UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Carlos Andres Ferrero

# Discovering Relevant Subtrajectories for Multidimensional Trajectory Classification

Florianópolis
2020

Carlos Andres Ferrero

# Discovering Relevant Subtrajectories for Multidimensional Trajectory Classification

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Doutor em Ciência da Computação.
Orientador: Prof. Dra. Vania Bogorny

Florianópolis
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Ferrero, Carlos Andres
Discovering Relevant Subtrajectories for Multidimensional Trajectory Classification / Carlos Andres Ferrero ; orientadora, Vania Bogorny, 2020.
118 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2020.

Inclui referências.

1. Ciência da Computação. 2. Multiple Aspect Trajectory. 3. Trajectory Classification. 4. Relevant Subtrajectories. 5. Movelets. I. Bogorny, Vania. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Carlos Andres Ferrero

**Discovering Relevant Subtrajectories for Multidimensional Trajectory Classification**

O presente trabalho em nível de doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Profa. Vania Bogorny, Dra.
Universidade Federal de Santa Catarina

Prof. José Antônio Fernandes de Macêdo, Dr.
Universidade Federal do Ceará

Prof. Mauro Roisenberg, Dr.
Universidade Federal de Santa Catarina

Prof. Gustavo Enrique de Almeida Prado Alves Batista, Dr.
University of New South Wales - UNSW Sydney

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Doutor em Ciência da Computação.

Documento assinado digitalmente
Vania Bogorny
Data: 14/02/2020 12:09:22-0300
CPF: 684.586.570-15

Profa. Dra. Vania Bogorny
Coordenador do Programa

Documento assinado digitalmente
Vania Bogorny
Data: 14/02/2020 12:09:50-0300
CPF: 684.586.570-15

Prof. Dra. Vania Bogorny
Orientador

Florianópolis, 2020.

Aos meus pais, Mónica e Carlos,
e ao meu amor, Anahi Macário

## AGRADECIMENTOS

Agradeço à minha orientadora Profa. Vania Bogorny pela amizade, pelos muitos ensinamentos, pela orientação, pelos desafios, pela paciência e pelo intentivo. Também ao Prof. Luis Otávio pela amizade, orientação, incentivo e apoio, bem como pela dedicação na leitura detalhada dos textos.

Aos professores do Programa de Pós-graduação em Ciência da Computação (PPGCC) da Universidade Federal de Santa Catarina (UFSC) e aos técnicos administrativos, especialmente à Katiana e ao Ivan, que me auxiliaram de forma muito prestativa.

Ao Instituto Federal de Santa Catarina (IFSC) pela oportunidade de afastamento integral para desenvolver o doutorado. Aos professores e tecnicos administrativos do IFSC, Campus Lages, que me incentivaram e me apoiaram em todo momento.

Aos professores que, como membros de banca de defesa, se dispusseram a ler, entender e discutir os meus textos. No seminário em andamento, aos Profes. Carina Dorneles e Ronaldo Mello. No exame de qualificação, aos Profes. Jose Antonio Macedo, José Leomar Todesco e Silvia Modesto Nassar. Na defesa da tese, aos Profes. Jose Antonio Macedo, Gustavo Batista e Mauro Roisenberg. Agradeço a todos pelas contribuições, críticas e sugestões.

À minha companheira da vida, meu amor, Anahi Macário de Eveche, que foi o pilar sentimental fundamental. Anahi me apoiou, incentivou e amou de maneira imensurável.

Aos meus pais, Mónica Sotuyo e Carlos Ferrero, pelo apoio, incentivo e amor que foi dado para afrontar as dificuldades desta etapa e para comemorar os bons momentos. Aos também familiares, Marlene e Victor Hugo, Larissa, Neri, Mariza e Raul, Rai e Raissa, por todo o apoio e incentivo.

Aos amigos e colegas do grupo de pesquisa com quem compartilhamos muitos momentos na sala de pesquisa, Lucas Alencar, Areli, André Salvaro, Ricardo, Marco, Denis, André Lehmann, Lucas Petry, Camila, Yuri, Ana e Tarlis.

Aos meus grandes amigos e irmãos da vida de Lages e Florianópolis, Felipe Schneider, Delcio, Raquel, Maurício, Cleber, Francine, Gabriel, Juliana, Jorge (*in memoriam*), Beta, Ivo, Clarinha, Giuliano, Luciele, Felipe e Mari, Priscila e Arthur.

Aos meus grandes amigos e irmãos da vida, André e Barbara, Willian e Dabna, Ricardo, Eduardo e Glaucia, Maksoel, Everton e Chris, Homero e Helen, Joylan e Ana, e Rafael.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

"A educação exige os maiores cuidados,
porque influi sobre toda a vida."
*Sêneca*

## RESUMO

Estamos vivendo a era do rastreamento do movimento de pessoas e de outros objetos móveis, em que uma grande quantidade de informação sobre a rotina diária das nossas vidas é coletada e armazenada em diferentes locais e formatos. Esses dados de movimento são chamados de trajetórias, e consistem em um dado complexo que envolve as dimensões espaço (onde o objeto está), tempo (quando), e em alguns casos semântica (o quê ou como faz). Mais recentemente, trajetórias estão sendo representadas por múltiplos aspectos, os quais permitem analisar o movimento do objeto móvel sob diferentes pontos de vista, como as interações em redes sociais ou a sequência de meios de transporte utilizados pelo objeto móvel. No entanto, analisar esse novo tipo de dado para descobrir padrões de movimentação é ainda uma questão de pesquisa em aberto. Uma das tarefas de descoberta de padrões mais importantes em Mineração de Dados é a classificação, que consiste em criar modelos matemáticos preditivos a partir de dados e usar esses modelos para predizer situações futuras. Classificação de Trajetórias é uma tarefa muito complexa, pois esses modelos devem ser construídos a partir de dados sequenciais envolvendo as dimensões de espaço, de tempo e, mais recentemente, semânticas. Nos últimos anos a classificação de trajetórias tem sido aplicada em problemas reais, considerando apenas as dimensões espaço e tempo, ou atributos criados a partir dessas dimensões, como velocidade e aceleração. No entanto, esses métodos não suportam trajetórias com dimensões além de espaço e tempo, como as provenientes de redes sociais, que envolvem diversas informações semânticas do movimento de um indivíduo. Como consequência, novos métodos para classificação de trajetórias são necessários para lidar com esse novo tipo de dado. Nesse sentido, o maior desafio consiste em identificar as partes de uma trajetória, chamadas de subtrajetórias, que melhor representam o movimento de indivíduos de uma classe em um problema de classificação. Nesta tese são propostos dois novos métodos para encontrar automaticamente as subtrajetórias mais relevantes de um conjunto de dados de trajetórias em um problema de classificação, sem a necessidade de passagem de parâmetros. O primeiro método, MOVELETS, utiliza uma abordagem baseada em distância entre subtrajetórias que considera todas as dimensões da trajetória de forma conjunta. Esse método é mais apropriado para problemas de classificação de trajetórias brutas, representadas apenas pelas dimensões espaço e tempo. O segundo método, MASTERMOVELETS, encontra as subtrajetórias mais relevantes e a melhor combinação de dimensões para cada subtrajetória, o que torna o método robusto para classificação de trajetórias com múltiplos aspectos. Os métodos propostos foram avaliados experimentalmente com conjuntos de dados reais de trajetórias. Os modelos preditivos construídos usando MOVELETS apresentaram melhor qualidade preditiva em relação aos métodos estado da arte, em quatro problemas clássicos classificação de trajetórias brutas, são eles: classificação de animais, de furações, de caminhões e de movimentação de pessoas (GeoLife). O método MASTERMOVELETS foi avaliado em problemas de classificação de trajetórias com múltiplos aspectos, provenientes de bases de dados de *check-ins* de três redes sociais, Gowalla, Brightkite e Foursquare. Os modelos construídos também apresentaram melhor qualidade preditiva em relação aos métodos existentes. Os resultados alcançados indicam que os métodos propostos superaram o estado da arte e são eficientes para classificação de trajetórias com múltiplos aspectos e promissores para a classificação de dados sequenciais multidimensionais.



**Palavras-chave:** Trajetórias Multiaspecto. Classificação de Trajetórias. Subtrajetórias Relevantes. Movelets. Classificação de Sequências Multidimensionais.

# RESUMO ESTENDIDO

## Introdução

Nos últimos anos uma grande quantidade de informação sobre o movimento e a rotina diária das nossas vidas é armazenada em diferentes locais e formatos. Esses dados de movimento são chamados de *trajetórias*. A trajetória bruta é a forma mais simples de representação de uma trajetória, que envolve as dimensões de espaço e tempo. Em 2007, Spaccapietra (SPACCAPIETRA et al., 2008) apresentou uma nova representação, na qual as trajetórias brutas são enriquecidas com o nome dos locais visitados pelo indivíduo e o tempo que o indivíduo esteve em cada local. Essa representação é chamada de *trajetória semântica* e estende a definição de trajetória bruta para três dimensões: *espaço*, *tempo* e *semântica*.

Com a disseminação das redes sociais, as trajetórias passaram a ser mais semânticas e heterogêneas, como mostra o exemplo da Figura 1. Nesta figura, um indivíduo inicia sua trajetória em casa, depois vai para o trabalho de carro e, posteriormente, vai jantar em um restaurante. Durante essa jornada as condições climáticas mudaram duas vezes, de chuva para nublado e de nublado para sol, e os meios de transporte utilizados para deslocamento foram a pé e de carro. Assim, o movimento do indivíduo é enriquecido com múltiplas dimensões de dados em diferentes formatos, que incluem espaço, tempo, texto, e outas informações categóricas e numéricas. Esse novo tipo de dado é chamado de *Trajetória Multiaspecto* e foi introduzida durante o desenvolvimento desta tese em (FERRERO; ALVARES; BOGORNY, 2016; MELLO et al., 2019).


**Figura 1.** Um exemplo de *Trajetória Multiaspecto*.

A *classificação* é uma tarefa de interesse em Mineração de Dados que consiste em aprender modelos matemáticos preditivos a partir de dados de um problema e usar esses modelos para classificar novas instâncias do problema. Classificar trajetórias consiste em predizer a classe de um objeto móvel baseado em sua trajetória (LEE; HAN; LI, 2008) e é uma tarefa muito complexa, já que os modelos preditivos devem ser construídos a partir de dados sequenciais envolvendo as dimensões de espaço, de tempo e, mais recentemente, semânticas. Nos últimos anos a classificação de trajetórias tem sido aplicada em problemas reais, considerando apenas as dimensões espaço e tempo, ou atributos criados a partir dessas dimensões, como velocidade e aceleração (LEE; HAN; LI, 2008; DODGE; WEIBEL; FOROOTAN, 2009; ZHENG et al., 2010; PATEL et al., 2012; XIAO et al., 2017; ETEMAD; JÚNIOR; MATWIN, 2018). No entanto, esses métodos não suportam trajetórias com dimensões além de espaço e tempo, como as provenientes de redes sociais, que envolvem diversas informações semânticas do movimento de um indivíduo.

Como consequência, novos métodos para classificação de trajetórias são necessários para lidar com esse novo tipo de dado. Nesse sentido, um dos maiores desafios em descoberta de padrões em trajetórias é identificar as partes de uma trajetória (subtrajetórias) que melhor representam o movimento de indivíduos de uma classe em um problema de classificação. Uma

subtrajetória relevante é aquela que trajetórias de uma classe passam perto e trajetórias de outras classes passam longe. A hipótese lançada neste trabalho é que a descoberta e extração das subtrajetórias mais relevantes, em problemas de classificação de trajetórias representadas por múltiplos aspectos, pode melhorar de forma significativa a acurácia de modelos preditivos para classificação de trajetórias. Nesse contexto, nesta tese são propostos novos métodos para encontrar automaticamente as subtrajetórias mais relevantes de um conjunto de dados de trajetórias em problemas de classificação.

**Objetivos**

O principal objetivo desta tese é propor novos métodos para classificação de trajetórias que suportem trajetórias representadas por múltiplos aspectos. Para atingir esse objetivo geral foram delineados os seguintes objetivos específicos:

- Propor novos métodos para extrair subtrajetórias relevantes sem realizar o particionamento prévio da trajetória;
- Propor novos métodos para encontrar a combinação de dimensões de cada subtrajetória que apresenta maior potencial de discriminação;
- Propor novos métodos para medir a relevância de subtrajetórias;
- Propor novos métodos para construir modelos de classificação usando as subtrajetórias relevantes.

**Metodologia**

Esta tese consiste em uma pesquisa científica classificada de acordo com (SILVA; MENEZES, 2001) pela sua *natureza*, *abordagem* e *procedimentos técnicos*. Quanto à *natureza* de pesquisa este trabalho é de pesquisa *básica*, pois consiste na criação de novos métodos para análise de dados sequenciais multidimensionais, os quais podem ser úteis em diferentes domínios. Quando à *abordagem* este trabalho é *quantitativo*, pois são utilizadas técnicas estatísticas para avaliar o desempenho dos métodos propostos, em termos de capacidade preditiva, tempo de execução e escalabilidade. Quantos aos *procedimentos técnicos* utilizados neste trabalho a pesquisa é *bibliográfica* e *experimental*, pois foi realizada uma revisão da literatura para compreender o estado da arte e encontrar o nicho de pesquisa e os novos métodos propostos foram avaliados experimentalmente e comparados com outros métodos existentes na literatura. Para alcançar os objetivos propostos nesta tese foram delineadas os seguintes tarefas.

1. Acompanhar as publicações sobre classificação de trajetórias nas conferências e periódicos científicos mais importantes da área.
2. Definir novos métodos para extrair subtrajetórias relevantes e introduzir o conceito de *movelets*, baseado no conceito de *shapelets* de séries temporais, para definir subtrajetórias relevantes para a classificação de trajetórias;
3. Definir novos métodos para mensurar a relevância de trajetórias: a relevância de *shapelets* em séries temporais é medida utilizando o ganho de informação, no entanto, medidas mais apropriadas para dados de trajetória podem auxiliar a encontrar melhores subtrajetórias e melhorar a acurácia dos classificadores;
4. Definir novos métodos para encontrar o alinhamento entre trajetórias e subtrajetórias considerando múltiplas dimensões: o alinhamento proposto para *shapelets* leva em conta apenas uma única dimensão, e novos métodos são necessários para encontrar esse alinhamento considerando múltiplas dimensões;
5. Definir os conjuntos de dados para experimentação;

6. Avaliar os métodos propostos experimentalmente com conjuntos de dados reais e compará-los com os métodos existentes na literatura;
7. Escrever o documento da tese descrevendo o estado da arte, os temas de pesquisa em aberto, as contribuições dos métodos propostos para esses temas de pesquisa e os avanços produzidos em relação ao estado da arte.

Nesta tese são propostos dois novos métodos para encontrar automaticamente as subtrajetórias mais relevantes de um conjunto de dados de trajetórias em um problema de classificação. O primeiro método é chamado de MOVELETS (FERRERO et al., 2018) e utiliza uma abordagem baseada em distância entre subtrajetórias que considera todas as dimensões da trajetória de forma conjunta. Esse método é mais apropriado para problemas de classificação de trajetórias brutas, representadas pelas dimensões espaço e tempo. Nesse método também foi proposta uma nova técnica para medir a relevância das subtrajetórias, denominada *Left Side Pure (LSP)*, que resolve um problema do uso do ganho de informação no contexto de dados de trajetórias.

O segundo método é chamado de MASTERMOVELETS e encontra as subtrajetórias mais relevantes e a melhor combinação de dimensões para cada subtrajetória, o que torna o método robusto para classificação de trajetórias com múltiplos aspectos (FERRERO et al., 2020). Nesse método também foram propostos duas novas técnicas, denominadas MASTERALIGNMENT e MASTERRELEVANCE, para encontrar o melhor alinhamento de trajetórias e subtrajetórias e para medir a relevância de subtrajetórias, respectivamente, no contexto trajetórias representadas por múltiplas dimensões com tipos de dados heterogêneos.

**Resultados e Discussão**

O métodos propostos nesta tese foram avaliados experimentalmente e comparados com outros métodos da literatura. O método MOVELETS foi avaliado experimentalmente com conjunto de dados reais de trajetórias brutas, envolvendo as dimensões espaço e tempo, para classificação de trajetórias de animais, de furações, de caminhões e de movimentação de pessoas (conjunto de dados GeoLife), e foi comparado com outros métodos existentes para classificação de trajetórias, como o TraClass (LEE et al., 2008), o TCPR (PATEL et al., 2012) e os propostos em (DODGE; WEIBEL; FOROOTAN, 2009; ZHENG et al., 2010; XIAO et al., 2017). Os resultados experimentais mostraram a eficiência de método proposto, em que MOVELETS melhorou por ampla margem a qualidade preditiva dos modelos de classificação em relação às abordagens existentes na literatura para todos os conjuntos de dados utilizados (FERRERO et al., 2018).

O método MASTERMOVELETS foi avaliado em problemas de classificação de trajetórias com múltiplos aspectos, provenientes de bases de dados de *check-ins* de três redes sociais, Gowalla, Brightkite e Foursquare. A qualidade preditiva dos modelos construídos utilizando MASTERMOVELETS foi comparada com outros métodos na literatura, como o BiTULER (GAO et al., 2017), proposto para classificação de trajetórias provenientes de redes sociais, e a classificadores baseados no algoritmo dos vizinhos mais próximos utilizando diferentes medidas de similaridade e distância para dados sequenciais multidimensionais: *Longest Common Subsequences* (LCSS) (VLACHOS; KOLLIOS; GUNOPULOS, 2002), *Edit Distance for Real Sequences* (EDR) (CHEN; ÖZSU; ORIA, 2005), *Dynamic Time Warping for Multidimensional Sequences* (MD-DTW) (HOLT; REINDERS; HENDRIKS, 2007), and *Multidimensional Similarity Measure* (MSM) (FURTADO et al., 2016). Os modelos construídos usando MASTERMOVELETS também apresentaram melhor qualidade preditiva em relação aos métodos existentes, reduzindo o erro de classificação entre 15% a 63%.

Apesar dos avanços alcançados na qualidade preditiva dos modelos utilizando os métodos MOVELETS e MASTERMOVELETS, ambos os métodos possuem uma alta complexidade de

tempo de execução, o que pode limitar a escalabilidade desses métodos, ou seja, a sua aplicação em cenários com grandes volumes de dados. Nesse sentido, foi realizada uma análise de escalabilidade para verificar o comportamento de tempo de execução desses métodos ao variar o número de trajetórias, o número de elementos das trajetórias e o número de dimensões que compõem as trajetórias. Adicionalmente, foram propostas duas novas variações dos métodos propostos que limitam o tamanho máximo das subtrajetórias a serem exploradas, chamados MOVELETS-LOG e MASTERMOVELETS-LOG, os quais reduzem de maneira significativa o tempo de execução dos métodos, mantendo a acurácia dos modelos preditivos.

**Considerações Finais**

Nesta tese foi abordado o problema de classificação de trajetórias, que é um tema importante em mineração de dados de mobilidade, com diversas aplicações. O tema foi abordado do ponto de vista de explorar e encontrar subtrajetórias relevantes para construir modelos de classificação. Foram propostos dois métodos que consideram diferentes cenários. O primeiro método, MOVELETS, encontra subtrajetórias relevantes considerando todas as dimensões da trajetória de forma conjunta. O segundo método, MASTERMOVELETS, procura pela melhor combinação de dimensões para encontrar cada subtrajetória relevante. Os métodos propostos foram avaliados experimentalmente com conjuntos de dados reais de trajetórias e comparados com outros métodos existentes na literatura, apresentando melhor desempenho em todos os conjuntos de dados utilizados. As subtrajetórias encontradas pelos métodos propostos são neutras em relação ao algoritmo de construção de modelos a ser utilizado, fáceis de visualizar, de descrever, de entender e de procurar em novos conjuntos de dados de trajetórias. Os métodos não necessitam que parâmetros sejam configurados e são independentes de domínio, o que significa que podem ser usados em qualquer problemas de classificação de sequências multidimensionais.

Trabalhos futuros incluem propor novos métodos para melhorar complexidade de tempo de MOVELETS e MASTERMOVELETS; usar técnicas de *word embeddings* para medir a distância entre elementos da trajetória baseado em contexto; e explorar a extração de atributos de subtrajetórias para descobrir *movelets*.

**Palavras-chave:** Trajetórias Multiaspecto. Classificação de Trajetórias. Subtrajetórias Relevantes. Movelets. Classificação de Sequências Multidimensionais.

## ABSTRACT

We are witnessing the era of movement tracking and mining, where huge volumes of data about our daily lives are being collected and stored in several sources and formats. These data are stored in the form of trajectories, that consist of a complex data type that involves space and time dimensions, and in some cases also semantic dimensions. More recently, trajectories can be represented by multiple aspects, leading to the analysis of the object movement from different points of view, such as the social network interactions, the sequence of transportation means used by the object on his/her movement, etc. However, analyze this type of data for knowledge discovery is an open research area. Classification is a Data Mining task that consists of learning models from a dataset and use these models to classify new samples. Trajectory classification is a very complex task, because the pattern discovery involves space, time, semantics, and sequences. 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, like speed, acceleration, and turning angle. 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 data. The main challenge is how to automatically explore, combine, and select the data dimensions and to discover the subtrajectories that better discriminate the class. In this thesis we define the concept of multiple aspect trajectory and we propose two new parameter-free methods for extracting the most relevant subtrajectories for trajectory classification. The first method, called MOVELETS, uses a distance-based approach that considers all dimensions together to find the most relevant subtrajectories. This method is very robust for classification of raw trajectories, which include only space and time dimensions, although it also works for multiple aspect trajectories. The second method, called MASTERMOVELETS, also finds the most relevant subtrajectories, but considering multiple and heterogeneous dimensions. This method automatically explores trajectory dimensions and finds the best dimension combination of subtrajectories, which makes it robust for high dimensional trajectory data. Experimental results show that MOVELETS outperforms state-of-the-art methods for raw trajectory classification and MASTERMOVELETS outperforms existing methods to classify multiple aspect trajectories, indicating that our proposals are effective and are very promising for multidimensional sequence data classification.



**Keywords:** Multiple Aspect Trajectory. Trajectory Classification. Relevant Subtrajectories. Movelets. Multidimensional Sequence Classification.

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF SYMBOLS

| | |
|---|---|
| $T$ | Trajectory |
| $class_T$ | Class label of trajectory $T$ |
| $c$ | Class label |
| $P, Q$ | Trajectory instances |
| $\mathbf{T}$ | Trajectory set |
| $e$ | Trajectory element |
| $p$ | Raw trajectory point |
| $x, y$ | Spatial location |
| $t$ | time instant |
| $D$ | Element dimension set |
| $C$ | Subset of $D$ |
| $d$ | Element dimension |
| $s, r$ | Subtrajectory instances |
| $S$ | Subtrajectory set |
| $S_T^w$ | All subtrajectories of length $w$ in trajectory $T$ |
| $S_T^*$ | All subtrajectories of any length in trajectory $T$ |
| $dist_e$ | Function to compute the distance value between two trajectory elements |
| $dist_{e_k}$ | Function to compute the distance value between two trajectory elements at dimension $k$ |
| $\alpha_{d_i}$ | Weight of dimension $d_i$ |
| $dist_s$ | Function to compute the distance value between two subtrajectories of equal length |
| $g_s^T$ | Distance of the best alignment of $s$ in $T$ |
| $min$ | Function that returns the minimum value of a value set |
| $\mathcal{M}$ | Movelet candidate from subtrajectory |
| $\mathbf{M}$ | Movelet candidate set |
| $\mathbf{M}'$ | Non-redundant movelet candidate set |
| $start$ | Position in $T$ where the subtrajectory of movelet candidate begins |
| $length$ | Subtrajectory length of movelet candidate |
| $G$ | Set of pairs $(g_T, class_T)$ to generate the orderline |
| $score$ | Relevance score of movelet candidate $M$ |

| | |
|---|---|
| $sp$ | Split point of movelet candidate $M$ |
| $A_1$ | Three dimensional array containing element distance values |
| $A_w$ | Three dimensional array containing subtrajectory distance values |
| $A[i, j, ..]$ | One dimensional array containing the distances of all possible alignments between a trajectory $T_i$ and a subtrajectory $s_j \in T$ |
| $i, j, k$ | Indexes for general use |
| $V$ | Distance vector between two trajectory elements |
| $v_k$ | Distance between two trajectory elements at the $k$th dimension |
| $\mathbf{v}_k$ | Distance between two subtrajectories of equal length at the $k$th dimension |
| $vdist_s$ | Function to compute the distance vector between two subtrajectories of equal length |
| $dist_{s_k}$ | Function to compute the distance between two subtrajectories of equal length at dimension $k$ |
| $\mathbf{V}$ | Distance vector between two subtrajectories of equal length |
| $W_T^{\mathbb{s}}$ | Distance vector of the best alignment of $s$ in $T$ |
| $\mathbb{M}$ | Multidimensional *movelet* candidate from subtrajectory |
| $\mathbb{W}$ | Set of pairs $(W_T, class_T)$ to generate the multidimensional orderline |
| $\mathbb{P}$ | Set of split points |
| $\mathbb{A}_1$ | Four dimensional array containing element distances values for all dimensions |
| $\mathbb{A}_w$ | Four dimensional array containing subtrajectory distance values for all dimensions |
| $\mathbb{A}[i, j, k, ..]$ | One dimensional array containing the distances of all possible alignments between a trajectory $T_i$ and a subtrajectory $s_j \in T$ at dimension $k$ |
| $C_d^*$ | Set of all dimension combination of $d \in D$ of any length |
| $R$ | Two dimensional array containing the distance ranking of the alignments between a subtrajectories and a trajectory for all dimensions |
| $\mathcal{D}$ | Dataset |

**CONTENTS**

# 1 INTRODUCTION

We are living the era of movement tracking and mining, where huge volumes of data about our daily lives are being collected and stored in several sources and formats. Some companies, such as Google and Apple, collect details about our daily routines, including the places we visit and the time we stay there. Facebook captures our location, stores our friendship relationships, as well as our thoughts and opinions about things and people. The Pokémon GO game emerged to capture our movement and photos of places we visit when capturing Pokémons, what certifies with a high accuracy where we are. In summary, when an individual is moving, his/her location is collected over time, in the form of sequences of spatio-temporal points, called *raw trajectories*. An example of raw trajectory is shown in Figure 1.1. A raw trajectory is a complex data type, represented as a sequence of points, and each point has *space* and *time* dimensions.


