Noname manuscript No.
(will be inserted by the editor)

MasterMovelets: Discovering Heterogeneous Movelets for Multiple Aspect Trajectory Classification

Carlos Andres Ferrero · Lucas May Petry · Luis Otavio Alvares · Camila Leite da Silva · Willian Zalewski · Vania Bogorny

Received: date / Accepted: date

Abstract In the last few years trajectory classification has been applied to many real problems, basically considering the dimensions of space and time or attributes inferred from these dimensions. However, with the explosion of social media data and the advances in the semantic enrichment of mobility data, a new type of trajectory data has emerged, and the trajectory spatio-temporal points have now multiple and heterogeneous semantic dimensions. By semantic dimensions we mean any type of information that is neither spatial nor temporal. As a consequence, new classification methods are needed to deal with this new type of spatio-temporal data. The main challenge is how to automatically select and combine the data dimensions and to discover the subtrajectories that better discriminate the class. In this paper we propose *MasterMovelets*, a new parameter free method for trajectory classification which finds the best trajectory partition and dimension combination for robust high dimensional trajectory classification. Experimental results show that our approach outperforms state-of-the-art methods by reducing the classification error up to 50%, indicating that our proposal is very promising for multidimensional sequence data classification.

Keywords multiple aspect trajectory · trajectory classification · relevant subtrajectories · multidimensional sequence classification · movelets · semantic trajectory classification

Carlos Andres Ferrero
Instituto Federal de Santa Catarina, IFSC, Lages, Santa Catarina, Brazil
and PPGCC/UFSC, Florianopolis, Brazil
E-mail: andres.ferrero@ifsc.edu.br

Willian Zalewski
Universidade Federal da Integração Latino-Americana, Foz do Iguaçu, Parana, Brazil

Lucas May Petry · Camila Leite da Silva · Luis Otavio Alvares · Vania Bogorny
Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, Brazil

1 Introduction

Trajectory classification is the data mining task for predicting the class labels of moving objects based on their trajectories (Lee et al., 2008). For instance, given a set of trajectories, (i) infer the class label of the transportation mode (e.g. car, taxi, bus, bike), (ii) the strength of a hurricane, (iii) the category of an animal specie, (iv) the user of a trajectory, (v) the type of a vessel, etc. Existing works, as Lee et al. (2008) and Patel et al. (2012), have considered trajectories as sequences of space-time points, called raw trajectories, as the example shown in Figure 1; or proposed methods for specific problems as transportation mode classification, as the works of Zheng et al. (2010), Xiao et al. (2017), and Etemad et al. (2018). A *raw trajectory* is defined as a sequence of elements $\langle e_1, e_2, \ldots, e_n \rangle$, where each element has three dimensions, x, y, t, where x and y correspond to the position of the object in space at time t.

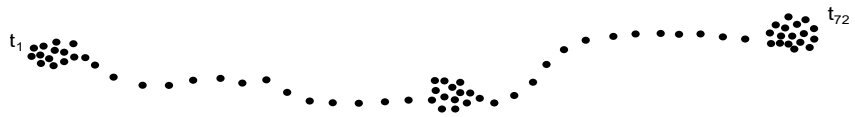

Fig. 1: Example of *raw trajectory*.

With the explosion of social media data as Foursquare, Twitter, Facebook, and Internet channels, as Weather Wunderground, trajectories are being enriched with more semantic and heterogeneous information, as the example shown in Figure 2. In this figure, an individual starts his/her trajectory at home, then he/she goes to work by car, and after work, he/she goes eating at a restaurant. During his/her movement the weather condition changes two times, from rainy to cloudy and then to sunny, and part of the trajectory is performed on foot and part by car. In addition, the individual uses different social networks (e.g. Twitter, Facebook, and Foursquare) to post his/her feelings. This new type of trajectory is called *Multiple Aspect Trajectory* (Ferrero et al., 2016; Mello et al., 2019), where the movement of an individual is enriched with *multiple* and *heterogeneous* data dimensions, including timestamps, spatial coordinates, and information as the category of the visited place, the name of the place, the rating of the visited place, the price, the weather condition, etc. We strongly believe that this new type of mobility data will be the research challenge of the next generation of trajectory pattern mining methods.

The main difference between a multiple aspect trajectory and a raw trajectory is the number and type of dimensions. A multiple aspect trajectory is a sequence of elements $\langle e_1, e_2, \ldots, e_m \rangle$, where each element has the dimensions x, y, t, A, where x and y correspond to the position of the object in space at

time t, and $A = \{a_1, a_2, ...a_k\}$ is a set of semantic dimensions, also called aspects or attributes. We call semantic dimensions any sort of information that is neither spatial nor temporal.

The number of heterogeneous data dimensions associated to multiple aspect trajectories increases the complexity of trajectory classification because of two main problems: (i) data dimensions cannot be fused in a single attribute value because of the nature of each dimension, and different distance functions are needed to compare heterogeneous attributes; and (ii) the trajectory must be segmented into subtrajectories in order to find patterns, since the entire trajectory may not discriminate the class (Lee et al., 2008).

Fig. 2: Example of Multiple Aspect Trajectory.

There are basically two ways to perform trajectory classification: to extract subtrajectories that discriminate the class or to extract features as speed, acceleration, direction change, etc. Several works such as Dodge et al. (2009), Zheng et al. (2010), Xiao et al. (2017), and Etemad et al. (2018) extract a set of spatio-temporal features either from the entire raw trajectory or from subtrajectories, as the average speed, average acceleration, and the number of points of subtrajectories with high and low speed. The main drawback of these approaches is that they are limited to numerical features, do not support the heterogeneous semantic dimensions of multiple aspect trajectories, and most of them are limited to a specific application as transportation mode classification.

According to Ferrero et al. (2018), for many applications the relevant subtrajectories do better discriminate the class than spatio-temporal features, but the challenge is how to discover these subtrajectories and which are the best dimensions and their combinations for the classification problem. Lee et al. (2008), for instance, extract subtrajectories considering only the spatial dimension, and the trajectory partition is based on the physical movement of the trajectory. Patel et al. (2012) extended the method of Lee et al. (2008) to consider both *space* and *time*, but the method is still limited to these dimensions. The main limitation of these methods is to perform trajectory partition based on a predefined criteria based on space or space/time and not dealing with semantic dimensions.

Ferrero et al. (2018) proposed the method MOVELETS for discovering subtrajectories without the need of a predefined splitting criteria, and showed that the method outperformed all previous works for raw trajectory classification

in several datasets. MOVELETS supports multiple dimensions, such as space, time, and semantics. The *main problem* of MOVELETS is that all the trajectory dimensions are considered together to find relevant subtrajectories, i.e., the dimensions are fused in a single feature. This fusion can hide discriminant dimensions, that if considered individually could better discriminate the class. This is also the case of some similarity measures that can be used for Nearest Neighbour Classification (kNN) as LCSS (Longest common subsequence) (Vlachos et al., 2002) and EDR (Edit Distance for Real Sequences) (Chen et al., 2005), and the distance measures MD-DTW (Multi-dimensional Distance Warping) (ten Holt et al., 2007b) and DTWa (Shokoohi-Yekta et al., 2017). For these measures the user must evaluate each dimension separately and all possible dimension combinations to find the most discriminant ones for classification problems.

In multiple aspect trajectories one class may be better characterized by subtrajectories with the dimensions spatial location and time, while another class may be better characterized by subtrajectories with a single dimension as the POI category or the POI category and its price. By observing Figure 2 we notice that the more dimensions a trajectory has, the more difficult it becomes to find the subtrajectories with the best dimensions to distinguish movement classes. We claim that finding the best dimension subset for each subtrajectory is a key issue to capture the relation among dimensions and to discover the most relevant subtrajectories for classification problems.

In this paper we propose a new method for discovering relevant subtrajectories for multiple aspect trajectory classification that automatically finds the best subtrajectory size and dimension combination to discriminate the class label. The proposed approach is generic and can be applied to multidimensional sequences of different application domains. We show with experiments on real data that this new method that deals with heterogeneous data dimensions, significantly reduces the trajectory classification error. The main contributions of our work are summarized as follows:

(i) a new domain independent and parameter-free method for discovering the most relevant subtrajectories with different and heterogeneous dimensions and of varying length for classification problems, called MASTERMOVELETS;
(ii) a new method for multidimensional alignment, called MASTERALIGNMENT, for discovering the part of a trajectory that is the most similar (or closest) to a given subtrajectory, taking into account multiple and different dimensions;
(iii) a new method to evaluate multidimensional subtrajectories for classification problems, called MASTERRELEVANCE; and
(iv) a robust experimental evaluation over real datasets for trajectory classification to demonstrate that MASTERMOVELETS outperforms the state-of-the-art methods on multiple aspect trajectories.

The remaining of this paper is structured as follows: Section 2 presents related work, Section 3 describes the proposed method, Section 4 shows the experiments, and Section 5 presents the conclusions and future work.

