate (TPR, FPR). The area under the ROC Curve (AUC) is a scalar value which provides a summary of ROC. AUC takes a value from the interval of [0:1], and a value less than 0.5 indicates that the prediction is negatively correlated with the true label. AUC is directly related to the U statistic of the rank-sum test. A larger AUC assures that one can reject, with a higher confidence level, the null hypothesis that the mean scores of the anomalies and the normal samples are equivalent. We evaluate the output of the ACE framework by AUC, and it is compared against that of the baseline ensemble approach.

7.6 Baseline methods

7.6.1 Feature bagging

For comparison, we employ feature bagging (FB) (Lazarevic and Kumar 2005) as a baseline ensemble method. FB generates its scores by combining those of *weak* detectors executed in randomly selected data subspaces. The higher scores signify probable anomalies. In Lazarevic and Kumar (2005), LOF (Breunig et al. 2000) was used as the *weak* detectors and two approaches were proposed for combining the scores of the *weak* detectors: Cumulative sum (CS) and the breadth-first (BF).

The random selection of subspaces in FB corresponds to choosing the resolutions to process the time series. We adapt FB to the multi-resolution setting by first randomly selecting a subset $\Theta_{FB} \subset \Theta$ and computing the individual scores of the *weak* detector over $\{S_\theta\}_{\theta \in \Theta_{FB}}$. Individual scores are then aggregated by CS or BF approaches.

We implement FB with a variety of *weak* methods, including: discord discovery (DD) (Yankov et al. 2008), Distance-based outlier detection (Angiulli et al. 2006) with partially-labeled data, and the angular velocity score, in addition to LOF in order to obtain a robust baseline performance. LOF is considered one of the best general outlier detection method (Noto et al. 2012), but its original formulation is known to be computationally expensive. DD is a highly scalable anomaly detection method developed for sequential data, such as time series and shapes. Both LOF and DD can benefit from sophisticated approximations in terms of efficiency (Ghoting et al. 2008; Yankov et al. 2008). For the purpose of our experiment, however, they provide sufficient baselines in terms of effectiveness without the approximations.

We introduce the distance-based outlier detection with partially-labeled data (DBPL) as a semi-supervised anomaly detection method. In the distance-based outlier detection, each sample is assigned a score associated with the nearest-neighbor distance (Angiulli et al. 2006). For DBPL, we define the score of an instance as the distance to the k_Ath nearest auxiliary data, to exploit the information of the normal instances.

The angular velocity is included as an application-specific measure for detecting the *pivoting* behavior. The change in the angle of the robot was suggested by a person with the knowledge of the behavior and the agent's controller as an intuitive indicator. Based on the cosine distance between the velocity at t and $t-1$, the angular velocity AV(t) is defined as follows.

$$\mathrm{AV}(t) = \sum_{\mathbf{v}(t) \in s(t)} 1 - \frac{\mathbf{v}(t) \cdot \mathbf{v}(t-1)}{|\mathbf{v}(t)||\mathbf{v}(t-1)|} \tag{20}$$

Though the suggestion was insightful, the *resolution* is also relevant to this feature as its definition (20) includes the velocity of the agent.

Except for the angular velocity score, the *weak* detection methods use the same numerical representation defined in Sect. 3.1. The LOF and Discord Discovery compute the score over the joint set of target and auxiliary data, but only those of the target data are considered in FB. The scores of the target data are evaluated in the ROC analysis. Instances with relatively higher scores are predicted as probable anomalies. We conduct the ROC analysis to evaluate the scores of the target data.

For LOF, the parameter $MinPts = 20$ was used after comparing values in the suggested range of 10–20 (Breunig et al. 2000) using T1. For distance-based outlier with partially labeled data, we compared the value of k_A from 5 to 25 for T1. $k_A = 10, 20$ yielded the highest AUC for CS and BF approaches, respectively. We refer to the parameter of DD, the number of non-trivial neighbors, as k_{DD}. We employ $k_{DD} = 10, 20$ which respectively achieved the highest AUC for T1 with CS and BF.

Although the problem setting is semi-supervised, *weak* detectors of unsupervised learning techniques are not at a complete disadvantage with regards to the information of normal instances. They are able to exploit the information to a degree by assuming that they are the significant majority, which holds true for this experiment. The SAC cannot be used in combination with FB because the output of the SAC is vectorial rather than the scalar score. Conversely, the *weak* detectors of FB cannot be used directly

in the proposed framework. Our experiment is thus designed to compare between the two ensemble frameworks rather than the individual *weak* detection methods.

Note that the techniques of outlier trajectory detection are omitted due to their dependence on the absolute coordinates/location. They primarily address trajectories extracted from surveillance images or over geographical tracking data (Lee et al. 2008), in which the trajectories overlapping with or traced closely by many other trajectories are considered *normal*. As Figs. 6 and 9 show, the agents can move with much degree of freedom and are unlikely to generate overlapping trajectories. For the problem at hand, the normality is defined instead at the behavior level, independently of the absolute coordinates. As such, the outlier trajectory detection techniques would not provide a good baseline performance.

7.6.2 Feature regression and classification

As a second baseline method, we employ another ensemble outlier detection framework called feature regression and classification (FRaC) which has shown *state-of-the-art* performances (Noto et al. 2012). The ensemble framework of FRaC takes after FB, more specifically the Cumulative sum method, for aggregating the scores of the weak outlier detection. The *weak* outlier detection in FRaC is a semi-supervised learning method based on the concept of feature modeling, using the regression models of individual feature values. Each regression model is trained to predict one feature using other features as the input using the training data that consists entirely of normal instances. To test a new instance, its feature values are substituted into each regression model to compute the probability of the output value. The negative logarithm of the probability is used as a numerical score quantifying the discrepancy from the normality. The cumulative score of an instance is computed by taking the summation of scores from all models.

For our experiment, the implementation of FRaC for continuous features with no missing values[3] were used. Additionally, FRaC requires the user to select the input and output variables of the regression models. We considered two settings: first, as suggested in Noto et al. (2012) for cases without feature-specific knowledge, the model of all features are learned respectively using all other features. Secondly, considering the different resolutions, each feature is modeled using all other features of the same resolution. We refer to the former as FRaC(A) and the latter as FRaC(R). The FRaC framework is directly applicable to the semi-supervised setting of our experiment, i.e., the auxiliary data is used for its training and the target data for its testing.

7.7 Results

First, we graphically analyze the typical output from the SAC and ACE. Figure 11 illustrates the *meta*-feature matrix $\mathbf{B}_\Theta$ and the summarized *meta*-feature matrices $\hat{\mathbf{B}}_\Theta$ from T1. In the top figure, the binary-valued elements of $\mathbf{B}_\Theta$ are indicated as black and white cells, respectively representing 1 and 0. Each column of the matrix represents

[3] http://bcb.cs.tufts.edu/frac/.