Figure 1.1 – Example of *raw trajectory*.

A raw trajectory is the most simple trajectory representation. In 2007, Spaccappietra (SPACCAPIETRA et al., 2008) introduced a new representation, in which raw trajectories can be enriched with the name of the places visited by an object and the amount of time the individual stays there. This representation is called *semantic trajectory*, and extends the definition of raw trajectory to support three dimensions: *space*, *time*, and *semantics*. By semantic dimensions we mean any type of information that is neither spatial nor temporal.

With the exploration of social media data and Internet channels in the era of Big Data, trajectories are becoming more semantic and more heterogeneous, as the example shown in Figure 1.2. 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 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*, and was introduced during the development of this thesis in (FERRERO; ALVARES; BOGORNY, 2016; MELLO et al., 2019). The movement of an individual is enriched with *multiple* and *heterogeneous* data dimensions, including timestamps, spatial locations, text, and other discrete and numeric information. It is clear from the figure that different distance functions are needed to deal with the heterogeneous data dimensions, but existing works for trajectory data mining, specially classification, have not considered these

multiple and heterogeneous dimensions so far.


Figure 1.2 – An example of Multiple Aspect Trajectory.

Trajectory classification is an important issue in mobility data mining, since it is used for several applications as discovering transportation modes, animal categories, hurricane strengths, etc. Most works on trajectory classification were developed for raw trajectories, i.e., trajectories with space and time information only. Among these works are (LEE et al., 2008; DODGE; WEIBEL; FOROOTAN, 2009; ZHENG et al., 2010; PATEL et al., 2012; XIAO et al., 2017; ETEMAD; JÚNIOR; MATWIN, 2018). In trajectory data mining, mainly for trajectory classification, an important step is to identify discriminant parts of trajectories (LEE et al., 2008; PATEL et al., 2012; FERRERO et al., 2018). These parts of trajectories, called subtrajectories, can characterize the behavior of an individual and discriminate it among others. Lee in (LEE et al., 2008) extracted discriminant subtrajectories for raw trajectory classification using only spatial information. Patel in (PATEL et al., 2012) extended the method of (LEE et al., 2008) to also consider the time duration information. However, these methods need to perform trajectory partitioning with predefined criteria, loosing important movement information that could discriminate the class.

On the other hand, in the context of trajectories represented by multiple and heterogeneous dimensions, as in *multiple aspect trajectories*, the problem is more difficult, because finding the discriminant parts of trajectories involves the analysis of other dimensions beyond space and time. For instance, trajectories generated by social networks, like Foursquare, may include other information about user check-ins beyond space and time, as the venue name, the category, the price, the rating, and the weather condition when the check-in happens (FERRERO et al., 2018; MELLO et al., 2019). Rossi in (ROSSI; MUSOLESI, 2014) and Gao (GAO et al., 2017) addressed the problem of building classification models for discovering the user (class) from user check-ins. In (ROSSI; MUSOLESI, 2014) the authors proposed a model that considers the check-ins spatial distance and the frequency. This approach is limited to identify users based on a set of check-ins (without any time order), i.e., it does not consider the sequence of check-ins, which is fundamental in trajectory analysis. In (GAO et al., 2017) the authors proposed a model that uses word embeddings (MIKOLOV et al., 2013) from check-in data and a Bidirectional Recurrent Neural Network (SCHUSTER; PALIWAL, 1997) to build the classifier. However, this approach is limited to trajectories represented by a sequence of check-in identifiers and it does not support other dimensions, such as time, space, category, price, or rating, available in *multiple aspect trajectories*.

In this work we define a relevant subtrajectory as a part of a trajectory with the capability for discriminating the classes of the classification problem. We claim that relevant subtrajectories for classification problems can include patterns that may not be discovered by methods that perform trajectory partitioning with predefined criteria or discretization, as has been done in the works of (LEE et al., 2008; PATEL et al., 2012; DODGE; WEIBEL; FOROOTAN, 2009; XIAO et al., 2017; ZHENG et al., 2010) and by methods that ignore the sequence of data, as in (ROSSI; MUSOLESI, 2014), or limit the sequential analysis to a single dimension, as in (GAO et al., 2017). So, our *main research question* is:

*How to discover relevant subtrajectories in trajectories represented by multiple and heterogeneous dimensions and use these subtrajectories to build trajectory classification models?*

A concept applied with success on time series classification that allows the identification of relevant parts of time series without performing either partitioning with predefined criteria or discretization is the *shapelet analysis*. This concept was introduced in (YE; KEOGH, 2011) and explored and improved in (MUEEN; KEOGH; YOUNG, 2011; HILLS et al., 2014; ZALEWSKI et al., 2016). However, the *shapelet analysis* cannot be directly applied to raw trajectory data, because it is defined for only one variable over time, and trajectory data have at least two variables to represent the space over time ($x$ and $y$), besides other movement features, such as speed, acceleration, apart from semantic dimensions. In addition, this analysis cannot be applied to trajectories represented by multiple and heterogeneous dimensions, like *multiple aspect trajectories*, because for this kind of data the method would need to explore different dimension combinations in order to find the best one for each subtrajectory.

To the best of our knowledge, there are no works in the literature for discovering and exploring the relevant parts of trajectories, with support to multiple and heterogeneous dimensions, for robust trajectory classification. In this thesis we propose new methods for discovering relevant parts of trajectories for classification, without performing either partitioning or discretization.

In order to answer our *main research question* we pose the hypothesis that *the discovery and extraction of the most relevant subtrajectories, from trajectories represented by multiple and heterogeneous dimensions, can significantly improve the accuracy of trajectory classification models*.

## 1.1 OBJECTIVES

The main objective of this thesis is to propose new methods for trajectory classification with support to multiple aspect trajectories. As specific objectives we have:

**O1:** Propose new methods to extract relevant subtrajectories without performing trajectory partitioning;

**O2:** Propose new methods to find the best dimension combination of subtrajectories;

**O3:** Propose new methods to measure the relevance of subtrajectories;

**O4:** Propose new methods to build trajectory classification models from relevant subtrajectories;

## 1.2 CONTRIBUTIONS

The main contributions of this thesis are:

**C1:** The definition of a new trajectory concept: multiple aspect trajectory (FERRERO; ALVARES; BOGORNY, 2016; MELLO et al., 2019) and its impact on mobility data analysis over different points of view;

**C2:** A *new method* to discover relevant subtrajectories, without performing trajectory partitioning with predefined criteria, for robust trajectory classification, called MOVELETS (FERRERO et al., 2018). This method finds relevant subtrajectories in multiple aspect trajectories using a distance-based approach that considers all dimensions together;

**C3:** A *new method* to discover relevant subtrajectories in multiple aspect trajectories, called MASTERMOVELETS (FERRERO et al., 2020). This method finds relevant subtrajectories using a ranking-based approach that allows discovering the best dimension combination for subtrajectories, in order to improve trajectory classification accuracy;

**C4:** An extensive experimental evaluation and comparison to the state of the art over both raw and multiple aspect trajectory datasets to demonstrate the potential of our methods.

## 1.3 METHODOLOGY, SCOPE AND STRUCTURE

To achieve the objectives we present the methodology as the following tasks:

1. Track the latest publications about trajectory classification, mainly using Google Scholar, focusing on high-impact journals and conferences (such as KBS, DMKD, VLDB, ICDE, IJCAI, ACMSAC, SIGKDD, SIGMOD, and others);
2. Define new methods to extract relevant subtrajectories for trajectory classification: we introduce the concept of *trajectory movelets*, based on *time series shapelets*, to find relevant subtrajectories and to perform trajectory classification;
3. Define new methods to measure the relevance of subtrajectories: the original relevance measure for *shapelets* is the information gain, however new relevance measures for *trajectory movelets* can help to find better subtrajectories and improve classification accuracy;

4. Define new methods to find the best alignment between trajectories and subtrajectories considering heterogeneous dimensions: the original algorithm for finding the best alignment in *shapelets* is limited to single dimension, but new methods can help to find the best alignment considering multiple dimensions;

5. Define a number of datasets;

6. Experimentally evaluate the proposed methods in real datasets and compare with state-of-the-art methods;

7. Write the thesis describing the state of the art, the research gaps, the contributions of our proposals for these research gaps and the advances over the state-of-the-art methods.

The scope of this thesis is limited to the definition of new methods for discovering and exploring relevant subtrajectories to improve trajectory classification over state of the art methods.

The rest of this document is organized as follows:

**Chapter 2 –** describes the state of the art on trajectory classification. Several related works are described and discussed in order to point the research gaps solved in thesis.

**Chapter 3 –** presents our proposal of a new method for discovering relevant subtrajectories *multiple aspect trajectory* for classification. Section 3.1 introduces basic definitions, Section 3.2 describes in details the method and the algorithms, and Section 3.3 presents the experiments.

**Chapter 4 –** presents our proposal of a new method for discovering relevant subtrajectories in *multiple aspect trajectories* for trajectory classification focusing on the discovery of the best dimension combination. Section 4.1 extends previous definitions, Section 4.2 introduces the new method and the algorithms to explore subtrajectory dimension combinations, and Section 4.3 presents the experiments.

**Chapter 5 –** presents the performance evaluation of the proposed methods in terms of processing time and scalability.

**Chapter 6 –** presents conclusions and future work for this thesis.

# 2 BASIC DEFINITIONS AND STATE OF THE ART

In this chapter we present basic definitions and the state of the art in trajectory classification, describing the main characteristics of the related works and discuss their limitations. Section 2.1 introduces the basic concepts, such as formal definitions of *trajectory* and *trajectory classification*, and Section 2.2 presents the state of the art about *trajectory classification* and *trajectory distance and similarity measures*.

## 2.1 BASIC CONCEPTS

A trajectory is represented as a sequence of points with multiple dimensions, including space and time. For many years a trajectory was represented and called raw trajectory, having a sequence of points $\langle p_1, p_2, ..., p_m \rangle$, where each point $p = (x, y, t)$ was a tuple with $x$ and $y$ representing the spatial dimension and $t$ representing time.

In 2007, Spaccapietra (SPACCAPIETRA et al., 2008) introduced the concept of semantic trajectory, where a trajectory is represented as a sequence of stops and moves. A stop is an important part of a trajectory where the moving object has stayed for a minimal amount of time, while the moves are the trajectory points between stops. What basically differs from this type of trajectory is that stops and moves are heterogeneous elements that represent trajectory parts, and that stops have besides the space and time dimensions, a third dimension representing the name of the stop. Since 2014, after the introduction of the CONSTANT model (BOGORNY et al., 2014), and more recently with the definition of multiple aspect trajectory (FERRERO; ALVARES; BOGORNY, 2016), semantic dimensions can be associated to each individual trajectory point. This semantic enrichment is specially important for trajectories inferred from social media data as Foursquare check-ins or Facebook.

A multiple aspect trajectory has besides space and time information, several semantic dimensions, but with the difference that these semantic dimensions can be associated to each trajectory point, and not to a subtrajectory as a stop or a move. All these dimensions are heterogeneous and may need different analysis functions when comparing trajectories. Therefore, in this thesis we represent a multiple aspect trajectory as a sequence of elements, where each element has a set of dimensions that include space, time, and semantics, according to the definition introduced by (FURTADO et al., 2016). For the sake of simplicity, in this thesis we call a multiple aspect trajectory as *trajectory*:

**Definition 2.1. Trajectory**. A *trajectory* $T$ is a multidimensional sequence of elements $\langle e_1, e_2, \ldots, e_m \rangle$, where each element has a set of $l$ dimensions $D = \{d_1, d_2, \ldots, d_l\}$.

A dimension can be the latitude, the longitude, the time, the name of the visited place, the temperature, the weather condition, etc.

In trajectory data mining, mainly in classification problems, we need to analyze parts of a trajectory instead of the entire trajectory, because some patterns may hold only on a small

portion of a trajectory. A part of a trajectory is called a *subtrajectory*.

**Definition 2.2. Subtrajectory**. Given a trajectory $T = \langle e_1, e_2, \ldots, e_m \rangle$ 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 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 $s_T^*$.

The problem we address in this thesis is *trajectory classification*, that consists in 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; STEINBACH; KUMAR, 2005), we define *trajectory classification* as:

**Definition 2.3. 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 $f(T) \rightarrow class_T$ that maps a sample trajectory $T$ to one of the predefined class labels.

There are two mechanisms to perform *classification*: *eager* and *lazy* (MITCHELL, 1997). The former consists of using supervised learning algorithms over training samples to build a model (classifier), and then use this classifier to assign a class label to new samples. Examples of this mechanism include Support Vector Machine (SVM) (CORTES; VAPNIK, 1995) and Decision Trees (DT) (QUINLAN, 1993). In contrast, *lazy* approaches do not involve a model construction and consist of assigning a class label to new samples based on the most similar samples into the training dataset. The most common algorithm used in this type of classification is the $k$ Nearest Neighbor ($k$NN), that uses a *distance or similarity measure* to compute the similarity between a new sample and all the samples in the training set, and assign the class label based on the $k$ most similar training samples (AHA; KIBLER; ALBERT, 1991).

A classifier is a function to perform classification. The evaluation of a classifier is very important to estimate how well the function classifies new instances of the problem, i.e. new trajectories. To evaluate the classifier the most used techniques are: *holdout* and *cross-validation* (WITTEN et al., 2016). *Holdout* consists of splitting the set of labelled samples in two subsets, training and test, where the first is used by a learning algorithm for training the classifier and the second for evaluating it. This technique allows to evaluate the classifier using samples never seen by the learning algorithm. *Cross-validation* consists of splitting the dataset into subsets (called folds), frequently 5 or 10, and generating a classifier for each of these subsets. Each subset is used to test the classifier and the remaining subsets are used to training it. This technique allows to evaluate the classifier by classifying all samples once.

Classification evaluation also needs a performance metric. The most common performance metric used in the literature is *accuracy*, but other metrics, like *precision*, *recall*, and *F-measure*, are also important to evaluate classifiers (SOKOLOVA; LAPALME, 2009),, which are very frequent in trajectory classification. These measures are calculated based on the con-

cepts of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN), defined for each class $c_i$ of the classification problem, which are presented as follows:

- $TP_i$ is the number of samples of class $c_i$ classified as $c_i$;
- $FP_i$ is the number of samples of class different from $c_i$ classified as $c_i$;
- $TN_i$ is the number of samples of class different from $c_i$ non classified as $c_i$;
- $FN_i$ is the number of samples of class $c_i$ non classified as $c_i$;

Based on these concepts we define and calculate the performance measures, accuracy, precision, recall, and F-measure. The *accuracy* is the overall effectiveness of a classifier and it is calculated as the number of correctly classified samples divided by the total number of samples, as in Equation 2.1.

$$acc = \frac{TP}{TP+TN} \tag{2.1}$$

Classification problems involving many classes also use the *acc top5*, which considers a sample as true positive if the true class is one of the five most probably classes predicted by the classifier.

*Precision* consists of the number of correctly classified samples of a class divided by the total number of samples classified as the class, as in Equation 2.2.

$$Precision = \frac{TP}{TP+FP} \tag{2.2}$$

*Recall* consists of the number of samples of a class correctly classified divided by the total number of samples of the class in the dataset. This measure is defined by Equation 2.3.

$$Recall = \frac{TP}{TP+FN} \tag{2.3}$$

*F-measure*, also called F1 score, consists of the combination of precision and recall and it is calculated as the harmonic mean between them, as in Equation 2.4.

$$F\text{-}measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.4}$$

In many classification problems the number of samples of each class is not different. When the difference in the number of examples among classes is expressive, it is important to consider the class frequency to calculate the performance measure, in order to reduce the effect of class imbalance. The weighted F-measure takes into account the class frequencies to calculate the F-measure and consists of the sum of the F-measure of each class weighted by the class frequency in the dataset, as in Equation 2.5,

$$wF\text{-}measure = \sum_i \frac{N_i}{N} \times F\text{-}measure_i \tag{2.5}$$

where $N_i$ is the number samples of class $c_i$, $N$ the total number of samples, and $F\text{-}measure_i$ the F-measure of class $c_i$.

The main problem addressed in this thesis is related to how to discover the class label of the moving object based on its trajectories, including both mechanisms: *eager* and *lazy*. In the following section, we present the state of the art in this issue.

## 2.2 STATE OF THE ART

In this thesis we deal with both mechanisms for trajectory classification: *eager* and *lazy*. As previously mentioned, the *lazy* mechanism needs a distance or similarity measure to perform the classification task. For that reason, besides presenting the related work about trajectory classification, we also include a section to present distance and similarity measures, that can be used to perform trajectory *lazy* classification (nearest neighbor). Section 2.2.1 presents the state of the art on *trajectory classification* and shows the main characteristics of related works and discusses their limitations. Then, Section 2.2.2 presents the state of the art of *trajectory distance and similarity analysis*.

### 2.2.1 Trajectory Classification

Trajectory classification is pointed by several surveys on trajectory data mining as one of the most important tasks (ZHENG, 2015; FENG; ZHU, 2016; MAZIMPAKA; TIMPF, 2016), because learning to classify the movement of objects can be a very complex task. Classify if an object is stopped or moving (SPACCAPIETRA et al., 2008) can be solved by analyzing only the movement speed, but classify an object by *how* he/she is moving is more difficult. For instance, how to discriminate private car drivers and taxi drivers based only on their movement, or how to discriminate individuals based on their check-ins in social media.

To find and organize related works we perform a systematic review on Google Scholar searching the following query: *"trajectory classification" OR "movement classification" OR "spatio-temporal classification"*. From the results we selected the first 100 papers to evaluate their relevance (because the others had very low relevance). We selected the most relevant papers and described them by the following characteristics:

1. Dimensions: which dimensions they support: space, time, and/or semantics;
2. Features: which global and/or local features are extracted from trajectories. Local features are extracted from trajectory points or subtrajectories, while global features are extracted from the entire trajectory;
3. Classification algorithm: which strategies were used to perform the classification task, such as Nearest Neighbor algorithm, Support Vector Machines, Decision Trees, etc;
4. Datasets: which datasets were used in the classification task;

5. Evaluation method: which methods are used to evaluate the quality of the classification tasks in the experiments.

Table 2.1 presents the comparison of related works, considering global and local *features* based on space, time, semantics, speed, acceleration, and direction, in addition to the extraction of relevant subtrajectories.

Table 2.1 – Comparative of features extracted on related work.

| Paper | Global Features | Local Features | Relevant Subtrajectories | Space | Time | Semantics |
|---|---|---|---|---|---|---|
| (LEE et al., 2008) | | ✓ | ✓ | ✓ | | |
| (DODGE; WEIBEL; FOROOTAN, 2009) | ✓ | ✓ | | | | |
| (ZHENG et al., 2010) | ✓ | ✓ | | | | |
| (SHARMA et al., 2010) | ✓ | ✓ | | | | |
| (LEE et al., 2011) | | ✓ | ✓ | | | ✓ |
| (SANTOS, 2011) | ✓ | ✓ | | | | |
| (BOLBOL et al., 2012) | ✓ | | | | | |
| (PATEL et al., 2012) | ✓ | ✓ | ✓ | ✓ | ✓ | |
| (SOLEYMANI et al., 2014) | ✓ | ✓ | | | | |
| (SOLEYMANI et al., 2015) | ✓ | ✓ | | | | |
| (VARLAMIS, 2015) | ✓ | | | | | |
| (ENDO et al., 2016) | ✓ | | | | | |
| (XIAO et al., 2017) | ✓ | ✓ | | | | |
| (GAO et al., 2017) | ✓ | ✓ | | | | |
| (ZHOU et al., 2018) | ✓ | ✓ | | | | |
| (ETEMAD; JÚNIOR; MATWIN, 2018) | ✓ | ✓ | | | | |
| **(FERRERO et al., 2018)** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| (SILVA; PETRY; BOGORNY, 2019) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **(FERRERO et al., 2020)** | | ✓ | ✓ | ✓ | ✓ | ✓ |

Most works in the literature perform trajectory classification using the *eager* mechanism, that consists of building classification models from a feature vector representation of trajectories. The feature vector is a set of global and local features extracted from trajectories, as mentioned in Table 2.1.

It is important to emphasize that trajectory classification methods do not propose new classification functions or models, but novelty relies on how and which features to extract from trajectories for classification problems.

Lee (LEE et al., 2008) proposed the first attempt of exploring subtrajectories as features for trajectory classification. The method consists of initially segment trajectories into subtrajectories when the movement direction changes rapidly. In a second step, this work groups similar subtrajectories (subtrajectories are similar when they are close in space), where each group has subtrajectories that can belong to different class labels. Then, the subtrajectories are divided in order to create subtrajectory groups of only one class label. After that, the groups are evaluated according to the relevance, which is measured by the average distance from a group to other groups of different classes, where the larger the distance is, the higher is the relevance. Thus, only those groups with a value greater than the median relevance are selected. Indeed, close groups with the same class can be merged, and only a representative subtrajectory of each group is considered as a local feature. Finally, each original trajectory is represented by a feature vector, where each entry of a feature vector is the frequency that each relevant subtrajectory happens in the trajectory (*frequency approach* for local features).

Dodge in (DODGE; WEIBEL; FOROOTAN, 2009) proposes a method to extract global and local features. Global features are minimum, maximum, mean, median, standard deviation, variance, and skewness of speed, acceleration, straightness, and direction. To extract local features it proposes the profile decomposition of trajectories, that consists of transforming a trajectory into a sequence of discrete values, as for instance, high or low speed. With that, statistics like mean, standard deviation, number of changes, and proportion, are extracted from the discrete sequence as local features. The profile decomposition is built also using acceleration, straightness, and turning angle, in addition to speed. Both global and local features are used to classify transportation means.

Zheng (ZHENG et al., 2010) considered only global features, as mean, variance, and maximum trajectory traveled distance, average speed, acceleration, and proposes three new features: heading change rate, stop rate, and velocity change rate. These features are also used to classify trajectories over transportation means.

Sharma (SHARMA et al., 2010) presented an alternative approach to the use of local and global features, using the mechanism of *lazy* classification, more specifically, the Nearest Neighbor (NN) classifier. This algorithm consists of assigning to an unlabeled new trajectory the label of the most similar trajectory in the training set. The similarity measure used to compare trajectories was the *route similarity*, proposed by Andrienko in (ANDRIENKO; ANDRIENKO; WROBEL, 2007), and the strategy was evaluated to classify vehicles.

Lee (LEE et al., 2011) proposed a method to extract relevant subtrajectories from trajectories represented by a sequence of roads followed by the object. First, the method transforms a trajectory into a sequence of roads, called *edges*. In this work, a subtrajectory is a subsequence of *edges*. Second, the authors select all subtrajectories that satisfied a maximum length and a minimum support. Third, using the $F$-score[1] to measure the subtrajectory relevance, the method selects the most relevant subtrajectories using a threshold for $F$-score. Finally, trajectories are

[1] $F$-score measures the discrimination of two sets of real numbers and it is commonly used for feature selection (CHEN; LIN, 2006).

represented using the *frequency approach* for local features, as in (LEE; HAN; LI, 2008). The method was evaluated using Support Vector Machines (SVM) to build models to classify trajectories of two taxi routes.

Santos (SANTOS, 2011) proposed a method that uses global and local features. To extract local features it transforms trajectory data into time series and applies a technique of time series analysis, called *motifs discovery*. The proposed method, first transforms trajectory data into a set of independent time series of speed, acceleration, direction, among others. Second, it uses the technique Symbolic Aggregate Approximation (SAX) (LIN et al., 2007) to transform each time series into a sequence of symbols, as for instance $a,a,a,b,b,a,...,b$. After that, all combinations of symbol subsequences with specific size are considered as local features. For example, for symbols *a* and *b*, the combinations of size 2 are $(aa)$, $(ab)$, $(ba)$, and $(bb)$. Finally, trajectories are represented using the *frequency approach* for each combination. The proposed method, in combination with global features, was evaluated using several algorithms, such as SVM, Bayesian, Logistic, and Decision Trees, to classify hurricanes, vessels, and animals, as in (LEE et al., 2008), reporting betters results.

