Artificial Intelligence 292 (2021) 103429

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

# Weakly-supervised sensor-based activity segmentation and recognition via learning from distributions

Hangwei Qian, Sinno Jialin Pan*, Chunyan Miao

*Nanyang Technological University, Singapore*

A R T I C L E   I N F O

*Article history:*
Received 11 February 2019
Received in revised form 28 June 2020
Accepted 10 November 2020
Available online 30 November 2020

*Keywords:*
Human activity recognition
Sensor readings segmentation
Kernel mean embedding

A B S T R A C T

Sensor-based activity recognition aims to recognize users' activities from multi-dimensional streams of sensor readings received from ubiquitous sensors. It has been shown that data segmentation and feature extraction are two crucial steps in developing machine learning-based models for sensor-based activity recognition. However, most previous studies were only focused on the latter step by assuming that data segmentation is done in advance. In practice, on the one hand, doing data segmentation on sensory streams is very challenging. On the other hand, if data segmentation is considered as a pre-process, the errors in data segmentation may be propagated to latter steps. Therefore, in this paper, we propose a unified weakly-supervised framework based on kernel embedding of distributions to jointly segment sensor streams, extract powerful features from each segment, and train a final classifier for activity recognition. We further offer an accelerated version for large-scale data by utilizing the technique of random Fourier features. We conduct experiments on four benchmark datasets to verify the effectiveness and scalability of our proposed framework.

## 1. Introduction

Activity recognition, as an important application of artificial intelligence [52], has been used in a wide spectrum of real-world applications, such as health care, smart homes, security and assisted living [8,19,5,23,11,39]. The goal of activity recognition is to automatically recognize human's activities or behaviors from a series of received signals. Traditionally, in computer vision, a large number of studies have been done on recognition of gestures and activities from still images and videos [49,9]. However, due to the privacy issue, vision-based activity recognition systems can only be used in some restricted environments. With the development of sensor technology, various wireless sensors, such as radio-frequency identification (RFID), WiFi, wearable sensors, etc, are widely available in our everyday lives. Recent efforts have been shifted to the development of sensor-based activity recognition systems.

In sensor-based activity recognition, the inputs are streams of low-level multivariate sensor readings and the outputs are labels of various activities. In the past two decades, machine learning-based methods have shown promising performance for sensor-based activity recognition. A typical procedure of training a learning-based activity recognition model includes three main stages: 1) data segmentation, which identifies those length-variate segments of the multivariate data streams that likely contain information about activities, 2) feature extraction, which transforms low-level sensor readings of each

* Corresponding author.
*E-mail addresses:* qian0045@e.ntu.edu.sg (H. Qian), sinnopan@ntu.edu.sg (S.J. Pan), ascymiao@ntu.edu.sg (C. Miao).

https://doi.org/10.1016/j.artint.2020.103429
0004-3702/© 2020 Elsevier B.V. All rights reserved.

segment to a vector of features that are discriminative to recognize activities, and 3) training a classifier, which builds a classifier to capture the relationship between features and activities or class labels. When data streams are perfectly segmented and powerful features are extracted, any standard classification algorithm can be used to train a classifier for activity recognition. Thus, data segmentation and feature extraction are the crucial stages in the procedure.

Regarding feature extraction from a segment of multivariate sensory streams, previous studies have been focused on constructing statistical or structural features [35,27], as well as spatial and temporal features [36,50,15]. However, these solutions fail to retain all the important information of a segment. In our preliminary work [37], we proposed a new feature extraction method for activity data based on kernel embedding of distributions [45,42], which is able to extract *infinite* orders of moments underlying each segment. Empirical results have shown that the features extracted by kernel embedding are powerful to distinguish instances among different classes (activities). However, similar to most feature extraction approaches, in [37], we assume data segmentation is performed in advance.

Compared with feature extraction, data segmentation of sensory streams of activity data is much less investigated [19,3,53]. To partition continuous steaming activity data, existing approaches typically move a sliding window over the data with static or dynamic sizes [43,33]. The difference between two adjacent windows is computed against some threshold to decide whether a breakpoint is found or not. However, how to identify the optimal window size remains an open problem [4]. One alternative approach is to detect activity transitions or boundaries. To achieve this, it is often assumed that the data adheres to some degree of homogeneity, such as constant [28,7], linear and polynomial models [13]. For these parametric models, the changepoints correspond to changes in the parameter(s). For activity data, it is often improper to feed the activity data into parametric models due to the complexity and large variations of activity data.

In this paper, we extend our preliminary work [37] to an end-to-end framework for sensor-based activity recognition, which enables joint learning of segmentation, feature extraction as well as final classification. Specifically, the continuous flow of activity data is first partitioned into segments with weak supervision. Kernel embedding of distributions is applied to extract sufficient statistical features of each segment, and to further learn a classifier for activity recognition. We cast the whole process as a weakly-supervised Support Measure Machines (SMM) [32,31] formulation to jointly update the learnable parameters of different components. Moreover, we adopt the technique of random Fourier features (RFF) [38] to develop an accelerated version to tackle the scalability issue.

To summarize, our contributions are 3-fold:

- We proposed an end-to-end weakly-supervised learning framework, denoted by S-SMM$_{AR}$, for activity recognition, where data segmentation, feature extraction and activity classification can be done jointly.
- We further propose an accelerated version, denoted by R-SMM$_{AR}$, to scale up the proposed framework.
- Extensive experiments are conducted to demonstrate the effectiveness of the proposed framework compared with state-of-the-art methods.

## 2. Related work and preliminary

### 2.1. Segmentation on sensory activity data

Sliding window is a commonly used approach to segment activity data streams. The size of a segmentation window is selected empirically and based on hardware limitations [8]. For example [2] proposed to determine the sizes of sliding windows based on whether there are sufficient differences between adjacent windows. However, an optimum window size varies depending on the characteristics of activities and a fixed window size may not suit all activities. Akbari et al. [1] proposed to first segment data with relatively large windows, and then split those segmentations that likely contain multiple activities into smaller sub-windows in order to fine-tune the label assignment. Methods with adaptive sliding windows have been also proposed [34]. Additionally environmental contextual information, such as location, time and ambient attributes of the objects, can also be utilized to segment activity data by a rule-based method [48]. Li et al. [24] applied existing multivariate segmentation methods to activity data, aiming to maximizing the likelihood of the data.

### 2.2. Segmentation on time series data

As sensor-based activity data is a type of time series data, here we also briefly review some classic approaches to time series data segmentation. Dynamic programming (DP) is usually utlized to find the optimal segmentation of time series data [18]. To further alleviate the computational burden, various pruning-based methods are proposed. The forward-backward DP algorithm [14] computes several most probable segment candidates based on the reversibility property of time series data. The cp3o method [54] removes less optimal candidate changepoints by comparing candidates to a specific solution during each iteration. The PELT method [21] limits the set of potential changepoints by removing those indices of data which cannot reduce the cost function performed at each iteration. The pDPA method [41] prunes the less optimal candidates based on a functional representation of cost functions which introduces one additional scalar parameter.

## 2.3. Feature extraction for activity recognition

Statistical approaches aim to compute and concatenate some orders of moments of a segment as features to represent the segment [23]. Structural approaches take into account the interrelationship among data. The ECDF approach [16,35] leverages distributions' quantile function to preserve the overall shape of the distribution as well as the spatial positions. Lin et al. [26] proposed the SAX method to discretize data into symbolic strings to represent equal probability mass. In addition, spatial and temporal features can be extracted as well [36,50,15]. The feature extraction component in our proposed framework aims to extract all orders of moments to form a concatenated feature vector for each segment, and thus falls into the category of statistical approaches.

## 2.4. Kernel embedding of distributions

As our proposed framework is based on the technique of kernel embedding of distributions, here we briefly introduce some important concepts of the technique. Consider an activity which lasts for $n$ timestamps, it can be modeled as a sample $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ drawn from a probability distribution $\mathbb{P}$, where each instance $\mathbf{x}_i$ is of $d$ dimensions. The technique of kernel embedding [45] for representing an arbitrary distribution is to introduce a mean map operation $\mu(\cdot)$ to map instances to a Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H}$, and to compute their mean in the RKHS as follows,

$$\boldsymbol{\mu}_{\mathbb{P}} := \mu(\mathbb{P}) = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}[\phi(\mathbf{x})] = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}[k(\mathbf{x}, \cdot)], \tag{1}$$

where $\phi : \mathbb{R}^d \to \mathcal{H}$ is a feature map, and $k(\cdot,\cdot)$ is the kernel function induced by $\phi(\cdot)$. In this way, the feature vector $\phi(x)$ is learned from raw data $x$. If the condition $\mathbb{E}_{\mathbf{x}\sim\mathbb{P}}(k(\mathbf{x},\mathbf{x})) < \infty$ is satisfied, then $\boldsymbol{\mu}_{\mathbb{P}}$ is also an element in $\mathcal{H}$. It has been proven that if the kernel $k(\cdot,\cdot)$ is characteristic, then the mapping $\mu : \mathcal{P} \to \mathcal{H}$ is injective [46]. The injectivity property indicates an arbitrary probability distribution $\mathbb{P}$ is uniquely represented by an element in a RKHS through the mean map. As each distribution can be mapped to $\mathcal{H}$, the operations defined in $\mathcal{H}$, such as inner product and distance measure, are capable of estimating similarity or distance between distributions.