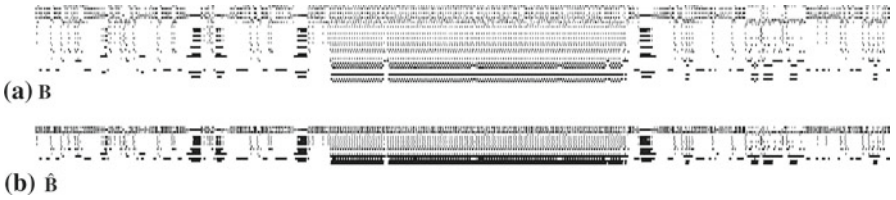


Fig. 11 Illustration of *Meta*-feature vectors (T1)

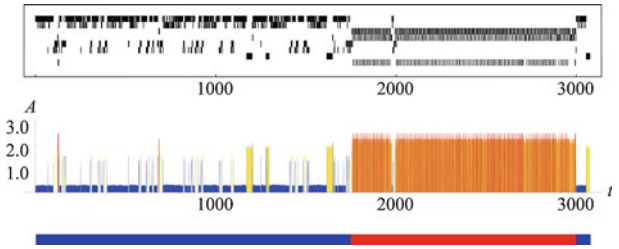


Fig. 12 Illustration of *Meta*-clusters and anomaly score (T1)

a *meta*-feature vector $\mathbf{m}(t)$. Vectors are aligned in the ascending order of t along the horizontal axis. The bottom figure illustrates the summarized matrix $\hat{\mathbf{B}}_{\Theta}$ in the same manner such that each column represents $\hat{\mathbf{m}}(t)$. Horizontal axes of the two figures are aligned such that $\hat{\mathbf{m}}(t)$ is placed under $\mathbf{m}(t)$ in the same horizontal position. The figures indicate that most instances are considered more or less anomalous in different *resolutions*.

Figure 12 illustrates the *meta*-clustering $\mathbf{c}(t)$, the anomaly scores $A(t)$, and the true label from T1 against time. In the top figure, the clustering of the *meta*-feature vectors is shown as a binary matrix. The column at t indicates the cluster assignment of $\mathbf{m}(t)$ by the row position of the black cell. For example, a black cell at the top row of the tth column indicates that the behavior instance is assigned to the first *meta*-cluster. If the black cell is at the bottom, the instance is assigned to the $\hat{k}$th cluster. The anomaly score is illustrated by the bar plot in the middle. The x- and y-axes indicate t and $A(t)$, respectively. The higher values of $A(t)$ are indicated by the warmer colors. The bar at the bottom illustrates the true label based on the human inspection. The red region corresponds to the occurrence of the *pivoting* behavior. Similarly, Figs. 13 and 14 illustrate typical outputs from T2 and T3.

The figures show that in all datasets, the *pivoting* behavior corresponds well with the high anomaly scores of ACE. In Fig. 12, the binary data matrix shows that the *meta*-clustering divides sequences in the *pivoting* behavior among three clusters. They can be seen as different phases of the *pivoting* behavior, e.g., stopping, restarting, and turning. Although non-*pivoting* behaviors were frequently labeled as anomalies in few *resolutions*, the patterns did not deviate collectively from the *meta*-reference vectors and yielded lower scores.

In Fig. 13, the bottom bar shows that T2 includes two independent occurrences of the *pivoting* behaviors. As shown in the binary data matrix, both are assigned to the same *meta*-cluster. Figure 14 shows that short sequences of the non-*pivoting* state in

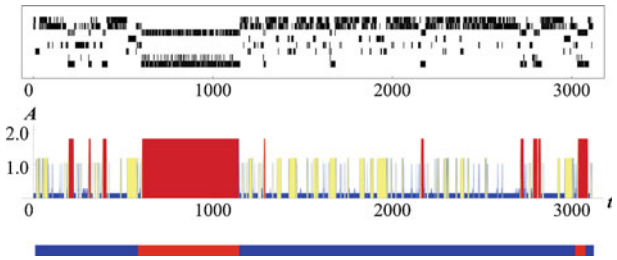


Fig. 13 Illustration of *Meta*-clusters and anomaly score (T2)

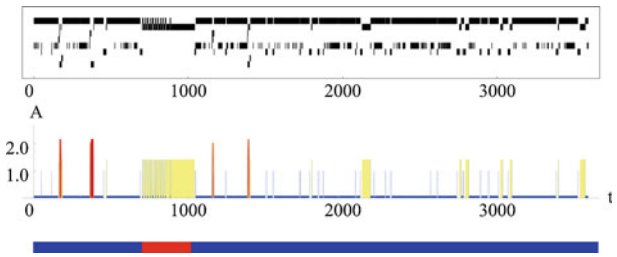


Fig. 14 Illustration of *Meta*-clusters and anomaly score (T3)

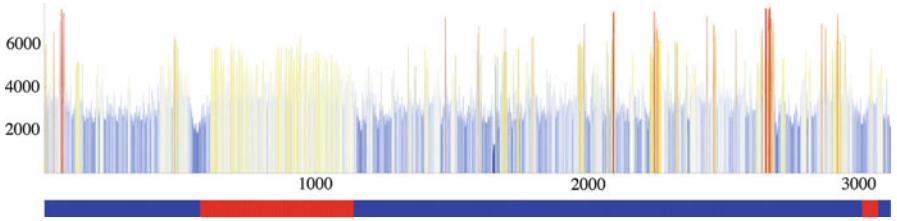


Fig. 15 Illustration of anomaly score of LOF ensemble (T2)

T3 are clustered together with the *pivoting* behavior. Examining the video log, we found that this counterintuitive output is caused by the *pivoting*-like behavior around $t = 1,500$ where the robot attempts to avoid another robot in its path.

Figure 14 also shows a small portion of the *pivoting* behavior included in the first cluster, which consists mostly of non-*pivoting* behaviors. Upon further review, we found that few times in the corresponding part of the log, the agent momentarily breaks from the *pivoting* behavior and advances toward the corner. Though it is arguable that these instances are mislabeled, we maintain the labels and the evaluation *as-is*, in order to avoid arbitrariness.

As typical examples of the baseline methods, LOF values are shown in Figs. 15 and 16. The bar plots illustrate the CS anomaly scores for T2 and T3, respectively. The bar on the bottom indicates the occurrence of the *pivoting* behavior in the similar manner as in Fig. 13. In Fig. 15, the ensemble of LOF scores during the *pivot* is slightly higher on average than the rest. However, the score is significantly inconsistent thus less practical compared to that of ACE. For T3, the difference in anomaly score during the *pivoting* behavior is less significant com-

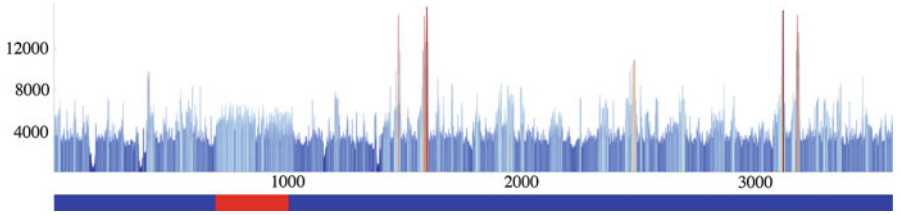


Fig. 16 Illustration of anomaly score of LOF ensemble (T3)

Table 1 Summary of AUC (T1-3)

	T1		T2		T3	
	μ	σ	μ	σ	μ	σ
LOF–CS	0.76	0.094	0.78	0.053	0.73	0.067
LOF–BF	0.70	0.00030	0.63	0.028	0.79	0.043
DD–CS	0.60	0.050	0.51	0.049	0.45	0.030
DD–BF	0.62	0.053	0.50	0.058	0.56	0.053
DBPL–CS	0.86	0.0073	0.69	0.027	0.62	0.028
DBPL–BF	0.77	0.0052	0.54	0.0052	0.48	0.027
AV–CS	0.92	0.030	0.82	0.024	0.71	0.040
AV–BF	0.82	0.0040	0.75	0.0068	0.71	0.0073
ACE	**0.97**	0.023	**0.93**	0.012	**0.87**	0.028
FRaC(A)	0.88	–	0.92	–	0.85	–
FRaC(R)	0.94	–	0.65	–	0.73	–

pared to that of ACE. From the graphical comparison, the results of the proposed framework are more consistent and discriminative for detecting the anomalous *pivot* behavior.