Patel (PATEL et al., 2012) extends the proposal of Lee (LEE et al., 2008) to consider, in addition to the *space dimension*, the time duration and the speed to find relevant subtrajectories. The method consists of two steps. First, a directed graph is built, where the vertices are micro-clusters of sample points and the directed edges represent the sequence of points. Initially, the vertices contain only a point. Then, the vertices can be merged with others according to their classes, and the edges as well. The merging process uses the *Minimum Description Length (MDL)*[2]. Second, many depth first searches in the graph are performed in order to recall the most relevant subtrajectories, to constitute the feature vector. The relevant subtrajectories are also combined with relevant regions and global features.

Bolbol (BOLBOL et al., 2012) proposed a method to perform feature selection on global features using statistical tests based on correlation and analysis of variance. This work used the features speed, acceleration, distance, and heading change rate, for transportation means classification.

Reumers (REUMERS et al., 2013) proposed a method to extract local features using only the time dimension. The proposal consists of extracting the start time of the trajectory and the time duration to build a Decision Tree. The method was evaluated for activity classification. Li in (LI, 2014) proposes new global features to measure the complexity of the movement: fractal dimension and entropy approximation. This work evaluates the effects of considering the complexity features, in addition to other features to classify transportation means.

Soleymani (SOLEYMANI et al., 2014) analyzes trajectory data into multiple scales of time and space. The method first finds the best scale on time and space to analyze trajec-

[2] The main idea of the MDL Principle proposed in (RISSANEN, 1978) is that "any regularity in a given set of data can be used to compress the data, i.e. to describe it using fewer symbols than needed to describe the data literally" (GRÜNWALD, 2000). In trajectory segmentation, the MDL is used to find the optimal trade-off between preciseness and conciseness, where conciseness is the number of segments used to represent the trajectory, and preciseness is how closely these segments are to the original trajectory.

tories, and then extracts statistics from duration, speed, as local features. This method was evaluated only on a classification problem of fish datasets by building SVM classifiers. In a later work, Soleymani in (SOLEYMANI et al., 2015) integrates several global features, such as speed, acceleration, and direction, with entropy approximation and the wavelet coefficients[3] extracted from traveled distance, and demonstrates that the integration of these new features can improve classification accuracy. Global features were evaluated on biology datasets, building classification models based on SVM and Decision Trees.

Varlamis (VARLAMIS, 2015) uses global features based on time, space, and semantics, and proposes evolutionary algorithms to reduce the number of samples required for training a classifier. The author claims that this approach reduces the effect of the class unbalance, and the approach is evaluated on transportation means classification.

Macdonald (MACDONALD; ELLEN, 2015) introduced two new global features for trajectory classification based on coefficients of Zernike Moments[4] and coefficients of Hu Moments[5]. These features are well used for image classification problems. In the case of trajectories, only spatial points are used to extract these coefficients. These new features are evaluated on transportation means classification over the GeoLife dataset. Endo (ENDO et al., 2016) introduces the use of deep learning techniques for feature extraction from images generated from raw trajectories. The weak point of the previous approaches is that the features are not interpretable, and can only be explored visually.

Xiao (XIAO et al., 2017) integrates the global and local features proposed by Dodge in (DODGE; WEIBEL; LAUBE, 2009) with other global features as the mode, interval between maximum and minimum, interquartile range, kurtosis, coefficient of variation, and autocorrelation coefficient of speed, acceleration, straightness, and direction. Local and global features were used to build ensemble classifiers and compare them with traditional classification methods (SVM, Decision Trees, and NN), achieving better results on the task of transportation means classification than previous works.

Gao in (GAO et al., 2017) and Zhou in (ZHOU et al., 2018) represent user trajectories by a sequence of check-in identifiers and represent each check-in by an embedding vector, like in text mining (MIKOLOV et al., 2013). These works generated models based on Recurrent Neural Networks to find local and global relationships among check-ins for classification problems. These works differ mainly on the features extracted to perform the classification task: global and/or local. Global features are in general easier and cheaper to extract, but they have

[3] Wavelet analysis is a technique to modeling a signal using both time domain and frequency domain. It differs from Fourier analysis that uses only the frequency domain. According to Soleymani in (SOLEYMANI et al., 2014) wavelet coefficients of trajectories can reveal temporal and periodicity patterns.

[4] Zernike Moments come from modeling a signal as a set of orthogonal polynomials, called Zernike Polyinomials, as basis functions. Basis functions are used to transform a signal to a function space, for instance, in Fourier analysis a signal is transformed as a set of sines and cosines as basis functions. Zernike Moments are rotation invariant (do not change for values of rotation) and are useful in image pattern recognition (ARVACHEH; TIZHOOSH, 2005).

[5] Hu Moments are similar to Zernike Moments, but with other basis functions that achieve rotation, scale, and translation invariance, and also very useful for image patterns recognition (HUANG; LENG, 2010).

the limitation of considering only a uniform movement in the entire trajectory. Many problems can have good solutions using only global features, such as to discriminate walking trajectories and car trajectories, where in general, only the mean of the speed is enough. However, to discriminate private cars and taxis, both global and local features are necessary (XIAO et al., 2017).

Most of the works focus on transportation modes classification (DODGE; WEIBEL; FOROOTAN, 2009; ZHENG et al., 2010; REUMERS et al., 2013; VARLAMIS, 2015; MACDONALD; ELLEN, 2015; XIAO et al., 2017), others focus on animals, vessels, and hurricanes (LEE et al., 2008; SANTOS, 2011; PATEL et al., 2012), and a few works on very specific applications (SHARMA et al., 2010; SOLEYMANI et al., 2014; SOLEYMANI et al., 2015; ETEMAD; JÚNIOR; MATWIN, 2018).

Local features provide more detailed information about the movement related to subtrajectories. The works of (DODGE; WEIBEL; FOROOTAN, 2009) and (SANTOS, 2011) transform the trajectories into a sequence of *discrete* values, such as low and high speed, and describe trajectories by how many times predefined parts (e.g. $\langle low, high, low \rangle$) occur into trajectories. However, by transforming a trajectory into a sequence of discrete values, details of movement (e.g. of speed) can be lost. For instance, Lee (LEE; HAN; LI, 2008) partitions trajectories by direction change, loosing information about the speed variation into the partition, and (PATEL et al., 2012) partitions trajectories by homogeneous regions, loosing information about the real location of each trajectory point. In addition, these methods are limited to only considering the dimensions *space* and *time*, without consider *semantics*, which makes these methods inappropriate for analyzing trajectories represented by multiple and heterogeneous dimensions, like *multiple aspect trajectories*.

In the last years, trajectories based on check-ins are becoming more common, because of the exploration 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 trajectories generated using the GPS (Global Positioning System), so global and local geometrical features such as speed and acceleration, used in previous works to build classification models, cannot be extracted because the path between check-ins is normally not recorded. In addition, check-ins provide more semantic and textual information beyond the spatial location and the place name, such as the place category, the price, the rating, the review, etc., which allows trajectories to be analyzed under multiple aspects or semantic dimensions.

Rossi in (ROSSI; MUSOLESI, 2014), Gao in (GAO et al., 2017), and Zhou in (ZHOU et al., 2018) addressed the problem of training a classification model for discovering the user (class) from the user' check-ins. In (ROSSI; MUSOLESI, 2014) the author proposed an hybrid model that considers the check-ins' spatial distance and the frequency for identifying users. This approach is limited to identify users based on a set of check-ins (without any time order), i.e., it does not consider the sequence of check-ins. In (GAO et al., 2017) the author proposed a model based approach, called BiTULER, that uses word embeddings from check-in data and a Bidirectional Recurrent Neural Network (RNN) to build the classifier. In (ZHOU et al., 2018)

the authors proposed the method TULVAE, that also uses word embeddings from check-in data and a Bidireccional RNN, but unlike BiTULER, it uses the technique Variational Autoencoder, which is designed for using both labeled and unlabelled data to help with the model training. However, this approach is limited to trajectories represented only by a sequence of check-in identifiers and it does not support other dimensions, such as time, space, rating, and review, as in multiple aspect trajectories. Other works only study the relation between check-ins visitation and user characteristics, such as gender, age, education background (ZHONG et al., 2015), friendship (CHO; MYERS; LESKOVEC, 2011), and motivations (BILOGREVIC et al., 2015), without considering the sequence of check-ins for user classification.

Until 2018 only the works of Lee in (LEE et al., 2008) and (LEE et al., 2011), and Patel in (PATEL et al., 2012) explored relevant subtrajectories as features to build classification models. However, these methods do not have support to more dimensions beyond the space and time. Only from the methods proposed in (FERRERO et al., 2018; FERRERO et al., 2020) it was possible to discover relevant subtrajectories considering multiple dimensions for trajectory classification. Silva in (SILVA; PETRY; BOGORNY, 2019) presented a survey on trajectory classification based on feature extraction.

Another way to perform trajectory classification consists of classifying new trajectories according to the most similar trajectory in the trajectory training set, based on a specific *distance or similarity measure*. In this case, the *classification model* is formed by a set of labeled trajectories and the distance or similarity measure, as explained in the following section.

### 2.2.2 Trajectory Distance and Similarity Analysis

Similarity Analysis is the other important topic of this thesis, as similarity or distance functions can be used for Nearest Neighbor classification. *Similarity* is a numeric value of the degree of how much two objects are alike, usually non-negative and often between 0 and 1, where 0 indicates no similarity and 1 complete similarity (TAN; STEINBACH; KUMAR, 2005). In many cases, distance functions are used to measure object similarity, where the closer two objects are, the higher is their similarity. Thus, *similarity measures* are functions to quantify the similarity degree between two objects. Similarity measures are very useful in several trajectory data mining tasks, such as clustering (VRIES; SOMEREN, 2010; ATEV; MILLER; PAPANIKOLOPOULOS, 2010; GAO et al., 2016; CAI; LEE; LEE, 2016), prediction (YING et al., 2010; YING et al., 2011), and classification (REUMERS et al., 2013; GAO et al., 2017).

Trajectory distance and similarity measures can be divided in two groups. The first group consists of measures proposed specifically for trajectories represented by dimensions time, space, and/or semantics, as in (YING et al., 2010; YING et al., 2011; LIU; SCHNEIDER, 2012; XIAO et al., 2014; RANU et al., 2015; FURTADO et al., 2018). The second group consists of distance and similarity measures for general purpose, that can be applied for trajectories with any number of dimensions, as in (BERNDT; CLIFFORD, 1994; VLACHOS; KOLLIOS; GUNOPULOS, 2002; HOLT; REINDERS; HENDRIKS, 2007; FURTADO et al.,

2016; SHOKOOHI-YEKTA et al., 2017; LEHMANN; ALVARES; BOGORNY, 2019; PETRY et al., 2019).

In the first group of distance and similarity measures Ying (YING et al., 2010) proposed the Maximal Semantic Trajectory Pattern (MSTP) similarity. In this work, a semantic trajectory is a sequence of places visited by a moving object, and MSTP consists of finding the major common subsequence of visited places between two trajectories, using the *Longest Common Subsequence* (LCS) algorithm (BERGROTH; HAKONEN; RAITA, 2000), and then calculate the similarity by dividing the length of the longest common subsequence by the mean length of two trajectories. The limitation of the *MSTP* is to calculate the similarity between semantic trajectories considering only the semantic dimension (name of visited places), without considering other dimensions. To support space, this work was extended in (YING et al., 2011). The authors propose to first perform a clustering task using the MSTP as similarity measure, and then calculate the spatial distance between visited places to predict the possible next location. Despite the fact that the authors include the space dimension, it is used after the similarity computation as a clustering refinement to predict the next location.

Liu (LIU; SCHNEIDER, 2012) proposed a *distance measure* that combines both space and semantics. The difference with the measure proposed by Ying in (YING et al., 2011) is that the spatial and semantic dimensions are both considered to calculate the distance between trajectories, not only as a clustering refinement step. On the one hand, for the space dimension the measure combines spatial information of visited places, the length of trajectories, and the cosine distance. On the other hand, for the semantic dimension, the measure calculates the similarity as the length of the LCS divided by the minimum length of the two trajectories.

Xiao (XIAO et al., 2014) proposed a similarity measure that also considers the time and semantic dimensions. This measure combines the following dimensions of trajectories: the visited places, the travelled time between visited places, and the popularity of these places. The idea of this approach is that if two trajectories visit the same places $\langle School, Cinema, Restaurant \rangle$ in the same order, the similarity between them must be greater than between other two trajectories that visit the same places but at different time or in different order. Additionally, two trajectories that visit less popular places must have a similarity value greater than other two that visit more popular places. The authors propose an algorithm to find a set of common subsequences of places, called *Maximal Travel Matches* (MTM), that considers the travel time between places. Different from the LCS technique, that returns only the longest common subsequence, in this approach more than one sequence can be found, what is interesting for long trajectories. The similarity between trajectories is given by the set of local common subsequences and the popularity of visited places. The main drawback of MTM is that space is not considered.

One of the objectives of this thesis is to classify *multiple aspect trajectories* by considering time, space, and semantics, taking into account the sequence of trajectory points. Table 2.2 summarizes the similarity/distance measures according to the capability to be applied over raw trajectories and multiple aspect trajectories, and the supported dimensions.

Table 2.2 – Comparative of trajectory similarity measures and distance functions.

| | **Trajectory Type** | | **Dimensions** | | |
|---|---|---|---|---|---|
| Measure | Raw | Semantic | Time | Space | Semantics |
| DTW Distance (VLACHOS; KOLLIOS; GUNOPULOS, 2002) | ✓ | | ✓ | ✓ | |
| LCSS Ratio (VLACHOS; KOLLIOS; GUNOPULOS, 2002) | ✓ | | ✓ | ✓ | |
| EDR Ratio (CHEN; ÖZSU; ORIA, 2005) | ✓ | | ✓ | ✓ | |
| MSTP (YING et al., 2010) | | ✓ | ✓ | | ✓ |
| (LIU; SCHNEIDER, 2012) | | ✓ | ✓ | ✓ | ✓ |
| (LV; CHEN; CHEN, 2013) | | ✓ | ✓ | | ✓ |
| MTM (XIAO et al., 2014) | | ✓ | ✓ | | ✓ |
| MSM (FURTADO et al., 2016) | | ✓ | ✓ | ✓ | ✓ |
| UMS (FURTADO et al., 2018) | ✓ | | ✓ | ✓ | |
| SMSM (LEHMANN; ALVARES; BOGORNY, 2019) | | ✓ | ✓ | ✓ | |
| MUITAS (PETRY et al., 2019) | | ✓ | ✓ | ✓ | ✓ |

In the second group of distance and similarity measures, which are for general purpose, the most common trajectory distance and similarity measures are Dynamic Time Warping (DTW) (BERNDT; CLIFFORD, 1994), Longest Common Subsequences (LCSS) (VLACHOS; KOLLIOS; GUNOPULOS, 2002), Edit Distance for Real Sequences (EDR) (CHEN; ÖZSU; ORIA, 2005), Multidimensional Similarity Measure (MSM) (FURTADO et al., 2016), Stops and Moves Similarity Measure (SMSM) (LEHMANN; ALVARES; BOGORNY, 2019), and Multiply Aspect Trajectory Similarity Measure (MUITAS) (PETRY et al., 2019).

DTW finds an optimal alignment between two sequences, which are warped in a nonlinear manner to match each other (BERNDT; CLIFFORD, 1994). Ten Holt in (HOLT; REINDERS; HENDRIKS, 2007) adapted DTW for multidimensional sequences, called MD-DTW, by transforming the distance values of all dimensions in only a distance value. Recently, Shokoohi-Yekta in (SHOKOOHI-YEKTA et al., 2017) proposed the DTWa, an adaptive approach of DTW for multidimensional sequence classification. However, DTW-based

distance measures are designed for numerical dimensions only. LCSS consists of finding the longest common subsequence between two trajectories (VLACHOS; KOLLIOS; GUNOPULOS, 2002). EDR consists of seeking the minimum number of edit operations (insert, delete, change) to transform one trajectory in another (CHEN; ÖZSU; ORIA, 2005).

MSM is a multidimensional similarity measure to be applied over trajectories represented by multiple dimensions, ignoring the order. This measure considers all dimensions together and also allows partial matching (FURTADO et al., 2016). MSM was initially proposed to measure the distance between trajectories represented by a sequence of stops, although it can be applied over social media check-ins. Lehmann in (LEHMANN; ALVARES; BOGORNY, 2019) proposed an extension of MSM, called SMSM, that considers both stop and moves to measure the similarity between semantic trajectories. The main problem related to these 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 DTW needs weights, the measures LCSS and EDR need thresholds, and MSM and SMSM need both thresholds and weights. These parameters are domain dependent and very difficult to estimate, even more in the presence of heterogeneous dimensions, because each dimension has its own way of measuring distance (e.g. the *space* dimension in meters, *time* in minutes, and *price* in price units).

Furtado in (FURTADO et al., 2018) proposed the similarity measure called UMS (Uncertain Movement Similarity) to analyze the physical movement of objects and retrieve which trajectories follow similar routes in space. UMS is limited to the spatial dimension, so focusing on spatial similarity, not being appropriate for multiple aspect trajectories.

MUITAS is the first attempt in the literature to measure the similarity between *multiple aspect trajectories* (PETRY et al., 2019). LCSS and EDR assume all dimensions as dependents, considering all dimensions together, while MSM assumes all dimensions as independent among dimensions, considering partial matchings. MUITAS supports both independent and dependent trajectory dimensions, allowing user-defined relationships of dimensions dependency, distance functions for each dimension and weights that represents the importance degree of each dimension. The main drawback of this method is the need of an expert to define the weights and relationships between dimensions.

Another drawback of distance based classifiers is that in general they do not perform any learning process to build a model. Therefore, they need to compare new test trajectories with all trajectories in the training set, and do not allow the extraction of the most relevant parts of trajectories, i.e., the discriminant subtrajectories.

Few works in the literature proposed methods for discovering relevant subtrajectories for classification, as in (LEE et al., 2008; PATEL et al., 2012), and these works perform trajectory partitioning with a predefined criterion, loosing important information. Only the work of Patel (PATEL et al., 2012) uses dimensions space and time, but it does not have support to other trajectory dimensions. In Chapters 3 and 4 we present our proposal for discovering relevant subtrajectories in trajectories represented by multiple and heterogeneous dimensions.

# 3 DISCOVERING RELEVANT SUBTRAJECTORIES FOR TRAJECTORY CLASSIFICATION

In this chapter we present the first contribution of this thesis: a new method to discover relevant subtrajectories for robust trajectory classification, called MOVELETS. Our method explores all subtrajectories from a set of trajectories and evaluates the relevance of each subtrajectory for selecting the most relevant ones. The method is parameter free and domain independent.

MOVELETS is based on a concept applied with success on time series classification, *shapelet analysis*. It allows the identification of relevant parts of time series, called *shapelets*, without performing either partitioning with predefined criteria or discretization. A *shapelet* is subsequence of a time series with the capability to discriminate classes in a time series classification problem, which can be used to build classification models (YE; KEOGH, 2011; HILLS et al., 2014). This concept was introduced in (YE; KEOGH, 2011) and was explored and improved in (MUEEN; KEOGH; YOUNG, 2011; HILLS et al., 2014; ZALEWSKI et al., 2016). However, *shapelet analysis* cannot be directly applied to trajectory data, because it is defined for only one variable over time, and trajectory data have two variables to represent the space over time ($x$ and $y$) and other variables to represent other semantic dimensions.

The scope of this chapter is limited to explore trajectories represented by multiple dimensions over time, such as spatial location and/or semantics. In Section 3.1, we introduce basic definitions. In Section 3.2, we describe the method MOVELETS, the algorithms for discovering trajectory movelets and the complexity analysis. Section 3.3 presents the experimental evaluation of our method over six raw trajectory datasets. In Section 3.4, we present the final considerations of this chapter, including drawbacks and limitations of our proposal.

## 3.1 BASIC DEFINITIONS

The MOVELETS method is based on the distance between subtrajectories. The general idea is to find subtrajectories of a class that are very close to a large number of trajectories of the same class and far from trajectories of other classes, characterizing a representative pattern of a given class. So, the key issue is to find these subtrajectories and their distances from other trajectories. To compute these distances, we need first to define how to measure the distance between two trajectory elements. This distance may consider one or more dimensions, since an element may have several information, such as spatial location, time, speed, acceleration, direction, etc. For that reason, we formally define the concept of *distance between elements*.

**Definition 3.1. Distance between two trajectory elements**. Given two trajectory elements $e_i$ and $e_j$ represented by $D$ dimensions, the distance between these trajectory elements $dist_e(e_i, e_j)$ must consider their distance in each dimension, and combine them into a non-negative value that respects the property of symmetry, $dist_e(e_i, e_j) = dist_e(e_j, e_i)$.

The idea behind Definition 3.1 is to allow the use of a distance function for each element dimension (such as spatial location, time, speed, acceleration, direction, and so on)

and combine them into a unique distance value. In other words, the distances in each dimension are computed and then encapsulated in a single distance value. Suppose two trajectories $P = \langle p_1, p_2, p_3, p_4 \rangle$ and $Q = \langle q_1, q_2, q_3 \rangle$, where each point is represented by $l$ dimensions. Figure 3.1 shows these two trajectories, where the dashed line indicates the distance between two trajectory elements. More details about how we compute the distance between two trajectory elements is given in Appendix 1.

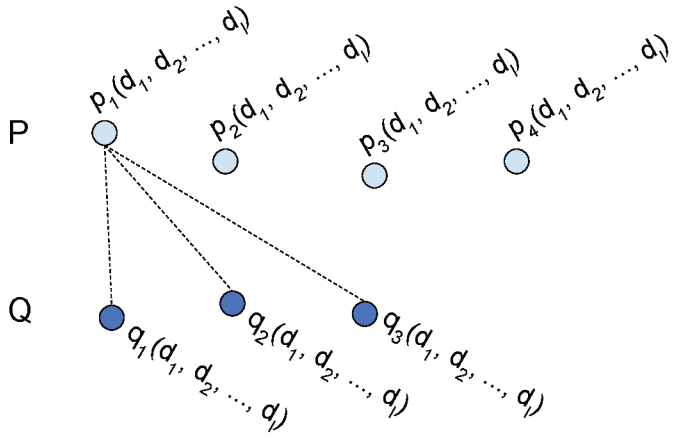



Figure 3.1 – Example of distance between trajectory elements $p$ and $q$ with $l$ dimensions.

Figure 3.1 shows, from left to right, the distances of three pairs of trajectory elements, $dist_e(p_1, q_1)$, $dist_e(p_1, q_2)$, and $dist_e(p_1, q_3)$, respectively. An example of a distance function that encapsulates the distances between all dimensions of two trajectory elements is given in Equation 3.1, where $dist_{e_k}$ is the distance between two trajectory elements at dimension $k$.

$$dist_e(p_1, q_1) = \sum_{i=1}^{l} dist_{e_k}(p_1, q_1) \tag{3.1}$$

The distance between two trajectory elements is fundamental, because it is used to compute the distance between two subtrajectories of equal length. The distance between two subtrajectories is given in Definition 3.2.

**Definition 3.2. Distance between two subtrajectories of equal length**. Given two subtrajectories $s_1$ and $s_2$ of equal length, $dist_s(s_1, s_2)$ computes the distance between their sequential elements, returns a non-negative value and respects the property of symmetry, $dist_s(s_1, s_2) = dist_s(s_2, s_1)$.

Considering two subtrajectories $s_1 = \langle p_1, p_2, ..., p_w \rangle$ and $s_2 = \langle q_1, q_2, ..., q_w \rangle$ of length $w$, their distance is given by the sum of the distances of all their elements, according to Equation 3.2.

$$dist_s(s_1, s_2) = \sum_{i=1}^{w} dist_e(p_i, q_i) \tag{3.2}$$

Figure 3.2 shows the comparison between subtrajectory $\langle p_1, p_2 \rangle$ of $P$ and the subtrajectories $\langle q_1, q_2 \rangle$ and $\langle q_2, q_3 \rangle$ of $Q$, where the dashed line indicates the distance between

subtrajectories. In this Figure, from left to right, the distances between the subtrajectories are $dist_s(\langle p_1,p_2\rangle,\langle q_1,q_2\rangle)$ and $dist_s(\langle p_1,p_2\rangle,\langle q_2,q_3\rangle)$, respectively. More details about how we compute the distance between subtrajectories can be found in Appendix 1.