In practice, an underlying probability distribution of a sample is unknown. In our case, a finite number of $n$ samples are collected for training. Hence, an unbiased empirical estimation is applied to approximate the mean map as follows,

$$\hat{\boldsymbol{\mu}}_{\mathbb{P}} = \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i) = \frac{1}{n}\sum_{i=1}^{n}k(\mathbf{x}_i, \cdot). \tag{2}$$

Though in theory, the dimension of $\hat{\boldsymbol{\mu}}_{\mathbb{P}}$ is potentially infinite, by using the kernel trick, the inner product of two probability distributions in a RKHS can be computed efficiently through a kernel function associated to the RKHS,

$$\langle \hat{\boldsymbol{\mu}}_{\mathbb{P}_x}, \hat{\boldsymbol{\mu}}_{\mathbb{P}_z} \rangle = \tilde{k}(\hat{\boldsymbol{\mu}}_{\mathbb{P}_x}, \hat{\boldsymbol{\mu}}_{\mathbb{P}_z}) = \frac{1}{n_x n_z}\sum_{i=1}^{n_x}\sum_{j=1}^{n_z} k(\mathbf{x}_i, \mathbf{z}_j), \tag{3}$$

where $\tilde{k}(\cdot,\cdot)$ is a linear kernel defined in the RKHS, $n_x$ and $n_z$ are the sizes of the samples $\mathbf{X}$ and $\mathbf{Z}$ drawn from $\mathbb{P}_x$ and $\mathbb{P}_z$, respectively. In general, $\tilde{k}(\cdot,\cdot)$ can be a nonlinear kernel defined as follows,

$$\tilde{k}(\hat{\boldsymbol{\mu}}_{\mathbb{P}_x}, \hat{\boldsymbol{\mu}}_{\mathbb{P}_z}) = \langle \psi(\hat{\boldsymbol{\mu}}_{\mathbb{P}_x}), \psi(\hat{\boldsymbol{\mu}}_{\mathbb{P}_z}) \rangle, \tag{4}$$

where $\psi(\cdot)$ is the associated feature mapping of the nonlinear kernel $\tilde{k}(\cdot,\cdot)$.

## 2.5. Random Fourier features

In kernel embedding of distributions, a kernel matrix needs to be computed, which is computationally expensive, especially when training data is large-scale. The technique of random Fourier features has been proposed to approximate the kernel computation efficiently [38]. Specifically, if a kernel is a shift-invariant kernel, i.e., $k(\mathbf{x},\mathbf{x}') = k(\mathbf{x}-\mathbf{x}')$, then one can construct a randomized feature map of $D$ dimensions, $\mathbf{z} = [\sqrt{2}cos(w_1^\top \mathbf{x} + b_1, ..., \sqrt{2}cos(w_D^\top \mathbf{x} + b_D]^\top$ to approximate values of the kernel via

$$k(\mathbf{x},\mathbf{x}') \approx \mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{x}'), \tag{5}$$

where $w_i \sim p(w)$, which is $k(\cdot,\cdot)$'s Fourier transform distribution on $\mathbb{R}^D$, and $b_i$ is sampled uniformly from $[0, 2\pi]$.

## 3. The proposed methodology

### 3.1. Problem statement

In our problem setting, we are given a stream of multivariate activity data $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^N$ where $\mathbf{x}_t \in \mathbb{R}^{d\times 1}$ is a vector of signals received from $d$ sensors at the $t$-th timestamp, which is referred to as a frame in the segment. Associated with signals are a sequence of $K$ activity labels $\mathbf{y} = \{y_k\}_{k=1}^K$, where $y_k \in \{Y_1, ..., Y_L\}$ is a set of predefined $L$ activity categories. Each $y_k$ may last for $n_k$ timestamps, and $n_k$ can be different, with the sum of all the $n_k$'s to be equal to the total duration $N$. This setting is referred to as a weakly supervised setting, as no ground-truth partition and ground-truth label on each segment is provided in training, while only a sequence of activity labels is available for the whole data stream. Note that this setting is more practically applicable, since human usually can remember effortlessly the sequence of activities conducted in a time period, but the exact starting and ending time require expensive annotation effort.

Our goal is to first find $K-1$ breakpoints indices $\mathbf{I} = \{I_k | 1 < I_k < N, I_k < I_{k+1}\}_{k=1}^{K-1}$ to segment the stream of activity data into $K$ adjacent segments $\{\mathbf{X}_k\}_{k=1}^K$, where $\mathbf{X}_k = \{\mathbf{x}_{I_{k-1}+1}, ..., \mathbf{x}_{I_k}\}$ such that each segment $\mathbf{X}_k$ is aligned with an activity $y_k$ of the sequences of activities sequentially. With the $K$ segments, each of which, $\mathbf{X}_i$, is aligned with a label $y_i \in \{Y_1, ..., Y_L\}$, we aim to train a classifier $f$ to map $\{\mathbf{X}_i\}$'s to $\{y_i\}$'s.

For testing, we suppose the segmentation is done, and we are given $m$ new unseen segments $\{\mathbf{X}_i^*\}_{i=1}^m$, each of which corresponds to an unknown label. We use the trained classifier $f$ to make predictions.

### 3.2. Problem formulation in weakly-supervised setting

In the weakly-supervised setting, given the data stream $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^N$, the ground-truth labels on each segment as well as breakpoints indices $\mathbf{I}$ are unknown, while only the sequences of activities $\mathbf{y} = \{y_k\}_{k=1}^K$ are available, where $K$ is the total number of activity segments in the stream consisting of $L$ classes of activities. We propose the following optimization problem to jointly learn the classifier $f$ and the segmentation in terms of $\mathbf{I}$,

$$\begin{aligned}
&\min_{f,\mathbf{I},\mathbf{C}} \frac{1}{K}\sum_{k=1}^{K}\sum_{j=1}^{L}\mathbf{C}_{kj}\ell(f(\mathbf{X}_k), Y_j; \mathbf{I}) + \lambda_1\Omega_1(\|f\|_{\tilde{\mathcal{H}}}) + \lambda_2\Omega_2(\mathbf{I}), \\
&\text{s.t. } f(\mathbf{X}_t) = y_t, \forall k \in \{1, ..., K\}, \\
&\quad \sum_{j=1}^{L}\mathbf{C}_{kj} = 1, \forall k \in \{1, ..., K\},
\end{aligned} \tag{6}$$

where $\ell(\cdot)$ is a data-dependent loss function, $\lambda_1, \lambda_2 > 0$ are the tradeoff parameters to control the impact of the regularization terms $\Omega_1(\cdot)$ and $\Omega_2(\cdot)$. $\tilde{\mathcal{H}}$ is a RKHS associated with the kernel $\tilde{k}(\cdot,\cdot)$, which will be explained later. The first term in the objective is the weighted average loss function on classification, and $\mathbf{C} \in \mathbb{R}^{K\times L}$ is the matrix of confidence scores, with each element $\mathbf{C}_{kj}$ being the confidence score of the $k$-th segment associated with the $j$-th activity class. The confidence score matrix leads the classifier $f$ to correctly learn those easy-to-classify segments first. A higher confidence score of a segment means a higher probability of a correct prediction by the classifier. Therefore, the classifier tends to predict the corresponding label correctly, or the weighted loss function is increased by a larger value compared to those with smaller confidence scores. The reason why we use the weighted average loss function is that we do not have a ground-truth label of each segment. Using the weighted average loss across all the possible classes is a reasonable approach to measure prediction errors. This idea is similar to the expectation loss used in e-SVM [55], which was originally proposed to address the object detection task under weak supervision, where only bounding box annotations for images are available.

The second term is a regularization term on the learned classifier to prevent overfitting. The form of $\Omega_1(\cdot)$ is chosen to be a strictly monotonically increasing function, with a special choice being the linear function as used in our previous work [37]. The last term is the regularization term on segmentation breakpoints to ensure the segmentation results to be reasonable, and is set to be the average of the MMD distance between segments with the same predicted label:

$$\begin{aligned}
&\Omega_2(\mathbf{I}) \\
&= \frac{1}{M}\sum_{\substack{1\le i<j\le N \\ f(\mathbf{X}_i)=f(\mathbf{X}_j)}} \mathrm{MMD}(\mathbf{X}_i, \mathbf{X}_j) \\
&= \frac{1}{M}\sum_{\substack{1\le i<j\le N \\ f(\mathbf{X}_i)=f(\mathbf{X}_j)}} \left\| \frac{1}{n_i}\sum_{k=1}^{n_i}(\phi(\mathbf{x}_k^i)) - \frac{1}{n_j}\sum_{k=1}^{n_j}(\phi(\mathbf{x}_k^j)) \right\|_2
\end{aligned} \tag{7}$$

$$= \frac{1}{M} \sum_{\substack{1 \le i < j \le N \\ f(\mathbf{X}_i) = f(\mathbf{X}_j)}} \left( \frac{1}{n_i^2} \sum_{k_1,k_2} k(x_{k_1}^i, x_{k_2}^j) - \frac{2}{n_i n_j} \sum_{k_1,k_2} k(x_{k_1}^i, x_{k_2}^j) + \frac{1}{n_j^2} \sum_{k_1,k_2} k(x_{k_1}^i, x_{k_2}^j) \right)^{\frac{1}{2}},$$

where $\mathbf{x}_i^k$ denotes the $k$-th instance in the $i$-th segment $\mathbf{X}_i$, and $n_i$ is the length of segment $\mathbf{X}_i$. The kernel $k(\cdot,\cdot)$ is induced by the feature map $\phi(\cdot)$.

The first constraint in (6) is to enforce that the sequence of predicted labels is aligned with the sequence of the ground-truth activities. The second constraint is to ensure that the summation of the confidences over all possible classes for each segment equals 1.