The comparison of ROC analyses is summarized in Table 1. As ACE and FB include stochastic processes, they are iterated from different initializations. The mean and the standard deviation of the AUC over ten iterations are shown in the columns indicated by μ and σ, respectively. In each column under μ, the largest value is indicated by bold font. Note that T1 is used for parameter selection. In both T2 and T3, ACE shows a significant advantage over the baseline methods. The lower AUC for T3 can be attributed to the previously discussed mislabeling. Among the baseline methods, the FRaC methods achieve the better AUC over FB methods overall. The advantage between the two FRaC methods differ by the data.

Table 2 summarizes the ROC analyses of the baseline *weak* detectors prior to the aggregation. It shows that LOF and DBPL perform as well as the proposed ensemble framework in the *best* setting, but substantially worse in many others. This variance demonstrates that some anomalies are detected more easily within a limited range of parameters as we had considered in our initial motivation. Further, the ROC analysis of the baseline ensemble method demonstrates its limitation in such a case.

Table 2 Summary of AUC at individual *resolutions*

	Min.	1Q	Med.	3Q	Max.	μ	σ
T1							
LOF	0.18	0.23	0.42	0.73	0.90	0.48	0.27
DD	0.12	0.32	0.57	0.68	0.75	0.50	0.46
DBPL	0.44	0.49	0.75	0.87	0.92	0.69	0.18
AV	0.52	0.64	0.65	0.66	0.71	0.64	0.24
FRaC	0.49	0.68	0.83	0.88	0.96	0.78	0.15
T2							
LOF	0.12	0.32	0.70	0.84	0.90	0.59	0.29
DD	0.12	0.29	0.40	0.70	0.80	0.46	0.48
DBPL	0.39	0.51	0.63	0.76	0.81	0.63	0.14
AV	0.57	0.57	0.58	0.61	0.65	0.60	0.19
FRaC	0.41	0.49	0.55	0.67	0.71	0.57	0.12
T3							
LOF	0.18	0.38	0.50	0.66	0.82	0.51	0.19
DD	0.22	0.43	0.50	0.64	0.67	0.50	0.37
DBPL	0.36	0.51	0.55	0.68	0.81	0.60	0.12
AV	0.53	0.53	0.53	0.57	0.61	0.55	0.17
FRaC	0.46	0.56	0.63	0.68	0.79	0.63	0.087

Table 3 Summary of AUC (Y1/2)

	Y1		Y2	
	μ	σ	μ	σ
DBPL–CS	0.79	0.039	0.70	0.021
LOF–BF	0.64	0.012	0.49	0.016
AV–BF	0.75	0.014	0.66	0.053
ACE	**0.85**	0.041	**0.75**	0.043
FRaC(A)	0.79	–	0.71	–
FRaC(R)	0.66	–	0.70	–

7.8 Results (validation data)

We apply ACE to Y1/Y2 datasets using the setting from the previous experiment, i.e., $k = 10$, $\alpha = 1.0 \times 10^{-5}$, $\hat{k} = 16$, $n_b = 100$. We also tested the combinations of the baseline ensemble schemes and the *weak* detection methods that performed better in the previous experiment: CS with DBPL and FB with LOF/AV. The parameters of DBPL and LOF are also the same from the previous experiment.

Table 3 summarizes the AUC of ACE and baseline methods for Y1/Y2 datasets. Each mean and standard deviation is computed over 20 iterations. In each column under μ, the largest value is indicated by bold font. ACE shows significant advantages

Table 4 Summary of AUC at individual *Resolutions* (Y1/2)

	Min.	1Q	Med.	3Q	Max.	μ	σ
Y1							
DBPL	0.60	0.69	0.82	0.85	0.85	0.77	0.10
LOF	0.52	0.58	0.61	0.65	0.70	0.61	0.053
AV	0.70	0.74	0.76	0.82	0.84	0.78	0.049
FRaC	0.40	0.43	0.48	0.71	0.82	0.56	0.024
Y2							
DBPL	0.62	0.69	0.75	0.79	0.81	0.73	0.065
LOF	0.46	0.49	0.52	0.56	0.61	0.53	0.047
AV	0.30	0.36	0.46	0.56	0.61	0.46	0.11
FRaC	0.64	0.65	0.65	0.67	0.76	0.66	0.0011

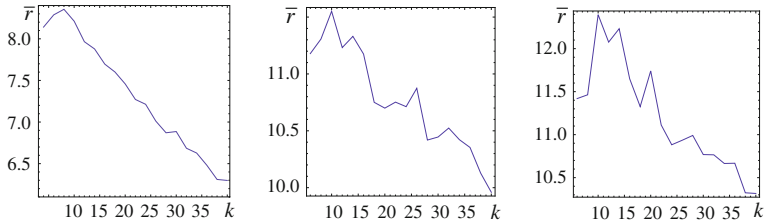


Fig. 17 Ratio of large intra-cluster distance

over the baseline methods for both datasets. Table 4 summarizes the AUC of the weak detectors for proposed and baseline methods. In most cases, BF and CS ensemble yield only slightly better or worse AUC than the mean of individual detectors. The FRaC framework, which employs CS for aggregating the individual scores, exhibits similar limitations. Although FRaC achieves lower standard deviation, its overall score is still significantly worse than the best individual AUC. Exceptionally for Y2, the ensemble of angular velocity achieves significantly better AUC than individual detectors. The above results demonstrate the difficulty of aggregating the scores by an unweighted summation for anomaly detection, and supports our motivation to employ the *meta*-feature representation.

7.9 Parameter selection scheme

In this section, we evaluate the performance of ACE using the parameter selection scheme described in Sect. 6. From the auxiliary dataset of the validation datasets, $\bar{r}$ is computed while changing k from 6 to 40 with the interval of 2. Here, it is assumed that the fundamental structure, namely the number of clusters, remain consistent in all *resolutions*. In general, the proposed scheme may be used to choose different k for respective *resolutions*.

Table 5 Summary of AUC (with different k_g)

	T1		T2		T3		Y1		Y2	
α	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0	0.95	0.035	0.93	0.012	0.87	0.028	0.82	0.031	0.74	0.040
20	0.97	0.023	0.93	0.036	0.82	0.065	0.80	0.025	0.74	0.033
50	0.97	0.031	0.93	0.026	0.82	0.039	0.81	0.028	0.76	0.045
100	0.94	0.039	0.91	0.041	0.78	0.040	0.82	0.046	0.73	0.052

In Fig. 17a–c, $\bar{r}$ is plotted against k for $T2$, $T3$ and $Y1$, respectively. The figures show that the largest $\bar{r}$ was achieved when clustering the auxiliary dataset with $k_x = 8$ for T1, $k_x = 10$ for T2/3 and $k_x = 12$ for Y1/2, respectively.