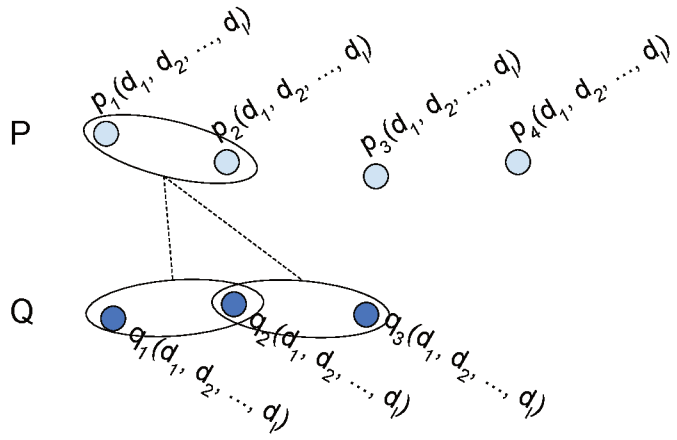



Figure 3.2 – Distance between subtrajectories of equal length.

Finding the part of a trajectory that is the most similar in relation to a subtrajectory is another essential part of our method, to know if the subtrajectory is more similar to trajectories of one class than trajectories of other classes. The most similar subtrajectory of a trajectory $T$ to a subtrajectory $s$ is called *best alignment*, and is a subtrajectory $r$ of $T$ with the minimum distance to $s$. This comparison is given in Definition 3.3.

**Definition 3.3. Distance between trajectory and subtrajectory.** Given a trajectory $T$ and a subtrajectory $s$ of length $w = |s|$, the distance between $T$ and $s$ is the best alignment of $s$ in $T$, which is defined by $g_T^s = min(dist_s(s,r) \mid r \in S_T^w)$, where $S_T^w$ is the set of all subtrajectories of length $w$ into $T$, and $min()$ returns the *distance value* of the *best alignment* between $s$ and all subtrajectories in $S_T^w$.

To better understand the concept of best alignment, let us consider the example in Figure 3.2, where there are only two possible alignments of the subtrajectory $\langle p_1,p_2\rangle$ to the trajectory $Q$. If the distance from the subtrajectory $\langle p_1,p_2\rangle$ and to the first subtrajectory in $Q$ is greater than to the second subtrajectory, i.e. $dist_s(\langle p_1,p_2\rangle,\langle q_1,q_2\rangle) > dist_s(\langle p_1,p_2\rangle,\langle q_2,q_3\rangle)$, then the best alignment of $\langle p_1,p_2\rangle$ with $Q$ is the subtrajectory $\langle q_2,q_3\rangle$.

Considering a trajectory classification problem of $n$ trajectories of length at most $m$, the number of possible subtrajectories of any length is at most $(n \times (m+m^2)/2$. By representing trajectories using all the subtrajectories as features, the induction of classification models is impracticable, because of the relation between instances and attributes. So, the selection of only the most *relevant subtrajectories* is necessary. We define a *movelet candidate* in Definition 3.4.

**Definition 3.4. Movelet Candidate**. A *movelet candidate* $\mathcal{M}$ from a subtrajectory $s$ is represented by a tuple $(T, start, length, G, score, sp)$, where:

- $T$ is the trajectory that origins the subtrajectory $s$;
- $start$ is the position in $T$ where the subtrajectory begins;

- *length* is the size of the subtrajectory;
- $G$ is a set of pairs $(g_{T_i}, class_{T_i})$, where $g_{T_i}$ is the distance value of the best alignment of $s$ into a trajectory $T_i$ (Definition 3.3) and $class_{T_i}$ is the class label of $T_i$;
- *score* is a relevance score of the movelet candidate, based on some criteria detailed later;
- *sp* is a distance value, called *split point*, used to measure the candidate relevance.

To evaluate the relevance of each candidate we use the set of distance values in $G$ and the concept of *orderline*. For example, let us consider the four trajectories shown in Figure 3.3, where $T_1$ and $T_2$ are of class $A$, and $T_3$ and $T_4$ are of class $B$. For the sake of simplicity we consider a single dimension. Also, let us consider, a *movelet candidate* $\mathcal{M}_1$ from $T_1$. The distance value of the best alignment between $\mathcal{M}_1$ and $T_1$ is zero, because there is a subtrajectory of equal length in $T_1$ that is identical to $s$, since this subtrajectory belongs to $T_1$. The shadow between the movelet candidate $\mathcal{M}_1$ and the trajectories in the figure shows the distance of the *best alignment* between $\mathcal{M}_1$ (subtrajectory in blue) and trajectories $T_2$, $T_3$, and $T_4$.

The distance value between $s$ and each trajectory is represented as an *orderline*. An *orderline* is a visual representation of the set of distances, where the values are sorted in ascending order. The distances $g_{T_1}, g_{T_2}, g_{T_3}, g_{T_4}$ in Figure 3.3 are displayed in an *orderline* from 0 (left) to $+\infty$ (right).

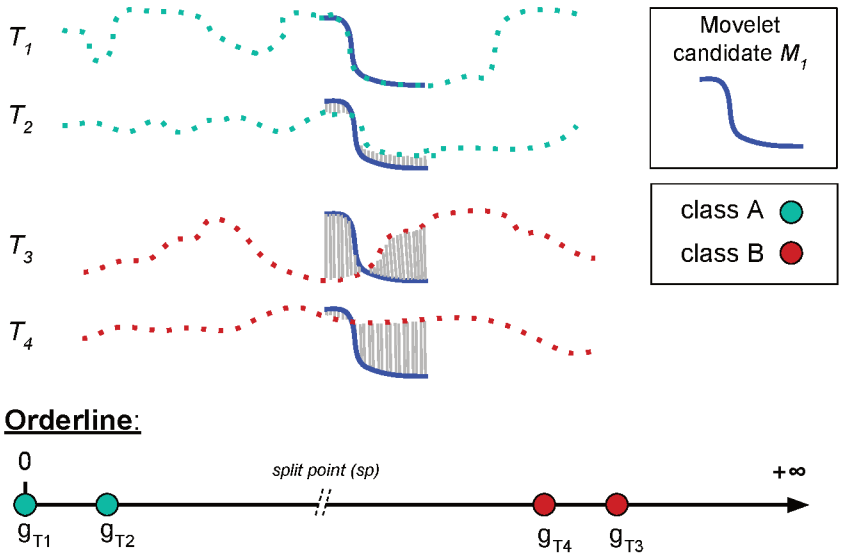



Figure 3.3 – Example of a candidate *orderline*.

One way to measure the *relevance* of a movelet candidate is to define a *split point sp* to divide the *distance orderline* into two disjoint subsets of distances, *left* and *right*, to separate the trajectories that are closer and farther to the movelet candidate. In Figure 3.3 the classes are well separated in the middle of distance values $g_{T_2}$ and $g_{T_4}$, where the subset on the *left* contains the distances of $\mathcal{M}_1$ to trajectories of class $A$ and the subset on the *right* contains the distances to trajectories of class $B$. In order to compute the relevance of a subtrajectory based on the orderline and the split point $sp$, we use the *Information Gain*. Any function can be used to

measure this relevance, but in this work we use the information gain, as described in Equation 3.3, where $prob_{c_i}$ is the probability of class $c_i$ from the classification problem, and $G_{left}$ and $G_{right}$ are the subset of $G$ where distance values are less or equal than the split point, $G|g_{Ti} \leq sp$ and grater than the split point,$G|g_{Ti} > sp$, respectively.

$$\begin{aligned} InformationGain(G,sp) &= Entropy(G) - Entropy(G|sp) \\ Entropy(G) &= -\sum_{c_i} prob_{c_i} * log_2\ prob_{c_i} \\ Entropy(G|sp) &= \frac{|G_{left}|}{|G|}\ Entropy(G_{left}) + \frac{|G_{right}|}{|G|}\ Entropy(G_{right}) \end{aligned} \tag{3.3}$$

In real scenarios with many trajectories, in general, there is not a single subtrajectory and split point that solves the problem. So the movelet candidate relevance is related to find a split point that better separates the classes and maximizes the number of distances of a class on the left side of the orderline. Based on the concept of *relevance* we define a *movelet* as in definition 3.5.

**Definition 3.5. Movelet**. Given a trajectory $T$ and a movelet candidate $M^s$ from $s \in T$, the candidate $M^s$ is a *movelet* if for each candidate $M^r$ from $r \in T$ that overlaps $s$ in at least one element, the $M^s$ has greater relevance score than $M^r$, i.e., $M^s.score > M^r.score$.

In other words, a candidate is a *movelet* if there is no other candidate overlapping it with more relevance. In addition, a *movelet* is a subtrajectory which trajectories of the same class pass near to the subtrajectory and trajectories of the other classes pass far to the subtrajectory. Based on the previous definitions, in the following section we present the proposed method to discover movelets for trajectory classification.

## 3.2 THE MOVELET METHOD

We propose a three step method, called MOVELETS, for discovering relevant subtrajectories: (1) *movelet* discovery; (2) *movelet* pruning; and (3) *movelet* transformation. Hereafter we refer to MOVELETS as the method and *movelets* as the relevant subtrajectories.

### 3.2.1 Step 1 – Movelet Discovery

The movelet discovery step consists of exploring all subtrajectories (movelet candidates) from a trajectory training set and selecting only the most relevant, i.e., the *movelets*. This step is detailed in Algorithm 1, that has as the unique input the trajectory training set **T**, without any parameter. The output is the set of *movelets*.

Algorithm 1 finds for each trajectory in the training set **T** the *movelets* (lines 2 to 23). The function $ComputeElementDistanceArray()$ computes all distances between each trajectory element in $T$ and all elements in the training set **T**, according to Equation 3.1, and stores

**Algorithm 1:** MOVELETDISCOVERY

```
   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 ← ComputeElementDistanceArray (T, T) ;
 5     for subtrajectory length w from 2 to T.length do
 6         A_w ← ComputeSubtrajectoryDistanceArray (T, T, A_{w-1}, A_1, w) ;
 7         for position j from 1 to (T.length − w + 1) do
 8             for trajectory i from 1 to |T| do
 9                 minDistance ← minvalue(A[i, j, ..]) ;
10                 g_{T_i} ← sqrt(minDistance/w) ;
11                 G[i] ← (g_{T_i}, class_{T_i}) ;
12             end
13             relevance ← AssessRelevance(G) ;
14             score ← relevance.score ;
15             sp ← relevance.splitPoint ;
16             M ← MoveletCandidate(T, j, w, G, score, sp) ;
17             candidates ← candidates ∪ M ;
18         end
19     end
20     SortByRelevance (candidates) ;
21     RemoveSelfSimilar (candidates) ;
22     movelets ← movelets ∪ candidates ;
23 end
24 return movelets
```

them into a 3-dimensional array of distances $A_1$ (line 4). Each value in $A_1[i, j, k]$ is the distance between the trajectory element of $T$ at position $j$ and the trajectory element of $T_i$ at position $k$. The array $A_1$ is precomputed in order to perform this computation only once. After that, the algorithm explores all subtrajectory lengths, one by one (lines 5 to 19). For a subtrajectory length $w$, the function *ComputeSubtrajectoryDistanceArray*() computes the distance array $A_w$ using the distance array computed for subtrajectory lengths $(w-1)$ and 1, represented by $A_{w-1}$ and $A_1$, respectively (line 6). This function is detailed in Section 3.2.1.1. The array $A_w$ contains the sums of the squares of element distances between all subtrajectories of equal length. Once $A_w$ is calculated, for each subtrajectory in $T$ of length $w$, the algorithm discovers the *best alignment* between the subtrajectory and trajectories in **T**, assesses the relevance, and adds the subtrajectory into the *movelet candidate set* (lines 7 to 18). In this loop the algorithm finds in each trajectory $T_i \in \mathbf{T}$ the *best alignment* of each subtrajectory in $T$ of size $w$ (lines 8 to 12). The *best alignment* is calculated based on $A_w$ and consists of the minimum value of the 1-dimensional array $A_w[i, j, ..]$, that contains the sums of the squares of element distances of all possible alignments between the subtrajectory and trajectory $T_i$ (line 9). Once the minimum distance value is performed, it is normalized (line 10) and stored into the distance vector $G$

along the class of $T_i$ (line 11). After computing the distance vector $G$, the algorithm assesses the relevance of the subtrajectory by the function *AssessRelevance* (line 13) and calculates both the *score* and the *split point* (lines 14 and 15). This function is detailed in Section 3.2.1.2. Then, it defines the *movelet candidate* **M** and adds it into the candidate set (line 17). Following the external loop, it sorts the trajectory movelet candidates by their relevance and removes those *self similar* (lines 20 to 21). Two candidates are *self similar* if they are overlapping on at least one element and the algorithm preserves the highest relevance candidate. Finally, it adds the remaining candidates to the *movelets* set.

In the following sections we detail the functions *ComputeSubtrajectoryDistanceArray*() and *AssessRelevance*(), implemented using a dynamic programming strategy.

#### *3.2.1.1 Computing Subtrajectory Distances*

The function *ComputeSubtrajectoryDistanceArray*() consists of computing the distance between all subtrajectories of length $w$ in $T$ and all subtrajectories in the set **T** with the same length. To perform that efficiently the algorithm uses the subtrajectory distance array $A_{w-1}$, calculated for length $(w-1)$, and the element distance values in $A_1$, to compute the subtrajectory distance values for length $w$. This function is detailed in Algorithm 2, that has as inputs the trajectories $T$ and $T_i$, the arrays $A_{w-1}$ and $A_1$, and the subtrajectory length $w$. The output of this algorithm is a new array $A_w$ containing the subtrajectory distance values for length $w$.

**Algorithm 2:** *ComputeSubtrajectoyDistanceArray*

```
   input : T, // a trajectory
           T, // the set of trajectories
           A_{w-1}, // array of subtrajectory distances for length w-1
           A_1, // array of element distances
           w // subtrajectory length
   output: A_w // array of subtrajectory distances for length w
 1 A_w ← ∅ ;
 2 for each trajectory i in (1,n) do
 3     for position j from 1 to (T.length − w + 1) do
 4         for position k from 1 to (T_i.length − w + 1) do
 5             distance_{w-1} ← A_{w-1}[i, j, k] ;
 6             elementDistance ← A_1[i, (j + w − 1), (k + w − 1)] ;
 7             distance_w ← distance_{w-1} + elementDistance ;
 8             A_w[i, j, k] ← distance_w ;
 9         end
10     end
11 end
12 return A_w
```

Algorithm 2 calculates for each trajectory $T_i \in \mathbf{T}$ the distance between all subtrajectory

distances of length $w$ between all subtrajectories in $T_i$ and all subtrajectories in $T$ and stores the distance values in a 3-dimensional array, called $A_w$ (lines 2 to 11). The algorithm explores each subtrajectory at starting position $j$ of trajectory $T$ (lines 3 to 10). For each subtrajectory in $T$ it explores all the subtrajectories in $T_i$ of the same length, from starting position $k$ (lines 4 to 9). In the internal loop the algorithm gets the previously calculated distance value between the subtrajectory of $T$ started at position $j$ of length $(w-1)$ and the subtrajectory of $T_i$ started at position $k$ of the same length (line 5). After that, it gets the element distance between the next trajectory elements of those subtrajectories (line 6). Once the previous subtrajectory distance and the element distance are gotten, it adds both the distance values for calculating subtrajectory distance value of length $w$ (line 7). This distance value corresponds to the distance between the subtrajectory of $T$ started at position $j$ of length $w$ and the subtrajectory of $T_i$ started at position $k$ of the same length and is stored in $A_w[i, j, k]$ (line 8).

Once the subtrajectory distance is computed for a subtrajectory of length $w$ we can find the best alignment of a subtrajectory starting at the $j$-th position of $T$ in trajectory $T_1$. Therefore, from the array $A_w$ between $T$ and each trajectory $T_i \in \mathbf{T}$ we can obtain the best alignment between any subtrajectory of length $w$ in $T$ and each trajectory $T_i \in \mathbf{T}$. Based on these best alignments the method assess the relevance of subtrajectories.

#### *3.2.1.2 Computing Subtrajectory Relevance*

The function *AssessRelevance*() in main MOVELETS Algorithm 1 has the objective to compute the relevance score of a subtrajectory $s$ for the classification problem, and the split point distance for the subtrajectory $s$ to decide which trajectories are close to the subtrajectory $s$.

The classical approach to measure the relevance of a *shapelet in time series* consists of the maximum information gain (YE; KEOGH, 2011; HILLS et al., 2014). However, this approach can return high split point values in order to achieve the maximum information gain, increasing the movelet variability. To avoid this problem, we propose a function that consists of the maximum information gain keeping the left side of the distance *orderline* pure, that we called *Left Side Pure* (LSP). The term *pure* indicates that all distances in the left side are of the same class. Algorithm 3 describes this process. This algorithm has as input the set of pairs $G$ of a subtrajectory, that contains the distance value $g_{T_i}$ of the best alignment of the subtrajectory in each trajectory $T_i$ in $\mathbf{T}$ and the class of $T_i$, in the form $(g_{T_i}, class_{T_i})$.

Algorithm 3 starts sorting the set of pairs $G$ in ascending order of distance values and stores the result in a vector, called *orderline* (line 1). In the *orderline* the $j$-th element contains the distance and the class of the $j$-th nearest trajectory in the set $\mathbf{T}$ to the subtrajectory, represented by $orderline[j].v$ and $orderline[j].class$, respectively. The first element of the *orderline* corresponds to the trajectory that origins the subtrajectory, because the best alignment between them has distance value equal to zero. The class of the trajectory that origins the subtrajectory is called *targetClass* (line 2). From the second element of the *orderline* the algorithm increases $i$ while the $i$th element has the same *class* of the *targetClass* (lines 3-6). Once the loop ends, the

**Algorithm 3:** *AssessRelevance*

**Input :** $G$ // the set of pairs $(g_{T_i}, class_{T_i})$
**Output:** *relevance* // the relevance of the subtrajectory

1 $orderline \leftarrow sort\ G\ in\ ascending\ order\ of\ g's$ ;
2 $targetClass \leftarrow orderline[1].class$ ;
3 $j = 2$ ;
4 **while** $orderline[j].class = targetClass$ **do**
5 $\quad j \leftarrow j+1$ ;
6 **end**
7 $sp \leftarrow \dfrac{orderline[j].v + orderline[j-1].v}{2}$;
8 $infogain \leftarrow InformationGain\ (orderline, sp)$;
9 $relevance.splitPoint \leftarrow sp$ ;
10 $relevance.score \leftarrow infogain$ ;
11 **return** $relevance$

algorithm calculates the *split point* and the respective information gain (lines 7 to 8). Finally, it stores the *split point* and the information gain value as the score into *relevance* (lines 9 to 10).

#### *3.2.1.3 Movelet Discovery Demonstration*

Figure 3.4 shows a demonstration of *movelets* discovery for a Hurricanes dataset with REDtwo classes: scale 2 in color red and scale 3 in color green. Figure 3.4(a) shows the *movelets* found for each class. Figure 3.4(b) shows an example of *movelet* of class green and focuses in the subtrajectory area (rectangle in Figure 3.4(b), where the relevant subtrajectory is in blue and the green trajectories are those passing near the *movelet*. Figure 3.4(c) shows the distance *orderline* for the *movelet*, where the vertical line is the split point that separates the closer trajectories on the *left side* and the more distant trajectories on the *right side*. Note that by assessing the relevance using LSP the algorithm keeps on the *left side* of the *ordeline* only distances to trajectories of the same class. Thus, hurricane trajectories that passing near to the movelet tends to be of scale 3 (green) and trajectories passing far to movelet tends to be of class 2 (red). Figure 3.4(d) shows the subtrajectories at the *left side* of the *orderline* in green and the *movelet* in blue. The black points in Figure 3.4(b) indicate the trajectory element where the trajectory starts and in Figure 3.4(d) the trajectory element where the subtrajectories start.

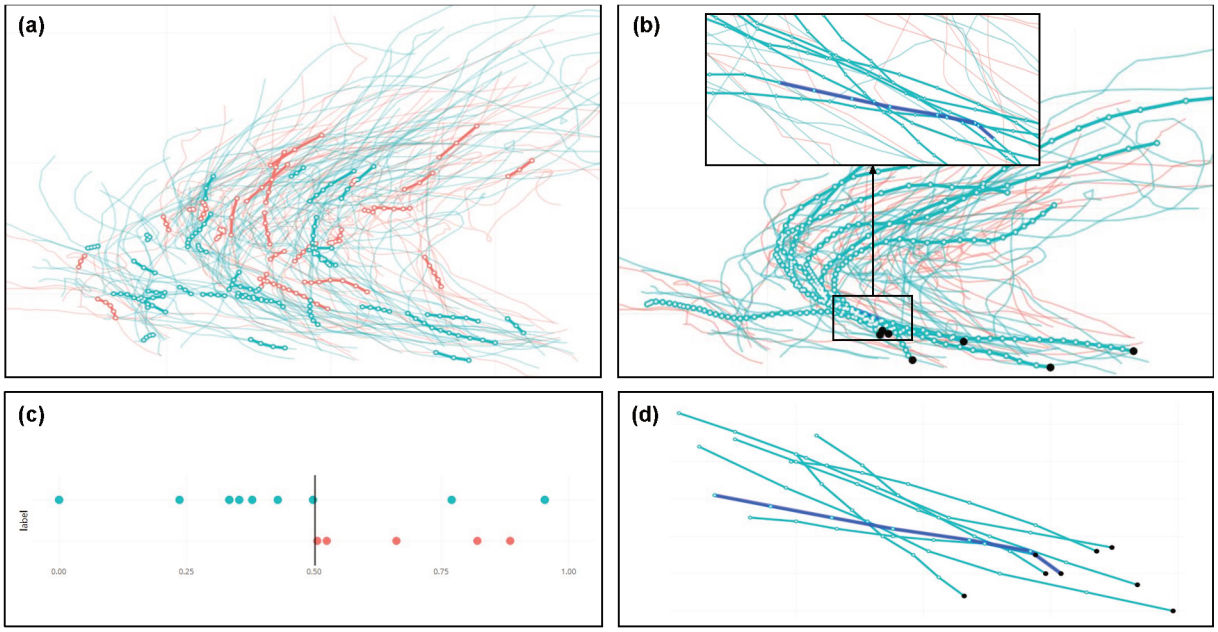



Figure 3.4 – Demonstration of movelets discovery in a Hurricane dataset.

### 3.2.2 Step 2 – Movelet Pruning

The *movelets* found by Algorithm 1 are the most relevant subtrajectories. However, the number of *movelets* found is $O(n \times m)$ (where $n$ is the number of trajectories and $m$ is the length of the longest trajectory), and this is not good for inducing classification models, because of the relation between the number of instances (that are the trajectories) and attributes (*movelets*). To reduce the number of *movelets*, the *pruning step* removes redundant *movelets*. We consider a *movelet* as redundant if the trajectories on the left side of the *ordeline* were already covered by other *movelets* with greater relevance score. To find the trajectories covered by a movelet $\mathcal{M}$ our method looks at the set of pairs $(g_{T_i}, class_{T_i})$ stored in $\mathcal{M}.G$ and finds each trajectory $T_i$ where the distance values $g_{T_i}$ is less or equal to the split point $\mathcal{M}.sp$. Algorithm 4 describes this step.

**Algorithm 4:** *Movelet Pruning*

**Input** : **M** // movelets sorted in descending order of relevance score
**Output:** **M**′ // non-redundant movelets

$\mathbf{M}' \leftarrow \emptyset$ ;
$CoveredT \leftarrow \emptyset$ ;
**for** *each movelet* $\mathcal{M}$ *in* **M do**
  $CoveredByMovelet \leftarrow Left\,side\,of\,\mathcal{M}.G$;
  **if** $(CoveredByMovelet \cap CoveredT) \neq \emptyset$ **then**
    $\mathbf{M}' \leftarrow \mathbf{M}' \cup \mathcal{M}$;
    $CoveredT \leftarrow CoveredT \cup CoveredByMovelet$;
  **end**
**end**
**return** **M**′

Algorithm 4 starts initializing the set of non-redundant *movelets* $\mathbf{M}'$ and the set of covered trajectories *CoveredT* (lines 1 and 2). It analyzes all *movelet* $\mathcal{M} \in \mathbf{M}$ one by one in descending order of relevance score (lines 3 to 9). Then, it finds the trajectories on the left side of the distance *orderline* of *movelet* $\mathcal{M}$ and stores them into *CoveredByMovelet* (line 4). Only if the *movelet* $\mathcal{M}$ covered at least one new trajectory, not present in the set *CoveredT*, the algorithm adds the *movelet* $\mathcal{M}$ to the set of non-redundant *movelets* $\mathbf{M}'$, and also adds the new covered trajectories into *CoveredT* (lines 5 to 8). By considering only the non-redundant *movelets* we reduce their number to $O(n)$, because each *movelet* in $\mathbf{M}'$ covered at least one new trajectory not covered by other *movelets* with greater relevance score in $\mathbf{M}'$.

### 3.2.3 Step 3 – Movelet Transformation

The *third step* of our method consists in representing a trajectory set in a matrix using the trajectories as rows, the *movelets* as attributes, and the distance of each trajectory to each *movelet* as the attribute values. Our approach is based on the time series shapelets transformation proposed by Hills in (HILLS et al., 2014). Algorithm 5 describes this step.