2 Related Work

Most works in trajectory classification as Lee et al. (2008); Dodge et al. (2009); Patel et al. (2012); Xiao et al. (2017) are limited to raw trajectories, basically considering the sequence of the time-ordered spatial locations of the moving object. These works extract global and/or local features of trajectories to build a classification model. The work of Xiao et al. (2017), for instance, uses global features for classifying transportation modes. Local features have been used in the works of Lee et al. (2008); Dodge et al. (2009); Zheng et al. (2010); Patel et al. (2012), which consist on using features extracted from trajectory parts, such as speed, acceleration, duration, and turning angle. However, as these methods are based on the spatial dimension of movement, they only support raw trajectory data and numerical dimensions, not being robust for highly dimensional trajectories.

All these previous methods for raw trajectory classification were outperformed by the recent work of Ferrero et al. (2018), in the datasets of hurricanes[1], animals (Rowland et al., 1997), vehicles (Frentzos et al., 2005), and transportation modes (Zheng et al., 2010). The method proposed by Ferrero et al. (2018), called MOVELETS, is parameter-free and can handle several dimensions, but its drawback is the transformation of all dimensions of a sub-trajectory in a single feature through a function that encapsulates all dimensions. This implies in considering all dimensions, ignoring the cases where some classes can be better discriminated by a subset of dimensions.

In the recent years, trajectories based on social networks are becoming more common, because of the explosion of social media and location based services, like Foursquare. These trajectories are more sparse with a low sampling of spatio-temporal points in comparison to raw GPS trajectories, so global and local geometrical features such as speed and acceleration, used in previous works to build classification models, cannot be extracted because the detailed movement between check-ins is not recorded. In addition, check-ins provide more semantic and textual information beyond the spatial location, such as the place category, the price, the rating, the reviews, etc., which allows trajectories to be analyzed under multiple aspects or semantic dimensions.

The work of Gao et al. (2017) was specifically developed for trajectories obtained from social media, but without exploring the number of heterogeneous semantic dimensions that characterize social media data. Gao et al. (2017) proposed BiTULER, a model that uses word embeddings and a Bidirectional Recurrent Neural Network, but it is limited to the sequence of check-in identifiers, not supporting other dimensions that characterize movement.

[1] http://weather.unisys.com/hurricanes

Another way to classify trajectories is using distance and similarity measures to perform Nearest Neighbor classification. This type of classification requires a trajectory distance function with support to sequential data and that is easily adaptable to multiple and heterogeneous dimensions. Most works in trajectory similarity measuring are limited to a fixed number of dimensions or are not adaptable to consider multiple and heterogeneous types of attributes.

The most well known distance and similarity measures developed for sequential data are Dynamic Time Warping (DTW) (Berndt and Clifford, 1994), Longest Common Subsequences (LCSS) (Vlachos et al., 2002), Edit Distance for Real Sequences (EDR) (Chen et al., 2005), and Multidimensional Similarity Measure (MSM) (Furtado et al., 2015). DTW and its variations as MD-DTW and DTWa were originally developed for time series and for numerical values only. DTW finds an optimal alignment between two sequences, which are warped in a nonlinear manner to match each other. ten Holt et al. (2007a) adapted DTW for multidimensional sequences, called MD-DTW, by transforming the distance values of all dimensions in only one distance value. Recently, Shokoohi-Yekta et al. (2017) proposed the DTWa, an adaptive approach of DTW for multidimensional sequence classification. LCSS consists of finding the longest common subsequence between two trajectories. EDR consists of seeking the minimum number of edit operations (insert, delete, change) to transform one trajectory into another. MSM is a multidimensional similarity measure developed for trajectories represented by multiple and heterogeneous dimensions, considering space, time, and semantics. MSM considers all dimensions as independent from each other and allows partial matching, but it ignores the sequence of the movement.

The main problem related to distances and similarity measures for classification is that, in the context of multiple and heterogeneous dimensions, we need to define weights and/or thresholds for each dimension. The measure MD-DTW uses weights, LCSS and EDR need thresholds, and MSM needs both thresholds and weights. These parameters are domain dependent and very difficult to estimate, becoming more challenging for heterogeneous dimensions where each one may use a specific distance function because of the nature of the data (e.g *space* distance is given in meters, *time* in minutes, *price* in price units, etc).

In the following section we propose a new method for multiple aspect trajectory classification problems.

3 Proposal: A Method for Discovering Heterogeneous Movelets for Trajectory Classification

In this section we detail our proposal for finding relevant subtrajectories in trajectories represented by multiple and heterogeneous dimensions for effective trajectory classification. Section 3.1 formulates the problem and introduces the basic definitions for the proposed method, Section 3.2 describes the method MasterMovelets, Section 3.3 presents a new method for heterogeneous subtrajectory alignment, Section 3.4 presents a new method for measuring

the quality of multidimensional subtrajectories, and Section 3.5 presents the complexity analysis.

3.1 Problem Statement and Main Definitions

To understand the problem of trajectory classification and the proposed approach, we first define a general concept of trajectory in Definition 1. This concept of trajectory was introduced by Furtado et al. (2015). From this point we will refer to Multiple Aspect trajectory as *trajectory*.

Definition 1 Trajectory. A trajectory T is a sequence of elements $\langle e_1, e_2, \ldots, e_m \rangle$, where each element has a set of l dimensions $D = \{d_1, d_2, \ldots, d_l\}$.

The problem we address in this paper is predicting the class labels of the moving objects based on their trajectories (Lee et al., 2008). Based on the concept of *classification* proposed by Tan et al. (2005), we define *trajectory classification* as follows:

Definition 2 Trajectory Classification. Given a trajectory set defined by a set of pairs $\mathbf{T} = \{(T_1, class_{T_1}), (T_2, class_{T_2}), \ldots, (T_n, class_{T_n})\}$, where each pair contains a trajectory and its class label, *trajectory classification* is the task of learning a prediction function f that maps each trajectory T_i of $\mathbf{T}$ to one of the predefined class labels.

Differently from conventional data classification, in trajectory classification the main problem is to find the best trajectory features to feed to the classifier, and these features can belong to a trajectory part, and not necessarily to the entire trajectory. In this work we want to find the most relevant parts of trajectories as the best features for trajectory classification. A trajectory part is called *subtrajectory*, and is given in Definition 3.

Definition 3 Subtrajectory. Given a trajectory T of length m, a *subtrajectory* $s = \langle e_a, \ldots, e_b \rangle$ is a contiguous subsequence of T starting at element e_a and ending at element e_b, where $1 \leq a \leq m$ and $a \leq b \leq m$. The subtrajectory s can be represented by all dimensions of D or a subset of dimensions $D' \subseteq D$. The length of the subtrajectory is defined as $w = |s|$. In addition, we also define the set of all *subtrajectories* of length w in T as S_T^w, and the *subtrajectories* of all lengths in T as S_T^*.

In order to find discriminant subtrajectories we start by defining the distance between two subtrajectories. This distance must consider the different dimensions. Since an element may have multiple and heterogeneous dimensions, we formally define the concept of *distance between elements*.

Definition 4 Distance vector between two multidimensional elements. Given two elements e_i and e_j represented by d dimensions, the distance between two multidimensional elements $dist(e_i, e_j)$ returns a *distance vector* $V = (v_1, v_2, ..., v_d)$, where each $v_k = dist\_e_k(e_i, e_j)$ is the distance between two elements at dimension k, that respects the property of symmetry $dist\_e_k(e_i, e_j) = dist\_e_k(e_j, e_i)$.

The idea behind Definition 4 is to allow using a distance function for each dimension, and storing the distance values of all dimensions into a *distance vector*. This approach differs from the method MOVELETS (Ferrero et al., 2018), where the function $dist(e_i, e_j)$ returns a single distance value, loosing the details about the distance along dimensions. For that reason, our proposal is to preserve the distance values along dimensions in a *distance vector* which is fundamental to compute the distance between two subtrajectories of equal length, preserving the distances on each dimension, that is given in Definition 5.

Definition 5 Distance vector between two subtrajectories of equal length. Given two subtrajectories s and r both of length w and d dimensions, $dist\_s(s, r)$ computes the pairwise distance between the subtrajectory elements in a *distance vector* $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d)$, where each $\mathbf{v}_k$ is the distance value between s and r at dimension k, which is obtained by a function over the w distances between the two subtrajectories at dimension k. Each distance $\mathbf{v}_k$ respects the property of symmetry $dist\_s_k(s, r) = dist\_s_k(r, s)$.

Finding the subtrajectory that is the most similar to a given subtrajectory is another essential part of our method. The most similar subtrajectory of a trajectory T to a subtrajectory s is called *best alignment*, and is the subtrajectory r from T with the minimum distance to s. This comparison is given in Definition 6.

Definition 6 Distance vector between trajectory and subtrajectory. Given a trajectory T and a subtrajectory s of length $w = |s|$, the distance between them is the best alignment of s into T, which is defined by $W_T^s = min_{r \in S_T^w}(dist\_s(s, r))$, where S_T^w is the set of all subtrajectories of length w into T, and $min()$ returns the *distance vector* of the *best alignment* between s and all subtrajectories in S_T^w.

The number of all possible subtrajectories of any length in a trajectory problem with n trajectories of length at most m, and l dimensions, is $O(n \times m^2 \times 2^l)$. Using all subtrajectories as input to a classifier is impracticable because of the high dimensionality. So the selection of only the most *relevant subtrajectories*, i.e., the *movelets*, is necessary. We define a *movelet candidate* in Definition 7.