We further evaluated the ROC of ACE using k_g for the SAC, with α ranging from 0 to 100. The results are summarized in Table 5. Based on the consistency of AUC among the tested values, we suggest k_g with $\alpha = 20$–50 as practical values for the general case.

7.10 Corruption of normal examples

In our next experiment, we investigate the robustness of the proposed algorithm in a practical scenario where the auxiliary dataset is corrupted, i.e., consists primarily of normal instances but contains a small number of anomalies. In Noto et al. (2012), FRaC is evaluated in the unsupervised anomaly detection problem of a similar setting. This experiment is more about robustness to auxiliary data corruption. In the ACE framework, the rank-sum test is the basis of robustness against such corruption. Provided that the portion of the anomalies are small, the distributions of distances would differ such that the null hypothesis cannot be rejected.

The target and the auxiliary data for this experiment are generated as follows. First, the target and the auxiliary data are divided as described in Sects. 7.3 and 7.4. Then, a small number of the anomalous instances is moved from the target data to the auxiliary data. These instances are chosen by the seeding algorithm of k-means++ (Arthur and Vassilvitskii 2007) to ensure that they are not uniformly similar and are scattered across the timeline. The amount moved is chosen such that it would account for 5 % of the new auxiliary data. The new target and auxiliary datasets are used in the same manner in the ACE framework. We refer to the modified datasets of the pivot behaviors and the collision behaviors as: T1C, T2C, T3C, Y1C, and Y2C, respectively. The number of the target/auxiliary data in respective datasets are 3063/1023, 1541/3100, 1775/3569, 2524/2482, 2218/2272.

The ROC analyses of ACE and FRaC in the corrupt setting are summarized in Table 6. In each column under μ, the largest value is indicated by bold font. Their parameters were carried over from the settings in Sect. 7.3. Their performances are fairly robust against the corrupted normal examples, exhibiting only slight degradation

Table 6 Summary of AUC (with corrupted auxiliary data)

	T1C		T2C		T3C		Y1C		Y2C	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ACE	**0.94**	0.063	**0.91**	0.039	**0.86**	0.051	**0.84**	0.045	**0.75**	0.050
FRaC(A)	0.87	–	**0.91**	–	0.81	–	0.80	–	0.71	–
FRaC(R)	**0.94**	–	0.64	–	0.71	–	0.67	–	0.70	–

Table 7 Summary of time series benchmarks

	Sequence length	# of classes	# of time series	Default train/test
Synthetic Functions (Saito 1994)	129	3	300	Yes
Control Chart (Pham and Chan 1995)	60	3	600	Yes
Handwriting (Williams et al. 2007)	108	20	2,549	No
Wall Following (Freire et al. 2009)	5,456	4	1	No

from the previous experiment. ACE has the overall advantage which can be attributed to its approach for aggregating multiple resolutions.

8 Application to time series data

So far, we have applied ACE to the time series of the behavioral feature generated from trajectories. Meanwhile, smoothing and segmentation are common procedures for learning from time series data, and thus are capable of causing a similar problem in multiresolution analysis. In this section, we investigate the capability of ACE in anomaly detection from time series and sequences in general. We extend our empirical study with fifteen anomaly detection problems based on artificial and real-world time series datasets from the publicdomain.

8.1 Data description

We employ two artificial datasets, which have been used frequently as benchmarks in temporal data mining research, and two real-world datasets made available at the UCI machine learning repository (MLR) (Bache and Lichman 2012). Originally, these datasets have been studied as benchmarks for multi-class classification. In previous studies of anomaly detection, such datasets have been used for evaluating detection methods. We follow their *one-against-rest* formalization to set up the target and auxiliary data for our experiment. First, one of the classes is chosen as the class of anomalies, and their samples are removed from the training set. We employ the modified training set as the auxiliary data. The test set is used as the target data without change. We show the summary of the datasets in Table 7 and describe their background and the multi-resolution setup in the following sections.

8.1.1 Artificial datasets

"Synthetic Functions" and "Control Chart" are datasets consisting of time series of different shapes, generated by formulations proposed in Saito (1994) and Pham and Chan (1995), respectively. The former includes three classes: bell, cylinder, and funnel, and the latter includes six classes: *normal*, *cyclic*, *increasing*, *decreasing*, *upward*, and *downward*. From each dataset, we set up the same number of *one-against-rest* problems as the classes, such that each class is a target anomalous class in one of the problems.

We employ moving average to smooth each time series and consider the window size m as one of the parameters of the *resolution*. We adopt the length of the time series as the second parameter related to the *resolution*. Considering that the observation of time series is sequential, it is practical for a recognition application to make a prediction once a sufficient amount of the time series has been observed, rather than to wait for it to complete. In our experiment, a time series instance of a *resolution* (l, m) is generated by truncating each time series to length l by removing the final values of each series and then taking the moving average by window size m.

8.1.2 Handwriting character trajectories

"Character Trajectories" consists of pen-tip trajectories of single *pen-down* characters, including four vowels and sixteen consonants. The length of trajectories differ from 109 to 205 depending on the character. The trajectories are captured using a tablet PC at 200 Hz and have been smoothed and shifted in the pre-process (Williams et al. 2007). We set up four *one-against-rest* anomaly detection problems where each vowel is designated as the class of anomalies and another problem in which the instances of all four vowels are considered anomalies.

We consider the combination of the re-sampling rate r and the trajectory length l as the *resolution*. Let $\mathbf{x}(i)$ denote a coordinate and $\mathscr{X} = \{(i)\}_{i=1}^{L}$ a character trajectory. At the *resolution* (l, r), the representation of a character instance $s_{l,r}$ is given as follows.

$$s_{l,r} = \{\mathbf{x}(i) : i \leq l, Mod_r(i) = 1\} \tag{21}$$

The distance between two trajectories is defined as the 2-norm of the differences of their coordinates. Let $\mathscr{X}_1 = \{\mathbf{x}_1(i)\}_{i=1}^{L}$ and $\mathscr{X}_2 = \{\mathbf{x}_2(i)\}_{i=1}^{L}$ denote the sets of coordinates from two character trajectories, respectively. The distance $D(\mathscr{X}_1, \mathscr{X}_2)$ is given as follows.

$$D(\mathscr{X}_1, \mathscr{X}_2) = \left\| \left[\mathbf{x}_1(1) - \mathbf{x}_2(1), \ldots, \mathbf{x}_1(L) - \mathbf{x}_2(L) \right] \right\|_2$$

8.1.3 Wall-following robot sensor data

"Wall-following Robot Navigation Data" is generated from the sensor readings observed in a single-robot experiment described in Freire et al. (2009). The robot is equipped with 24 ultrasound sensors, each recorded at a rate of 9 samples per second. The robot is controlled by a simple rule-based program for navigating through a closed

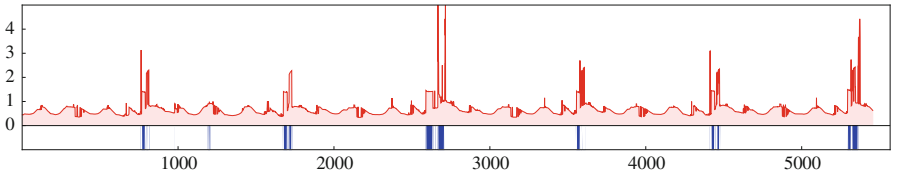


Fig. 18 Actions and sensor readings of wall-following robot

environment. The controller continually chooses one of four actions: *Move-Forward*, *Slight-Right-Turn*, *Sharp-Right-Turn*, and *Slight-Left-Turn*.