**Algorithm 5:** *MoveletTransformation*

**Input :** $\mathbf{T}$ // the trajectory set
$\mathbf{M}'$ // the movelet set
**Output:** *dataset* // an attribute-value representation

1 $dataset \leftarrow \emptyset$ ;
2 **for** *trajectory* $i$ ***from*** 1 ***to*** $|\mathbf{T}|$ **do**
3  **for** *movelet* $j$ ***from*** 1 ***to*** $|\mathbf{M}'|$ **do**
4  $distance \leftarrow BestAlignment(\mathbf{M}'[j], \mathbf{T}[i])$ ;
5  $maxDistance \leftarrow MaxDistance(\mathbf{M}'[j].G)$ ;
6  $splitPoint \leftarrow \mathbf{M}'[j].sp$ ;
7  **if** $distance \leq splitPoint$ **then**
8  $normDistance \leftarrow \frac{distance}{splitPoint}$ ;
9  **else**
10  $normDistance \leftarrow 1 + \frac{distance}{maxDistance}$ ;
11  **end**
12  $dataset[i, j] \leftarrow normDistance$ ;
13  **end**
14 **end**
15 **return** *dataset*

The input of Algorithm 5 is a set of trajectories $\mathbf{T}$ and a set of *movelets* $\mathbf{M}'$. For each trajectory and each *movelet* it finds the best alignment of the *movelet* over the trajectory, according to Definition 3.3 (line 4). It normalizes the distance value to keep the values between 0 and 2, where the values in the interval $[0,1]$ are lower or equal than the *movelet* split point and in the interval $(1,2]$ are greater than the split point (lines 6 to 11). After that, it stores the

normalized distance value into the dataset (line 12). It finalizes by returning a dataset in the attribute-value format.

The idea behind the distance value transformation between 0 and 2 is to reduce the large variation among attribute values. For instance, suppose two movelets $\mathcal{M}_1$ and $\mathcal{M}_2$, where the distance values from a set of trajectories for $\mathcal{M}_1$ are between 0.00 and 0.01 and for $\mathcal{M}_2$ are between 0.00 and 1.00. Note that the interval of distance values for $\mathcal{M}_2$ is one hundred times greater than for $\mathcal{M}_1$. By performing normalization the algorithm keeps the values of both movelets in the same interval, where the values in the interval $[0,1]$ corresponds to distance values lower or equal than the split point and the values in the interval $(1,2]$ to distance values greater than the split point. For example, a normalized distance value equal to 0.20 indicates that the distance represents 20% of the split point value.

Table 3.1 shows the attribute-value visual representation of the *dataset*, where $v_{i,j}$ is the distance from trajectory $T_i$ to *movelet* $\mathbf{M}[j]$, $dataset[i,j]$, and the attribute *class* represents the class $class_{T_i}$ of the trajectory $T_i$.

| Trajectory | $\mathbf{M}[1]$ | $\mathbf{M}[2]$ | $\ldots$ | $\mathbf{M}[\lvert\mathbf{M}\rvert]$ | *class* |
|---|---|---|---|---|---|
| $T_1$ | $v_{1,1}$ | $v_{1,2}$ | $\ldots$ | $v_{1,\lvert\mathbf{M}\rvert}$ | $class_{T_1}$ |
| $T_2$ | $v_{2,1}$ | $v_{2,2}$ | $\ldots$ | $v_{2,\lvert\mathbf{M}\rvert}$ | $class_{T_2}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $T_n$ | $v_{n,1}$ | $v_{n,2}$ | $\ldots$ | $v_{n,\lvert\mathbf{M}\rvert}$ | $class_{T_n}$ |

Table 3.1 – Attribute-value representation of movelet transformation matrix.

From the attribute-value representation of trajectories we can build classifiers using several techniques, such as SVM, Decision Trees, Bayesian, Neural Networks. We can also combine the movelet transformation matrix with other global and local features. For instance, Table 3.2 shows the combination of movelets transformation with three global features: time duration ($Z^1$), traveled distance ($Z^2$), and average speed ($Z^3$). In this table the value $z_{i,k}$ corresponds to the value of the feature $Z^k$ for the $i$-th trajectory.

| Trajectory | Time duration | Distance traveled | Average speed | $\mathbf{M}[1]$ | $\mathbf{M}[2]$ | $\ldots$ | $\mathbf{M}[\lvert\mathbf{M}\rvert]$ | *class* |
|---|---|---|---|---|---|---|---|---|
| $T_1$ | $z_{1,1}$ | $z_{1,2}$ | $z_{1,3}$ | $v_{1,1}$ | $v_{1,2}$ | $\ldots$ | $v_{1,\lvert\mathbf{M}\rvert}$ | $class_{T_1}$ |
| $T_2$ | $z_{2,1}$ | $z_{2,2}$ | $z_{2,3}$ | $v_{2,1}$ | $v_{2,2}$ | $\ldots$ | $v_{2,\lvert\mathbf{M}\rvert}$ | $class_{T_2}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $T_n$ | $z_{n,1}$ | $z_{n,2}$ | $z_{n,3}$ | $v_{n,1}$ | $v_{n,2}$ | $\ldots$ | $v_{n,\lvert\mathbf{M}\rvert}$ | $class_{T_n}$ |

Table 3.2 – Attribute-value representation of movelet transformation.

The datasets created in this step are neutral for classification methods, so we can use this step to create first a trajectory training dataset in order to build a classification model, and then a trajectory test dataset to evaluate the classification model.

### 3.2.4 Complexity Analysis

The complexity of our method is dominated by Algorithm 1. In terms of memory space, it keeps storing three matrices $A_1$, $A_{w-1}$, and $A_w$ simultaneously for calculating the set distances for subtrajectories of length $w$, using $O(n \times m^2)$, where $n$ is the number of trajectories and $m$ is the length of the longest trajectory. Indeed, its stores $O(m \times log\ m)$ candidates for each trajectory. Therefore, the space complexity is $O(n \times m^2)$. In terms of time, the most costly methods are *ComputeElementDistanceArray* and *ComputeSubtrajectoryDistanceArray*, that are $O(n \times m^2)$. While the first is performed only $n$ times, the methods *ComputeSubtrajectoryDistanceArray* is performed $O(n \times m)$ times. So the overall movelet discovering process requires $O(n^2 \times m^3)$ of time complexity, which is the same complexity of time series shapelets.

The process of movelet discovering needs to be performed only once, to train the classification model. The time complexity to classify a new trajectory consist of finding the distances of the best alignment of the movelets in the trajectory and use these distances to run the classifier. To find the best alignment of $n_M$ movelets with length at most $m_M$ in a new trajectory of length $m'$ the time complexity is $O(n_M \times m_M \times m')$ and to run the classifier depends on the classification model trained.

## 3.3 MOVELET EXPERIMENTAL EVALUATION

In this section we present the experimental evaluation of the method MOVELETS. We performed two experiments. The first experiment consists of comparing the classification accuracy of MOVELETS over five datasets, among which three are of different trajectory categories (hurricanes, animals, and vehicles). These datasets were used in the experimental evaluation of previous works, so we compare our results with the same datasets in order to make a fair comparison. This experiment is presented Section 3.3.2 and shows that MOVELETS statistically outperforms the state of the art and that it is a promising strategy for domain independent trajectory classification problems.

The second experiment is an evaluation for transportation mode classification, that is one of the most common problems in trajectory classification. As several works for trajectory classification were specifically developed for the transportation mode problem, we show that MOVELETS is general enough to improve trajectory classification in specific domains. This experiment is detailed in Section 3.3.3.

All the algorithms and datasets used in the experimental evaluations are publicity available at the author website[1]. In the next section we detail the datasets used in these experiments.

[1] https://bitbucket.org/anfer86/acmsac2018_movelets/

### 3.3.1 Datasets

The first experiment is performed ov: scale 2 (red) and scale 3ergfives of three different trajectory categories: hurricanes, animals, and vehicles. These datasets were used in (PATEL et al., 2012) and are described in TaREDble 3.3 by dataset name, number of trajectories, average length, and class proportion.

Table 3.3 – Datasets description.

| Dataset | # Traj | Length avg (sd) | | Classes | |
|---|---|---|---|---|---|
| $\mathbb{D}_1$ Hurricane$_{2,3}$ | 135 | 42.11 | (17.21) | Scale 2<br>Scale 3 | (46%)<br>(64%) |
| $\mathbb{D}_2$ Hurricane$_{1,4}$ | 210 | 34.84 | (17.33) | Scale 1<br>Scale 4 | (71%)<br>(29%) |
| $\mathbb{D}_3$ Hurricane$_{0,45}$ | 354 | 27.44 | (17.03) | Scale 0<br>Scale 4,5 | (76%)<br>(24%) |
| $\mathbb{D}_4$ Animals | 102 | 146.96 | (62.51) | Elk<br>Deer<br>Cattle | (37%)<br>(30%)<br>(33%) |
| $\mathbb{D}_5$ Vehicle | 421 | 467.98 | (250.53) | Bus<br>Truck | (34%)<br>(66%) |

Datasets $\mathbb{D}_1$, $\mathbb{D}_2$, and $\mathbb{D}_3$ are related to Atlantic Hurricanes Database collected between 1950 and 2008[2], classified using the Saffir-Simpson scale from 0 to 5, where 0 is the weakest and 5 is the strongest. The problem of classifying the scale (from 0 to 5) of a hurricane based on its trajectory is very complex, because there are 6 classes to predict. So, from the entire dataset, Patel in (PATEL et al., 2012) generates three dataset for binary classification, that are less complex problems. In this thesis we used the same dataset used in (PATEL et al., 2012). Dataset $\mathbb{D}_1$ contains the trajectories of scale 2 and 3 hurricanes, $\mathbb{D}_2$ contains the trajectories of scale 1 and 4, and $\mathbb{D}_3$ trajectories of scale 0, 4 and 5.

The dataset $\mathbb{D}_4$ is related to animals movement generated during the Starkey project[3], and contains trajectories of three species observed in June 1995: elk, deer, and cattle. The number of trajectories for each specie is 38 (7,117 points), 30 (4,333 points), and 34 (3,540 points), respectively.

The dataset $\mathbb{D}_5$ contains two categories of vehicles moving around the Athens metropolitan area[4]. The trajectories were collected by school buses and trucks, and the number of trajectories for each class is 145 (66,096 points) and 276 (112,203 points), respectively.

The second experiment was performed over the GeoLife dataset (ZHENG et al., 2010) for transportation mode classification. As the original dataset used in (XIAO et al., 2017) was not provided by the authors, we generate a similar size dataset to perform the comparison. From the total of 14,718 labeled trajectories, only 9,606 have at least a record. We first took the same six classes used by (XIAO et al., 2017). Then we removed trajectories with less than 100 points and we split trajectories with time gaps grater than 300 seconds. Finally, we selected 20% of

[2] http://weather.unisys.com/hurricane/atlantic/
[3] http://www.fs.fed.us/pnw/starkey/data/tables/
[4] http://www.chorochronos.org/

trajectories from each class, 1,763, with the following class distribution *walk* (38%), *bus* (22%), *bike* (17%), *car* (9%), *subway* (6%), *taxi* (5%), and *train* (3%).

### 3.3.2 Experimental Evaluation

This experiment consists of comparing MOVELETS with state-of-the-art methods. We compare MOVELETS with (i) the results reported in (PATEL et al., 2012) for the methods RB-TB (LEE et al., 2008) and TCPR (PATEL et al., 2012); and (ii) the feature sets proposed by Dodge (DODGE; WEIBEL; FOROOTAN, 2009), Zheng (ZHENG et al., 2010), and Xiao (XIAO et al., 2017), reimplemented for this comparison and publicly available.

To generate the Profile Decomposition proposed by Dodge (DODGE; WEIBEL; FOROOTAN, 2009), the threshold value for sinuosity was defined as the mean between minimum and maximum sinuosity values. To fit the best threshold values for the features proposed by Zheng (heading change rate, stop rate and velocity change rate), we generated decision tree models for each feature, as suggested in (ZHENG et al., 2010). For MOVELETS we only use the spatial dimension, which has demonstrated to be the best dimension, and we add the same global features used in (PATEL et al., 2012): traveled distance, time duration, and average speed.

To build the models we use the algorithms Naive Bayes, C4.5, and SVM, implemented in Weka (WITTEN et al., 2016) with default parameters. To evaluate the models we use the weighted average of F-measure ($wF$), as in (PATEL et al., 2012). This average consists of the F-measure per class weighted by the class proportion, over a 5-fold cross-validation evaluation.

In this experiment we performed two evaluations. The first evaluation is a cross-validation evaluation, where we compare MOVELETS with the results reported by (PATEL et al., 2012) and the methods proposed by (DODGE; WEIBEL; FOROOTAN, 2009; ZHENG et al., 2010; XIAO et al., 2017), that we reimplemented. This experiment is shown in section 3.3.2.1. The second evaluation, detailed in section 3.3.2.2, is a holdout evaluation, where we compared MOVELETS with the methods (DODGE; WEIBEL; FOROOTAN, 2009; ZHENG et al., 2010; XIAO et al., 2017). This experiment is important for the future reproduction of the results reported.

#### *3.3.2.1 Cross-validation Evaluation Results*

Tables 3.4, 3.5, and 3.6 show the results of cross-validation evaluation for each classifier SVM, C4.5, and Bayes, respectively. The best results for each dataset are highlighted in bold.

As can be seen in the three tables, independently of the classifier (SVM, C4.5, and Bayes), MOVELETS outperformed almost all existing works in all datasets, loosing only marginally of 0.01 to the method TCPR (PATEL et al., 2012) with SVM and ending in a tie with TCPR for the classifier C4.5 over the vehicle dataset. An overall evaluation indicates

Table 3.4 – Cross-validation evaluation for SVM.

| | SVM | | | | | |
|---|---|---|---|---|---|---|
| **Dataset** | **TB-RB** | **TCPR** | **Dodge** | **Zheng** | **Xiao** | **MOVELETS** |
| $\text{Hurricane}_{2,3}$ | 0.46 | 0.55 | 0.50 | 0.50 | 0.52 | **0.75** |
| $\text{Hurricane}_{1,4}$ | 0.72 | 0.78 | 0.72 | 0.77 | 0.78 | **0.86** |
| $\text{Hurricane}_{0,45}$ | 0.71 | 0.87 | 0.85 | 0.85 | 0.86 | **0.90** |
| Animals | 0.79 | 0.89 | 0.74 | 0.79 | 0.87 | **0.97** |
| Vehicle | 0.94 | **0.99** | 0.94 | 0.84 | 0.98 | 0.98 |

Table 3.5 – Cross-validation evaluation for C4.5.

| | C4.5 | | | | | |
|---|---|---|---|---|---|---|
| **Dataset** | **TB-RB** | **TCPR** | **Dodge** | **Zheng** | **Xiao** | **MOVELETS** |
| $\text{Hurricane}_{2,3}$ | 0.53 | 0.56 | 0.47 | 0.49 | 0.60 | **0.62** |
| $\text{Hurricane}_{1,4}$ | 0.69 | 0.73 | 0.72 | 0.76 | 0.74 | **0.78** |
| $\text{Hurricane}_{0,45}$ | 0.71 | 0.83 | 0.83 | 0.83 | 0.81 | **0.85** |
| Animals | 0.80 | 0.89 | 0.74 | 0.83 | 0.81 | **0.96** |
| Vehicle | 0.94 | **0.98** | 0.85 | 0.94 | 0.92 | **0.98** |

Table 3.6 – Cross-validation evaluation for Bayes.

| | Bayes | | | | | |
|---|---|---|---|---|---|---|
| **Dataset** | **TB-RB** | **TCPR** | **Dodge** | **Zheng** | **Xiao** | **MOVELETS** |
| $\text{Hurricane}_{2,3}$ | 0.35 | 0.45 | 0.53 | 0.55 | 0.48 | **0.76** |
| $\text{Hurricane}_{1,4}$ | 0.74 | 0.79 | 0.70 | 0.70 | 0.66 | **0.86** |
| $\text{Hurricane}_{0,45}$ | 0.71 | 0.85 | 0.80 | 0.82 | 0.78 | **0.87** |
| Animals | 0.70 | 0.77 | 0.51 | 0.70 | 0.77 | **0.91** |
| Vehicle | 0.92 | 0.97 | 0.71 | 0.60 | 0.74 | **0.99** |

that from the 15 cases (3 classifiers $\times$ 5 datasets) MOVELETS wins or ties in 14 (93.3%) and looses marginally in only 1 (6.7%). We perform a statistical analysis using the *Friedman's Aligned Rank Test* (GARCÍA et al., 2010), with level of significance $\alpha = 0.05$, that results in a *p-value* $< 0.05$. Then, we perform the *post hoc test (control vs. all)* with MOVELETS as the control and we obtain all *p-values* $< 0.05$. This result indicates that the probability of the performance differences between MOVELETS and the other methods has occurred by chance is less than 5%. Therefore, in this experiment our method significantly outperforms the state of the art approaches.

#### *3.3.2.2 Holdout Evaluation Results*

The second evaluation consists in comparing MOVELETS with features used in (DODGE; WEIBEL; FOROOTAN, 2009; ZHENG et al., 2010; XIAO et al., 2017), performing a holdout evaluation, dividing each dataset into training (60%) and test (40%) sets. We also use the weighted average of F-measure over the test set as the performance metric. Tables 3.7, 3.8, and 3.9, show the results of holdout evaluation for each classifier SVM, C4.5, and

Bayes, respectively. The best results for each dataset are highlighted in bold.

For the SVM classifier our method outperforms all state-of-the-art methods in all datasets. For the C4.5 classifier our method outperforms or ties existing works in all datasets, loosing only to the work of Zheng in one dataset. MOVELETS improves the results for dataset $D_2$ from 0.81 (Dodge) to 0.85, $D_4$ from 0.76 (Zheng) to 0.93, and $D_5$ from 0.94 (Xiao) to 0.96. For the Bayes classifier our method significantly outperformed existing approaches in 4 datasets (for $D_1$ MOVELETS improves the performance from 0.56 (Zheng) to 0.60, for $D_2$ from 0.72 (Dodge) to 0.80, for $D_4$ from 0.76 (Xiao) to 0.93, and for $D_5$ from 0.81 (Xiao) to 0.97). Only for $D_3$ Zheng achieves 0.86 and MOVELETS 0.84.

An overall evaluation indicates that from the 15 cases (3 classifiers $\times$ 5 datasets) MOVELETS wins or ties 13 (86.7%) and marginally looses in 0.02 over 2 datasets (13.3%). We also perform a statistical analysis that results in a *p-value* $< 0.05$. The *post hoc test (control vs. all)* results in all the *p-values* $< 0.05$, showing that MOVELETS outperforms the state-of-the-art methods.

Table 3.7 – Holdout evaluation for SVM.

| | SVM | | | |
|---|---|---|---|---|
| **Dataset** | **Dodge** | **Zheng** | **Xiao** | **MOVELETS** |
| $\text{Hurricane}_{2,3}$ | 0.56 | 0.59 | 0.53 | **0.60** |
| $\text{Hurricane}_{1,4}$ | 0.79 | 0.75 | 0.77 | **0.85** |
| $\text{Hurricane}_{0,45}$ | 0.87 | 0.87 | 0.86 | **0.88** |
| Animals | 0.67 | 0.68 | 0.81 | **0.90** |
| Vehicle | 0.89 | 0.78 | 0.98 | **0.99** |

Table 3.8 – Holdout evaluation for C4.5.

| | C4.5 | | | |
|---|---|---|---|---|
| **Dataset** | **Dodge** | **Zheng** | **Xiao** | **MOVELETS** |
| $\text{Hurricane}_{2,3}$ | 0.39 | **0.62** | 0.51 | **0.62** |
| $\text{Hurricane}_{1,4}$ | 0.81 | 0.71 | 0.76 | **0.85** |
| $\text{Hurricane}_{0,45}$ | 0.84 | **0.85** | 0.82 | 0.83 |
| Animals | 0.67 | 0.76 | 0.74 | **0.93** |
| Vehicle | 0.73 | 0.90 | 0.94 | **0.96** |

Table 3.9 – Holdout evaluation for Bayes.

| | Bayes | | | |
|---|---|---|---|---|
| **Dataset** | **Dodge** | **Zheng** | **Xiao** | **MOVELETS** |
| $\text{Hurricane}_{2,3}$ | 0.54 | 0.56 | 0.54 | **0.60** |
| $\text{Hurricane}_{1,4}$ | 0.72 | 0.68 | 0.68 | **0.80** |
| $\text{Hurricane}_{0,45}$ | 0.76 | **0.86** | 0.81 | 0.84 |
| Animals | 0.63 | 0.64 | 0.76 | **0.93** |
| Vehicle | 0.76 | 0.47 | 0.81 | **0.97** |

### 3.3.3 Transportation Mode Classification

The works of (ZHENG et al., 2010; XIAO et al., 2017) were specifically developed for transportation mode classification, so the features used in these works solve a specific problem. We compare MOVELETS to these works using also the global features mean and standard deviation of speed and acceleration with *movelets*, since they are basic common features in transportation mode classification. We perform the experiments over the GeoLife database (ZHENG et al., 2010), the same used by state-of-the-art methods.

From the total of $1,763$ transportation mode trajectories, we separate 70% for training Random Forest models and 30% to test, to perform a holdout evaluation. We used the same train and test sets proportion used in (XIAO et al., 2017). Table 3.10 shows the classification results of F-measure for each class over the testing set for the approaches MOVELETS, Dodge, Zheng, and Xiao. The results show that Zheng features wins for transportation mode *train*, Xiao features wins for *train*, *walk*, *car* and *bike*, and MOVELETS wins for *taxi*, *bus*, and *subway*.

Table 3.10 – Transportation mode classification results.

| Class | **Dodge** | **Zheng** | **Xiao** | **MOVELETS** |
|---|---|---|---|---|
| Train | 0.85 | **0.94** | **0.94** | 0.72 |
| Walk | 0.90 | 0.90 | **0.92** | 0.88 |
| Taxi | 0.15 | 0.24 | 0.31 | **0.61** |
| Bus | 0.76 | 0.75 | 0.79 | **0.80** |
| Subway | 0.67 | 0.72 | 0.73 | **0.85** |
| Car | 0.64 | 0.77 | **0.78** | 0.71 |
| Bike | 0.84 | 0.79 | **0.88** | 0.83 |

For a further evaluation, we combine MOVELETS with the features of Dodge, Zheng, and Xiao, as shown in Table 3.11. As can be seen, the use of MOVELETS improved the results over methods Dodge and Xiao for all cases, except the ties for method Dodge to predict class *walk* and for method Xiao to predict classes *train* and *walk*. Using MOVELETS over Zheng method improved the result for classes *taxi*, *bus*, *subway* and *bike*, but present worst results for classes *train*, *walk* and *car*. The results indicate that the best F-measure values for this problem is achieved by the combination of MOVELETS with the global features used by Xiao, as it obtains the best F-measure values for all transportation modes.

We conclude that even though MOVELETS was not developed for this specific classification problem, it has demonstrated to be very robust as a new general solution for finding relevant subtrajectories for trajectory classification.

## 3.4 CONSIDERATIONS

In this chapter we proposed a new method for extracting relevant subtrajectories for trajectory classification, called MOVELETS. Our method does neither perform trajectory discretization nor partition, so not loosing important information for class discrimination. The

Table 3.11 – Using Movelets with other features.

| Class | Dodge | Dodge + MOVELETS | Zheng | Zheng + MOVELETS | Xiao | Xiao + MOVELETS |
|---|---|---|---|---|---|---|
| Train | 0.85 | **0.89** | **0.94** | 0.83 | **0.94** | **0.94** |
| Walk | **0.90** | **0.90** | **0.90** | 0.88 | **0.92** | **0.92** |
| Taxi | 0.15 | **0.68** | 0.24 | **0.62** | 0.31 | **0.71** |
| Bus | 0.76 | **0.81** | 0.75 | **0.79** | 0.79 | **0.84** |
| Subway | 0.67 | **0.86** | 0.72 | **0.84** | 0.73 | **0.86** |
| Car | 0.64 | **0.73** | **0.77** | 0.76 | 0.78 | **0.84** |
| Bike | 0.84 | **0.86** | 0.79 | **0.84** | 0.88 | **0.89** |

proposed approach is parameter-free and domain independent, what is very important since the parameter definition is a well known problem in data mining, being difficult to estimate and directly affecting the data mining results. *Movelets* are very flexible, since they can be combined with any trajectory global or local features without any need of extending or changing the method, so new trajectory features may improve even further movelet based classification. In addition, *movelets* are neutral to classification methods, are easy to visualize, to describe, to understand, and to find them in new trajectory datasets.

We presented an experimental evaluation of MOVELETS demonstrating the effectiveness of the proposed method, that largely outperformed most existing approaches for trajectory classification in almost all datasets. We also demonstrate the effectiveness of our proposal for assessing subtrajectory relevance and finding the split point orderline, called *Left Side Pure* (LSP).