Definition 7 Movelet Candidate. A *movelet candidate* $\mathcal{M}$ from a subtrajectory s is represented by a tuple $\mathcal{M} = (T, start, length, C, \mathbb{W}, score, sp)$, where T is the trajectory that originates the candidate; *start* is the position in T where the subtrajectory s begins; *length* is the subtrajectory length; C contains the candidate dimensions; $\mathbb{W}$ is a set of pairs $(W_{T_i}^s, class_{T_i})$, where $W_{T_i}^s$ is the distance vector of the best alignment of s into a trajectory T_i and $class_{T_i}$ is the class label of T_i; *score* is a relevance score; and *sp* is a set of distance values (called split points), one per dimension, used to measure the candidate relevance for the classification task.

Evaluating the *relevance* of each candidate is fundamental to discover *movelets*. In classification problems this relevance is given by the capability to differentiate trajectories of one class (*target class*) from trajectories of other classes. In other words, it is expected that a relevant subtrajectory appears in trajectories of the *target class* and does not appear in trajectories of other classes. This defines a *movelet candidate* as *discriminant*. Based on the concept of relevance *score*, that will be detailed in Section 3.4, we define a *movelet* as given in Definition 8.

Definition 8 Movelet. Given a trajectory T and a movelet candidate $\mathcal{M}^s$ from $s \in T$, the candidate $\mathcal{M}^s$ is a *movelet* if for each candidate $\mathcal{M}^r$ from $r \in T$ that overlaps s in at least one element, $\mathcal{M}^s.score > \mathcal{M}^r.score$.

In other words, a candidate is considered as a *movelet* if there is no other candidate overlapping it with more relevance.

3.2 MasterMovelets: Multiple Aspect Trajectory Movelets Discovering Algorithm

In this section we propose an algorithm for discovering heterogeneous *movelets* called MasterMovelets (Multiple ASpect TrajEctoRy Movelets). Our proposal is an extension of Movelets, proposed by Ferrero et al. (2018), to explore the multiple aspects for *movelet* discovering. MasterMovelets consists of exploring all the *movelet candidates* from a trajectory dataset and selecting only the *movelets*. It explores each subtrajectory by finding the *best alignment* of the subtrajectory to all trajectories in the dataset (process detailed in Section 3.3) and then computes the relevance *score* (detailed in Section 3.4). After that, the selected *movelets* are used as input to train a trajectory classifier. The trajectory dataset used to discover the selected *movelets* and to train the classifier is called trajectory *training set*, and the dataset used to evaluate the classifier is called trajectory *test set*.

Algorithm 1 details the method MasterMovelets, that has as the unique input the trajectory training set $\mathbf{T}$, without any parameter. The output is the set of *movelets*. It starts by exploring each trajectory T in the training set $\mathbf{T}$ (lines 2 to 38). The function $ComputeElementDistanceVectors()$ computes the distance between all trajectory elements in T and all trajectories in $\mathbf{T}$, and stores them into a 4-dimensional array, A_1 (line 4). Each value $A_1[i, j, d, k]$ is the distance between the element of T at position j and the element of $T_i \in \mathbf{T}$ at position k, considering dimension d.

The next step consists of exploring all subtrajectory lengths, one by one (lines 5 to 34). For a subtrajectory length w, the function $ComputeSubtrajectoryDistanceVectors()$ computes the distance between the subtrajectories in T and in each $T_i \in \mathbf{T}$ by just adding the values of A_{w-1} and A_1, and stores them into A_w (line 7).

In the loop of lines 9-33, for each subtrajectory in T of size w, the algorithm uses A_w to discover its best dimension combination, and adds it into the movelet *candidates* set . In this loop, the algorithm first computes for each

Algorithm 1: MasterMovelets

```
   Input  : T // trajectory training set
   Output: movelets // set of relevant subtrajectories
 1 movelets ← ∅ ;
 2 for each trajectory T in T do
 3 |   candidates ← ∅;
 4 |   A_1 ← ComputeElementDistanceVectors (T, T) ;
 5 |   for subtrajectory length w from 1 to T.length do
 6 |   |   if w > 1 then
 7 |   |   |   A_w ← ComputeSubtrajectoryDistanceVectors (T, T, A_{w−1}, A_1, w) ;
 8 |   |   end
 9 |   |   for position j from 1 to (T.length − w + 1) do
10 |   |   |   R ← ∅ ;
11 |   |   |   for trajectory i from 1 to |T| do
12 |   |   |   |   for dimension d from 1 to |D| do
13 |   |   |   |   |   R[i, d, ..] ← Rank(A_w[i, j, d, ..]) ;
14 |   |   |   |   end
15 |   |   |   end
16 |   |   |   bestScore ← 0 ;
17 |   |   |   for each dimension combination C in C*_d do
18 |   |   |   |   W ← ∅;
19 |   |   |   |   for trajectory i from 1 to |T| do
20 |   |   |   |   |   W_i ← min MasterAlignment(R[i, C, ..], A_w[i, j, C, ..]) ;
21 |   |   |   |   |   W ← W ∪ (W_i, T[i].class) ;
22 |   |   |   |   end
23 |   |   |   |   relevance ← assess MasterRelevance(W, T.class) ;
24 |   |   |   |   if relevance.score > bestScore then
25 |   |   |   |   |   bestScore ← relevance.score ;
26 |   |   |   |   |   bestSp ← relevance.sp ;
27 |   |   |   |   |   bestW ← W ;
28 |   |   |   |   |   bestC ← C ;
29 |   |   |   |   end
30 |   |   |   end
31 |   |   |   M ← MoveletCandidate(T, j, w, bestC, bestW, bestScore, bestSp) ;
32 |   |   |   candidates ← candidates ∪ M ;
33 |   |   end
34 |   end
35 |   SortByRelevance (candidates) ;
36 |   RemoveSelfSimilar (candidates) ;
37 |   movelets ← movelets ∪ candidates ;
38 end
39 return movelets
```

subtrajectory in T the distance ranking R among all subtrajectories in the ith trajectory, at dimension k (lines 10 to 15). Once R is computed, the algorithm explores each dimension combination C (lines 17 to 30). In this loop, it finds the distance vector of the *best alignment* between each subtrajectory in T to each trajectory T_i, using a specific method for multidimensional alignment, called MasterAlignment (detailed in Section 3.3), and stores the distance vector into $\mathbb{W}$ (lines 18 to 22). After computing $\mathbb{W}$, the algorithm measures the relevance of each subtrajectory based on $\mathbb{W}$ by using a specific function called MasterRelevance (detailed in Section 3.4). Finding the dimension

combination with the highest relevance score the algorithm also preserves the split points, the distance vectors, and the dimension combination (lines 23 to 29). Then, it defines the subtrajectory candidate as the subtrajectory with the most relevant dimension combination and stores it into the set *candidates* (lines 31 and 32).

Following the external loop, it sorts the trajectory candidates by their relevance and removes those *self similar* (lines 35 to 36). Two candidates are *self similar* if they are overlapping on at least one trajectory element and the algorithm preserves the highest relevance candidate. Finally, it adds the remaining candidates to the *movelets* set.

Two key points to perform *movelets* discovery in multiple aspect trajectories are: finding the best alignment of the subtrajectory into a trajectory, performed by the method MasterAlignment, and measuring the subtrajectory relevance, performed by the method MasterRelevance. These key points substantially change the way to discover *movelets* and are detailed in the next sections.

3.3 Multidimensional Alignment of a Subtrajectory into a Trajectory

The problem of movelet alignment is defined as follows: given a subtrajectory and a trajectory, the best alignment of the subtrajectory in relation to the trajectory consists of finding the most similar (closest) part of the trajectory to the subtrajectory. The function $min()$ in Definition 6 performs the best alignment and returns the distance. In the case of one-dimensional alignment ($|D| = 1$) the function returns only the minimum distance value, but in the case of $|D| > 1$ all distance values of the dimensions D must be considered, in the form of a distance vector. A naive solution consists of transforming each *distance vector* in only a distance value by applying a function, but this solution brings two major drawbacks. The first is the designing of a transformation function to encapsulate the distances, which is domain dependent, and the second is the loss of distance information in each dimension. To exemplify this scenario, Figure 3a shows an example of a subtrajectory s and Figure 3b a trajectory T. The subtrajectory s we want to align in T is: "*Users that visit a Café of price $$$ around 12:30am and after go to work around 13:30am*".

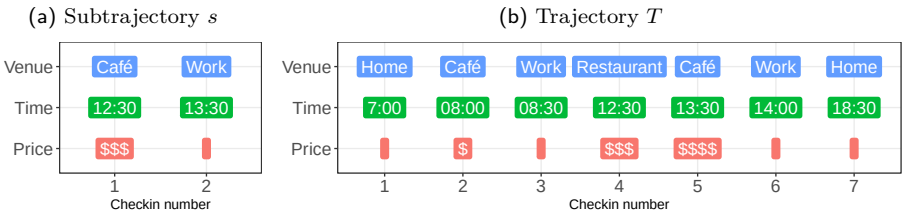

Fig. 3: Example of a subtrajectory s and a trajectory T.