### 3.3. Alternating optimization for joint segmentation and classification

Note that the optimization problem (6) is a joint learning framework, where the breakpoints influence the formation of segments of activities, while the predicted labels further influence the detection of breakpoints. Therefore, in this section, we propose an alternating optimization algorithm to solve the problem. In the sequel, we denote our proposed joint learning algorithm for activity segmentation and classification by S-SMM$_{AR}$. The overall algorithm is shown in Algorithm 1.

**Algorithm 1** The proposed S-SMM$_{AR}$ algorithm.

```
Input: A data sequence $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \in \mathbb{R}^{d\times 1}$, a coarse label sequence $\mathbf{y} = \{y_k\}_{k=1}^K \in \{Y_1, ..., Y_L\}$, the number of breakpoints $K-1$
Output: the breakpoints indices $\mathbf{I}$ and the classifier $f$
1: procedure MAIN
2:     Randomly initialize breakpoints indices $\mathbf{I} = \{I_k\}_{k=1}^{K-1}$ and the matrix of confidence scores $\mathbf{C}$
3:     while not convergent do
4:         Fix $\mathbf{I}$ and $\mathbf{C}$, update $f$ with (11)
5:         Fix $f$, update $\mathbf{C}$ as described in the 2nd paragraph in Section 3.3.2
6:         Update candidate range of breakpoints with (15) and (16)
7:         Fix $f$ and $\mathbf{C}$, update $\mathbf{I}$ by solving (14)
8:     return $\mathbf{I}$, $f$, and $\mathbf{C}$
```

#### 3.3.1. Learning the classifier $f$ with fixed $\mathbf{I}$ and $\mathbf{C}$

With $\mathbf{I}$ and $\mathbf{C}$ fixed, the $K$ segmentations $\mathbf{X_i}$'s from $\mathbf{X}$ are known, and their corresponding class labels are also known by aligning them with the sequence of activities $\mathbf{y}$. Therefore, the optimization problem (6) is reduced as the following unconstrained optimization problem,

$$\min_f \frac{1}{K} \sum_{k=1}^{K} \mathbf{C}_{k\hat{k}} \ell(f(\mathbf{X}_k), y_k) + \lambda_1 \Omega_1(\|f\|_{\tilde{\mathcal{H}}}), \tag{8}$$

where $\hat{k}$ is the index of $y_k$ in $\{Y_1, ..., Y_L\}$.

To construct a classifier, in most standard classification methods, the input is required to be a feature vector of fixed dimensionality, and the output is a label. However, in our problem setting, the input $\mathbf{X}_i$ is a matrix. Moreover, the sizes of the different segments can be different. Therefore, standard classification methods cannot be directly applied. As discussed, a commonly used solution is to decompose the matrix $\mathbf{X}_i$ to $n_i$ vectors or frames $\{\mathbf{x}_k^i\}_{k=1}^{n_i}$, and assign the same label $y_i$ to each vector. In this way, for each segment, one can construct $n_i$ input-output pairs $\{(\mathbf{x}_k^i, y_i)\}_{k=1}^{n_i}$. By combining such input-output pairs from all the segments, one can apply standard classification methods to train a classifier $f$. A major drawback of this approach is that a single frame of a segment fails to represent an entire activity that lasts for a period of time.

Another approach is to aggregate the $n_i$ frames of a segment $\mathbf{X}_i$ to generate a feature vector of fixed dimensionality to represent the segment. For example, one can use the mean vector $\bar{\mathbf{x}}_i = \sum_{k=1}^{n_i} \mathbf{x}_k^i$ to represent a segment $\mathbf{X}_i$. This approach can capture some global information of a segment, but in practice, one needs to manually generate a very high-dimensional vector to fully capture useful information of each segment. For example, one may need to generate a set of vectors of different orders of moments for a segment, and then concatenate them to construct a unified feature vector to capture rich statistic information of the segment, which is computationally expensive.

Different from previous approaches, we consider each segment $\mathbf{X}_i$ as a sample of $n_i$ instances drawn from an unknown probability $\mathbb{P}_i$, and all $\{\mathbb{P}_i\}_{i=1}^n \subseteq \mathcal{P}$, where $\mathcal{P}$ is the space of probability distributions. By borrowing the idea from kernel embedding of distributions, we can map all samples to a RKHS through a characteristic kernel, and then use a potentially infinite-dimensional feature vector to represent each sample, and thus each segment. As the kernel embedding with characteristic kernel is able to capture any order of moments of the sample, the feature vector is supposed to capture all statistical moments information of the segment. With the new feature representations for each segment in the RKHS, we can train a classifier with their corresponding labels for activity recognition.

To be specific, firstly, each segment or sample $\mathbf{X}_i$ is mapped to a RKHS with a kernel $k(\mathbf{x}^i_{k_1}, \mathbf{x}^i_{k_2}) = \langle \phi(\mathbf{x}^i_{k_1}), \phi(\mathbf{x}^i_{k_2}) \rangle$ via an implicit feature map $\phi(\cdot)$, and represented by an element $\boldsymbol{\mu}_i$ in the RKHS via the mean map operation:

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{k=1}^{n_i} \phi(\mathbf{x}^i_k). \tag{9}$$

As a result, we have $K$ pairs of input-output in the RKHS $\{(\boldsymbol{\mu}_1, y_1), ..., (\boldsymbol{\mu}_K, y_K)\}$. Then our goal becomes to learn a classifier $f$ by solving

$$\min_f \frac{1}{K} \sum_{k=1}^{K} \mathbf{C}_{k\hat{k}} \ell(f(\boldsymbol{\mu}_k), y_k) + \lambda_1 \Omega_1(\|f\|_{\tilde{\mathcal{H}}}). \tag{10}$$

As shown in our preliminary work [37], by using the representer theorem in [32], the solution of the functional $f(\cdot)$ in (10) can be represented by

$$f = \sum_{i=1}^{K} \alpha_i \psi(\boldsymbol{\mu}_i), \tag{11}$$

where the weights $\mathbf{C}_{k\hat{k}}$'s are incorporated into $\alpha_i$'s, the feature map $\psi : \mathcal{H} \to \tilde{\mathcal{H}}$ is used for classification, and $\tilde{\mathcal{H}}$ is another RKHS with a kernel $\tilde{k}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = \langle \psi(\boldsymbol{\mu}_i), \psi(\boldsymbol{\mu}_j) \rangle$ defined by $\psi(\cdot)$. If $\tilde{\mathcal{H}} = \mathcal{H}$, then a linear kernel on $\{\boldsymbol{\mu}_i\}$'s is used, i.e., $\tilde{k}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = \langle \boldsymbol{\mu}_i, \boldsymbol{\mu}_j \rangle$, and (11) can be reduced as

$$f = \sum_{i=1}^{K} \alpha_i \boldsymbol{\mu}_i, \text{ where } \alpha_i \in \mathbb{R}. \tag{12}$$

By specifying (11) or (12) using the Support Vector Machines (SVMs) formulation,[1] we reach the following optimization problem, which is known as Support Measure Machines (SMMs) [32],

$$\begin{aligned} \min_f \ & \frac{1}{2}\|f\|^2_{\tilde{\mathcal{H}}} + \eta \sum_{i=1}^{K} \xi_i, \\ s.t. \ & y_i f(\boldsymbol{\mu}_i) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \\ & 1 \leq i \leq K, \end{aligned} \tag{13}$$

where $\tilde{\mathcal{H}}$ is a RKHS associated with the kernel $\tilde{k}(\cdot,\cdot)$ on $\mathcal{P}$, $\{\xi_i\}_{i=1}^n$ are slack variables to absorb tolerable errors, and $\eta > 0$ is a tradeoff parameter. When the forms of the kernels, $k(\cdot,\cdot)$ and $\tilde{k}(\cdot,\cdot)$, are specified,[2] many optimization techniques developed for standard linear or nonlinear SVMs can be applied to solve the optimization problem of SMMs.

After the classifier $f(\cdot)$ is learned, given a test segment $\mathbf{X}^*_p$, one can first represent it using the mean map operation

$$\boldsymbol{\mu}^*_k = \frac{1}{n_p} \sum_{k=1}^{n_p} \phi(\mathbf{x}^{p^*}_k),$$

and then use $f(\cdot)$ to make a prediction $f(\boldsymbol{\mu}^*_k)$.

### 3.3.2. *Update* **I** *and* **C** *with fixed* $f$

After obtaining the updated classifier $f$, we now show how to update $\mathbf{I}_{\mathbf{bkps}}$ and $\mathbf{C}$. With $f$ fixed, the optimization problem (6) becomes

$$\begin{aligned} \min_{\mathbf{I},\mathbf{C}} \ & \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{L} \mathbf{C}_{kj} \ell(f(\boldsymbol{\mu}_k), Y_j; \mathbf{I}) + \lambda_2 \Omega_2(\mathbf{I}), \\ \text{s.t.} \ & f(\mathbf{X}_t) = y_t, \forall k \in \{1, ..., K\}, \\ & \sum_{j=1}^{L} \mathbf{C}_{kj} = 1, \forall k \in \{1, ..., K\}, \end{aligned} \tag{14}$$

[1] Note that one can also specify (11) or (12) using other loss functions, which result in different particular approaches.

[2] Recall that the kernel $k(\cdot,\cdot)$ is defined on $\{\mathbf{X}_i\}$'s to perform a mean map operation for generating $\{\boldsymbol{\mu}_i\}$'s, and the kernel $\tilde{k}(\cdot,\cdot)$ is defined on $\{\boldsymbol{\mu}_i\}$'s for final classification.

where the regularization term $\Omega_2(\mathbf{I})$ is defined in (7).