Although the proposed method does not repeat distance computations to find the *movelets* for each trajectory, the number of distance computations is very high. Therefore, the time complexity is the main *drawback* of our proposal. Besides, MOVELETS also has other limitations. The first limitation is that it uses a distance function between subtrajectories that supports only subtrajectories of equal length, in order to compute the distance in linear time, $O(w)$, where $w$ is the subtrajectory length. This limitation can be solved by using a time-warp distance, such as Dynamic Time Warping (DTW), increasing this time complexity from linear to quadratic time. The second limitation is that the method uses only one distance function between trajectory elements, that encapsulates the distances of all dimensions in a single distance value. As a consequence, besides the difficulty of balancing the importance of each dimension to build this function, it may include non-discriminant dimensions in the element distance, reducing the chance to find discriminant subtrajectories as the number of dimensions increase. In this limitation we have found a new issue for studying with a great potential of exploring and discovering new relevant subtrajectories in multidimensional sequences, which is the main subject of the next chapter.

During the development of MOVELETS we also tried to use other strategies for movelet pruning, such as using correlation. In this approach we implemented Pearson and Spearman correlation to detect if two movelets have high correlation. We considered two movelets highly

correlated when their distances to trajectories are correlated. If two movelets are very correlated we only preserve the one with the highest quality. Although removing redundant movelets is an important issue, this process can result in a class of individuals represented by only one relevant subtrajectory, indicating only one point of view about the instances of this class. For that reason, we also implemented a technique to control the redundancy of the relation between trajectories and movelets, i.e., to control the minimum number of times that a trajectory needs to be covered by movelets. With that we obtain many points of view to build the classification model. For instance, a trajectory can be covered by the movelets $M_1$, $M_2$, and $M_4$, instead of only one of them. However, this approach can only be used for methods to build classifiers that deal better with redundant attributes, such as ensemble of decision trees, like Random Forest, and Neural Networks. This approach did not improve the classification accuracy of MOVELETS models.

# 4 EXPLORING DIMENSIONS FOR DISCOVERING RELEVANT SUBTRAJECTORIES

The method MOVELETS, proposed in the previous chapter, is a parameter-free method and supports multiple dimensions, such as space, time, and semantics. This method uses a distance measure between trajectory elements and considers all dimensions together for discovering the subtrajectories that discriminate the class. However, for analyzing trajectories represented by multiple dimensions, this method has two limitations. The *first limitation* is that the distance measure needs to encapsulate the distances of all dimensions in a single distance value, which requires the design of a transformation function. This situation is even more complicated in multiple aspect trajectory data, because the dimensions are heterogeneous, e.g., time, space, and semantics, where each dimension needs a specific distance measure, and designing the transformation function is very complex. This is substantially different than the analysis of other sequential multidimensional data, like time series, where all dimensions are merely numeric and, in general, the use of normalization techniques is enough for encapsulating the distances in a single value by using only one distance measure. The *second limitation* is that encapsulating all dimensions to measure the distance between trajectory elements makes it more difficult to find discriminant subtrajectories as the number of dimensions increase, because an individual must perform the same movement with respect to all dimensions.

In this chapter, we propose a new method for finding relevant subtrajectories considering multiple and heterogeneous dimensions without encapsulating all dimensions in a single distance value. The new method covers both mentioned *limitations* by using a distance measure for each dimension, for preserving the distance information of each dimension, and exploring the subtrajectory dimension combinations in order to find the best dimension combination to represent each subtrajectory.

Section 4.1 introduces basic definitions. Section 4.2 describes the proposed method MASTERMOVELETS for exploring *movelet* dimension and its complexity analysis. Section 4.3 presents the experimental evaluation of our method over three multiple aspect trajectory datasets. Section 4.4 presents the final considerations of this chapter, mentioning drawbacks and limitations of our proposal.

## 4.1 BASIC DEFINITIONS

Let us consider the definition of multidimensional trajectory and subtrajectory as in Definition 2.1 and Definition 2.2 of Section 2.1. Figure 4.1 shows an example of a multidimensional trajectory $T_1$, like generated by Foursquare users, represented by dimensions *Venue*, *Time*, and *Price*. The trajectory is denoted by a sequence of seven elements (check-ins) considering the three dimensions, as $T_1 = \langle e_1, e_2, \ldots, e_7 \rangle$, where the first element $e_1$ is $\langle \text{Home}, 07{:}00, \emptyset \rangle$, the second element $e_2$ is $\langle \text{Café}, 08{:}00, \$ \rangle$, and the last element $e_7$ is $\langle \text{Home}, 18:30, \emptyset \rangle$, where the empty set symbol ($\emptyset$) means that the information of Price is not available.

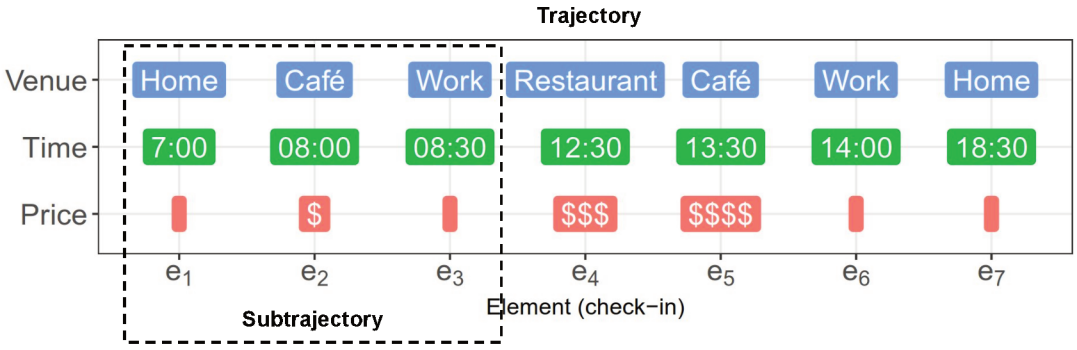



Figure 4.1 – An example of a multiple aspect trajectory and a subtrajectory.

Figure 4.1 highlights in dashed line an example of a subtrajectory $s_1 = \langle e_1, e_2, e_3 \rangle$ containing the first three multidimensional elements of $T_1$.

In order to find discriminant parts of a trajectory we need first to define the distance between two multidimensional elements. Since an element may have multiple and heterogeneous dimensions, we formally define the concept of *distance vector between two multidimensional elements*.

**Definition 4.1. Distance vector between two multidimensional elements**. Given two elements $e_i$ and $e_j$ represented by $l$ dimensions, the distance between two multidimensional elements $vdist(e_i, e_j)$ returns a *distance vector* $V = (v_1, v_2, ..., v_l)$, 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 $dist\_e_k(e_i, e_j) = dist\_e_k(e_j, e_i)$.

Note that this definition is different from the analogous definition for the method MOVELETS. Instead of a single value, the distance is represented as a vector.

Figure 4.2 shows an example of how we compute the distance between two trajectory elements: $e_1$ from $T_1$ and $e_2$ from trajectory $T_2$.

In this figure, the element $e_1$ of $T_1$ has the values $(Home, 09{:}00, \emptyset)$ and the element $e_2$ of $T_2$ has the values $(\text{Café}, 08{:}00, \$)$. In relation to the distance between the elements, for dimension *Venue* they have distance 1, because *Home* and Café are different venues[1]. For the dimension *Time* the elements have distance 60 minutes, because of the difference between the two day times, 09:00 and 08:00. And for the dimension *Price* the difference is one price unit[2]. So, the distance vector between these two elements is $V_1 = (1, 60, 1)$.

The idea behind Definition 4.1 is to allow using a distance function for each dimension and storing the distance values of all dimensions into a *distance vector* to help computing the distance between two subtrajectories of equal length, which is given in Definition 4.2.

**Definition 4.2. Distance vector between two subtrajectories of equal length**. Given two subtrajectories $s_1$ and $s_2$ both of length $w$ and represented by $l$ dimensions, $vdist\_s(s, r)$ computes the pairwise distance between the subtrajectory elements in a *distance vector* $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_l)$, where each $\mathbf{v}_k = dist\_s_k(s_1, s_2)$ is the distance value between $s_1$ and $s_2$ at dimension $k$, which is obtained by a func-

[1] In addition, another distance measure can be used in this case, for instance, considering the venue hierarchy of Foursquare.

[2] We are considering that a missing value has a value of zero.

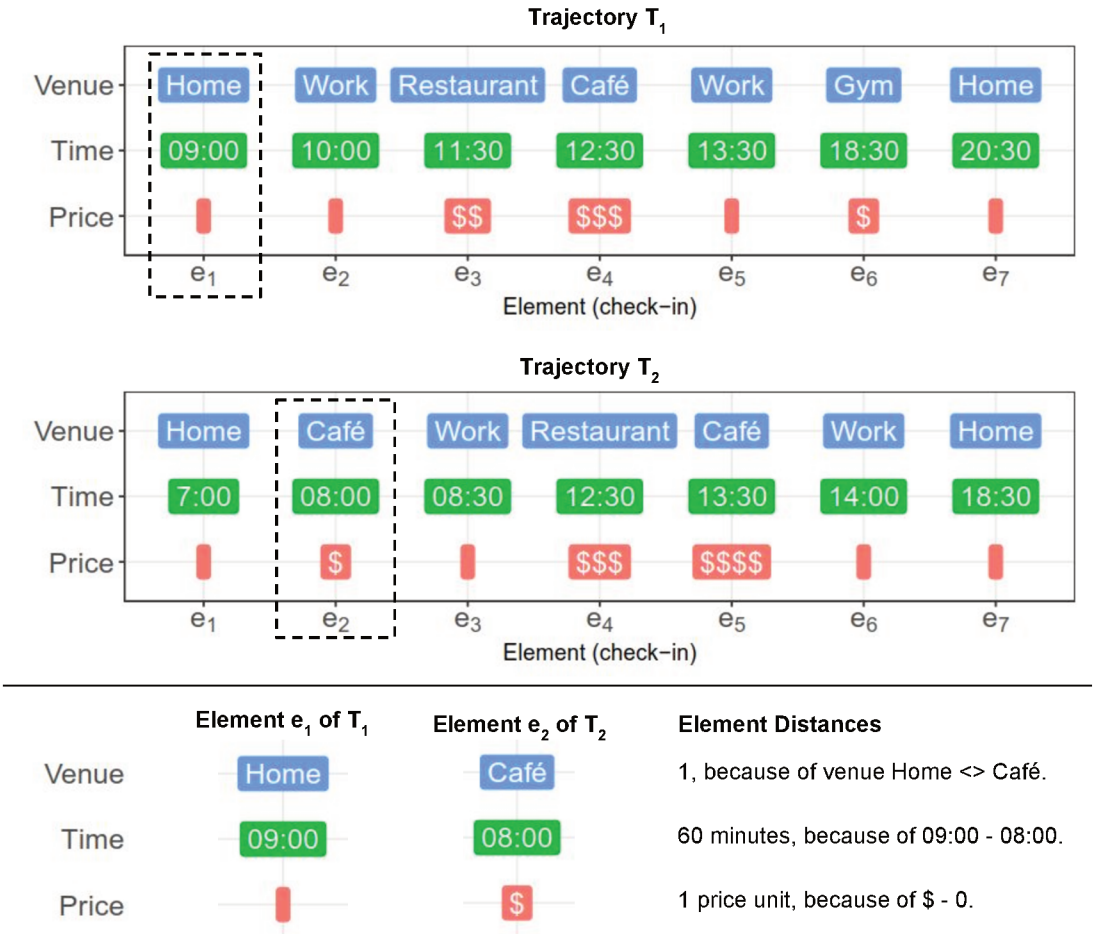



Figure 4.2 – An example of the distance between two multidimensional elements.

tion over the $w$ distances between the two subtrajectories at dimension $k$. Each distance $\mathbf{v}_k$ respects the property of symmetry $dist\_s_k(s_1, s_2) = dist\_s_k(s_1, s_2)$.

Figure 4.3 shows an example of how we compute the distance between two subtrajectories, $s_4$ of $T_1$ and $s_1$ of $T_2$, using the Manhattan distance to compute the distance between elements on the same dimension, i.e., the sum of the absolute distance values.

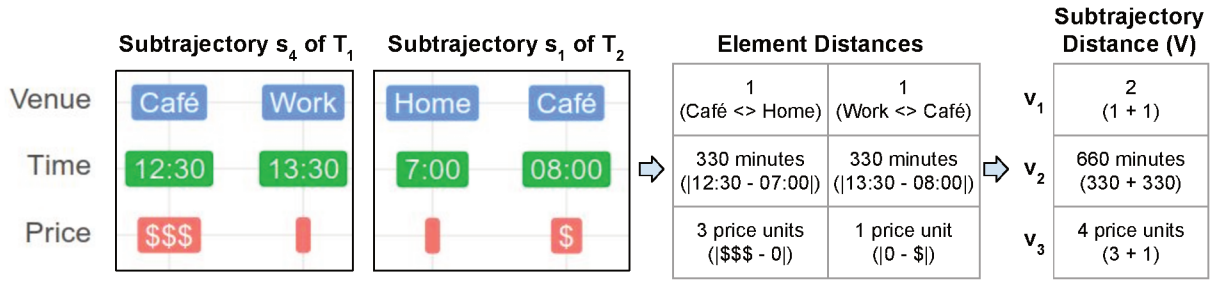



Figure 4.3 – An example of distance between two multidimensional subtrajectories.

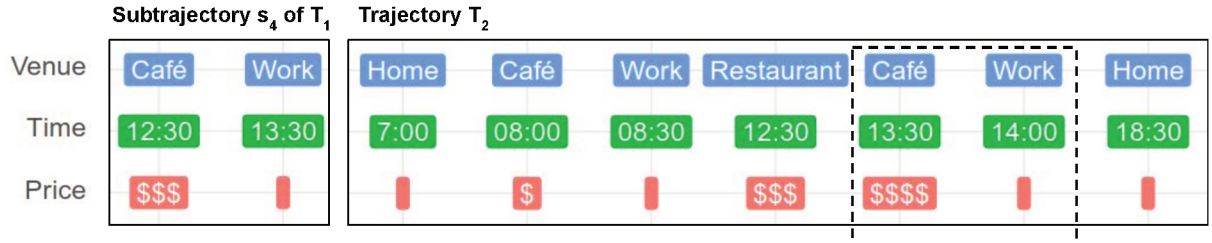
According to Figure 4.3, the element distances are calculated in a pairwise way, i.e., the first element in $s_4$ of $T_1$ with the first element in $s_1$ of $T_2$, and the second element in $s_4$ of $T_1$ with the second element in $s_1$ of $T_2$. Then, the subtrajectory distance is computed as a vector of the sum of the distances on each dimension. For instance, for dimension Venue the

distance between these subtrajectories is 2, because of both element distances have distance 1, i.e., $dist\_s_{Venue}(Cafe, Home) = 1$ and $dist\_s_{Venue}(Work, Cafe) = 1$. So, the distance between these two subtrajectories is a vector containing distance 2 for dimension *Venue*, 660 minutes for *Time*, and 4 price units for *Price*.

Finding the part of a trajectory that is the most similar in relation to a subtrajectory is another essential part of our method, to know if the subtrajectory is more similar to trajectories of one class than trajectories of other classes. The most similar subtrajectory of a trajectory $T$ to a subtrajectory $s$ is called *best alignment*, and is a subtrajectory $r$ with the minimum distance. This comparison is given in Definition 4.3.

**Definition 4.3. 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 $W_T^s = min\_vector(vdist\_s(s,r) \mid r \in s_T^w)$, where $s_T^w$ is the set of all subtrajectories of length $w$ into $T$, and $min\_vector()$ returns the *distance vector* of the *best alignment* between $s$ and all subtrajectories in $s_T^r$.

Figure 4.4 shows and example of computing the distance vector between a subtrajectory $s_4$ of $T_1$, and trajectory $T_2$. In this figure, we show the subtrajectory distance vector between $s_4$ of $T_1$ and each subtrajectory of equal length in $T_2$. We considered $V_5$ as the distance vector *best alignment*, where the subtrajectories $s_4$ of $T_1$ and $s_5$ of $T_2$ differs in 0 venues, 90 minutes, and 1 price unit. In Section 4.2.2 we describe this process in details.




**Subtrajectory Distances** — **Best alignment**

| $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ |
|---|---|---|---|---|---|
| 2 venues | 0 venues | 2 venues | 2 venues | 0 venues | 2 venues |
| 660 minutes | 570 minutes | 300 minutes | 0 minutes | 90 minutes | 390 minutes |
| 4 price units | 2 price units | 6 price units | 4 price units | 1 price unit | 6 price units |

Figure 4.4 – An example of distance between multidimensional trajectory and subtrajectory.

An important consideration is that the number of all possible subtrajectories of any length in a trajectory mining problem with $n$ trajectories of length at most $m$, and $l$ dimensions, is $(n^2 \times (m + m^2)/2 \times 2^l)$. By representing trajectories using all subtrajectories as features, the induction of classification models is impracticable, because of the relation between instances and attributes. So, the selection of only the most *relevant subtrajectories*, i.e., the *movelets*, is necessary. In order to consider the distance vectors during the movelet discovery process, we

present a new definition of movelet candidate, called *multidimensional movelet candidate*, in Definition 4.4.

**Definition 4.4. Multidimensional Movelet Candidate**. A *multidimensional movelet candidate* $\mathbb{M}$ from a subtrajectory $s$ is represented by a tuple $\mathbb{M} = (T, start, length, C, \mathbb{W}, score, \mathbb{P})$, where $T$ is the trajectory that origins 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 $(W_{T_i}, class_{T_i})$, where $W_{T_i}$ 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 $\mathbb{P}$ is a set of distance values (called split points), one per dimension, used to measure the candidate relevance.

We call this movelet candidate as *multidimensional movelet candidate* because we keep for each trajectory $T_i$ the distance vector of the best alignment of $s$ with $T_i$, while in the method presented in the previous chapter these distances of all dimensions were encapsulated in a single value.

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 is detailed in Section 4.2.3, we define a *multidimensional movelet* as given in Definition 4.5.

**Definition 4.5. Multidimensional Movelet**. Given a trajectory $T$ and a *multidimensional movelet candidate* $\mathbb{M}^s$ from $s \in T$, the candidate $\mathbb{M}^s$ is a *multidimensional movelet* if for each candidate $\mathbb{M}^r$ from $r \in T$ that overlaps $s$ in at least one element, the $\mathbb{M}^s$ has greater relevance score than $\mathbb{M}^r$, $\mathbb{M}^s.score > \mathbb{M}^r.score$.

In other words, a candidate is considered as a *movelet* if there is no other candidate overlapping it with more relevance on any dimension combination. This strategy selects the dimension combination of the candidate with highest relevance score.

## 4.2 MASTERMOVELETS: A METHOD FOR DISCOVERING RELEVANT MULTIPLE ASPECT SUBTRAJECTORIES

In this section we present the algorithm for discovering heterogenous *movelets*, called MASTERMOVELETS (Multiple ASpect TrajEctoRy Movelets). Our proposal is an extension of MOVELETS, proposed in (FERRERO et al., 2018), to explore the multiple dimensions for *movelet* discovering. In the remaining of this chapter, we use the terms *movelet* and *movelet candidate* in the context of multidimensional movelet and candidate, respectively. MASTERMOVELETS consists of exploring all the *movelet candidates* from a trajectory dataset and selecting only the *movelets*. It explores each *movelet candidate* by finding the *best alignment* of the subtrajectory to all trajectories in the dataset (process detailed in Section 4.2.2) and then computes the relevance *score* (detailed in Section 4.2.3).

**Algorithm 6:** MASTERMOVELETS

**Input :** $\mathbf{T}$ // trajectory training set
**Output:** *movelets* // set of relevant subtrajectories

$movelets \leftarrow \emptyset$ ;
**for** ***each*** *trajectory* $T$ ***in*** $\mathbf{T}$ **do**
  $candidates \leftarrow \emptyset$;
  $\mathbb{A}_1 \leftarrow ComputeElementDistanceVectors\ (T, \mathbf{T})$ ;
  **for** *subtrajectory length* $w$ ***from*** $1$ ***to*** $T.length$ **do**
    **if** $w > 1$ **then**
      $\mathbb{A}_w \leftarrow ComputeSubtrajectoryDistanceVectors\ (T, \mathbf{T}, \mathbb{A}_{w-1}, \mathbb{A}_1, w)$ ;
    **end**
    **for** *position* $j$ ***from*** $1$ ***to*** $(T.length - w + 1)$ **do**
      $R \leftarrow \emptyset$ ;
      **for** *trajectory* $i$ ***from*** $1$ ***to*** $|\boldsymbol{T}|$ **do**
        **for** *dimension* $d$ ***from*** $1$ ***to*** $|D|$ **do**
          $R[i,d,..] \leftarrow Rank(\mathbb{A}_w[i,j,d,..])$ ;
        **end**
      **end**
      $bestScore \leftarrow 0$ ;
      **for** ***each*** *dimension combination* $C$ ***in*** $C_d^*$ **do**
        $\mathbb{W} \leftarrow \emptyset$;
        **for** *trajectory* $i$ ***from*** $1$ ***to*** $|\mathbf{T}|$ **do**
          $W_i \leftarrow min$ MASTERALIGNMENT$(\mathbb{A}_w[i,j,C,..], R[i,C,..])$ ;
          $\mathbb{W} \leftarrow \mathbb{W} \cup (W_i, \mathbf{T}[i].class)$ ;
        **end**
        $relevance \leftarrow assess$ MASTERRELEVANCE$(\mathbb{W}, class_T)$ ;
        **if** $relevance.score > bestScore$ **then**
          $bestScore \leftarrow relevance.score$ ;
          $best\mathbb{P} \leftarrow relevance.\mathbb{P}$ ;
          $best\mathbb{W} \leftarrow \mathbb{W}$ ;
          $bestC \leftarrow C$ ;
        **end**
      **end**
      $\mathbb{M} \leftarrow MoveletCandidate(T, j, w, bestC, best\mathbb{W}, bestScore, best\mathbb{P})$ ;
      $candidates \leftarrow candidates \cup \mathbb{M}$ ;
    **end**
  **end**
  $SortByRelevance\ (candidates)$ ;
  $RemoveSelfSimilar\ (candidates)$ ;
  $movelets \leftarrow movelets \cup candidates$ ;
**end**
**return** *movelets*

Algorithm 6 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 $ComputeElementDistanceVectors()$, detailled in Section 4.2.1, computes the distance between all trajectory elements in $T$ and all trajectories in $\mathbf{T}$, and stores them into a 4-dimensional array, $\mathbb{A}_1$ (line 4). Each value $\mathbb{A}_1[i,j,d,k]$ 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 length $w$, the function $ComputeSubtrajectoryDistanceVectors()$, detailed in Section 4.2.1, computes the distance between the subtrajectories in $T$ and the subtrajectories in each $T_i \in \mathbf{T}$ by just adding the values of $\mathbb{A}_{w-1}$ and $\mathbb{A}_1$, and stores them into $A_w$ (line 7).

In the loop of lines 9 to 33, for each subtrajectory in $T$ of size $w$, the algorithm uses $\mathbb{A}_w$ to discover its best dimension combination, and adds it into the movelet *candidates* set. In this loop, the algorithm first computes for each subtrajectory in $T$ the distance ranking $R$ among all subtrajectories in the $i$th 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 4.2.2), 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 4.2.3). 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.

### 4.2.1 Computing Element and Subtrajectory Distance Vectors

Computing distance between trajectory elements and subtrajectories taking into account multiple dimensions in an efficient way requires using a dynamic programming strategy, to avoid repeating distance computation. The function $ComputeElementDistanceVectors()$ computes the distance between all dimensions of trajectory elements in $T$ and all elements in $\mathbf{T}$, and is detailed in Algorithm 7. This algorithm has as input the trajectory $T$ and the training set $\mathbf{T}$, and as the output a 4-dimensional array $\mathbb{A}_1$ containing all element distance values.

Algorithm 7 computes for each trajectory $T_i \in \mathbf{T}$ the distance between all trajectory elements in $T_i$ and all trajectory points in $\mathbf{T}$ and stores the distance values in a 4-dimensional

**Algorithm 7:** *ComputeElementDistanceVectors*

```
   input  : T, // a trajectory
            T // the set of trajectories
   output: A_1 // array of element distances
 1 A_1 ← ∅ ;
 2 for trajectory i from 1 to |T| do
 3 |   T_i ← T[i] ;
 4 |   for position j from 1 to T.length do
 5 |   |   for dimension d from 1 to |D| do
 6 |   |   |   for position k from 1 to T_i.length do
 7 |   |   |   |   distance ← ElementDistance(T[j], T_i[k], d) ;
 8 |   |   |   |   A_1[i, j, d, k] ← distance × distance ;
 9 |   |   |   end
10 |   |   end
11 |   end
12 end
13 return A_1
```

array, called $\mathbb{A}_1$ (lines 2 to 12). In other words, it computes the distance between all subtrajectories of size 1 for all dimensions. This is one of the most time consuming step in the whole process. The algorithm explores each trajectory point in $T$ at position $j$ (lines 4 to 11) and for the $j$-th trajectory point in $T$ it explores, for each dimension $d$, each trajectory point in $T_i$ at position $k$ (lines 5 to 10). In the most internal loop it performs the function $ElementDistance()$ between the $j$-th trajectory point in $T$ and the $k$-th trajectory point in $T_i$, represented by $T[j]$ and $T_i[k]$, respectively, at dimension $d$ (line 7). After computing the distance value, it calculates and stores the square of the distance value into $\mathbb{A}$. The 4-dimensional array $\mathbb{A}$ is indexed by $i$, $j$, $d$, and $k$, in order to store for the $i$-th trajectory the distance values between all pairs $T[j]$ and $T_i[k]$, at dimension $d$.