Note in Figure 3 that the user of T does not perform the exact sequence of s, considering all dimensions. But, he/she goes to the venue Café and then to Work twice, at starting position 2 (check-in number 2) and 5 (check-in number 5). In addition, at starting position 4 (check-in number 4) the user of T performs check-ins at 12:30am and 13:30am. Considering this situation, which of these starting positions (2, 4, and 5) represent the *best alignment*? We *claim* that the best alignment is represented by the sequence of check-ins starting at position 5, as highlighted in Figure 4, because besides the venues sequence being the same $\langle \text{Café}, Work \rangle$, the dimensions *Time* and *Price* are also quite similar.

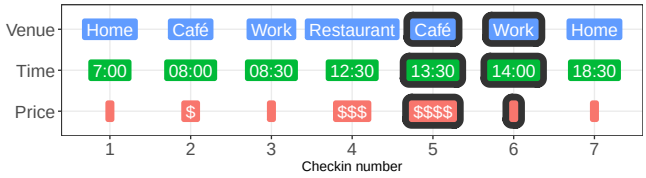

Fig. 4: Best subtrajectory alignment highlighted in the trajectory.

In this work we introduce the method MasterAlignment, that consists in ranking the distances of each dimension individually and getting the position of the minimum average rank to determine the position of the best alignment. Let us consider $\mathbf{V}_1, \ldots, \mathbf{V}_6$ as the distance vectors of all the alignments, where $\mathbf{V}_i$ corresponds to the alignment between s and the subtrajectory in T starting at the ith-position. Table 1a presents the values of the distance vectors. The column at starting position 1, shows the distance vector $\mathbf{V}_1$ between the sequence $s = \langle (\text{Café}, 12{:}30, \$\$\$), (\text{Work}, 13{:}30,) \rangle$ and the sequence of T at starting position 1, represented by $r_1 = \langle (\text{Home}, 07{:}00, \emptyset), (\text{Café}, 08{:}00, \$)$. For $Venue$ dimension the distance is 2, because $\langle \text{Café}, \text{Work} \rangle$ differs from the sequence $\langle \text{Home}, \text{Café} \rangle$ in both check-ins. On $Time$ dimension the distance is 660, because the sum of the time differences in minutes is: $|(12{:}30 - 07{:}00)| = 330$ *minutes* and $|(13{:}30 - 08{:}00)| = 330$ *minutes*, totaling 660. And, on *Price* dimension the distance is 4 price units, because the sum of the difference between price values is: $|(\$\$\$ - \emptyset)| = 3$ *price units* and $|(\emptyset - \$)| = 1$ *price unit*, totaling 4 *price units*. The method performs the same distance calculation for the next starting positions in an sliding way.

After that, we rank the distance values for each dimension. Table 1b shows the ranking values. For instance, for $Time$ dimension the distance values are $(660, 570, 300, 0, 90, 390)$ and the ranking values are $(6, 5, 3, 1, 2, 4)$. This ranking indicates the distance 660 has the worst ranking value, 6, and the distance 0 has the best ranking value, 1. Note on the $Venue$ dimension that the method also supports fractional ranks in case of tie, such as 1.5 at starting positions 2 and 5. Then, the method computes the average rank at each starting position (last row in Table 1b), resulting in $(5.0, 2.8, 4.5, 3.3, \underline{1.5}, 3.8)$. So, Master-

Table 1: Finding the best alignments from the distance vectors.

(a) Distance values.

	Starting position					
Distance	*1*	*2*	*3*	*4*	*5*	*6*
Venue	2	0	2	2	0	2
Time	660	570	300	0	90	390
Price	4	2	6	4	1	3
Vector	$\mathbf{V}_1$	$\mathbf{V}_2$	$\mathbf{V}_3$	$\mathbf{V}_4$	$\mathbf{V}_5$	$\mathbf{V}_6$

(b) Distance rankings.

	Starting position					
Ranking	*1*	*2*	*3*	*4*	*5*	*6*
Venue	4.5	1.5	4.5	4.5	1.5	4.5
Time	6	5	3	1	2	4
Price	4.5	2	6	4.5	1	3
Avg. rank	5.0	2.8	4.5	3.3	<u>1.5</u>	3.8

Alignment considers the best alignment as the lowest average rank, that is 1.5 (underlined in Table 1b) and corresponds to the 5th starting position. Finally, the method returns the *distance vector* $\mathbf{v}_5 = (0, 90, 1)$, that represents the distances (of the best alignment) between the subtrajectory s and the trajectory T, which is denoted by W_T^s.

The function MasterAlignment is detailed in Algorithm 2, that has as input two 2-dimensional arrays, V and R, containing the distance values and the distance rankings, respectively.

Algorithm 2: MasterAlignment

```
   Input  : V, // distance values of subtrajectory alignments
            R // distance rankings of subtrajectory alignments
   Output: W // distance values of the best subtrajectory alignment
 1 Y ← ∅ ;
 2 l ← |R[.., 1]| ;
 3 posMinAvgRank = 1 ;
 4 for position j from 1 to |R[1, ..]| do
 5 |   sumRank =← ∅ ;
 6 |   for dimension d from 1 to l do
 7 |   |   sumRank = sumRank + R[d, j] ;
 8 |   end
 9 |   avgRank = sumRank / l ;
10 |   Y[j] = avgRank ;
11 |   if Y[j] < Y[posMinAvgRank] then
12 |   |   posMinAvgRank ← j ;
13 |   end
14 end
15 W ← V[.., posMinAvgRank] ;
16 return W
```

Algorithm 2 starts initializing the set to store the average rank values Y, the number of dimensions l, and the initial position of the minimum average rank, *posMinAvgRank* (lines 1 to 3). Then, it computes the average rank along dimensions (lines 4 to 14). In this loop it sums the rank values along dimensions and then performs the average rank, storing the result in the set Y (lines 5 to 10). After computing the average rank, it compares the current average rank and the minimum average rank found. If the current average rank

is lower, it updates the position of the minimum average rank (lines 11 to 13). Finally, the algorithm gets the distance vector of the best multidimensional alignment based on that position and stores the vector in W (line 15). This is the distance vector between the subtrajectory s and trajectory T.

Since we can measure the distance of a subtrajectory to any trajectory considering *multiple* and *heterogeneous dimensions*, we can also measure the relevance of the subtrajectory. This process is detailed in the following section.

3.4 Relevance Measuring for Multidimensional Subtrajectory Candidates

The relevance of a subtrajectory is related to the number of trajectories of the same class that perform similar movement. We analyze the distances of the best alignment between a subtrajectory and all trajectories in the dataset, in order to define which trajectories of the same class perform similar movement. The most common approach consists of representing the distances of the best alignments in an *orderline* and finding a *split point* to separate the distances into two groups: the nearest (left side) and the farthest (right side), where the nearest perform similar movement and the farthest not. Several techniques have been proposed to find the *split point*, such as the maximum information gain (Ye and Keogh, 2011), the Kruskal-Wallis and Mood's Median (Lines and Bagnall, 2012), and the Left Side Pure (LSP) (Ferrero et al., 2018). However, these techniques are limited to finding the *split point* in one dimensional orderline. Let us consider an example with $T_1, T_2, \ldots, T_8$ of classes L_1 and L_2, represented by $Time$ and $Venue$ dimension, where T_1, T_2, T_3, and T_4 are of class L_1 and T_5, T_6, T_7, and T_8 are of class L_2, and a movelet candidate $\mathcal{M}$ extracted from T_1 of class L_1. The set $\mathbb{W}$ of $\mathcal{M}$ (Definition 7) contains the pairs $\{(W_{T_1}, class_{T_1}), (W_{T_2}, class_{T_2}), \ldots, (W_{T_8}, class_{T_8})\}$, where W_{T_i} and $class_{T_i}$ are the distance vector of the *best alignment* to trajectory T_i and the *class* of T_i, respectively. Figure 5 shows the orderlines for $\mathbb{W}$, where each point indicates the distance value to the ith trajectory for each dimension and the symbols X and O are the classes. Note that trajectory T_1 is the first distance value on each dimension because the movelet candidate $\mathcal{M}$ comes from T_1.

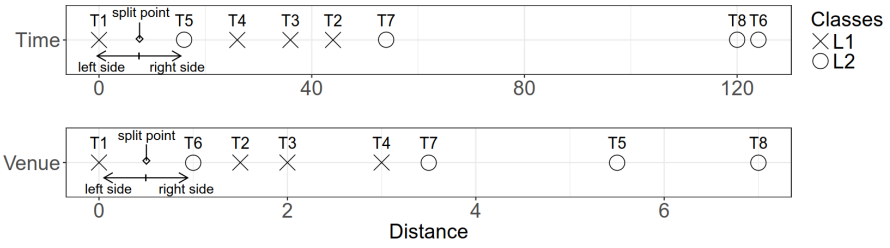

Fig. 5: Orderlines for dimensions Time and Venue.