Regarding updating the matrix $\mathbf{C}$, the confidence score $\mathbf{C}_{kj}$ is expected to measure the confidence of the segment $X_k$ that belongs to the class $Y_j$. In the supervised setting where the ground-truth labels are available, the confidence score can be easily obtained by calculating the accuracy of predicted labels of each segment. However, as we discussed, the annotation effort on segmentation is highly expensive as the exact start and end time stamps of each activity need to be marked for training. In our proposed weakly-supervised setting, where we only have access to the coarse activities sequence, we aim to make the confidence score of a predicted segment depending on the distance to the decision boundary. Specifically, for classification of $L$ classes activities, a common practice is to learn $L$ classifiers by one-vs-rest mechanism, which transforms the problem into learning multiple binary classifiers. For each binary classifier, the distance of the data point to the decision boundary matters in the way that a larger distance reflects the easier classification of the data point. Therefore, we set the confidence score to be $\frac{1}{1+exp(Af(\boldsymbol{\mu})+B)}$ in the binary case, and further normalize the scores in the multi-class case. The confidence score is similar to the Platt's probabilistic output [25], where $A$ and $B$ are decided by the data distribution prior.

Regarding updating $\mathbf{I}$, Dynamic Programming (DP) can be applied to find breakpoints one by one sequentially, but the candidate range of a new breakpoint is from the former breakpoints to the end of a time series, which is computationally expensive. Therefore, in the literature, various algorithms have been proposed to alleviate the computational cost by limiting the searching range of each breakpoint. Specifically, the computational cost is alleviated by pruning the set of candidate breakpoint locations and finding the next breakpoint in the restricted set. However, as discussed in Section 2.2, existing algorithms are supposed to work under non-trivial assumptions on the data property or the model.

In our proposed algorithm, we also aim to prune the candidate set to reduce the complexity, but we do not have any assumptions on the data or the model. Different from other pruning methods for DP, our method prunes the candidates set from a probabilistic point of view. Specifically, for each segment $\mathbf{X}_k$ with breakpoints indices $I_{k-1}+1$ and $I_k$ being the starting and ending locations, respectively, there is a vector of confidence scores, $\mathbf{C}_{k*}$ (the $k$-th row of $\mathbf{C}$), to represent the probabilities over all the classes for this segmentation. A larger $\mathbf{C}_{kj}$ indicates the higher probability of the segment $k$ that belongs to the class $Y_j$. Intuitively, for a good segment, there should exist a $i$ such that the corresponding confidence score $\mathbf{C}_{ki}$ is large and all the other confidence scores $\{\mathbf{C}_{kj}, i \neq j\}$'s are small. Thus, we set the confidence score of the segment $k$ to be the maximum of $\{\mathbf{C}_{kj} | 1 \leq j \leq L\}$, i.e., $\max_j \mathbf{C}_{kj}$. Our proposed method prunes the candidate range of a breakpoint with its neighbors' status, i.e., the candidate range of $I_k$ is the range of low confidence score neighbors with different labels $[\mathbf{I}_{left}, \mathbf{I}_{right}]$:

$$\mathbf{I}_{left} = \max(\mathbf{I}_m | m < k, y_m \neq y_k, \text{and } \max_j(\mathbf{C}_{mj}) < \epsilon), \tag{15}$$

and

$$\mathbf{I}_{right} = \min(\mathbf{I}_m | m > k, y_m \neq y_k, \text{and } \max_j(\mathbf{C}_{mj}) < \epsilon), \tag{16}$$

where $\epsilon$ is a threshold. In the next iteration, the location indices with lower confidence scores are more likely to be modified. And the breakpoints indices with high confidence scores are kept unchanged. In this way, the complexity of DP is reduced by pruning the candidate sets of breakpoints as well as reducing the number of modified breakpoints.

After specifying the candidate range of breakpoints, the next step is to go through each candidate range to search for an updated breakpoint location for each of the modified breakpoints by minimizing the optimization problem (14) with the updated $\mathbf{C}$ fixed. Note that the computational cost of the regularization term in (14) can be further reduced by reusing the precomputed kernel values in the previous classifier training step.

Precise segmentation of activity stream data is an essential prerequisite for learning an accurate classifier. Once the segmentation of activity data is corrupted, the extracted features are no longer representative for the corresponding activity class, hence the learning process of classification would be hindered. From another perspective, a low similarity measure (as shown in (4)) between two segments from the same activity indicates two possibilities: 1) the classifier is not trained properly, and/or 2) the data is not segmented correctly. Thus, with the same similarity measure applied in both the segmentation and the prediction phases, the two modules can interact and boost each other in an iterative manner.

### *3.4. R-SMM$_{AR}$ for large-scale activity recognition*

Note that the technique of kernel embedding of distributions used in S-SMM$_{AR}$ makes a feature vector of each segment be able to capture sufficient statistics of the segment. This is useful for calculating similarity or distance metric between segments. However, it needs to compute two kernels, one is for kernel embedding of the frames within each segment, and the other is for estimating similarity between segments. This makes S-SMM$_{AR}$ computationally expensive when the number of segments is large and/or the number of frames within each segment is large. To scale up S-SMM$_{AR}$, in this section, we present an accelerated version using fandom Fourier features to construct an explicit feature map instead of using the kernel trick.

To be specific, based on (9) and (5), the empirical kernel mean map on a segment $\mathbf{X}_i$ with explicit random Fourier features can be written by

**Table 1**
Statistics of the four datasets. Note that in the table, "Seg." denotes segments, "En." denotes average number of frames per segment, "Fea." denotes feature dimensions, "C." denotes classes, "freq" denotes frequency in Hz (sampling rates of sensors may be various, but we assume the frequency of all sensors in a dataset is the same after preprocessing), "Sub." denotes subjects, and "$\frac{\#Seg.}{\#C.}$" denotes the average number of segments that each class of activity has.

| Datasets | # Seg. | # En. | # Fea. | # C. | freq | # Sub. | $\frac{\#Seg.}{\#C.}$ |
|---|---|---|---|---|---|---|---|
| Skoda | 1,447 | 68.8 | 60 | 10 | 14 | 1 | 144.7 |
| WISDM | 389 | 705.8 | 6 | 6 | 20 | 36 | 64.8 |
| HCI | 264 | 602.6 | 48 | 5 | 96 | 1 | 52.8 |
| PS | 1,614 | 100.0 | 9 | 6 | 50 | 4 | 269 |

$$\boldsymbol{\mu}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} \mathbf{z}(\mathbf{x}_k^i),$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^D$. We aim to learn a classifier $f(\cdot)$ in terms of parameters $\mathbf{w}$. If $f(\cdot)$ is linear with respect to $\{\boldsymbol{\mu}_i\}$'s, then the form of $f(\cdot)$ can be parameterized as

$$f(\boldsymbol{\mu}_i) = \mathbf{w}^\top \boldsymbol{\mu}_i. \tag{17}$$

If $f(\cdot)$ is a nonlinear classifier, then it can be written as

$$f(\boldsymbol{\mu}_i) = \mathbf{w}^\top \tilde{\mathbf{z}}(\boldsymbol{\mu}_i), \tag{18}$$

where $\tilde{\mathbf{z}} : \mathbb{R}^D \rightarrow \mathbb{R}^{\tilde{D}}$ is another mapping of Random Fourier Features. (17) is a special case of (18) when $\tilde{\mathbf{z}}$ is an identity mapping. The resultant optimization problem on learning a classifier is reformulated accordingly as follows,

$$\min_{\mathbf{w}\in\mathbb{R}^{\tilde{D}}} \frac{1}{n}\sum_{k=1}^{K} \mathbf{C}_{k\hat{k}}\ell(\mathbf{w}^\top \tilde{\mathbf{z}}(\boldsymbol{\mu}_k), y_k) + \lambda\|\mathbf{w}\|_2^2. \tag{19}$$

As $\tilde{\mathbf{z}}(\cdot)$ is an explicit feature map, standard linear SVMs solvers can be applied to solve (19), which is much more efficient than solving (13). Accordingly, in the sequel, we denote this accelerated version of S-SMM$_{AR}$ with random Fourier features by R-SMM$_{AR}$.

## 4. Experiments

In this section, we investigate three different experimental settings: 1) different segmentation methods with fixed feature extraction; 2) joint segmentation and classification scenario; 3) feature extraction and classification under the perfect segmentation scenario. We conduct comprehensive experiments on four real-world activity recognition datasets to evaluate the effectiveness and scalability of our proposed S-SMM$_{AR}$ and its accelerated version R-SMM$_{AR}$.

### 4.1. Datasets

The overall statistics of the four benchmark datasets used in our experiments are listed in Table 1.

**Skoda** [47] contains 10 gestures performed during car maintenance scenarios. 20 sensors are placed on the left and right arms of the subject. The features are accelerations of 3 spatial directions of each sensor. Each gesture is repeated about 70 times.

**WISDM** is collected using accelerometers built into phones [22]. A phone was put in each subject's front pants leg pockets. Six regular activities were performed, i.e., walking, jogging, ascending stairs, descending stairs, sitting and standing.

**HCI** focuses on variations caused by displacement of sensors [10]. The gestures are arm movements with the hand describing different shapes, e.g., a pointing-up triangle, an upside-down triangle, and a circle. Eight sensors are attached to the right lower arm of each subject. Each gesture is recorded for over 50 repetitions, and each repetition for 5 to 8 seconds.