To exemplify, we present a running example of element distance vectors computation between two trajectories. Let us consider the trajectories $T_1$ and $T_2$ in Figure 4.2 both containing 7 elements, represented by dimensions *Time*, *Venue*, and *Price*. These trajectories belong to a trajectory training set **T**. For calculating the part of the array $A_1$ containing the element distances between $T_1$ and $T_2$, we calculate the distance between each pair of trajectory elements $T_1[j]$ and $T_2[k]$, as in Figure 4.5. Figures 4.5(a-c) show the distance values for dimensions *Time*, *Venue*, and *Price*, individually, and Figure 4.5(d) shows the 3-dimensional array $\mathbb{A}_1[2,..,..,..]$ with 7 rows and 7 columns, where $\mathbb{A}_1[2, j, d, k]$ is the distance between the $j$-th trajectory element of $T_1$ and the $k$-th trajectory element of $T_2$ at dimension $d$.

For instance, for the first element of trajectory $T_1$, represented by $(Home, 09:00, \emptyset)$, and the second element of trajectory $T_2$, represented by $(\text{Café}, 08:00, \$)$, the distance between these points at dimension *Time* is 60 minutes, that corresponds to the distance value at row $j = 1$ and column $k = 2$ in Figure 4.5(a). For dimensions *Venue* and *Price* the distances are 1 venue and 1 price unit, as in Figures 4.5(b) and (c). Therefore, the element distance vector is

| $\mathbb{A}_1$ [2,..,1,..] | | k | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| j | 1 | 120 | 60 | 30 | 210 | 270 | 300 | 570 |
| | 2 | 180 | 120 | 90 | 150 | 210 | 240 | 510 |
| | 3 | 270 | 210 | 180 | 60 | 120 | 150 | 420 |
| | 4 | 330 | 270 | 240 | 0 | 60 | 90 | 360 |
| | 5 | 390 | 330 | 300 | 60 | 0 | 30 | 300 |
| | 6 | 690 | 630 | 600 | 360 | 300 | 270 | 0 |
| | 7 | 810 | 750 | 720 | 480 | 420 | 390 | 120 |

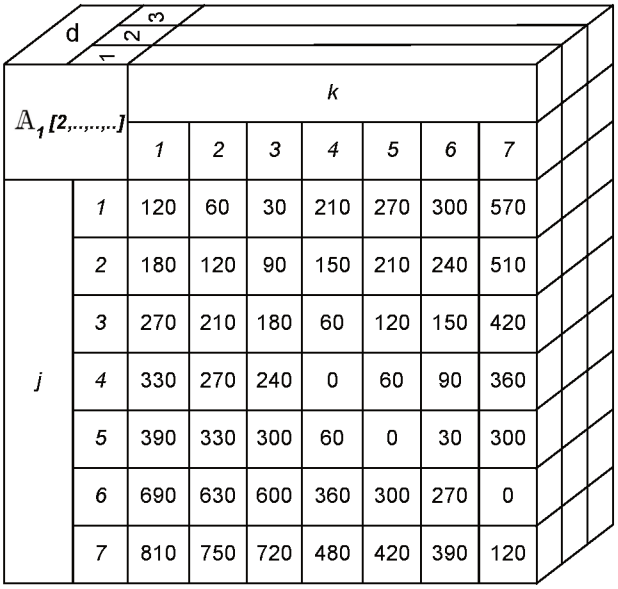
(a) Element distances for the 1st dimension, *Time*.

| $\mathbb{A}_1$ [2,..,2,..] | | k | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| j | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| | 2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| | 3 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| | 4 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| | 5 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| | 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 7 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

(b) Element distance for the 2nd dimension, *Venue*

| $\mathbb{A}_1$ [2,..,3,..] | | k | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| j | 1 | 0 | 1 | 0 | 3 | 4 | 0 | 0 |
| | 2 | 0 | 1 | 0 | 3 | 4 | 0 | 0 |
| | 3 | 2 | 1 | 2 | 1 | 2 | 2 | 2 |
| | 4 | 3 | 2 | 3 | 0 | 1 | 3 | 3 |
| | 5 | 0 | 1 | 0 | 3 | 4 | 0 | 0 |
| | 6 | 1 | 0 | 1 | 2 | 3 | 1 | 1 |
| | 7 | 0 | 1 | 0 | 3 | 4 | 0 | 0 |

(c) Element distances for the 3th dimension, *Price*.




(d) Array of distances between $T_1$ and $T_2$.

Figure 4.5 – Running example of computing element distances between two trajectories.

$(60,1,1)$. Figure 4.5(d) shows the 3-dimensional array obtained from the three 2-dimensional arrays. This array is the basis for computing the subtrajectory distance vectors between $T_1$ and $T_2$.

The function $ComputeSubtrajectoryDistanceVectors()$ consists of computing the distance between all subtrajectories of length $w$ in $T$ and all subtrajectories in the set $\mathbf{T}$ with the same length. To perform that efficiently the algorithm uses a dynamic programming strategy that uses the subtrajectory distance values calculated for length $(w-1)$ and the element distance values in $\mathbb{A}_1$ to compute the subtrajectory distance values for length $w$. This function is detailed in Algorithm 8, that has as input the arrays $\mathbb{A}_{w-1}$ and $\mathbb{A}_1$, which are the subtrajectory distance values calculated for length $(w-1)$ and the element distance values, respectively, where $w$ is the length of the subtrajectory distances to be calculated. The output of this algorithm is a new

array $\mathbb{A}_w$ containing the distance values for subtrajectory of length $w$.

**Algorithm 8:** *ComputeSubtrajectoryDistanceVectors*

```
   input : T, // a trajectory
           T, // the set of trajectories
           A_{w-1}, // array of subtrajectory distances for length w-1
           A_1, // array of element distances
           w // subtrajectory length
   output: A_w // array of subtrajectory distances for length w
 1 A_w ← ∅ ;
 2 for trajectory i from 1 to |T| do
 3     T_i ← T[i] ;
 4     for position j from 1 to (T.length − w + 1) do
 5         for dimension d from 1 to |D| do
 6             for position k from 1 to (T_i.length − w + 1) do
 7                 distance_{w-1} ← A_{w-1}[i, j, d, k] ;
 8                 distance_1 ← A_1[i, (j + w − 1), d, (k + w − 1)] ;
 9                 distance_w ← distance_{w-1} + distance_1 ;
10                 A_w[i, j, d, k] ← distance_w ;
11             end
12         end
13     end
14 end
15 return A_w
```

Algorithm 8 explores each subtrajectory at starting position $j$ of trajectory $T$ (lines 4 to 13). For each subtrajectory in $T$ and dimension $d$, it explores all the subtrajectories in $T_i$ of the same length (lines 6 to 11). In the internal loop the algorithm gets the previously calculated distance value between the subtrajectory of $T$ started at position $j$ of length $(w-1)$ and the subtrajectory of $T_i$ started at position $k$ of the same length, considering the dimension $d$ (line 7). After that, it gets the element distance between the next trajectory elements of those subtrajectories (line 8). Once the previous subtrajectory distance and the element distance are gotten, it adds both the distance values for calculating subtrajectory distance value of length $w$ (line 9). This distance value corresponds to the distance between the subtrajectory of $T$ started at position $j$ of length $w$ and the subtrajectory of $T_i$ started at position $k$ of the same length at dimension $d$ and is stored in $\mathbb{A}_w[i, j, d, k]$ (line 10).

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

### 4.2.2 Multidimensional Alignment of a Subtrajectory in 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 $min\_vector()$ in Definition 4.3 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 4.6(a) shows an example of a subtrajectory $s$ and Figure 4.6(b) 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*".

(a) Subtrajectory $s$. (b) Trajectory $T$.

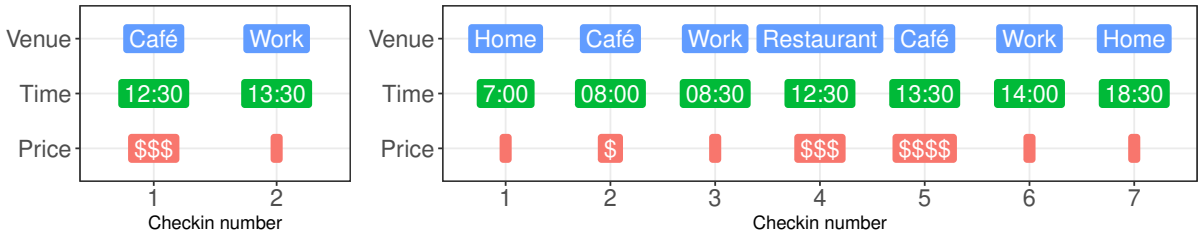



Figure 4.6 – Example of a subtrajectory $s$ (left) and a trajectory $T$ (right).

Note in Figure 4.6 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 4.7, because besides the venues sequence being the same $\langle$Café, *Work*$\rangle$, the dimensions *Time* and *Price* are also quite similar.

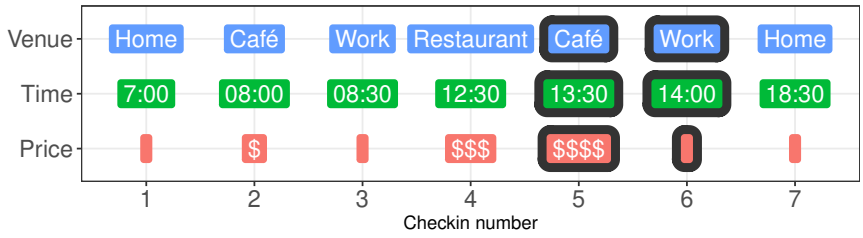



Figure 4.7 – Best subtrajectory alignment of $s$ in $T$ highlighted in the trajectory.

To find the best alignment between a subtrajectory and a trajectory, we propose the

method MASTERALIGNMENT (Multiple Apect SubTrajEctoRy Alignment), 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 $i$th-position. Table 4.2 presents the values of the distance vectors for the example in Figure 4.6. 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 $r_1 = \langle(\text{Home}, 07{:}00, \emptyset), (\text{Café}, 08{:}00, \$)\rangle$. 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.

Table 4.1 – Finding the best alignments from the distance vectors.

Table 4.2 – 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 | $\mathbf{V}_1$ | $\mathbf{V}_2$ | $\mathbf{V}_3$ | $\mathbf{V}_4$ | $\mathbf{V}_5$ | $\mathbf{V}_6$ |

Table 4.3 – 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 |

After that, we rank the distance values for each dimension. Table 4.3 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 4.3), resulting in $(5.0, 2.8, 4.5, 3.3, \underline{1.5}, 3.8)$. So, MASTERALIGNMENT considers the best alignment as the lowest average rank, that is 1.5 (underlined in Table 4.3) 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 9, that has as input two 2-dimensional arrays, $V$ and $R$, containing the distance values and the distance rankings, respectively.

Algorithm 9 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

**Algorithm 9:** MASTERALIGNMENT

**input :** $V$, // distance values of subtrajectory alignments
$R$ // distance rankings of subtrajectory alignments
**output:** $W$ // distance values of the best subtrajectory alignment

1 $Y \leftarrow \emptyset$ ;
2 $l \leftarrow |R[..,1]|$ ;
3 $posMinAvgRank = 1$ ;
4 **for** *position* $j$ ***from*** 1 ***to*** $|R[1,..]|$ **do**
5 $\quad sumRank =\leftarrow \emptyset$ ;
6 $\quad$ **for** *dimension* $d$ ***from*** 1 ***to*** $l$ **do**
7 $\quad\quad sumRank = sumRank + R[d,j]$ ;
8 $\quad$ **end**
9 $\quad avgRank = \dfrac{sumRank}{l}$ ;
10 $\quad Y[j] = avgRank$ ;
11 $\quad$ **if** $Y[j] < Y[posMinAvgRank]$ **then**
12 $\quad\quad posMinAvgRank \leftarrow j$ ;
13 $\quad$ **end**
14 **end**
15 $W \leftarrow V[..,posMinAvgRank]$ ;
16 **return** $W$

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 measure the relevance of the subtrajectory, that is detailed in the following section.

### 4.2.3 Relevance Measuring for Multidimensional Subtrajectory Candidates

The relevance of a subtrajectory is related to the number of trajectories of the same class that performs similar movement. To do that 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 putting 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; KEOGH, 2011), the Kruskal-Wallis

and Mood's Median (LINES; 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* dimensions, 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 $\mathbb{M}$ extracted from $T_1$ of class $L_1$. The set $\mathbb{W}$ of $\mathbb{M}$ (Definition 4.4) 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 4.8 shows the orderlines for $\mathbb{W}$, where each point indicates the distance value to the $i$th 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 $\mathbb{M}$ comes from $T_1$.

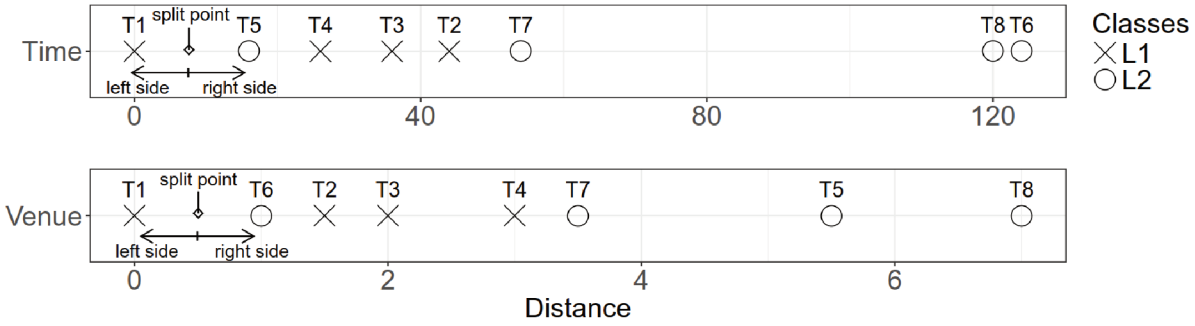



Figure 4.8 – Orderlines for dimension Time and Venue.

Figure 4.8 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 4.8 that the distance values of $T_7$ may be a good estimators 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 maximizes the relevance of the movelet candidate. The method, called MASTERRELEVANCE (Multilple Aspect SubTrajectoERy Relevance), consists of three steps and it is exemplified by Figure 4.9.

Figure 4.9(a) shows the distance values presented in Figure 4.8, in a scatter plot, where each point indicates the distances of a best alignment 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 prune *covered* points until only *non-covered* points have remained. This problem is related to multi-objective optimization problems and its is equivalent to determining the maximal vector problem in computational geometry or the Pareto optimal

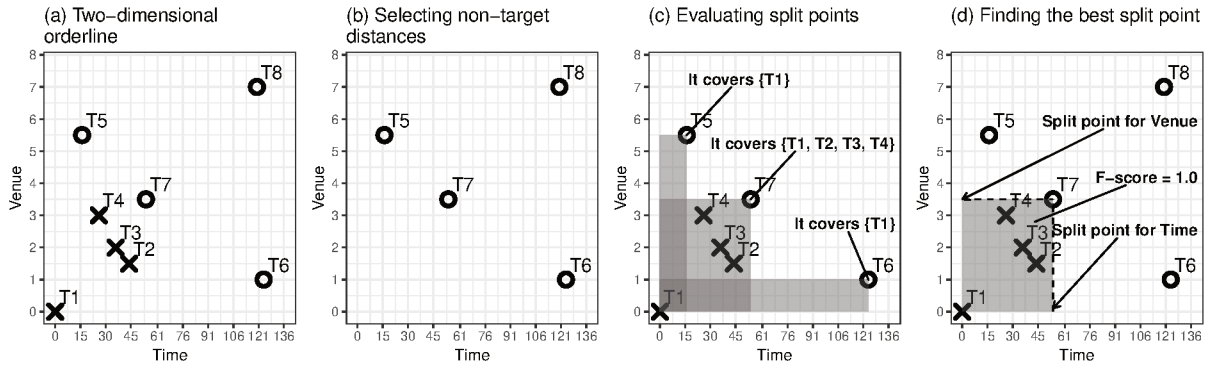



Figure 4.9 – Example of finding split points in a *multidimensional orderline*.

set (KALYVAS; TZOURAMANIS, 2017)[3]. In Figure 4.9(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 to the *number of covered points* of each class, considering both dimensions. Figure 4.9(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 points that belongs to the *target class*, and the *recall* is the proportion of *points* covered by the split points that belongs to the *target class* in relation to all the *points* of the *target class*. As shown in Figure 4.9(d), the best split points are the values defined by the point $T_7$ and the score is 1.

Algorithm 10 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 10 starts by randomly separating half of the pairs in $\mathbb{W}$, in a stratified way, for the split point finding (line 2), to avoid overfitting. The algorithm stores in $nonTargetWtrain$ all the split points in $\mathbb{W}train$ that do not correspond to the target (lines 4 to 8). Then, the algorithm removes from $nonTargetWtrain$ all the split point sets $W_j$ covered by another split point set $W_i$ (lines 10 to 16). After that, the remaining split point sets are stored in $\mathbb{P}candidates$, which contains the candidates to become the movelet split points (line 18). 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 20 to 34). After computing the statistics, the algorithm computes the F-score of each split point candidate (line 33). To finalize, it finds the split point set that achieves the maximum F-score (line 35) and stores both the split point set and the F-score value, in the variable *relevance* (lines 36 and 37).

[3] There are many other strategies in the literature to find exact and approximated solutions for this specific problem. More details can be found in (KUNG; LUCCIO; PREPARATA, 1975; VELDHUIZEN; LAMONT, 2000; MARLER; ARORA, 2004; KALYVAS; TZOURAMANIS, 2017).

**Algorithm 10:** MASTERRELEVANCE

**input :** $\mathbb{W}$, // set of distance vectors of the best alignments
*targetClass* // the target class, i.e. the movelet candidate class
**output:** *relevance* // subtrajectory relevance

// Step 1: pruning points with greater distance values
$\mathbb{W}train \leftarrow selectHalfForTraining(\mathbb{W})$ ;
$nonTargetWtrain \leftarrow \emptyset$ ;
**for** *pair* $(W_i, class_i)$ ***of*** $\mathbb{W}train$ **do**
  **if *not*** $class_i = targetClass$ **then**
    $nonTargetWtrain \leftarrow nonTargetWtrain \cup \{W_i\}$ ;
  **end**
**end**
// Step 2: evaluating split points
**for** $W_i$ ***in*** $nonTargetWtrain$ **do**
  **for** $W_j$ ***in*** $nonTargetWtrain$ **do**
    **if** $W_j$ *isCoveredBy* $W_i$ **then**
      $nonTarget\mathbb{W} \leftarrow nonTarget\mathbb{W} - \{W_j\}$ ;
    **end**
  **end**
**end**
// Step: choosing the best split point
$\mathbb{P}candidates \leftarrow nonTargetWtrain$ ;
$statistics \leftarrow \emptyset$
**for** $W_i$ ***in*** $\mathbb{P}candidates$ **do**
  **for** *pair* $(W_k, class_k)$ ***in*** $\mathbb{W}$ **do**
    **if** $W_k$ *isCoveredBy* $W_i$ **then**
      **if** $class_k = targetClass$ **then**
        $statistics[i].TruePositive$ += 1 ;
      **else**
        $statistics[i].FalsePositive$ += 1 ;
    **else**
      **if** $class_k = targetClass$ **then**
        $statistics[i].FalseNegative$ += 1 ;
      **else**
        $statistics[i].TrueNegative$ += 1 ;
  **end**
  $statistics[i].Fscore \leftarrow calculateFscore(statistics[i])$ ;
**end**
$indexBestFscore \leftarrow \arg\max_i(statistics)$ ;
$relevance.score \leftarrow statistics[indexBestFscore].Fscore$ ;
$relevance.\mathbb{P} \leftarrow \mathbb{P}candidates[indexBestFscore]$ ;
**return** *relevance*

### 4.2.4 Trajectory Attribute-value Representation

Algorithm 6 extracts from a trajectory *training set* $\mathbf{T}$ the set of *movelets* $\mathbf{M}$ of size $|\mathbf{M}|$. Then, these *movelets* are used to set up an *attribute-value* representation for the trajectory set, where the *movelets* are the *attributes* and the distances from each trajectory to each movelet are the *attribute values*. This representation can be used to build classification models (HILLS et al., 2014). Table 4.4 shows the *attribute-value* visual representation, where $z_{i,j}$ is the distance from trajectory $T_i$ to *movelet* $\mathbf{M}[j]$, and the attribute *class* represents the class of trajectory $T_i$. In this work each value $z_{i,j}$ is a binary value, that is 0 if the movelet $\mathbf{M}[j]$ *covers* the trajectory $T_i$ (considering the split points and dimensions of $\mathbf{M}[j]$), and 1 otherwise.

| Trajectory | $\mathbf{M}[1]$ | $\mathbf{M}[2]$ | ... | $\mathbf{M}[\lvert\mathbf{M}\rvert]$ | *class* |
|---|---|---|---|---|---|
| $T_1$ | $z_{1,1}$ | $z_{1,2}$ | $\cdots$ | $z_{1,\lvert\mathbf{M}\rvert}$ | $class_{T_1}$ |
| $T_2$ | $z_{2,1}$ | $z_{2,2}$ | $\cdots$ | $z_{2,\lvert\mathbf{M}\rvert}$ | $class_{T_2}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $T_n$ | $z_{n,1}$ | $z_{n,2}$ | $\cdots$ | $z_{n,\lvert\mathbf{M}\rvert}$ | $class_{T_n}$ |

Table 4.4 – Attribute-value representation of trajectories.

So we first use this representation for the trajectory *training set* $\mathbf{T}$ in order to build the classification model, and then we use the same representation for the trajectory *test set* to evaluate the classification model.

### 4.2.5 Complexity Analysis

In terms of memory space, MASTERMOVELETS (Algorithm 6) keeps storing the arrays $\mathbb{A}_1$, $\mathbb{A}_{w-1}$, and $\mathbb{A}_w$ 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 $l$ the number of dimensions. Also, it stores at most $m^2$ candidates for each trajectory. Therefore, the space complexity is $O(n \times m^2 \times l)$.

In terms of time, the algorithm MASTERMOVELETS repeats the function *Rank* by $(n^2 \times (m+m^2)/2 \times l)$ times and MASTERRELEVANCE by $(n \times (m+m^2)/2 \times 2^l)$ 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 chose to build the classifier, considering as input $n$ trajectories as samples and $O(n \times m)$ *movelets* as attributes.

Note the process of discovering movelets needs to be performed only once, to train the classification model. Then, to classify a new trajectory we need to find the distances of the best alignment of the movelets in the trajectory and use these distances to run the classifier. Considering $n_M$ movelets with at most $m_M$ trajectory elements and $l_M$ movelet dimensions in a new trajectory of length $m'$, to perform the algorithm MASTERALIGNMENT to find the best

alignment between the movelets and the trajectory the time complexity is $O(n_M \times m_M \times l_M \times m' \, log \, m')$ and to run the classifier depends on the model trained for classification.

## 4.3 CLASSIFYING MULTIPLE ASPECT TRAJECTORIES USING MASTERMOVELETS

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; MYERS; LESKOVEC, 2011) and Brightkite (CHO; MYERS; LESKOVEC, 2011), used by Gao et al. (2017) to evaluate the method BiTULER, and a third dataset of Foursquare (YANG et al., 2015) check-ins, 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; KOLLIOS; GUNOPULOS, 2002), EDR (CHEN; ÖZSU; ORIA, 2005), MD-DTW (HOLT; REINDERS; HENDRIKS, 2007), and MSM (FURTADO et al., 2016). We do not compare MASTERMOVELETS with the methods developed for raw trajectory classification (LEE et al., 2008; DODGE; WEIBEL; FOROOTAN, 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).

In the following sections we present the experimental results for each dataset. In each section we include an introduction of the dataset, the description of its dimensions, and the presentation and discussion of experimental results.

### 4.3.1 Evaluation with the Gowalla dataset

The first experiment uses the dataset from Gowalla (CHO; MYERS; LESKOVEC, 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 4.5 shows details about each dimension, such as data type, value range, and the distance measure used to compare two dimension values.

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

Table 4.5 – Gowalla trajectory dimension description.

| Dimension | Type | Range or examples | Distance measure |
|---|---|---|---|
| Space | Spatial | 40.82651 -73.95039 | Euclidean Distance |
| Time | Temporal | [00:00,23:59] | Difference in minutes |
| Weekday | Ordinal | {Mon, Tue, ..., Sun} | Weekday Distance[4] (0 or 1) |
| Place identifier | Nominal | Any nominal value | Binary Distance (0 or 1) |

We evaluate the methods performing a 5-fold cross-validation. It is not possible to run a traditional cross-validation since for each training set we need to generate new features for MOVELETS and MASTERMOVELETS, and define the thresholds for similarity/distance measures. Therefore, we manually performed a 5-fold cross-validation, generating, in a stratified way, five files for each dataset, and executed 5 times the experiments using four files for training and one for testing, changing each time the test file. We reported the average precision of these 5 executions for each measure in each dataset.