Figure 5 indicates the split points obtained by applying the technique LSP (Ferrero et al., 2018), that requires the left side of the orderline has only distance values of the *target* class, L_1. But note this requirement only separates T_1 on the *left side* in both orderlines, which means that the movelet candidate has very low relevance, because only one trajectory of the same class performs similar movement. Instead of defining the split point for each dimension independently, we propose a method to analyze all dimensions together. Note in Figure 5 that the distance values of T_7 may be a good estimator of the split points, because all the distance values of class L_1 have values lower than T_7 in both dimensions. Our method deals with the problem of distances on multiple dimensions and finds the *split points* that maximize the relevance of the movelet candidate. The method, called MasterRelevance, consists of three steps and it is exemplified by Figure 6.

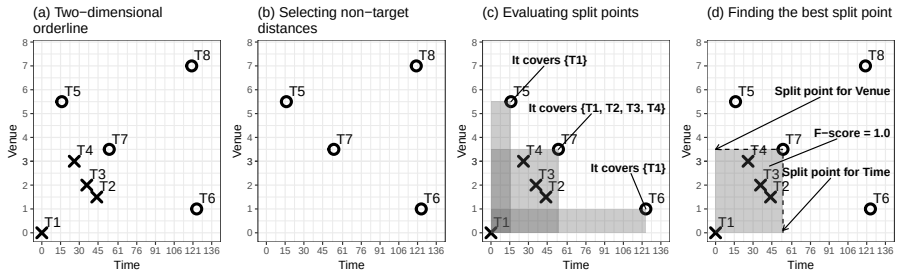

Fig. 6: Example of finding split points in a *multidimensional orderline*.

Figure 6(a) shows the distance values presented in Figure 5, in a scatter plot, where each point indicates the distances in both dimensions, $Time$ and $Venue$, on the abscissa and ordinate, respectively. The *first step* consists of selecting only the points of the *non-target* class L_2 and then pruning the points with greater distance values than some other in both dimensions $Time$ and $Venue$, called *covered* points. In this work, we consider the obvious solution of iteratively pruning *covered* points until only *non-covered* points remain[2]. In Figure 6(b) the point of T_8 is pruned because it has greater distance values than T_7 in both dimensions. In the *second step*, the method evaluates the unpruned points according the *number of points* of each class that are covered, considering both dimensions. Figure 6(c) shows that by using the point values of T_5 or T_6 as the *split points* they only cover T_1 of class L_1, but by using the point values of T_7 they cover T_1, T_2, T_3, and T_4. In the *final step* it chooses the *split points* that have the highest relevance *score*. To calculate the score we use F-measure that is the harmonic average of the *precision* and the *recall*. In this context, the *precision* is the proportion of *points* covered by the split

[2] There are many other strategies in the literature to find exact and approximate solutions for this specific problem. More details can be found in Kung et al. (1975); Veldhuizen and Lamont (2000); Marler and Arora (2004).

points that belongs to the *target class*, and the *recall* is the proportion of *points* covered by the split points that belong to the *target class* in relation to all the *points* of the *target class*. As shown in Figure 6(d), the best split points are the values defined by the point T_7 and the score is 1.

Algorithm 3: MasterRelevance

```
   Input  : 𝕎, // set of distance vectors of the best alignments
            targetClass // the target class, i.e. the movelet candidate class
   Output: relevance // subtrajectory relevance
 1 // step 1: pruning points with greater distance values
     𝕎train ← selectHalfForTraining(𝕎) ;
 2 nonTargetWtrain ← ∅ ;
 3 for pair (W_i, class_i) of 𝕎train do
 4 |   if not class_i = targetClass then
 5 |   |   nonTargetWtrain ← nonTargetWtrain ∪ {W_i} ;
 6 |   end
 7 end
 8 for W_i in nonTargetWtrain do
 9 |   for W_j in nonTargetWtrain do
10 |   |   if W_j isCoveredBy W_i then
11 |   |   |   nonTarget𝕎 ← nonTarget𝕎 − {W_j} ;
12 |   |   end
13 |   end
14 end
15 // step 2: Evaluating split points
16 ℙcandidates ← nonTargetWtrain ;
17 statistics ← ∅
18 for W_i in ℙcandidates do
19 |   for pair (W_k, class_k) in 𝕎 do
20 |   |   if W_k isCoveredBy W_i then
21 |   |   |   if class_k = targetClass then
22 |   |   |   |   statistics[i].TruePositive  += 1 ;
23 |   |   |   else
24 |   |   |   |   statistics[i].FalsePositive += 1 ;
25 |   |   else
26 |   |   |   if class_k = targetClass then
27 |   |   |   |   statistics[i].FalseNegative += 1 ;
28 |   |   |   else
29 |   |   |   |   statistics[i].TrueNegative  += 1 ;
30 |   end
31 |   statistics[i].Fscore ← calculateFscore(statistics[i]) ;
32 end
33 // step 3: choosing the best split point
34 indexBestFscore ← argmax_i(statistics) ;
35 relevance.score ← statistics[indexBestFscore].Fscore ;
36 relevance.ℙ ← ℙcandidates[indexBestFscore] ;
37 return relevance
```

Algorithm 3 details the method MasterRelevance. The algorithm has as input the target class (the class of the movelet candidate) and the set of pairs $\mathbb{W}$ containing the distance vector of the best alignments and the class of each trajectory. Algorithm 3 starts by randomly separating half of the pairs in

$\mathbb{W}$, in a stratified way, for the split point finding (line 1), to avoid overfitting. The algorithm stores in $nonTargetWtrain$ all the split points in $\mathbb{W}train$ that do not correspond to the target (lines 3 to 7). Then, the algorithm removes from $nonTargetWtrain$ all the split point sets W_j covered by another split point set W_i (lines 8 to 14). After that, the remaining split point sets are stored in $\mathbb{P}candidates$, which contains the candidates to become the movelet split points (line 16). For each candidate W_i in $\mathbb{P}candidates$ the algorithm computes the statistics of classifying each distance vector W_k in $\mathbb{W}$ as true positive, false positive, true negative or false negative. (lines 18 to 32). After computing the statistics, the algorithm computes the F-score of each split point candidate (line 31). To finalize, it finds the split point set that achieves the maximum F-score (line 34) and stores both the split point set and the F-score value, in the variable *relevance*(lines 35 and 36).

3.5 Complexity Analysis

In terms of memory space, MasterMovelets (Algorithm 1), keeps storing the arrays A_0, A_{w-1}, and A_w simultaneously, using $O(n \times m^2 \times l)$, where n is the number of trajectories, m is the length of the longest trajectory, and l is the number of dimensions of the dataset. Also, it stores at most $O(m \times log\ m)$ candidates for each trajectory. Therefore, the space complexity is $O(n \times m^2 \times l)$.

In terms of time, MasterMovelets repeats the function $Rank$ by $O(n^2 \times m^2 \times l)$ times and MasterRelevance by $O(n \times m^2 \times 2^l)$ times. The former costs $O(m\ log\ m)$ and the latter $O(n^2 \times l)$. So, the overall complexity is $O(n^3 \times m^3\ log\ m \times 2^l)$. The number of movelets extracted by MasterMovelets is $O(n \times m)$. So, the cost to build the classification model depends on the complexity of the algorithm chosen to build the classifier, considering as input n trajectories as samples and $O(n \times m)$ *movelets* as attributes.

4 Experimental Evaluation

We begin by noting that the MasterMovelets source code, the datasets and the results of the experiments are available at Ferrero (2019).

We evaluate MasterMovelets with three real trajectory datasets, the Gowalla (Cho et al., 2011) and Brightkite (Cho et al., 2011), used by Gao et al. (2017) to evaluate the method BiTULER, and a third dataset of Foursquare (Yang et al., 2015) checkins, with more data dimensions. We compare MasterMovelets to the state-of-the-art method BiTULER (Gao et al., 2017) because it was developed for social media data, and to nearest neighbor classifiers using the following distance and similarity measures: LCSS (Vlachos et al., 2002), EDR (Chen et al., 2005), MD-DTW (ten Holt et al., 2007a), and MSM (Furtado et al., 2015). We do not compare MasterMovelets with the methods developed for raw trajectory classification (Lee et al., 2008; Dodge et al., 2009; Zheng et al., 2010; Patel et al., 2012; Xiao et al., 2017) because

these works do not support semantic dimensions, and because they were outperformed by MOVELETS for raw trajectories in Ferrero et al. (2018) over four classical datasets (animals, hurricanes, trucks, and Geolife).

4.1 Evaluation with the Gowalla dataset

The first experiment uses the dataset from Gowalla (Cho et al., 2011), that is a location-based social network, where users share their locations by checking-in. Each check-in contains the anonymized user id, the timestamp, the spatial location (latitude and longitude), and the check-in venue (place). This dataset was used in Gao et al. (2017) to classify users based on their check-in identifiers. From the original dataset containing more than 6 million check-ins, collected between 2009 and 2010, we selected places with at least 10 check-ins. We segmented trajectories in weekly trajectories with at least 10 check-ins and users with at least 10 trajectories, resulting in 33,816 weekly trajectories of 1,952 users. We randomly selected 300 users for experimental evaluation obtaining 5,329 trajectories. The class labels are the 300 user identifiers. Table 2 shows details about each dimension, such as data type, value range, and the distance measure used to compare two dimension values.