**PS** is collected by four smartphones on four body positions: [44]. The smartphones are embedded with accelerometers, magnetometers and gyroscopes. Four participants were asked to conduct six activities for several minutes: walking, running, sitting, standing, walking upstairs and downstairs.

### 4.2. Evaluation metric

For segmentation, we adopt an indicator Ind to indicate whether the method can find the exact correct number of breakpoints. We also adopt the $F_1$ score as our evaluation metric for classification. As the activity recognition datasets are imbalanced and of multiple classes, we adopt both micro-$F_1$ score (miF) and weighted macro-$F_1$ score (maF) to evaluation the performance of different methods. Note that the *Null* class is included during training and testing, and is always considered as a "negative" class when computing miF and maF. More specifically, miF is defined as follows,

$$\mathrm{miF} = \frac{2 \times \mathrm{precision}_{all} \times \mathrm{recall}_{all}}{\mathrm{precision}_{all} + \mathrm{recall}_{all}},$$

where $\mathrm{precision}_{all}$ and $\mathrm{recall}_{all}$ are computed from the pooled contingency table of all the positive classes as follows,

$$\mathrm{precision}_{all} = \frac{\sum_i \mathrm{TP}_i}{\sum_i \mathrm{TP}_i + \sum_i \mathrm{FP}_i}, \text{ and } \mathrm{recall}_{all} = \frac{\sum_i \mathrm{TP}_i}{\sum_i \mathrm{TP}_i + \sum_i \mathrm{FN}_i},$$

where $i$ denotes the $i$-th class of a set of predefined activity categories (i.e., positive classes), and $\mathrm{TP}_i$, $\mathrm{FP}_i$, and $\mathrm{FN}_i$ denote true positive, false positive, and false negative with respect to $i$-th positive class, respectively. Different from miF, maF is defined as follows,

$$\mathrm{maF} = \sum_i w_i \frac{2 \times \mathrm{precision}_i \times \mathrm{recall}_i}{\mathrm{precision}_i + \mathrm{recall}_i},$$

where $w_i$ is the proportion of the $i$-th positive class.

## 4.3. Experiments for segmentation

### 4.3.1. Experimental setup

In this section, we compare the segmentation performance of our proposed method with several state-of-the-art baselines. The feature extraction is fixed, and our proposed feature extraction method is applied after segmentation. There is no splitting of training and testing phase in this experiment. All the raw data as well as the coarse label sequence of the activities are available, and the goal is to decide the changepoints between each activity.

### 4.3.2. Baselines

We compare our proposed method with the following state-of-the-art methods.

- Binseg [12]: binary segmentation method, which finds one breakpoint in the dataset first, then splits the data into two subsegments, and the same procedure is applied recursively to subsegments.
- BottomUp [20]: contrary to binary segmentation, bottom-up method starts with many breakpoints and successively removes less important ones.
- Window [4]: fixed-size sliding window method with step size to be half the window size.
- KCpE [18]: a kernel-based nonparametric segmentation method which segments multi-dimensional data by minimizing intra-segment scatter. Dynamic programming is applied to recursively find breakpoints.
- KCpA [17]: a kernel-based test statistic based upon the maximum kernel Fisher discriminant ratio as a measure of homogeneity between segments. Sliding windows are running along the data.
- PELT [21]: a pruning DP method with exact optimal solution under certain conditions.
- E-Divisive [29]: a nonparametric technique which combines bisection and divergence measure to form a hierarchical statistical testing.
- e-cp3o and ks-cp3o [54]: dynamic programming with search space pruning. Two popular nonparametric goodness-of-fit metrics are utilized as cost functions, namely E-statistics and the Kolmogorov-Smirnov statistics.
- pDPA [40,41]: a functional pruning method which can only handle scalar data. Hence in the experiments, we only use the first dimension of the data.

### 4.3.3. Experimental results

The overall comparison results are listed in Table 2. Our proposed method achieves the best segmentation performance on three out of four datasets. All the RI values are quite close, but the classification performance of our proposed method is greater than the best baselines with the margin of 9% and 57% on Skoda and PS dataset respectively. It is interesting to find out that the performance of the proposed method seems to be relevant to the number of $\frac{\#Seg.}{\#C.}$ as listed in Table 1. The larger the average number of segments that each class of activity has, the better the performance of segmentation. This may shed light on the reason of the so much higher classification performance of the proposed method in PS dataset, since the $\frac{\#Seg.}{\#C.}$ value is much greater than that of other datasets. This is reasonable, since the more repetitions of activities in the dataset, the more accurate the class-wise similarity measure in the proposed method.

### 4.3.4. Runtime analysis

The runtime of each method is listed in Table 3. The Window method is the quickest baseline. It is expected that kernel-based methods, such as S-SMM$_{AR}$, KCpE and KCpA, take more runtime than other baselines due to the computation of

**Table 2**
Overall comparison results of segmentation performance on the four datasets (unit: %). NaN indicates that the produced results are infeasible.

| Datasets / Methods | Ind | Skoda | | WISDM | | HCI | | PS | |
|---|---|---|---|---|---|---|---|---|---|
| | | miF±std | maF±std | miF±std | maF±std | miF±std | maF±std | miF±std | maF±std |
| S-SMM$_{AR}$ | yes | **55.24±2.35** | **46.91±2.37** | **29.88±3.52** | **28.37±3.74** | 24.04±4.39 | 15.10±4.32 | **86.27±3.34** | **85.63±3.58** |
| Binseg | yes | 33.31±14.02 | 25.60±10.87 | 24.10±5.95 | 17.97±6.17 | 26.12±3.87 | 15.91±3.40 | 28.56±5.71 | 27.84±5.61 |
| BottomUp | yes | 46.43±6.87 | 36.95±8.91 | 28.80±2.86 | 26.86±4.50 | 21.15±2.65 | 11.22±4.45 | 19.84±4.69 | 17.49±4.19 |
| KCpE | yes | 45.35±22.25 | 38.41±18.85 | 25.24±6.78 | 20.25±4.71 | 24.04±4.39 | 15.10±4.32 | 20.37±6.56 | 19.54±5.97 |
| KCPA | no | 32.53±17.97 | 24.11±13.82 | 11.80±5.58 | 9.35±4.40 | **28.04±8.65** | 21.13±10.37 | 25.58±6.06 | 21.95±5.20 |
| PELT | no | 12.13±9.41 | 6.82±5.32 | 28.85±3.63 | 24.48±1.12 | 26.12±10.85 | **25.90±10.06** | 22.01±3.60 | 18.69±3.72 |
| Window | no | 0.77±1.08 | 0.71±0.96 | 28.42±2.02 | 26.55±4.45 | 13.62±4.53 | 12.96±1.89 | 16.15±4.89 | 15.88±4.78 |
| e.divisive | yes | 23.15±1.25 | 16.73±1.04 | 17.56±5.52 | 18.66±5.45 | 19.23±0.00 | 6.20±0.00 | 13.41±3.61 | 6.17±1.63 |
| ks.cp3o | yes | 20.81±0.76 | 13.29±1.25 | 13.73±2.45 | 13.14±1.98 | 19.23±0.00 | 6.20±0.00 | 14.65±2.02 | 3.79±1.05 |
| e.cp3o | yes | 22.77±2.51 | 13.72±2.38 | 22.53±8.54 | 16.92±4.22 | 20.83±0.50 | 7.20±0.28 | 14.38±2.56 | 5.89±0.73 |
| pDPA | no | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

**Table 3**
Runtime comparisons of all methods on the Skoda dataset.

| Methods | S-SMM$_{AR}$ | Binseg | BottomUp | KCpE | KCPA | PELT | Window | e.divisive | ks.cp3o | e.cp3o |
|---|---|---|---|---|---|---|---|---|---|---|
| Runtime | 19min23s | 67min20s | 25s | >24h | 34min56s | 35s | 19s | 1min48s | 139min38s | 2min50s |