The dataset includes the time series of the nominal actions and the simplified distance, a feature proposed in the original study. The lengths of both time series are 5,456. The simplified distance is defined as the minimum value of readings among the sensors located within 60 degree arcs at the front parts of the robot. It is used in a classification problem to predict the action of the robot.

In the following experiment, we attempt to detect the least frequent class of action, "Slight-Left-Turn", as the anomalies, based on the preceding sequence of simplified distance. Figure 18 illustrates the time series of simplified distance and the actions of the robot. The horizontal and vertical axes represent the time and the value of simplified distance. The red line indicates the time series of simplified distance and the blue bars indicate the occurrence of the Slight-Left-Turn action.

We employ moving average of window size m for smoothing and sliding window of window size l to generate the segmented instances. The *resolution* is represented by a combination (l, m).

8.2 General setup

For the problems of "Synthetic Functions" and "Control Chart" datasets, the sequence length l is taken from {70, 90, 110, 129} and {40, 50, 60}, respectively. For the "Wall-following Robot" dataset, the subsequence length l is taken from {80, 90, 100}. The window size m for taking the moving average of the above three datasets is taken from {1, 3, 7, 15}. For the "Character Trajectories" dataset, the sampling rate r and the trajectory length l are taken from {1, 2, 3} and {72, 84, 96, 108}, respectively.

The properties of the respective anomaly detection problems are summarized in Table 8. The ratios of auxiliary data and the anomaly class examples are taken against the number of target data. The source code for generating all datasets are provided at the author's website, and the instructions are provided in the supplementary material.[4]

8.3 Results

Table 9 shows the AUC achieved by the baseline and the proposed methods in all fifteen anomaly detection problems. The mean and the standard deviation are taken

[4] http://keis.ei.st.gunma-u.ac.jp/~ando/expdat_dami.html.

Table 8 Anomaly detection problem setup

	Anomaly class	Auxiliary data ratio	Anomaly class ratio
Syntehtic Function	Bell	0.857	0.143
	Cylinder	0.857	0.143
	Funnel	0.857	0.143
Control Chart	Normal	0.833	0.167
	Cyclic	0.833	0.167
	Increasing	0.833	0.167
	Decreasing	0.833	0.167
	Upward	0.833	0.167
	Downward	0.833	0.167
Handwriting Trajectory	All Vowels	0.594	0.200
	'a'	0.698	0.059
	'e'	0.698	0.059
	'o'	0.698	0.059
	'u'	0.698	0.059
Wall Following Robot	Slight-Left-Turn	0.510	0.060

over 20 runs. The problems from "Synthetic Functions" are indicated by the class name of the anomalous function. Those from "Control Chart" are abbreviated as CC-s, string s indicating the class of anomalies. The problems of "Character Trajectories" are abbreviated as HW-c, c being the anomalous vowel character.

Table 9 shows that ACE achieves the best AUC in 10 problems, which is more than any of the baselines. It also achieves the next to best AUC in the rest of the problems. Looking at the individual results, the Cumulative Summation approach performs well when all weak methods perform generally well. In combination with DBPL, it achieves the best AUC in four problems which is the second most to ACE. BF approach performs relatively well when the variance is large, in which case the Cumulative Summation performs poorly.

9 Discussion

9.1 Anomaly detection by SAC

Clustering is an essential function for the SAC, as it aims to detect agglomerated anomalies based on the cluster analysis. We first discuss the issue of *meaningless* clustering, which is a critical caveat for STS clustering. It has been reported in Keogh and Lin (2005) that the subsequences generated by the sliding window exhibit correlations that are independent of the original data. The sliding window generates subsequences which continuously transition from an initial pattern to the final pattern over any segment of the time series. As a result, for all subsequences, there exist significantly similar subsequences in the adjacent windows. Subsequences that are similar by virtue

Table 9 Summary of AUC (time series anomaly detection)

	FB						ACE
	LOF		DD		DBPL		
	CSUM	BF	CSUM	BF	CSUM	BF	
Bell							
μ	0.526	0.661	0.794	0.675	0.871	0.611	**0.923**
σ	0.0326	0.0373	0.0290	0.0614	0.0163	0.0429	0.0424
Cylinder							
μ	0.514	0.550	**0.995**	0.838	0.961	0.852	0.966
σ	0.00813	0.0157	0.00101	0.0242	0.00681	0.0178	0.00351
Funnel							
μ	0.521	0.500	0.709	0.432	0.702	0.427	**0.891**
σ	0.0564	0.0181	0.0464	0.0718	0.0314	0.0460	0.0352
CC-NRM							
μ	0.726	0.643	0.509	0.499	0.772	0.855	**0.934**
σ	0.0152	0.0111	0.0409	0.0196	0.0232	0.0153	0.0534
CC-CYC							
μ	0.529	0.576	0.522	0.497	**0.966**	0.741	0.946
σ	0.0672	0.0343	0.0526	0.0300	0.00992	0.0401	0.0443
CC-INC							
μ	0.599	0.586	0.500	0.559	0.870	0.456	**0.947**
σ	0.0199	0.0124	0.0174	0.0176	0.0155	0.0244	0.0539
CC-DEC							
μ	0.600	0.528	0.442	0.496	0.837	0.424	**0.968**
σ	0.0175	0.0125	0.00810	0.0377	0.0507	0.0283	0.0350
CC-UWD							
μ	0.499	0.455	0.652	0.470	0.**964**	0.517	0.918
σ	0.0427	0.0294	0.0113	0.0323	0.0123	0.0475	0.0572
CC-DWD							
μ	0.445	0.505	0.635	0.496	0.**952**	0.489	0.920
σ	0.0250	0.0128	0.00898	0.0183	0.0121	0.0239	0.0563
HW-All							
μ	0.530	0.552	0.449	0.492	0.545	0.644	**0.868**
σ	0.00246	0.00146	0.00238	0.0146	0.00612	0.0381	0.0391
HW-a							
μ	0.502	0.517	0.458	0.489	0.459	0.495	0.**876**
σ	0.00211	0.00150	0.00424	0.0137	0.00461	0.0783	0.0493
HW-e							
μ	0.512	0.512	0.701	0.623	0.**918**	0.863	0.865
σ	0.00340	0.00146	0.003685	0.0201	0.000519	0.0386	0.0444

Table 9 continued

	FB						ACE
	LOF		DD		DBPL		
	CSUM	BF	CSUM	BF	CSUM	BF	
HW-o							
μ	0.543	0.562	0.475	0.510	0.525	0.592	**0.873**
σ	0.0113	0.00198	0.00141	0.0168	0.00911	0.0118	0.0550
HW-u							
μ	0.557	0.621	0.442	0.560	0.524	0.676	**0.895**
σ	0.0149	0.00284	0.0143	0.0502	0.0103	0.0459	0.0587
Slight-Left-Turn							
μ	0.551	0.660	0.518	0.571	0.818	0.817	**0.904**
σ	0.00830	0.0198	0.0115	0.0131	0.00736	0.00881	0.0388

of adjacency are called the *trivial matches* in Keogh and Lin (2005). Since the density-based approach finds anomalies in the sparse region of the dataset, it is problematic that the *trivial matches* fill the gap between the normal and anomalous patterns. It is discussed in Chiu et al. (2003) and Yankov et al. (2008) in detail, that the transitional patterns critically affect the density-based anomaly detection methods.