Table 2: Gowalla trajectory dimension description.

Dimension	**Type**	**Range or examples**	**Distance measure**
Space	Composite (*lat lon*)	40.82651 -73.95039	Euclidean Distance
Time	Temporal (HH:MM)	[00:00,23:59]	Difference in minutes
Weekday	Ordinal	{Mon, Tue, . . . , Sun}	Weekday Distance[3] (0 or 1)
Place identifier	Nominal	Any nominal value	Binary Distance (0 or 1)

We evaluate the methods performing a 5-fold cross-validation. For evaluating MASTERMOVELETS we limit the maximum size of the movelets to the size of the smallest trajectory of the dataset, and we build classification models using Neural Networks (NN) and Random Forests Decision Trees (RF). The *former* is a Single-hidden layer Neural Network with 100 units and to train it we used the same parameters used in Gao et al. (2017), a dropout rate of 0.5, and an Adam optimizer with the following values of *learning rate* 10^{-4} (*number of epochs*): 9.5(80), 7.5(50), 5.5(50), 2.5(30), and 1.5(20). The *latter* consists of an ensemble of 300 decision trees.

For BiTULER (Gao et al., 2017) we build a Bidirectional Neural Network from word embeddings extracted from the entire dataset. BiTULER is limited to consider only one dimension, the place identifier. For the distance and similarity measures LCSS, EDR, and MSM we define three threshold values for

[3] The weekday distance between two values returns 0 if both are weekdays or weekends, and 1, otherwise.

each dimension with non-binary distance: 30, 60, and 120 minutes for the *Time* dimension, and 100, 300, and 500 meters for the *spatial* dimension. We keep all dimensions with the same weights. For the distance measure MD-DTW we normalize the non-binary distances, using the threshold values mentioned above, dividing the distance value by the threshold value. For distance and similarity measures we performed 5×2-fold cross validation on the training set to find the best parameter configuration and use this configuration to calculate the classification accuracy on the test set.

For MOVELETS and the similarity/distance measures we used all available dimensions. As the objective of this paper is to compare our work to approaches that deal with multiple dimensions we did not manually explore individual dimensions or combinations of only some dimensions in the experiments, although we know that in many cases better results are obtained with less dimensions.

Table 3 shows the classification accuracy (*acc*) and the accuracy on the top 5 most probable classes (*acc top5*) on the test set. The best result is presented in bold and the second best result is underlined.

Table 3: Evaluation results on Gowalla dataset.

							MASTER MOVELETS	
Measure	MD-DTW	LCSS	EDR	MSM	BiTULER	MOVELETS	NN	RF
acc	75.8	90.0	87.2	92.1	63.0	52.2	**95.2**	<u>93.3</u>
acc top5	88.8	95.7	93.4	96.2	74.1	77.3	**98.2**	<u>97.9</u>

The results show that MASTERMOVELETS (NN) achieves the best accuracy, 95.2% (4.8% of error). MASTERMOVELETS (RF) achieves the second best accuracy, 93.3% (6.7% of error). Among the state of the art methods, the similarity measure MSM achieves the best results, 92.1% of accuracy (7.9% of error). MASTERMOVELETS (NN) and (RF) reduce the classification error in comparison to MSM in 39.2% $(1 - 4.8/7.9)$ and 15.2% $(1 - 6.7/7.9)$, respectively.

As expected, because of the number of dimensions, the worst results were achieved by MOVELETS, that encapsulates the distances of all dimensions in a single value; and BiTULER, because it supports only a single dimension, what is not the best solution for classifying semantically rich trajectories.

4.2 Evaluation with the Brightkite Dataset

The second experiment uses the dataset from Brightkite (Cho et al., 2011). This dataset was also used in Gao et al. (2017) to classify users based on their check-in identifiers. Each check-in contains the anonymized user id, the timestamp, the spatial location (latitude and longitude), and the check-in venue, without any other information about checking-in. From the original dataset

containing more than 4.5 million check-ins, collected between 2008 and 2010, we selected places with at least 10 check-ins. We segmented trajectories in weekly trajectories with at least 10 check-ins and users with at least 10 trajectories, resulting in 54,247 weekly trajectories of 2,042 users. We randomly selected 300 users for experimental evaluation obtaining 7,911 trajectories. The class labels are the 300 user identifiers. The trajectory dimension description is the same as the Gowalla dataset (shown in Table 2).

The experimental setup is the same of the previous dataset.

Table 4: Evaluation results on Brightkite dataset.

							MasterMovelets	
Measure	MD-DTW	LCSS	EDR	MSM	BiTULER	Movelets	NN	RF
acc	91.2	94.2	94.0	95.2	90.8	64.5	**96.6**	96.3
acc top5	97.1	97.7	97.3	98.2	95.4	89.3	**99.1**	99.1

Table 4 shows the results, where MasterMovelets (NN) achieves the best accuracy, 96.6% (3.4% of error). MasterMovelets (RF) achieves the second best accuracy, 96.3% (3.7% of error). Among the state of the art methods, the similarity measure MSM achieves the best results, 95.2% of accuracy (4.8% of error). The results indicate that in comparison to MSM, MasterMovelets reduces the classification error in 29.2% and 22.9% with NN and RF models, respectively. The worst results were also achieved by Movelets and BiTULER.

4.3 Evaluation with the Foursquare Dataset

For the experiment with the Foursquare (Yang et al., 2015) dataset we considered check-ins (mostly in New York city) between 2012 and 2013. The original dataset has 227,428 check-ins of 1,083 distinct users. Each check-in is composed of the anonymized user id, the timestamp of the check-in, and the corresponding Foursquare venue id. We extract the weekdays from time and enriched the check-ins with venue information collected from the Foursquare API[4] and with historical weather data (the weather condition) collected via the Weather Wunderground API[5], in order to explore the relation between user mobility and weather information. In this experiment we removed the spatial dimension, and used only the most general venue category instead of the venue identifier, to make the problem more difficult. Table 5 presents the six dimensions used in this dataset, the description of each dimension, and the respective distance function.

[4] https://developer.foursquare.com/

[5] https://www.wunderground.com/weather/api/

We preprocessed the dataset by applying the following steps: we removed check-ins belonging to broad categories such as roads, rivers, cities, neighborhoods, etc, and duplicated check-ins (considering a 10-minutes threshold); we segmented the trajectories into weekly trajectories and selected those with at least 10 check-ins and users with at least 10 trajectories, resulting in 3,079 weekly trajectories of 193 users. The class label is the user identifier.

Table 5: Foursquare trajectory dimension description.

Dimension	**Type**	**Range or examples**	**Distance measure**
Time	Temporal (HH:MM)	[00:00,23:59]	Difference in minutes
Weekday	Ordinal	{Mon, Tue,... , Sun}	Weekday Distance (0 or 1)
Venue Category	Nominal	Foursquare categories[6]	Binary Distance (0 or 1)
Price	Numeric	$\{1, 2, 3, 4\}$	Manhattan Distance
Rating	Numeric	$[0.0, 9.9]$	Manhattan Distance
Weather condition	Nominal	{Clear, Cloudy, Fog, Haze, Rainy, Snow}	Binary Distance (0 or 1)

The experimental configuration was the same used with the previous datasets. For the similarity measures we use the thresholds: 30, 60, and 120 minutes for *Time*; 0, 1, and 2 *price units* for *Price*; and 0.5, 1.0, and 1.5 *rating values* for the *Rating*.

Table 6 shows the classification accuracy (*acc*) and the accuracy on the top 5 most probable classes (*acc top5*) on the test set. The best result is highlighted in bold and the second best result is underlined.

Table 6: Evaluation results on Foursquare dataset.

							Master Movelets	
Measure	MD-DTW	LCSS	EDR	MSM	BiTULER	Movelets	NN	RF
acc	20.9	29.3	32.0	47.8	30.9	29.0	**80.7**	<u>72.3</u>
acc top5	40.1	54.2	56.3	71.4	58.1	49.5	**92.5**	<u>89.1</u>

As in the previous experiments, MasterMovelets with both NN and RF achieves the best results, with 80.7% of accuracy (19.3% of error) and 72.3% (27.7% of error), respectively. We notice that in all three experiments, the best results were achieved with MasterMovelets NN. Indeed, as in the previous datasets, apart from MasterMovelets, the second best method was MSM, that achieved 47.8% of accuracy (52.2% of error). The classification

[6] Foursquare categories are Shop & Service, Professional & Other Places, Food, Travel & Transport, Outdoors & Recreation, Arts & Entertainment, Residence, Nightlife Spot, Event, College & University.

error improvement of MasterMovelets in relation to MSM is 63.0% (NN) and 46.9% (RF).

The worst results on this dataset were achieved with MD-DTW, Movelets and LCSS. MD-DTW and Movelets have the same problem: they depend on a transformation function to encapsulate the distances of all multiple and heterogeneous dimensions in a single distance value. BiTULER is limited to the place identifier (place category in this experiment), what shows that this dimension is not sufficient to characterize the class label. LCSS and EDR do not present good classification accuracy because the number of matchings decrease as the number of dimensions increase. MSM presents better results than LCSS and EDR because it allows partial matching among dimensions.