**Table 4**
Overall comparison results on joint segmentation and feature extraction on four datasets (unit:%).

| Datasets / Methods | Skoda | | WISDM | | HCI | | PS | |
|---|---|---|---|---|---|---|---|---|
| | miF±std | maF±std | miF±std | maF±std | miF±std | maF±std | miF±std | maF±std |
| S-SMM$_{AR}$ | **51.65±6.18** | **42.98±6.33** | **28.18±4.13** | **27.61±4.33** | **23.88±4.14** | **15.15±3.97** | **86.44±3.44** | **85.81±3.70** |
| ECDF-5 | 16.29±7.99 | 9.48±4.68 | 26.16±3.18 | 32.08±3.44 | 14.42±3.99 | 13.44±3.93 | 18.11±5.24 | 17.63±5.33 |
| ECDF-15 | 22.91±7.86 | 17.19±5.41 | 16.83±1.86 | 21.60±1.42 | 12.82±7.06 | 11.65±5.31 | 16.71±2.99 | 16.36±2.85 |
| ECDF-30 | 23.51±7.51 | 19.79±6.40 | 10.95±2.61 | 13.30±3.79 | 11.86±6.25 | 9.91±4.28 | 16.43±1.73 | 16.11±1.66 |
| ECDF-45 | 25.96±5.58 | 23.36±5.54 | 10.43±3.57 | 11.18±4.05 | 10.74±6.86 | 8.67±3.95 | 15.87±1.98 | 15.48±2.01 |
| SAX-3 | 3.92±3.25 | 3.48±2.81 | 16.06±2.56 | 19.46±3.52 | 23.88±7.80 | 14.56±9.54 | 15.13±2.20 | 14.58±2.04 |
| SAX-6 | 2.11±1.89 | 2.06±1.86 | 15.28±2.57 | 18.39±3.24 | 19.39±3.77 | 9.48±3.46 | 16.95±1.46 | 15.87±1.51 |
| SAX-9 | 4.15±3.60 | 3.99±3.46 | 15.98±2.80 | 19.24±3.33 | 20.83±1.57 | 9.90±2.88 | 15.48±2.30 | 14.91±2.30 |
| SAX-10 | 2.69±2.09 | 2.65±2.10 | 15.27±3.14 | 18.73±3.53 | 22.44±5.82 | 11.67±6.34 | 16.43±1.11 | 15.49±1.05 |
| miFV-3 | 3.08±5.61 | 2.04±3.50 | 13.43±0.19 | 3.18±0.08 | 19.23±0.00 | 6.20±0.00 | 11.95±0.05 | 2.55±0.02 |
| miFV-6 | 18.22±7.58 | 13.70±4.99 | 13.43±0.19 | 3.18±0.08 | 19.23±0.00 | 6.20±0.00 | 11.95±0.05 | 2.55±0.02 |
| miFV-9 | 37.38±4.10 | 30.55±3.30 | 13.43±0.19 | 3.19±0.08 | 19.23±0.00 | 6.20±0.00 | 11.95±0.05 | 2.55±0.02 |
| miFV-10 | 33.57±4.67 | 27.30±4.37 | 13.43±0.19 | 3.18±0.08 | 19.23±0.00 | 6.20±0.00 | 11.95±0.05 | 2.55±0.02 |

kernel matrices. Pruned dynamic programming methods, i.e., PELT, e-cp3o and ks-cp3o, can save a lot of runtime compared with the original DP method KCpE. Our proposed method also saves a lot of runtime compared with the original dynamic programming method KCpE, and surpasses other kernel-based methods.

## 4.4. Experiments for joint segmentation and feature extraction

### 4.4.1. Experimental setup

In this scenario, we investigate the joint segmentation and classification performance of our method. For baseline methods, we apply the segmentation methods mentioned in their papers (for miFV method, sliding window methods are applied), and then conducted the corresponding feature extraction methods. The segmented data is randomly split into training and testing sets with a ratio of 70% : 30%. Both the training and testing data are set to contain activities of all classes. All the results are reported by taking average values together with the standard deviation over 6 repeated experiments. We compare the proposed method with the state-of-the-art baselines. In order to compare segmentation and feature extraction methods, to minimize the impact of classifiers, SVM is chosen as the unique classifier, and we use LIBSVM [6] for implementation.

### 4.4.2. Baselines

- **ECDF-*d***. ECDF-$d$ extracts $d$ descriptors per sensor per axis. The range is set to $d \in \{5, 15, 30, 45\}$ following the settings in [16].
- **SAX-*a***. Following the settings in [26], we set $N$ to be the number of frames of the segment, $n$ to be the dimension of features (thus no dimension reduction), alphabet_size $a \in \{3, \ldots, 10\}$.
- **miFV-*c***. miFV [51] is a state-of-the-art multi-instance learning method. It treats each segment of frames as a bag of instances, and adopts Fisher kernel to transform each bag into a vector. We follow the parameter tuning procedure in [51] with PCA energy set to 1.0 and the number of centers $c \in \{3, 6, 9, 10\}$.

### 4.4.3. Experimental results

As listed in Table 4, our proposed S-SMM$_{AR}$ has the best performance on all four datasets. The results clearly demonstrate the efficacy of our proposed unified framework to do segmentation and feature extraction. One potential reason of our proposed method surpassing other baselines may come from two aspects: 1) our segmentation methods are more accurate than the preprocessing step in baselines, which supports our motivation that segmentation is a crucial preprocessing step; 2) our feature extraction has no information loss but the baselines can only extract limited number of features. Nevertheless, the trade-off between performance and runtime is inevitable, i.e., our proposed method takes more runtime (around 30 minutes) than baseline methods (less than 10 minutes).

## 4.5. Experiments for classification with perfect segmentation

### 4.5.1. Experimental setup

In this scenario, we are given the ground truth segments of the raw data beforehand, and hence we focus on the classification performance of our proposed method. In our experiments, each dataset is randomly split into training and testing sets using a ratio of 70% : 30%. Missing values are replaced by the mean values of the certain class in the training data. PCA is conducted as preprocessing with 90% variance kept. All the results are reported by taking average values together with the standard deviation over 6 repeated experiments. We use SVMs as the base classifier, and LIBSVM [6] for implementation. For overall comparisons between our proposed methods and baseline methods, we use the RBF kernel $k(x, x') = \exp(-\gamma \|x - x'\|^2)$. Note that in S-SMM$_{AR}$, we use RBF kernels for both kernel embedding within each segment and classifier learning over different segments. We will further investigate different choices of kernels in S-SMM$_{AR}$. We tune the kernel parameter $\gamma$ as well as the tradeoff parameter $C$ in LibSVM, and choose optimal parameter settings based on 5-fold cross-validation on the training set. We compare S-SMM$_{AR}$ with the following baseline methods.

### 4.5.2. Baselines

Segment-based methods: this type of methods aims to aggregate sensor-reading segments of variable-lengths into feature vectors of a fixed-length. In order to compare feature extraction methods, to minimize the impact of classifiers, SVM is chosen as the unique classifier for different feature extraction methods.

- **Moment-*x***. All the frames in a segment are aggregated by extracting different orders of moments to concatenate a single feature vector to be fed to SVMs. We use Moment-$x$ to denote up to $x$ orders of moments (inclusive) are extracted to generate a feature vector.
- **ECDF-*d***. As introduced in previous section.
- **SAX-*a***. As introduced in previous section.
- **miFV**. As introduced in previous section. Optimal results are displayed with the number of centers from 1 to 10.

Frame-based methods: This type of methods considers each frame as an individual instance, whose class label is as the same as the corresponding segment's.

- **SVM-f** apply a SVM on frame-level data.
- **KNN-f** apply a kNN classifier on frame-level data, where the value of $k$ is tuned in the range of $\{1, \ldots, 10\}$.

### 4.5.3. Overall experimental results

The overall comparison results of proposed methods along with all the baseline methods are presented in Table 5. It is clear that with the perfect segmentation, the classification results are much better than those in Table 4. As can be seen from the table, on average, the performance of S-SMM$_{AR}$/Moment-$x$/ECDF-$d$ methods is much more stable than that of other methods. For example, SAX-$a$ methods perform very well on Skoda, but perform very poor on all the other datasets. And our proposed S-SMM$_{AR}$ performs best on three out of four datasets. This illustrates the effectiveness of using kernel embedding technique to generate feature vectors in a RKHS for capturing any order of moments of a segment. Moreover, we can also observe from the table that in general, SVMs trained on feature vectors that contain more moment information perform better. For instance, on average, Moment-10 > Moment-5 > Moment-2 > Moment-1 on the datasets Skoda, WISDM, and HCI. One might notice that miFV performs very poor on all the four datasets. The reason is that it's not robust enough with respect to imbalanced class and the Null class interruption in the activity data. If the activity data is arranged into a balanced manner, the performances of miFV improve about 10%. If the Null class is removed, the performances improve about 30%.

### 4.5.4. Impact on orders of moments

To further investigate impact of different orders of moments to be used for constructing feature vectors on activity recognition, we conduct experiments on HCI as shown in Fig. 1. In the figure, different curve denotes different sampling frequency on sensor readings, which results in different numbers of frames per segment on average. The x-axis indicates up to what orders of moments are used. Though the recognition results are more or less effected by using different sampling

**Table 5**
Overall comparison results on the four datasets (unit: %). The perfect prediction on HCI lies in the fact that the large # En. from Table 1. It means much more accurate record of each activity. WISDM has the same advantage, but the problem lies in the large # Sub., which greatly enlarges variance of each class, thus affects the prediction.