The presence of transitional patterns provides additional motivations for the cluster-based and the semi-supervised approaches of the SAC. More specifically, it finds anomalies (1) as a cluster of instances rather than as an isolated instance, and (2) based on the relative densities from the set of normal instances. Let us consider the case that the target data are contaminated and thus contains the anomalies and the transitional patterns to normal subsequences. With regards to (1), the transitional patterns are likely to be clustered together with the relevant anomalous subsequence. With regards to (2), given that the auxiliary dataset contains none of or a significantly small number of anomalous patterns, the approximate density of the anomalous and transition pattern subsequences are smaller in relativity. The SAC is therefore capable of identifying the cluster containing the anomalous pattern in these cases. In the alternative case that the target dataset does not contain anomalous patterns, the trivial matches are generated identically for the target and auxiliary data by the sliding window, and do not cause difference in their distributions of subsequences.

It is also discussed in Chiu et al. (2003), Keogh and Lin (2005) that smooth patterns generally have more *trivial matches* than noisy or high frequency patterns. This is a cause for a concern that the clusters of such patterns can be overlooked by the clustering algorithm. To investigate this concern, we visually examine the cluster centers obtained from the experimental data. Figure 19 shows the cluster centers from a typical clustering of subsequences from T1 with $k = 10$. In each plot, horizontal and vertical axes indicate the time and the velocity norm, respectively. The values over the plot indicate the cardinality of the cluster and the mean and the standard deviation of within-cluster distances. From Fig. 19, we can observe that larger clusters do not

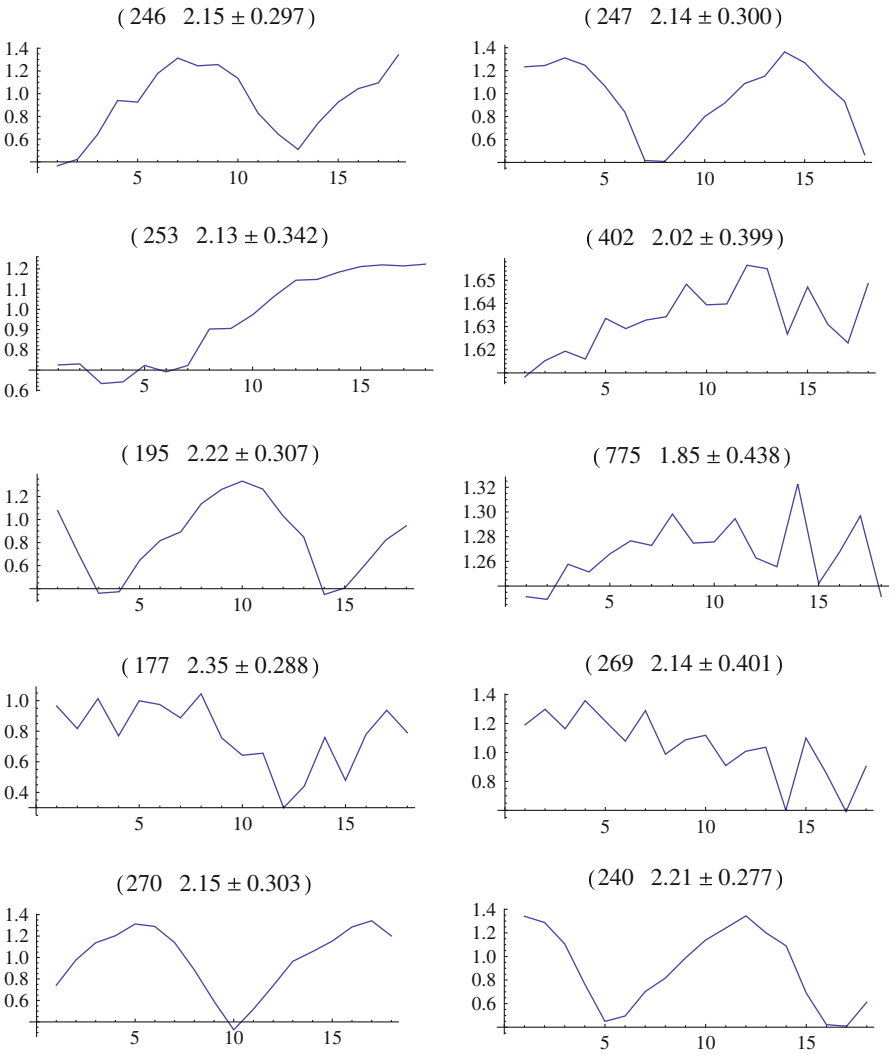


Fig. 19 Typical cluster centroids of SAC

necessarily have smoother centers. Furthermore, the degree of smoothness does not appear to affect the within-cluster variance. Based on these observations, the clustering algorithm is not compromised by trivial matches and capable of identifying noisy or high frequency pattern as cluster centers.

The use of cluster analysis allows the SAC to summarize the data and substantially reduce its complexity, which is $O(nk)$ in the worst case for k clusters. It is an important contribution for conducting an instance-based approach in multiple *resolutions*. Further benefits of the SAC include: (1) the efficiency of *re*-using the distance matrix computed in clustering, (2) directly comparing the densities of target and auxiliary data clusters from two sets of nearest-neighbor distances. Due to (1), no additional

evaluations of distance function is necessary after clustering. Due to (2), we avoid individual approximations of densities over target and auxiliary datasets. It is an important advantage since density approximation is a difficult problem individually, and comparing two approximate densities can increase the perplexity significantly.

9.2 Ensemble approach for multi-resolution anomaly detection

The conventional ensemble approach for anomaly detection has been designed to counteract the randomly added noise of the numerical features (Lazarevic and Kumar 2005). Unlike the random noise, varying the data-processing parameters causes *weak* learners to perform inconsistently, as demonstrated in the empirical results. Because the conventional approach exploits the scalar score outputs from each *resolution*, the inconsistency inevitably limits the effectiveness of the ensemble. In contrast, ACE considers the predictions of *weak* learners in a *meta*-feature representation and their orientation of biases with regards to the candidate *resolutions*. The empirical study, e.g., Figs. 12–14, showed that the *meta*-feature representation provides sufficient discriminative information for partitioning anomalous behaviors and normal patterns.

The manner of addressing multiple resolutions in the ensemble approach can be considered a *divide-and-conquer*, because the cluster-based anomaly detection is conducted in respective resolutions, and the outputs are aggregated afterwards. The motivation for employing such an approach derives from the issue of transitional patterns discussed in the previous section. Due to their presence, intuitively anomalous patterns in a time series are not isolated among the subsequences (Chiu et al. 2003; Lin et al. 2007; Yankov et al. 2008), and the issue can occur in all candidate *resolutions*. Alternatively, one may consider a *simultaneous* approach to analyze all *resolutions* at once, e.g., by representing the behavior at t by a concatenation of $s_{\delta,\lambda}(t)$ over all (δ, λ). The effect of the transitional patterns may generate connectivity between the anomalies and the normal samples over multiple dimensions of such a representation, and subsequently increase the difficulty of detecting the anomalies. From above discussion, the *divide-and-conquer* is more preferable over the *simultaneous* approach for the problem at hand.