In relation to the Neural Network and Random Forest models, the NN models capture the relation between the *movelets* and the classes better than RF models. In general, Neural Network models deal better with high dimensional spaces than symbolic models as decision trees, in detriment of interpretability.

4.4 General Analysis over all Datasets

In this section we analyze the capability of the models to best discriminate classes. For this comparison we considered only MasterMovelets NN, that was better than MasterMovelets RF. Figure 7 shows a bar plot for each dataset, indicating for how many classes each classifier presents the best F-measure score[7].

In Figure 7a, for the Gowalla dataset, the best model is MasterMovelets, that achieves the best F-measure score in 206 (of 300) classes, followed by MSM, that achieves 133 classes.

Figure 7b shows the bar plot for Brightkite. In this dataset, MasterMovelets, LCSS, MSM and EDR were very similar achieving the best F-measure score in between 188 and 178 (of 300) classes.

On the other hand, the bar plot for the Foursquare dataset in Figure 7c shows that MasterMovelets is significantly better than state of the art methods. MasterMovelets achieves the best F-measure in 186 (of 193) classes, and the other methods in less than 5 classes.

4.5 Movelet Interpretation and Dimension Analysis

One of the most interesting aspect related to MasterMovelets, but which has not been explored in this paper, is its capability to provide subtrajectories of different lengths and with different dimension combinations that can be used to understand what distinguishes one moving object behavior from another one. This information represents knowledge about the trajectories of the same

[7] A classifier presents the best F-measure performance for a class if there is no other classifier with better F-measure score and there are at least a classifier with lower score. In addition, the sum of the bars in bar plot exceed the number of classes, because of ties.

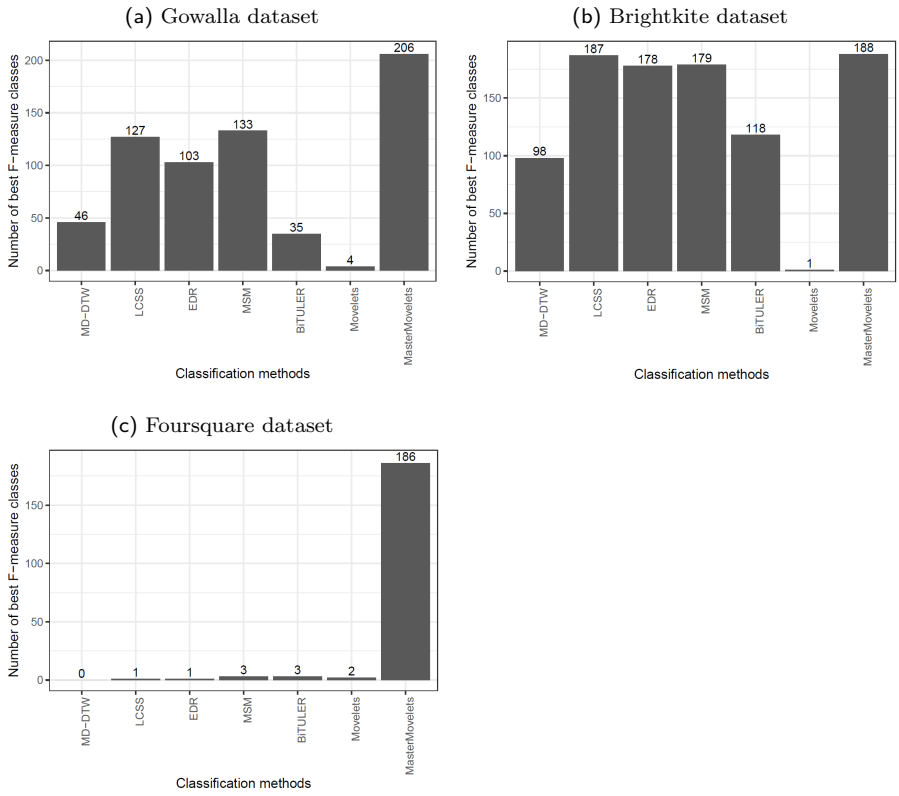

Fig. 7: Bar plots indicating for how many classes each classifier presents the best F-measure.

class (same moving object), i.e., the behaviour pattern of a single individual, and it can be used for other types of analysis as, for instance, trajectory anonymization or pattern interpretation.

The capability of MasterMovelets to automatically capture the best dimension combination and element sequence length makes it very robust for trajectory classification problems of a variety of datasets with a high number of dimensions, what is a tendency nowadays.

In order to understand a little more about the dimensions that are among relevant movelets, we show two movelets of three different classes in Table 7, extracted from the Foursquare dataset. In this table, -1 means that the dimension has no value for that trajectory point. We may notice that the movelets are heterogenous, i.e., contain different dimensions. For instance, the class label 12 has two movelets of the same length (2), and the first movelet has the dimensions Rating, Time, and Venue Category, while the second movelet has the dimensions Price and Time. The movelets of class 25 have different lengths (1 and 3). The first one has the dimensions Rating, Time and Venue Category, and the second one has the dimensions Rating, Venue Category and Weekday.

Table 7: Examples of movelets extracted from the Foursquare dataset

class label	movelet length	Price	Rating	Time	Weather	Venue Category	Weekday
12	2		-1	06:50		Food	
12	2		5.3	19:09		Shop & Service	
12	2	1		07:05			
12	2	-1		18:43			
25	1		9	07:55		Arts & Entertainment	Wednesday
25	3		9.2	06:30		Outdoors & Recreation	
25	3		-1	08:07		Outdoors & Recreation	
25	3		9.6	08:11		Outdoors & Recreation	
50	1		7.9	21:13	Clear	Shop & Service	Tuesday
50	4	-1		21:46		Travel & Transport	Friday
50	4	-1		21:50		Outdoors & Recreation	Friday
50	4	1		00:04		Food	Saturday
50	4	1		09:44		Nightlife Spot	Saturday

For the class 50, the first movelet has length one and five dimensions (Rating, Time, Weather, Venue Category and Weekday). The second movelet has length four with four dimensions (Price, Time, Venue Category and Weekday). With the few examples shown in the figure we observe that in the Foursquare dataset, that does not have the spatial dimension and the POI instance, the movelets of the same class are characterized by many dimension combinations.

4.6 MasterMovelets Processing Time Evaluation

In this section we evaluate the movelets accuracy as we limit the maximum size of the movelets (Section 4.6.1) and its scalability (Section 4.6.2).

4.6.1 Maximum Movelet size Evaluation

A simple way to reduce the processing time of MasterMovelets is to avoid exploring all possible subtrajectory lengths, limiting the maximum size of the movelets. This strategy is used in the time series domain for the method shapelets. The problem is that finding the appropriate maximum size is sometimes challenging. Therefore, we evaluate the accuracy of MasterMovelets by exploring the *movelet* maximum length. In this experiment we limited the maximum length of the movelets to 3 values: (i) the $log_e(trajectory\_length)$, i.e., the size of the movelets generated from a trajectory is limited to the natural logarithm of the length of that trajectory, (ii) the smallest trajectory length in the dataset, and (iii) the length of the trajectory that is generating the movelet, i.e., without limit. We used the Random Forest classifier in this experiment with a hold-out 70/30.

Table 8 presents the results of the experiment. We observe that the classification accuracy does not have a high variation when the maximum size of the movelets increase. The behaviour changes according to the considered

Table 8: Comparing the accuracy for different maximum size of the movelets

Dataset	Log	Shortest trajectory	Without limit
Gowalla	91.0	92.2	92.4
Brightkite	95.6	96.0	96.1
Foursquare	68.0	69.3	67.3

dataset. By limiting the maximum movelet size to the log_e of the trajectory, or to the smallest trajectory, the accuracy does not change significantly, because in general the movement patterns that distinguish users are characterized by short subtrajectories. In practice this means that human routines are given by sequences of a few visited places. Therefore, limiting the maximal size of the movelet to the log_e of the trajectory size is a good strategy to reduce the processing time of the method without expressive loss of accuracy and without depending on a user defined parameter.

4.6.2 Scalability Analysis

Social media trajectories normally have less points when compared to GPS trajectories, because the number of places daily visited and/or checked-in by users in general is not high. The points are sparse in space. This is important to define the size of the trajectories in the scalability analysis. We evaluate MasterMovelets when increasing the length of the trajectories, i.e., the number of trajectory points, the amount of trajectories, and the number of dimensions. We compare the processing time of the MasterMovelets, Master-Log, that limits the maximal length of the movelets to $log_e(trajectory\_length)$, and the method movelets that is simpler and does not explore the dimensions. Figure 8 presents the results of this experiment. In Figure 8 (a) we generated 200 trajectories of 1 dimension, and varied the length of the trajectories from 10 points to 400 points. In Figure 8 (b) we generated trajectories with 50 points and 1 dimension, and varied the number of trajectories from 100 to 4.000 trajectories. For the experiment in Figure 8 (c) we generated 200 trajectories with 50 points each, and varied the number of dimensions from 1 dimension to 5. The scalability experiments were performed in an Intel Core i7-6700 CPU @ 3.40GHz with 4 cores, and 32GB of memory.