| | Skoda | | WISDM | | HCI | | PS | |
|---|---|---|---|---|---|---|---|---|
| Methods | miF | maF | miF | maF | miF | maF | miF | maF |
| S-SMM$_{AR}$ | **99.61±.24** | **99.60±.25** | 55.87±2.66 | 56.09±3.03 | **100±0** | **100±0** | **96.74±1.20** | **96.72±1.22** |
| Moment-1 | 92.46±1.97 | 92.39±2.01 | 38.30±4.10 | 44.63±12.22 | 91.35±2.28 | 91.32±2.33 | 93.90±.94 | 93.85±.93 |
| Moment-2 | 92.27±1.47 | 92.14±1.49 | 52.55±1.46 | 57.21±7.22 | 96.47±.79 | 96.47±.77 | 95.95±.86 | 95.94±.86 |
| Moment-5 | 94.49±1.66 | 94.45±1.70 | 57.31±5.91 | **62.52±9.81** | 97.76±.79 | 97.77±.78 | 93.31±.99 | 93.42±.93 |
| Moment-10 | 95.24±.63 | 95.23±.64 | **57.79±3.97** | 62.44±8.02 | 98.72±.79 | 98.72±.79 | 91.93±1.44 | 92.00±1.36 |
| ECDF-5 | 92.96±1.57 | 92.95±1.52 | 52.77±2.73 | 56.22±7.33 | **100±0** | **100±0** | 95.63±1.07 | 95.63±1.06 |
| ECDF-15 | 93.62±1.34 | 93.60±1.36 | 54.01±3.09 | 57.47±7.65 | **100±0** | **100±0** | 93.97±.96 | 94.04±.97 |
| ECDF-30 | 93.25±1.11 | 93.21±1.15 | 55.33±4.50 | 58.26±7.13 | **100±0** | **100±0** | 90.82±.53 | 91.05±.57 |
| ECDF-45 | 92.20±1.07 | 92.20±1.13 | 53.46±2.84 | 57.77±7.02 | **100±0** | **100±0** | 87.15±1.32 | 87.23±1.59 |
| SAX-3 | 94.54±1.28 | 94.48±1.21 | 32.90±1.47 | 23.62±1.81 | 21.15±0 | 7.39±0 | 50.28±2.40 | 41.30±3.89 |
| SAX-6 | 96.13±1.57 | 96.10±1.55 | 35.49±3.11 | 28.77±2.82 | 21.15±0 | 7.39±0 | 52.95±2.54 | 46.86±.68 |
| SAX-9 | 97.36±1.33 | 97.31±1.34 | 32.43±1.16 | 23.84±1.61 | 21.15±0 | 7.39±0 | 51.70±1.14 | 43.58±1.52 |
| SAX-10 | 96.22±.84 | 96.18±.83 | 32.57±1.48 | 26.89±2.39 | 21.15±0 | 7.39±0 | 52.81±1.08 | 44.60±1.52 |
| miFV | 61.40±3.24 | 53.63±2.50 | 14.61±2.04 | 4.72±2.13 | 21.64±1.58 | 18.78±2.24 | 15.32±4.28 | 7.65±5.83 |
| SVM-f | 93.46±1.20 | 92.65±1.38 | 27.49±2.71 | 18.70±2.88 | 99.52±.53 | 99.52±.53 | 95.22±1.10 | 95.21±1.10 |
| kNN-f | 93.17±1.44 | 92.93±1.45 | 28.48±2.15 | 17.96±2.84 | 99.04±1.22 | 99.05±1.21 | 94.73±.65 | 94.72±.65 |

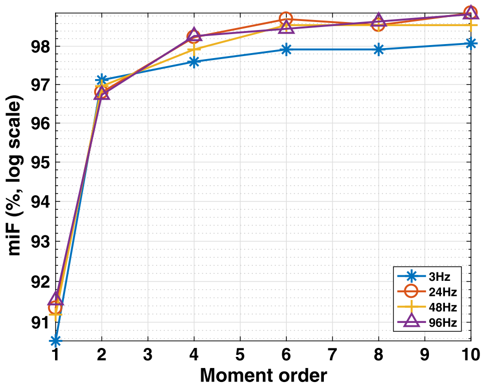

**Fig. 1.** Comparison results of Moment-*x* in terms of miF on HCI by varying moments and frequencies.

frequencies on sensor readings, their increasing trends with more orders of moments are the same. These favourably prove our idea that incorporating more moment information in the feature vectors benefits the activity recognition performance. Hence the proposed method is likely to perform the best since all orders of moments information are utilized in the proposed method.

#### 4.5.5. *Impact of sampling frequency on sensor readings*

Maurer et al. [30] found that when increasing the sampling frequency, there is no significant gain in accuracy above 20Hz for activities. Here, we conduct experiments to analyze the impact of sampling frequency on the classification performance of S-SMM$_{AR}$. Fig. 2 shows the miF performance of S-SMM$_{AR}$ on Skoda under different sampling rates varying from 0.5Hz to 14Hz, resulting in average numbers of frames per segment varying from 3 to 68. The classification performance increases with larger average number of frames per segment, then becomes stable between 10 to 70 frames/segment. Therefore, our suggestion is that to use S-SMM$_{AR}$ for activity recognition, each segment needs to contain 10 or more frames, which is reasonable in practice.

#### 4.5.6. *Impact on different choices of kernels*

In S-SMM$_{AR}$, there are two types of kernels: $k(\cdot,\cdot)$ for kernel embedding within each segment (3) and $\tilde{k}(\cdot,\cdot)$ for training a nonlinear classifier (4). In this section, we conduct experiments to investigate the impact of different combinations of kernels on the final classification performance of S-SMM$_{AR}$. The results are shown in Table 6, where linear kernel (LIN), polynomial kernel of degree 3 (POLY3), RBF kernel and sigmoid kernel (SIG) are used. When S-SMM$_{AR}$ uses the RBF kernel for both $k(\cdot,\cdot)$ and $\tilde{k}(\cdot,\cdot)$, it performs best. Moreover, when the sigmoid kernel is used for kernel embedding, S-SMM$_{AR}$ performs worst. This may be because sigmoid kernel is not positive semi-definite, thus not characteristic, which may not be able to capture sufficient statistics for each segment (or sample).

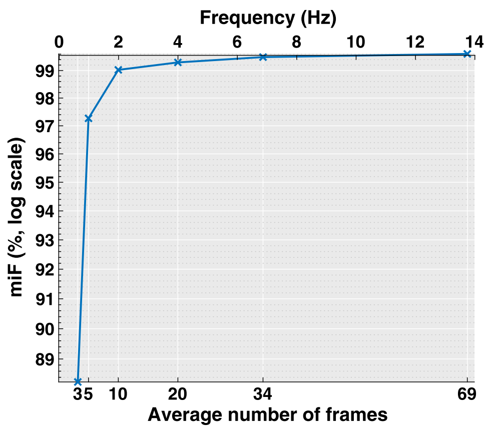

**Fig. 2.** The miF performance on Skoda under different sampling frequencies and different average numbers of frames for each segment. The x-axis on the top and the x-axis are relevant as a lower sampling frequency on sensor readings leads to a smaller number of frames per segment.

**Table 6**
Comparison performance in terms of miF of S-SMM$_{AR}$ on Skoda with different combinations of kernels.

| | | $\tilde{k}(\cdot,\cdot)$ | | | |
|---|---|---|---|---|---|
| | | LIN | POLY3 | RBF | SIG |
| $k(\cdot,\cdot)$ | LIN | 91.4300 | 91.3852 | 91.3632 | 28.6446 |
| | POLY3 | 98.1202 | 98.0728 | 98.1556 | 92.0938 |
| | RBF | 98.1422 | 90.8818 | **98.8950** | 98.3728 |
| | SIG | 87.7026 | 87.0830 | 90.4140 | 90.4176 |

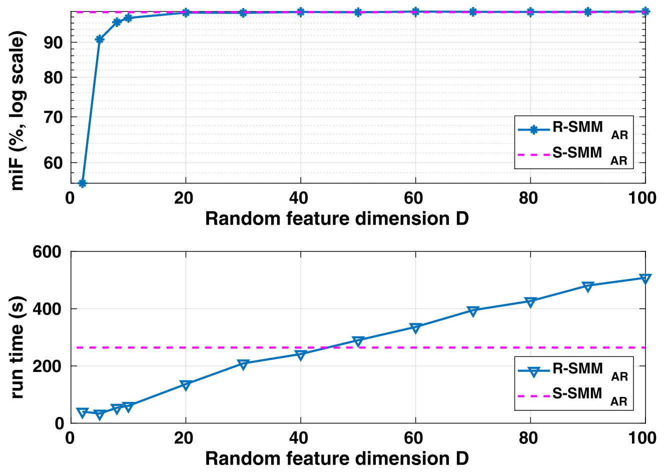

**Fig. 3.** Comparison results between S-SMM$_{AR}$ and R-SMM$_{AR}$ in terms of runtime and miF score on Skoda.

### *4.5.7. Experimental results on R-SMM$_{AR}$*

In our final series of experiments, we test the scalability and effectiveness of our proposed accelerated version R-SMM$_{AR}$. Fig. 3 illustrates the trends of performance and runtime with increasing sizes of random feature dimension $D$, respectively. The experiments are conducted on a Linux computer with Intel(R) Core(TM) i7-4790S 3.20GHz CPU. The runtime in seconds shown in the figure is the total runtime in both training and testing. As can be seen that with the increase of $D$, the runtime of R-SMM$_{AR}$ increases accordingly, and performance in terms of miF becomes higher. Note that the best performance of S-SMM$_{AR}$ in terms of miF on Skoda is 99.61%, with runtime of 264 seconds. R-SMM$_{AR}$ is able to achieve a comparable miF score with small standard deviation when $10 \le D \le 40$, while requires much less runtime. Therefore, compared with S-SMM$_{AR}$, R-SMM$_{AR}$ is an efficient and effective approximation approach, which is suitable for large-scale datasets. It saves a large proportion of runtime, and at the mean time, achieves comparable performance.

## 5. Conclusion and future work

In this paper, we propose a novel unified weakly-supervised framework, S-SMM$_{AR}$, to jointly segment the activity data and extract all statistical moments of the activity data. This is the very first work to apply the idea of kernel embedding in the context of activity recognition problems. We investigate the performance of general time-series segmentation methods on the specific activity data. We conduct extensive experiments and demonstrate the effectiveness of S-SMM$_{AR}$ compared with a number of baseline methods. Moreover, we also present an accelerated version R-SMM$_{AR}$ to solve large-scale problems. In the future, besides statistical information, we plan to extend the proposed method to capture temporal information of each segment for learning feature representation of each segment.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This research is partially supported by the NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020, Singapore MOE AcRF Tier-1 grant 2018-T1-002-143, the National Research Foundation-Prime Minister's office, Republic of Singapore under its IDM Futures Funding Initiative, the Singapore Ministry of Health under its National Innovation Challenge on Active and Confident Ageing (NIC Project No. MOH/NIC/COG04/2017 and MOH/NIC/HAIG03/2017), and the Interdisciplinary Graduate School, Nanyang Technological University under its Graduate Research Scholarship.