10 Conclusion

In this paper, we addressed the problem of multi-*resolution* data mining, which involves not only learning from data but also finding the appropriate settings of data processing for learning. We presented a novel ensemble framework for anomaly detection, which exploits the outputs from multiple *resolution* collectively as *meta*-features reflecting the orientation of biases regarding the data and the *resolutions*.

The results of our empirical study using real-world trajectory data indicate that some anomalous behaviors are associated with specific *resolutions*, and that identifying such anomalies in unsupervised and semi-supervised settings are difficult for the conventional ensemble approach. The ROC performance of individual *weak* detectors were inconsistent, and the effectiveness of the aggregated score was understandably limited. This issue was addressed in the ACE framework by the *meta*-feature representation of

behaviors. We also demonstrated the capability of the ACE framework in the practical setting where the auxiliary data can contain anomalies in small amount. Furthermore, we showed that the multi-resolution anomaly detection is effective in the time series domain using a collection of benchmarks, particularly when the performance of the weak methods are inconsistent in different resolutions.

As part of the ACE framework, we introduced the SAC, which efficiently integrates clustering with density-based anomaly detection to reduce the computational burden for mining in multiple *resolutions*. The use of statistical test contributes to a robust and scalable direct approximation of the relative density. Collectively, ACE provides an intuitive and efficient means to process temporal data and discover temporal patterns, which are two essential and interrelated tasks for mining temporal data.

Acknowledgments The authors would like to thank the handling editor and the anonymous reviewers for their valuable comments and suggestions. Parts of this study are supported by the Strategic International Cooperative Program funded by Japan Science and Technology Agency and the Grant-in-Aid for Scientific Research on Fundamental Research (B) 21300053, (B) 25280085, and (C) 22500119 by the Japanese Ministry of Education, Culture, Sports, Science and Technology.

References

Ailon N, Charikar M, Newman A (2008) Aggregating inconsistent information: ranking and clustering. J ACM 55, 23:1–23:27

Ando S, Thanomphongphan T, Hoshino D, Seki Y, Suzuki E (2011) ACE: anomaly clustering ensemble for multi-perspective anomaly detection in robot behaviors. In: Proceedings of the tenth SIAM international conference on data mining, pp 1–12

Angiulli F, Basta S, Pizzuti C (2006) Distance-based detection and prediction of outliers. IEEE Trans Knowl Data Eng 18(2):145–160

Angiulli F, Fassetti F (2010) Distance-based outlier queries in data streams: the novel task and algorithms. Data Min Knowl Discov 20(2):290–324

Anjum N, Cavallaro A (2008) Multifeature object trajectory clustering for video analysis. IEEE Trans Circuits Syst Video Technol 18(11):1555–1564

Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, pp 1027–1035

Bache K, Lichman M (2012) UCI machine learning repository. University of California, Irvine, School of Information and Computer Science. http://archive.ics.uci.edu/ml. Accessed Mar 2012

Banerjee A, Langford J (2004) An objective evaluation criterion for clustering. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 515–520

Blanchard G, Lee G, Scott C (2010) Semi-supervised novelty detection. J Mach Learn Res 11:2973–3009

Bonchi F, Castillo C, Donato D, Gionis A (2009) Taxonomy-driven lumping for sequence mining. Data Min Knowl Discov 19(2):227–244

Breunig MM, Kriegel HP, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. SIGMOD Rec 29(2):93–104

Bu Y, Chen L, Fu AWC, Liu D (2009) Efficient anomaly monitoring over moving object trajectory streams. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 159–168

Budhaditya S, Pham DS, Lazarescu M, Venkatesh S (2009) Effective anomaly detection in sensor networks data streams. In: Proceedings of the 2009 ninth IEEE international conference on data mining, ICDM'09. IEEE Computer Society, Washington, DC, pp 722–727

Castro N, Azevedo PJ (2010) Multiresolution motif discovery in time series. In: Proceedings of tenth SIAM international conference on data mining. SIAM, pp 665–676

Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):1–58

Chiu B, Keogh E, Lonardi S (2003) Probabilistic discovery of time series motifs. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 493–498

Cotofrei P, Stoffel K (2002) Classification rules + time = temporal rules. In: Proceedings of the international conference on computational science-Part I. Springer-Verlag, London, pp 572–581

Daubechies I (1992) Ten lectures on wavelets. Society for Industrial and Applied Mathematics, Philadelphia

Dereszynski E, Dietterich T (2007) Probabilistic models for anomaly detection in remote sensor data streams. In: Proceedings of the twenty-third conference annual conference on uncertainty in artificial intelligence, UAI-07. AUAI Press, Corvallis, pp 75–82

Dietterich TG (2000) Ensemble methods in machine learning. In: Proceedings of the first international workshop on multiple classifier systems. Springer-Verlag, London, pp 1–15

Ester M, Kriegel HP, Sander Jö, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the second international conference on knowledge discovery and data mining (KDD-96). AAAI Press, Portland, pp 226–231

Fern XZ, Brodley CE (2004) Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the twenty-first international conference on machine learning. ACM, New York, pp 36–43

Fraley C, Raftery AE (1998) How many clusters? Which clustering method? Answers via model-based cluster analysis. Comput J 41(8):578–588

Freire A, Barreto G, Veloso M, Varela A (2009) Short-term memory mechanisms in neural network learning of robot navigation tasks: a case study. In: Proceedings of the 6th Latin American Robotics, Symposium (LARS2009), pp 1–6

Gaffney S, Smyth P (1999) Trajectory clustering with mixtures of regression models. In: Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 63–72

Ghoting A, Parthasarathy S, Otey ME (2008) Fast mining of distance-based outliers in high-dimensional datasets. Data Min Knowl Discov 16(3):349–364

Giannotti F, Nanni M, Pinelli F, Pedreschi D (2007) Trajectory pattern mining. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 330–339

Gionis A, Mannila H, Tsaparas P (2007) Clustering aggregation. ACM Trans Knowl Discov Data 1(1):1–30

Han J, Lee JG, Gonzalez H, Li X (2008) Mining massive RFID, trajectory, and traffic data sets (Tutorial). In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York

Hido S, Tsuboi Y, Kashima H, Sugiyama M, Kanamori T (2011) Statistical outlier detection using direct density ratio estimation. Knowl Inf Syst 26:309–336

Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844

Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323

Jiang S, Ferreira J, Gonzälez M (2012) Clustering daily patterns of human activities in the city. Data Min Knowl Discov 25:478–510

Johnson N, Hogg D (1995) Learning the distribution of object trajectories for event recognition. In: Proceedings of the sixth british conference on machine vision B, vol 2. BMVA Press, Surrey, pp 583–592

Keogh E, Lin J (2005) Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowl Inf Syst 8(2):154–177

Keogh E, Lin J, Fu A (2005) HOT SAX: efficiently finding the most unusual time series subsequence. In: Proceedings of the fifth IEEE international conference on data mining. IEEE Computer Society, Washington, DC, pp 226–233