We may observe in the figure that the running time of MasterMovelets grows in all scenarios, but limiting the maximum length of the movelets as in Master-Log, we notice that the running time significantly decreases. The best scenario for Master-Log is in Figure 8(a) where the length of the trajectories increase, i.e., their number of points, and in Figure 8(c) where the number of dimensions increase. The results of Master-Log are less impacting in the experiment in Figure 8 (b) where the number of trajectories increase, because in that scenario the number of points of the trajectories is not so high, and by

consequence the gain of limiting the maximum size of the movelets to log_e of the length of the trajectories are not so expressive.

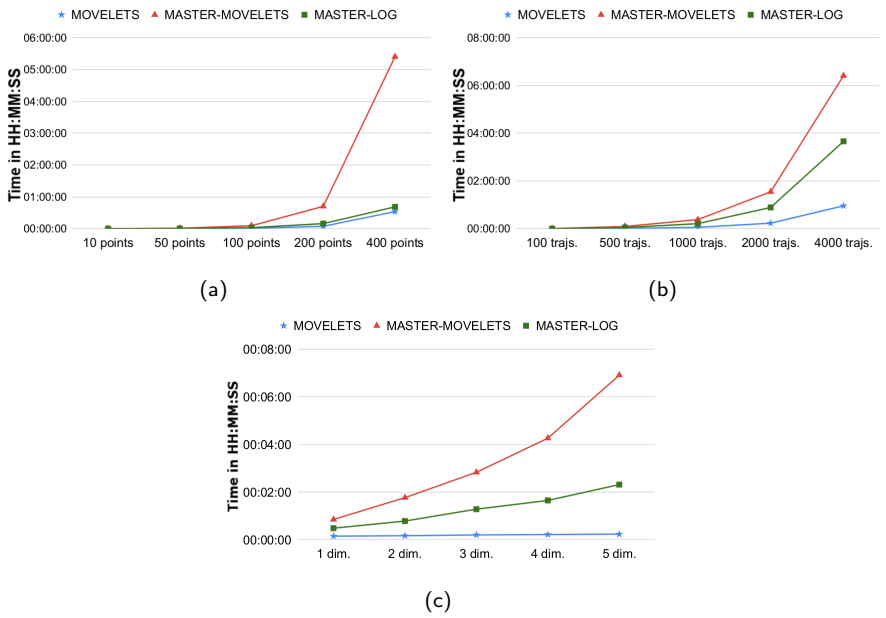

Fig. 8: Computational time spent by MasterMovelets without limits and using the log limitation (Master-Log), and Movelets, over three synthetic dataset configuration, respectively: (a) varying the length of the trajectories, (b) varying the number of trajectories and (c) varying the number of dimensions.

5 Conclusions

In this paper we proposed a new method for extracting relevant subtrajectories for multiple aspect trajectory classification, called MasterMovelets. The method consists in an extension of a previous work to extract relevant subtrajectories from multidimensional trajectories. Our method finds the most relevant subtrajectories leading with the problem of exploring dimension combinations. MasterMovelets is parameter-free and domain independent, which is very important since parameter values are difficult to estimate in many problems and directly affect the data mining results.

We evaluate our method using three public datasets to classify trajectories and compare it with methods in the literature that support multiple data dimensions with different characteristics, including space, time, and semantics. Experiments demonstrate that *MasterMovelets* outperformed existing approaches by reducing the classification error between 15% and 63%. For

the best of our knowledge this is the first work in the literature for classifying trajectories represented by multiple and heterogeneous dimensions, recently introduced as Multiple Aspect Trajectories.

The main drawback of our method is the time complexity, although it is performed only once, before the classification task. Preliminary experiments have shown that the movelets that best characterize the class label are short movelets, i.e., subtrajectories with a few elements. Therefore, one simple way to reduce the processing time is to search for movelets until a limited size (for instance $log_e(length(m))$, where m is the number of trajectory elements, as discussed in Section 4.6.1).

Future works include the proposal of approaches to reduce the time complexity of movelets discovery, and a new algorithm for representing distances between trajectories and subtrajectories by a degree of membership, in order to improve the representation.

Acknowledgements This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and the Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC).

References

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: KDD workshop, AAAI Press, Seattle, WA, USA, vol 10, pp 359–370

Chen L, Özsu MT, Oria V (2005) Robust and fast similarity search for moving object trajectories. In: Proc. of the ACM International conference on Management of data (SIGMOD), ACM, New York, NY, USA, pp 491–502

Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. In: Proc. of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 1082–1090

Dodge S, Weibel R, Forootan E (2009) Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. Computers, Environment and Urban Systems 33(6):419–434

Etemad M, Soares Júnior A, Matwin S (2018) Predicting transportation modes of gps trajectories using feature engineering and noise removal. In: Advances in Artificial Intelligence: 31st Canadian Conference on Artificial Intelligence, Canadian AI 2018, Toronto, ON, Canada, May 8–11, 2018, Proceedings 31, Springer, pp 259–264

Ferrero CA (2019) MasterMovelets Code. `https://github.com/anfer86/dmkd_masterMovelets_results`, [accessed 16-July-2019]

Ferrero CA, Alvares LO, Bogorny V (2016) Multiple aspect trajectory data analysis: Research challenges and opportunities. In: XVII Brazilian Sym-

posium on Geoinformatics, GEOINFO, Campos do Jordão, SP, Brazil, GEOINFO '16, pp 1–12

Ferrero CA, Alvares LO, Zalewsky W, Bogorny V (2018) Movelets: Exploring relevant subtrajectories for robust trajectory classification. In: Proc. of the 33rd ACM SAC, ACM, Pau, France, pp 1–8

Frentzos E, Gratsias K, Pelekis N, Theodoridis Y (2005) Nearest neighbor search on moving object trajectories. In: Proceeedings of the International Symposium on Spatial and Temporal Databases, Springer, pp 328–345

Furtado AS, Kopanaki D, Alvares LO, Bogorny V (2015) Multidimensional similarity measuring for semantic trajectories. Transactions in GIS 20:280–298

Gao Q, Zhou F, Zhang K, Trajcevski G, Luo X, Zhang F (2017) Identifying human mobility via trajectory embeddings. In: Proc. of the 26th International Joint Conference on Artificial Intelligence (IJCAI), AAAI Press, Melbourne, Australia, pp 1689–1695

ten Holt GA, Reinders MJ, Hendriks E (2007a) Multi-dimensional dynamic time warping for gesture recognition. In: Proc. of the 13th Annual Conference of the Advanced School for Computing and Imaging, vol 300, p 1

ten Holt GA, Reinders MJ, Hendriks EA (2007b) Multi-dimensional dynamic time warping for gesture recognition. In: Thirteenth annual conference of the Advanced School for Computing and Imaging

Kung HT, Luccio F, Preparata FP (1975) On finding the maxima of a set of vectors. Journal of the ACM (JACM) 22(4):469–476

Lee JG, Han J, Li X, Gonzalez H (2008) Traclass: Trajectory classification using hierarchical region-based and trajectory-based clustering. VLDB 1(1):1081–1094, DOI 10.14778/1453856.1453972

Lines J, Bagnall A (2012) Alternative quality measures for time series shapelets. In: Proc. of the 13th International Conference on Intelligent Data Engineering and Automated Learning, Springer-Verlag, Berlin, Heidelberg, pp 475–483

Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization 26(6):369–395

Mello RdS, Bogorny V, Alvares LO, Santana LHZ, Ferrero CA, Frozza AA, Schreiner GA, Renso C (2019) Master: A multiple aspect view on trajectories. Transactions in GIS 23(4), DOI 10.1111/tgis.12526, URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12526`

Patel D, Sheng C, Hsu W, Lee ML (2012) Incorporating duration information for trajectory classification. In: Proc. of the 28th ICDE, IEEE, Washington, DC, USA, pp 1132–1143, DOI 10.1109/ICDE.2012.72

Rowland MM, Bryant LD, Johnson BK, Noyes JH, Wisdom MJ, Thomas JW (1997) Starkey project: history facilities, and data collection methods for ungulate research. Tech. rep., Portland, Or.: US Dept. of Agriculture, Forest Service, Pacific Northwest Research Station

Shokoohi-Yekta M, Hu B, Jin H, Wang J, Keogh E (2017) Generalizing dtw to the multi-dimensional case requires an adaptive approach. Data mining and knowledge discovery 31(1):1–31

Tan PN, Steinbach M, Kumar V (2005) Introduction to Data Mining, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA

Veldhuizen DAV, Lamont GB (2000) Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. Evolutionary computation 8(2):125–147

Vlachos M, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: Proc. of the 18th International Conference on Data Engineering, IEEE, San Jose, CA, USA, pp 673–684

Xiao Z, Wang Y, Fu K, Wu F (2017) Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. ISPRS Int J Geo-Inf 6(2):57

Yang D, Zhang D, Zheng VW, Yu Z (2015) Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. IEEE Transactions on Systems, Man, and Cybernetics: Systems 45(1):129–142

Ye L, Keogh EJ (2011) Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min Knowl Discov 22(1-2):149–182

Zheng Y, Chen Y, Li Q, Xie X, Ma WY (2010) Understanding transportation modes based on gps data for web applications. ACM Transactions on the Web (TWEB) 4(1):1–36