## References

[1] A. Akbari, J. Wu, R. Grimsley, R. Jafari, Hierarchical signal segmentation and classification for accurate activity recognition, in: UbiComp/ISWC Adjunct, ACM, 2018, pp. 1596–1605.
[2] S. Aminikhanghahi, T. Wang, D.J. Cook, Real-time change point detection with application to smart home time series data, IEEE Trans. Knowl. Data Eng. 31 (2019) 1010–1023.
[3] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, P.J.M. Havinga, Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: a survey, in: ARCS Workshops, 2010, pp. 167–176.
[4] O. Baños, J.M. Galvez, M. Damas, H. Pomares, I. Rojas, Window size impact in human activity recognition, Sensors 14 (2014) 6474–6499.
[5] A. Bulling, U. Blanke, B. Schiele, A tutorial on human activity recognition using body-worn inertial sensors, ACM Comput. Surv. 46 (2014) 33:1–33:33.
[6] C. Chang, C. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 27:1–27:27.
[7] J. Chen, A.K. Gupta, Parametric Statistical Change Point Analysis: With Applications to Genetics, Medicine, and Finance, Springer Science & Business Media, 2011.
[8] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, Y. Liu, Deep learning for sensor-based human activity recognition: overview, challenges and opportunities, CoRR, arXiv:2001.07416 [abs], 2020.
[9] L. Chen, J. Hoey, C.D. Nugent, D.J. Cook, Z. Yu, Sensor-based activity recognition, IEEE Trans. Syst. Man Cybern., Part C 42 (2012) 790–808.
[10] K. Förster, D. Roggen, G. Tröster, Unsupervised classifier self-calibration through repeated context occurences: is there robustness against sensor displacement to gain?, in: ISWC, 2009, pp. 77–84.
[11] J. Frank, S. Mannor, D. Precup, Activity and gait recognition with time-delay embeddings, in: AAAI, 2010.
[12] P. Fryzlewicz, et al., Wild binary segmentation for multiple change-point detection, Ann. Stat. 42 (2014) 2243–2281.
[13] E. Fuchs, T. Gruber, J. Nitschke, B. Sick, Online segmentation of time series based on polynomial least-squares approximations, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 2232–2245.
[14] Y. Guédon, Exploring the latent segmentation space for the assessment of multiple change-point models, Comput. Stat. 28 (2013) 2641–2678.
[15] N.Y. Hammerla, S. Halloran, T. Plötz, Deep, convolutional, and recurrent models for human activity recognition using wearables, preprint, arXiv:1604.08880, 2016.
[16] N.Y. Hammerla, R. Kirkham, P. Andras, T. Ploetz, On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution, in: ISWC, 2013, pp. 65–68.
[17] Z. Harchaoui, F.R. Bach, E. Moulines, Kernel change-point analysis, in: NIPS, 2008, pp. 609–616.
[18] Z. Harchaoui, O. Cappé, Retrospective mutiple change-point estimation with kernels, in: IEEE/SP Workshop on SSP, 2007, pp. 768–772.
[19] M. Janidarmian, A.R. Fekr, K. Radecka, Z. Zilic, A comprehensive analysis on wearable acceleration sensors in human activity recognition, Sensors 17 (2017) 529.
[20] E. Keogh, S. Chu, D. Hart, M. Pazzani, An online algorithm for segmenting time series, in: ICDM, 2001, pp. 289–296.
[21] R. Killick, P. Fearnhead, I.A. Eckley, Optimal detection of changepoints with a linear computational cost, J. Am. Stat. Assoc. 107 (2012) 1590–1598.
[22] J.R. Kwapisz, G.M. Weiss, S. Moore, Activity recognition using cell phone accelerometers, SIGKDD Explor. 12 (2010) 74–82.
[23] O.D. Lara, M.A. Labrador, A survey on human activity recognition using wearable sensors, IEEE Commun. Surv. Tutor. 15 (2013) 1192–1209.
[24] K. Li, R. Habre, H. Deng, R. Urman, J. Morrison, F.D. Gilliland, J.L. Ambite, D. Stripelis, Y.-Y. Chiang, Y. Lin, et al., Applying multivariate segmentation methods to human activity recognition from wearable sensors' data, JMIR mHealth uHealth 7 (2019).
[25] H. Lin, C. Lin, R.C. Weng, A note on platt's probabilistic outputs for support vector machines, Mach. Learn. 68 (2007) 267–276.
[26] J. Lin, E.J. Keogh, L. Wei, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, Data Min. Knowl. Discov. 15 (2007) 107–144.
[27] J.W. Lockhart, G.M. Weiss, Limitations with activity recognition methodology & data sets, in: UbiComp, 2014, pp. 747–756.
[28] R. Maidstone, T. Hocking, G. Rigaill, P. Fearnhead, On optimal multiple changepoint algorithms for large data, Stat. Comput. 27 (2017) 519–533.
[29] D.S. Matteson, N.A. James, A nonparametric approach for multiple change point analysis of multivariate data, J. Am. Stat. Assoc. 109 (2014) 334–345.
[30] U. Maurer, A. Smailagic, D.P. Siewiorek, M. Deisher, Activity recognition and monitoring using multiple sensors on different body positions, in: BSN, 2006, pp. 113–116.
[31] K. Muandet, From Points to Probability Measures: A Statistical Learning on Distributions with Kernel Mean Embedding, Ph.D. thesis, University of Tübingen, Germany, 2015.

[32] K. Muandet, K. Fukumizu, F. Dinuzzo, B. Schölkopf, Learning from distributions via support measure machines, in: NIPS, 2012, pp. 10–18.
[33] Q. Ni, T. Patterson, I. Cleland, C.D. Nugent, Dynamic detection of window starting positions and its implementation within an activity recognition framework, J. Biomed. Inform. 62 (2016) 171–180.
[34] M.H.M. Noor, Z.A. Salcic, K.I. Wang, Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer, Pervasive Mob. Comput. 38 (2017) 41–59.
[35] T. Plötz, N.Y. Hammerla, P. Olivier, Feature learning for activity recognition in ubiquitous computing, in: IJCAI, 2011, pp. 1729–1734.
[36] H. Qian, S.J. Pan, B. Da, C. Miao, A novel distribution-embedded neural network for sensor-based activity recognition, in: IJCAI, 2019, pp. 5614–5620, ijcai.org.
[37] H. Qian, S.J. Pan, C. Miao, Sensor-based activity recognition via learning from distributions, in: AAAI, 2018, pp. 6262–6269.
[38] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: NIPS, 2007, pp. 1177–1184.
[39] N. Ravi, N. Dandekar, P. Mysore, M.L. Littman, Activity recognition from accelerometer data, in: AAAI, 2005, pp. 1541–1546.
[40] G. Rigaill, Pruned dynamic programming for optimal multiple change-point detection, preprint, arXiv:1004.0887, 2010.
[41] G. Rigaill, A pruned dynamic programming algorithm to recover the best segmentations with 1 to k_max change-points, J. Soc. Fr. Stat. 156 (2015) 180–205.
[42] B. Schölkopf, A.J. Smola, Learning with Kernels: support vector machines, regularization, optimization, and beyond, 2002.
[43] A. Shahi, B.J. Woodford, H. Lin, Dynamic real-time segmentation and recognition of activities using a multi-feature windowing approach, in: PAKDD (Workshops), 2017, pp. 26–38.
[44] M. Shoaib, H. Scholten, P.J.M. Havinga, Towards physical activity recognition using smartphone sensors, in: UIC/ATC, 2013, pp. 80–87.
[45] A.J. Smola, A. Gretton, L. Song, B. Schölkopf, A Hilbert space embedding for distributions, in: ALT, 2007, pp. 13–31.
[46] B.K. Sriperumbudur, K. Fukumizu, A. Gretton, G.R.G. Lanckriet, B. Schölkopf, Kernel choice and classifiability for RKHS embeddings of probability distributions, in: NIPS, 2009, pp. 1750–1758.
[47] T. Stiefmeier, D. Roggen, G. Tröster, Fusion of string-matched templates for continuous activity recognition, in: ISWC, 2007, pp. 41–44.
[48] D. Triboan, L. Chen, F. Chen, Z. Wang, A semantics-based approach to sensor data segmentation in real-time activity recognition, Future Gener. Comput. Syst. 93 (2019) 224–236.
[49] M. Vrigkas, C. Nikou, I.A. Kakadiaris, A review of human activity recognition methods, Front. Robot. AI 2015 (2015).
[50] J. Wang, Y. Chen, S. Hao, X. Peng, L. Hu, Deep learning for sensor-based activity recognition: a survey, Pattern Recognit. Lett. 119 (2019) 3–11.
[51] X. Wei, J. Wu, Z. Zhou, Scalable algorithms for multi-instance learning, IEEE Trans. Neural Netw. Learn. Syst. 28 (2017) 975–987.
[52] Q. Yang, Activity recognition: linking low-level sensors to high-level intelligence, in: IJCAI, 2009, pp. 20–25.
[53] J. Yin, D. Shen, Q. Yang, Z. Li, Activity recognition through goal-based segmentation, in: AAAI, 2005, pp. 28–34.
[54] W. Zhang, N.A. James, D.S. Matteson, Pruning and nonparametric multiple change point detection, in: ICDM Workshops, 2017, pp. 288–295.
[55] J. Zhu, J. Mao, A.L. Yuille, Learning from weakly supervised data by the expectation loss SVM (e-svm) algorithm, in: NIPS, 2014, pp. 1125–1133.