Khalid S, Naftel A (2005) Classifying spatiotemporal object trajectories using unsupervised learning of basis function coefficients. In: Proceedings of the third ACM international workshop on video surveillance & sensor networks. ACM, New York, pp 45–52

Khalid S, Naftel A (2006) Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. Multimed Syst 12(3):227–238

Kim S, Cho NW, Kang B, Kang SH (2011) Fast outlier detection for very large log data. Expert Syst Appl 38(8):9587–9596

Knorr EM, Ng RT, Tucakov V (2000) Distance-based outliers: algorithms and applications. VLDB J 8(3–4):237–253

Kröger T (2010) On-line trajectory generation in robotic systems, springer tracts in advanced robotics, vol 58. Springer, Berlin

Kumar S, Nguyen HT, Suzuki E (2010) Understanding the behaviour of reactive robots in a patrol task by analysing their trajectories. In: Proceedings of the 2010 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology, vol 02, WI-IAT'10. IEEE Computer Society, Washington, DC, pp 56–63

Lazarevic, A., Kumar V (2005) Feature bagging for outlier detection. In: Proceeding of the eleventh ACM SIGKDD international conference on knowledge discovery in data mining. ACM Press, New York, pp 157–166

Lee JG, Han J, Li X (2008) Trajectory outlier detection: a partition-and-detect framework. In: Proceedings of the 2008 IEEE 24th international conference on data engineering, ICDM'08. IEEE Computer Society, Washington, DC, pp 140–149

Lehmann EL (2006) Nonparametrics: statistical methods based on ranks (revised edition). Springer, New York

Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144

Liu Z, Yu JX, Chen L, Wu D (2008) Detection of shape anomalies: a probabilistic approach using hidden markov models. In: Proceedings of the 2008 IEEE 24th international conference on data engineering. IEEE Computer Society, Washington, DC, pp 1325–1327

Markou M, Singh S (2003a) Novelty detection: a review—part 1: statistical approaches. Signal Process 83:2481–2497

Markou M, Singh S (2003b) Novelty detection: a review—part 2: neural network based approaches. Signal Process 83:2499–2521

Markou M, Singh S (2006) A neural network-based novelty detector for image sequence analysis. IEEE Trans Pattern Anal Mach Intell 28(10):1664–1677

Morris B, Trivedi M (2008) Learning, modeling, and classification of vehicle track patterns from live video. IEEE Trans Intell Transp Syst 9(3):425–437

Morris B, Trivedi M (2009) Learning trajectory patterns by clustering: experimental studies and comparative evaluation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 312–319

Morris BT, Trivedi MM (2008) A survey of vision-based trajectory learning and analysis for surveillance. IEEE Trans Circuits Syst Video Technol 18(8):1114–1127

Nguyen HV, Ang HH, Gopalkrishnan V (2010) Mining outliers with ensemble of heterogeneous detectors on random subspaces. In: Proceedings of the 15th international conference on database systems for advanced applications, DASFAA'10, vol I. Springer, Berlin, pp 368–383

Noto K, Brodley C, Slonim D (2012) FRaC: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. Data Min Knowl Discov 25(1):109–133

Pelekis N, Kopanakis I, Kotsifakos EE, Frentzos E, Theodoridis Y (2011) Clustering uncertain trajectories. Knowl Inf Syst 28(1):117–147

Pham DT, Chan AB (1998) Control chart pattern recognition using a new type of self-organizing neural network. In: Proceedings of the institution of mechanical engineers, part I. J Syst Control Eng 212(2):115–127

Piciarelli C, Foresti GL (2006) On-line trajectory clustering for anomalous events detection. Pattern Recogn Lett 27(15):1835–1842

Piciarelli C, Foresti GL (2007) Anomalous trajectory detection using support vector machines. In: Proceedings of the 2007 IEEE conference on advanced video and signal based surveillance. IEEE Computer Society, Washington, DC, pp 153–158

Piciarelli C, Micheloni C, Foresti G (2008) Trajectory-based anomalous event detection. IEEE Trans Circuits Syst Video Technol 18(11):1544–1554

Porikli F, Haga T (2004) Event detection by eigenvector decomposition using object and frame features. In: Conference on computer vision and pattern recognition workshop, 2004. CVPRW '04, p 114

Roddick JF, Spiliopoulou M (2002) A survey of temporal knowledge discovery paradigms and methods. IEEE Trans Knowl Data Eng 14:750–767

Rokach L (2010) Ensemble-based classifiers. Artif Intell Rev 33:1–39

Rosswog J, Ghose K (2012) Detecting and tracking coordinated groups in dense, systematically moving, crowds. In: Proceedings of the twelfth SIAM international conference on data mining, pp 1–11

Saito N (1994) Local feature extraction and its applications using a library of bases. Ph.D. Thesis, Yale University, New Haven

Steland A (1998) Bootstrapping rank statistics. Metrika 47:251–264

Strehl A, Ghosh J (2003) Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J Mach Learn Res 3:583–617

Suzuki N, Hirasawa K, Tanaka K, Kobayashi Y, Sato Y, Fujino Y (2007) Learning motion patterns and anomaly detection by human trajectory analysis. In: IEEE international conference on systems, man and cybernetics, ISIC2007, pp 498–503

Wan L, Ng WK, Dang XH, Yu PS, Zhang K (2009) Density-based clustering of data streams at multiple resolutions. ACM Trans Knowl Discov Data 3, 14:1–14:28

Wang Q, Megalooikonomou V, Faloutsos C (2010) Time series analysis with multiple resolutions. Inf Syst 35(1):56–74

Wang X, Li G, Jiang G, Shi Z (2011) Semantic trajectory-based event detection and event pattern mining. Knowl Inf Syst. doi:10.1007/s10115-011-0471-8

Webb A, Copsey K (2011) Statistical pattern recognition. Wiley, New York

Williams BH, Toussaint M, Storkey AJ (2007) A primitive based generative model to infer timing information in unpartitioned handwriting data. In: Proceedings of the 20th international joint conference on artifical intelligence, IJCAI'07. Morgan Kaufmann Publishers Inc., San Francisco, pp 1119–1124

Xiong Y, Yeung DY (2002) Mixtures of ARMA models for model-based time series clustering. In: Proceedings of the 2002 IEEE international conference on data mining. IEEE Computer Society, Washington, DC, pp. 717–720

Xu R, Wunsch D (2005) Survey of clustering algorithms. IEEE Trans Neural Netw 16(3):645–678

Yamanishi K, Takeuchi J, Williams G, Milne P (2004) On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. Data Min Knowl Discov 8(3):275–300

Yang Q (2009) Activity recognition: linking low-level sensors to high-level intelligence. In: Proceedings of the 21st international joint conference on artifical intelligence, IJCAI'09. Morgan Kaufmann Publishers Inc., San Francisco, CA, pp 20–25

Yang Y, Chen K (2011) Temporal data clustering via weighted clustering ensemble with different representations. IEEE Trans Knowl Data Eng 23:307–320

Yankov D, Keogh E, Rebbapragada U (2008) Disk-aware discord discovery: finding unusual time series in terabyte sized datasets. Knowl Inf Syst 17(2):241–262

Zheng Y, Zhou X (2011) Computing with spatial trajectories, 1st edn. Springer Publishing Company, Incorporated, New York