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 MOVELETS we also used a RF classifier with 300 decision trees.

For BiTULER (GAO et al., 2017) we built 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 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 kept 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 $5 \times 2$-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 the experimental evaluation 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 4.6 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 4.6 – Cross-validation 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, which is not the best solution for classifying semantically rich trajectories.

### 4.3.2 Evaluation with the Brightkite dataset

The second experiment uses the dataset from Brightkite (CHO; MYERS; LESKOVEC, 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 4.5).

The experimental setup is the same of the previous dataset.

Table 4.7 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.

Table 4.7 – Cross-validation evaluation results on Brightkite dataset.

| Measure | MD-DTW | LCSS | EDR | MSM | BiTULER | MOVELETS | MASTER MOVELETS NN | MASTER MOVELETS 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 |

### 4.3.3 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[5] and with historical weather data (the weather condition) collected via the Weather Wunderground API[6], 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 4.8 presents the six dimensions used in this dataset, the description of each dimension, and the respective distance function.

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 4.8 – Foursquare trajectory dimension description.

| Dimension | Type | Range or examples | Distance measure |
|---|---|---|---|
| Time | Temporal | [00:00,23:59] | Difference in minutes |
| Weekday | Ordinal | {Mon, Tue, ..., Sun} | Weekday Distance (0 or 1) |
| Venue Category | Nominal | Foursquare categories[7] | 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) |

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

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

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

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 4.9 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 4.9 – Cross-validation 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 error improvement of MASTERMOVELETS in relation to MSM is 63.0% (NN) and 46.9% (RF).

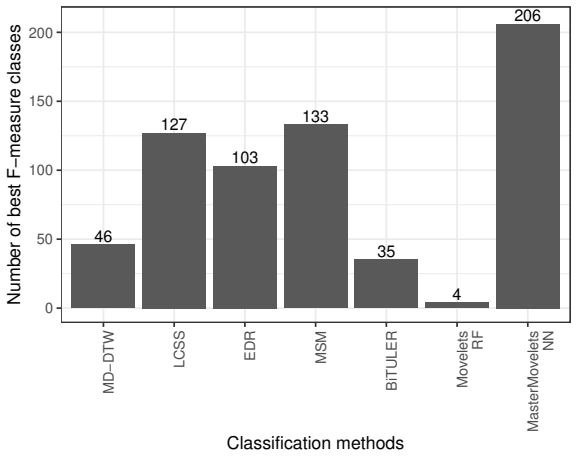
The worst results on this dataset were achieved with MD-DTW, MOVELETS and LCSS. MD-DTW and MOVELETS have the same problem: they depend on a transformation function to encapsulate the distances of all multiple and heterogeneous dimensions in a single distance value. BiTULER is limited to the place identifier (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.

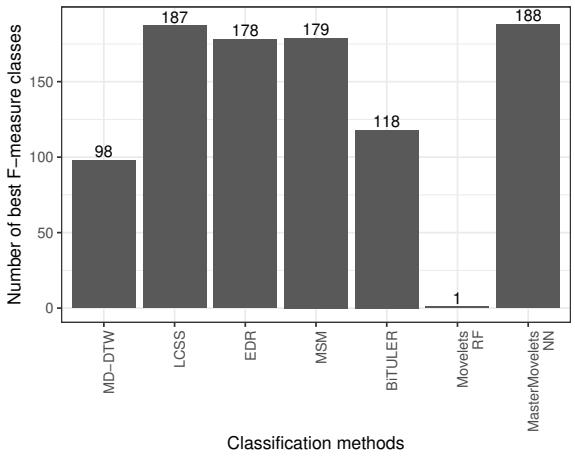
### 4.3.4 General Analysis over all datasets

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 4.10 shows a bar plot for each dataset, indicating for how many classes

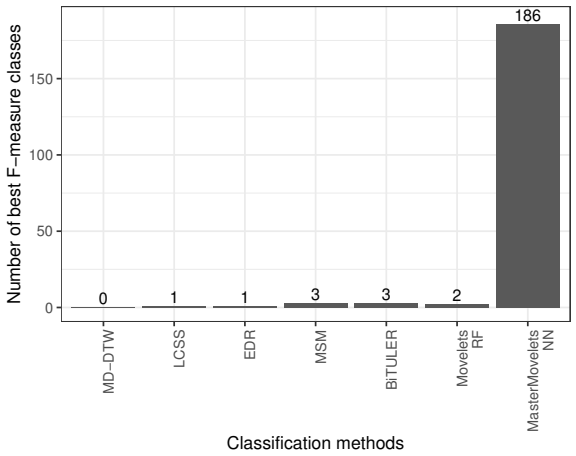
each classifier presents the best F-measure score[8].




(a) Gowalla dataset.




(b) Brightkite dataset.




(c) Foursquare dataset.

Figure 4.10 – Bar plots indicating for how many classes each classifier presents the best F-measure.

In Figure 4.10(a), 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 4.10(b) 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 4.10(c) 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.

We performed an statistical analysis over the capability of these classifiers to discriminate classes, using the Friedman's Aligned Rank Test (GARCÍA et al., 2010), with level of sig-

[8] 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.

nificance $\alpha = 0.05$, resulting in *p-values* $< 0.05$ for all three datasets, Gowalla, Brightkite, and Foursquare, indicating statistical difference among classifiers. Then, we performed the *post hoc* test to find out the significant statistical difference between MASTERMOVELETS and the other classifiers. Table 4.10 shows the result of the *post hoc* test, considering MASTERMOVELETS NN as control.

Table 4.10 – *p-values* of the Friedman's Aligned Rank Statistical Test using MASTERMOVELETS NN as control.

| Dataset | MD-DTW | LCSS | EDR | MSM | BiTULER | MOVELETS | MASTER MOVELETS NN |
|---|---|---|---|---|---|---|---|
| Gowalla | $< 0.01$ | $< 0.01$ | $< 0.01$ | 0.08 | $< 0.01$ | $< 0.01$ | - |
| Brightkite | $< 0.01$ | 0.04 | 0.01 | 0.28 | $< 0.01$ | $< 0.01$ | - |
| Foursqaure | $< 0.01$ | $< 0.01$ | $< 0.01$ | $< 0.01$ | $< 0.01$ | $< 0.01$ | - |

### 4.3.5 Movelet Interpretation and Dimension Analysis

One of the most interesting aspects 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 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, which 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 4.11, 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 heterogeneous, 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. 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).

Table 4.11 – 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 |

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.

## 4.4 CONSIDERATIONS

In this chapter we proposed a new method for extracting relevant subtrajectories for multiple aspect trajectory classification, called MASTERMOVELETS. The proposed method 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 described in details how MASTERMOVELETS explores different dimension combinations in order to find the best combination for a *movelet candidate*. Despite exploring dimension combination, our method generates a number of movelets of the same order of the previous proposal MOVELETS. The *movelets* extracted from MASTERMOVELETS are also neutral to classification methods, are easy to visualize, to describe, to understand, and to find them in new trajectory datasets.

We evaluate our method using three 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 demonstrated that MASTERMOVELETS outperformed existing approaches by reducing the classification error between 10% and 58%.

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 length (e.g. $\log_e m$, where $m$ is the length of each trajectory).

When working on MASTERMOVELETS we also tried other approaches to improve the classification results, but without success. We present and discuss these approaches:

**New Distance Measures for Point of Interest:** in this thesis we compute distance between point of interests (POIs), like check-in venues, using a binary function, where two POIs have distance zero if they are the same POI and one otherwise. We tried to use other two approaches to measure the distance between POIs. The *first approach* consisted of using a hierarchy of POI types as additional information to perform distance calculation. We used the algorithm Lowest Common Antecessor (LCA), that finds the first common hierarchy path between two POIs. Suppose a POI *Brazilian Restaurant* represented on the hierarchy by the sequence $\gamma_1$ =(*Food, Latin American Restaurant, South American Restaurant, Brazilian Restaurant*) and another POI *Argentinian Restaurant* represented by the sequence $\gamma_2$ =(*Food, Latin American Restaurant, South American Restaurant, Argentinian Restaurant*). The LCA of both sequences is (*Food, Latin American Restaurant, South American Restaurant*). So the distance is calculated as $1 - length(LCA(\gamma_1, \gamma_2)/(length(\gamma_1) + length(\gamma_2))$. In our example the distance is 0.25, so the distance between *Brazilian Restaurant* and *Argentinian Restaurant* is 0.25, instead of 1 provided by the binary distance measure used in the experiments. Despite the coherence of the distance value with respect to the semantics of POIs, does this approach not improve the classification results.

The second approach was using as distance between POIs a distance based on a Word Embeddings of the POIs. Word Embeddings are very useful in text mining applications. Using Word Embeddings we can represent each POI as a vector in a $theta$-dimensional space. The vector values for each POI are fitted according to the context, i.e., the POIs visited before and after by an individual, such that minimizing the distance between POIs with the same context in the $theta$-dimensional space. The most common distance function between embedding vectors is the cosine distance that we used to measure the distance between POIs to movelet discovery. However, using this approach has also not improved the classification results. One reason for that is that the classical approach of Word Embeddings is a no supervised learning, so the method does not use the class to fit the embeddings as part of the context. For instance, suppose that it is very common people go to a *Bar* or a *Cafe* after *Work*, then the embedding vectors of *Bar* and *Cafe* are very similar because they happen in similar contexts, i.e., after *Work*. But this rule may not be true for all the users. Maybe, training embeddings using the class as part of the context can improve the classification results.

**Exploring the Multidimensional Orderline:** in this thesis we find the split point for each dimension by using half of the points in the multidimensional orderline and the other half of points for evaluation. During this thesis, we also tried other approaches to explore the multidimensional orderline. The first approach consisted of calculating a confidence

interval for the split points. To do that, we sampled the orderline in a stratified way many times, commonly 10, and then we calculated the mean, the standard deviation and the confidence interval of the mean, for each dimension split point. From this approach we divided the multidimensional orderline in three decision regions: closely (lower than the lower bounds of the confidence interval in all dimensions), middle (into the confidence interval in some dimension) and far (upper than the upper bounds of the confidence interval in all dimensions). This approach has also not improved the classification results. Another approach not explored yet consist of building fuzzy rules to define membership functions based on the multidimensional orderline points distribution to define a degree of membership of a trajectory to the movelet.

# 5 PROCESSING TIME EVALUATION

The main drawback of MOVELETS and MASTERMOVELETS is their processing times. In this chapter we make some analysis considering this point. In Section 5.1 we evaluate these methods by limiting the maximum movelet length. In Section 5.2 we present a scalability evaluation.

## 5.1 MAXIMUM MOVELET LENGTH EVALUATION

A simple way to reduce the processing time of MOVELETS and MASTERMOVELETS is to avoid exploring all possible subtrajectory lengths, limiting the maximum length of the movelets. This strategy is used in the time series domain for the method shapelets. The problem is that finding the appropriate maximum length is sometimes challenging. Therefore, we evaluate the accuracy of MOVELETS and MASTERMOVELETS by exploring the *movelet* maximum length.

In this experiment we explored the maximum length of the movelets by limiting its length to 2, 4, 6, $log_e$ and without limit. For MOVELETS and MASTERMOVELETS we used the spatial dimension for the datasets Animals and Vehicles, and we used all available dimensions for Gowalla and Foursquare datasets (see Tables 4.5 and 4.8). We build classification models using Random Forest considering 300 as the number of trees and we evaluate them with a hold-out strategy, 70% for training e 30% to test. For MOVELETS we used only the movelets as attributes[1], and have not considered global trajectory features. The experiment was performed using 16 cores in two sockets Intel Xeon E5-4627 v2 @ 3.30GHz, 8 cores per socket, and 256 GB RAM.

Table 5.1 presents the results of the experiment for the method MOVELETS in terms of classification accuracy, processing time in seconds (secs) and number of movelets extracted. The results show that in general we can limit the movelet length without expressive loss of accuracy, significantly reducing the processing time and in some cases keeping the same accuracy as exploring all movelet lengths. In addition, in all datasets, the number of movelets is reduced as the maximum movelet length increases. This happens because our proposal selects only the best non-overlapping movelets, and by limiting the movelets length the number of movelets is at most $(n \times m/m')$, where $n$ is the number of trajectories, $m$ the maximum trajectory length and $m'$ is the maximum movelet length.

For the Animals dataset the best accuracy value was 95.2% and was achieved by both without limiting movelet length and by limiting it using $log_e$. In addition, the processing time was significantly reduced. Note that the approach without limit takes 14.1 secs and by limiting movelet length the processing time was reduced at least 7 times, to 2.0 secs.

[1] Different from the experiments presented in Chapter 3 when we combine the movelets with other global features to build classifiers.

Table 5.1 – MOVELETS performance evaluation.

| Dataset | Measure | Max length 2 | Max length 4 | Max length 6 | Max length $log_e$ | Without limit |
|---|---|---|---|---|---|---|
| *Animals* | Accuracy (%) | 92.7 | 92.9 | 92.9 | 95.2 | 95.2 |
| | Processing time (secs) | 1.4 | 1.5 | 1.7 | 2.0 | 14.1 |
| | Number of movelets | 3,099 | 2,890 | 2,783 | 2,760 | 2,440 |
| *Vehicles* | Accuracy (%) | 98.1 | 98.1 | 98.1 | 98.1 | 98.1 |
| | Processing time (secs) | 169.6 | 197.6 | 207.8 | 233,3 | 2,480.7 |
| | Number of movelets | 62,675 | 57,565 | 55,257 | 54,023 | 45,393 |
| *Gowalla* | Accuracy (%) | 48.8 | 48.9 | 50.1 | 49.0 | 49.0 |
| | Processing time (secs) | 166.1 | 176.6 | 187.7 | 183.0 | 493.0 |
| | Number of movelets | 5,259 | 5,323 | 5,339 | 5,329 | 5,366 |
| *Foursquare* | Accuracy (%) | 23.5 | 26.0 | 25.8 | 25.9 | 26.9 |
| | Processing time (secs) | 108.5 | 116.2 | 119.9 | 117.6 | 139.3 |
| | Number of movelets | 1,174 | 1,320 | 1,333 | 1,327 | 1,383 |

For the Vehicles dataset the accuracy was the same for all the movelet length approaches. One more time, the processing time was significantly reduced by limiting the movelet length. Extracting movelets without limiting the length takes 2,480.7 secs and by limiting the movelet length using $log_e$, the processing time was reduced more than 10 times, to 233.3 secs.

For the Gowalla dataset the best accuracy was achieved for maximum movelet length 6, 50.1%, with a difference of 1.1% to the second best accuracy. And for the Foursquare dataset the best accuracy was achieved without limiting movelet length (26,9), with a difference of 1.0% to the second best accuracy. These results show that depending on the dataset the results may vary. Nevertheless, all accuracy results for both datasets were quite low due to the limitation of the method MOVELETS for dealing with multiple and heterogeneous dimensions.

We considered that $log_e$ is a good approach to balance the accuracy and the processing time for extracting movelets. Depending on the length of trajectories in a classification problem the use of a constant movelet length, e.g. 2 trajectory elements to compose the movelet, can be insignificant in terms of trajectory movement, representing a very small part of movement. So, the proposal of using $log_e$ MOVELETS is more independent to the length of trajectories.

Table 5.2 presents the results of the experiment for the method MASTERMOVELETS in terms of classification accuracy, processing time in seconds (secs) and number of movelets. We observe that the classification accuracy does not have a high variation when the maximum length of the movelets increases. The behaviour changes according to the dataset. By limiting the maximum movelet length to the $log_e$ of the 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 length of the movelet to the $log_e$ of the trajectory

length 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. By limiting the maximal length of the movelet to $log_e$ we reduced the number of candidates evaluated by MASTERMOVELETS from $(n \times m^2 \times 2^l)$ to $(n \times m\ log_e\ m \times 2^l)$ and consequently the overall complexity of movelet discovering decreased from $O(n^3 \times m^3\ log\ m \times 2^l)$ to $O(n^3 \times m^2\ log^2\ m \times 2^l)$.

Table 5.2 – MASTERMOVELETS performance evaluation.

| Dataset | Measure | Max length 2 | Max length 4 | Max length 6 | Max length $log_e$ | Without limit |
|---|---|---|---|---|---|---|
| *Animals* | Accuracy (%) | 71.4 | 71.4 | 69.0 | 69.0 | 66.7 |
| | Processing time (secs) | 5.9 | 7.9 | 11.6 | 13.7 | 69.5 |
| | Number of movelets | 4,736 | 4,030 | 3,654 | 3,630 | 2,511 |
| *Vehicles* | Accuracy (%) | 98.7 | 98.7 | 98.7 | 98.7 | 98.7 |
| | Processing time (secs) | 484.0 | 790.1 | 1,025.7 | 1,535.4 | 22,275.2 |
| | Number of movelets | 69,863 | 58,204 | 52,781 | 50,156 | 35,071 |
| *Gowalla* | Accuracy (%) | 91.8 | 92.2 | 92.0 | 91.0 | 92.4 |
| | Processing time (secs) | 4,370.7 | 8,237.3 | 11,403.9 | 9,751.4 | 21,399.1 |
| | Number of movelets | 56,037 | 51,720 | 50,435 | 51,252 | 49,397 |
| *Foursquare* | Accuracy (%) | 67.9 | 67.8 | 68.9 | 68.0 | 67.3 |
| | Processing time (secs) | 3,947.9 | 7,507.0 | 10,522.4 | 9,313.4 | 25,169.3 |
| | Number of movelets | 28,560 | 21,532 | 19,889 | 20,718 | 18,123 |

For the Animals dataset the best accuracy was achieved by using maximum movelet length 2 and 4, 71.4%, and the lowest accuracy without limiting the movelet length, 66%. These accuracy values were around 20% lower than the results reported in Table 5.1 for the method Movelets. The method MOVELETS compose the attribute-value table using the distance value between trajectories and movelets (as detailed in 3.2.3), leaving for the classifier (Random Forest in these experiments) to decide how to use distance values to predict the class label. However, because MASTERMOVELETS deals with multiple and heterogeneous dimensions we cannot use a single distance value to compose the attribute-value table, and we only use a binary value, 0 if the distances between each trajectory and each movelet is lower than the movelet split points for all dimensions, and 1 otherwise. This situation can influence the accuracy of the classification models depending on the problem. An interesting research topic is how to build a membership probability function based on the distances of dimensions and use the probability values to build the attribute-value table.

For the Vehicles dataset the accuracy was the same for all the approaches, 98.7. The processing time was significantly reduced from 22,275.2 secs without limiting movelet length to 1,535.4 secs, more than 14 times, using $log_e$ as maximum movelet length.

For the Gowalla dataset the best accuracy was achieved without limiting the movelet length, i.e. 92.4%, with a difference of 1.4% to the worst accuracy, 91.0%. For the Foursquare

dataset the best accuracy was achieved limiting the movelet length to 6 (68.9), with a difference of 1.6% to the second best accuracy, 67.3%. These results show that the accuracy values are quite similar for each dataset and the decreasing of the processing time is significant, considering that for these datasets we need to explore all the trajectory dimensions. The order of processing time reduction is at least 2.3 times, between without limiting movelet length and limiting by $log_e$.

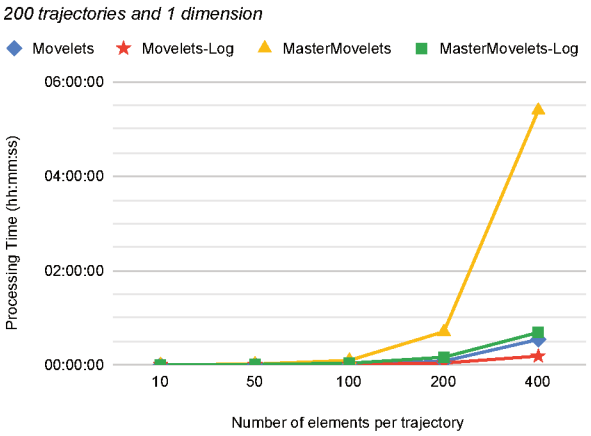
## 5.2 SCALABILITY ANALYSIS

Social media trajectories normally have less points when compared to raw trajectories generated using the GPS (Global Positioning System), 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 length 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.

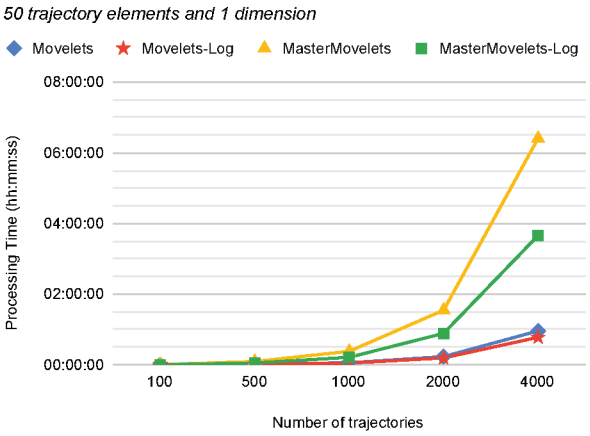
In this experiment we compare the processing time of MOVELETS and MASTERMOVELETS to MOVELETS-LOG and MASTERMOVELETS-LOG, that limit the maximal length of the movelets to $log_e(m_\alpha)$, where $m_\alpha$ is the length of each trajectory in the dataset.

Figure 5.1 presents the results of this experiment. In Figure 5.1 (a) we generated 200 trajectories of 1 dimension, and varied the length of the trajectories from 10 points to 400 points. In Figure 5.1 (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 5.1 (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 using an Intel Core i7-6700 CPU @ 3.40GHz with 4 cores, and 32GB of memory.

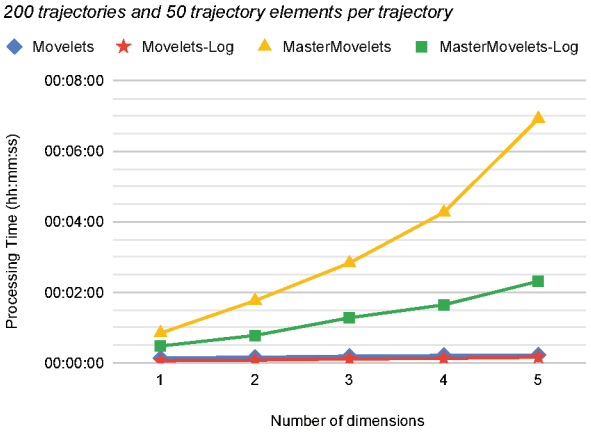
We may observe in the figure that the running time of MASTERMOVELETS grows in all scenarios, but limiting the maximum length of the movelets as in MASTERMOVELETS-LOG, we notice that the running time significantly decreases. The best scenario for MASTERMOVELETS-LOG is in Figure 5.1(a) where the length of the trajectories increase, i.e., their number elements increase, and in Figure 5.1(c) where the number of dimensions increases. The results of MASTERMOVELETS-LOG are less impacting in the experiment in Figure 5.1 (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 length of the movelets to $log_e$ of the length of the trajectories are not so expressive.




(a) varying the length of the trajectories.




(b) varying the number of trajectories.




(c) varying the number of dimensions.

Figure 5.1 – Computational time spent by MASTERMOVELETS without limits and using the log limitation (MASTERMOVELETS-LOG), MOVELETS, and MOVELETS-LOG, 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.

# 6 CONCLUSIONS

We are witnessing the era of movement tracking and mining, where huge volumes of data about our daily lives are being collected and stored in several sources and formats. These data are stored in the form of trajectories, that is a complex data type represented as a sequence of points, where each point has a *spatial dimension*, commonly represented by latitude and longitude; a *time dimension*, described by a timestamp; and more recently *semantic* dimensions. In this context, the movement of an individual can be enriched with *multiple* and *heterogeneous* dimensions, beyond space and time, like weather condition, transportation modes, checking-in information, among others. This new type of data was introduced in this thesis as *Multiple Aspect Trajectories*. Because it is a new data type, there are no methods in the literature for mining this kind of data.

Trajectory classification is an important issue in mobility data mining, since it is used for discovering movement patterns for many applications. 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. For trajectories represented by these two dimensions the classification task consists of extracting global and local features from trajectories, like average speed, acceleration, turning angle, among others, and build a classifier using these features to describe each trajectory sample. With the explosion of social media data and the advances in the semantic enrichment of mobility data, a new type of trajectory has emerged, and the trajectory spatio-temporal points have now multiple and heterogeneous semantic dimensions. There are no classification methods in the literature to perform trajectory classification for this type of data. As a consequence, new classification methods are needed.

In this thesis we addressed the problem of trajectory classification by exploring relevant subtrajectories for building classification models. We proposed two methods considering different scenarios. In the first scenario finding relevant subtrajectories considering all trajectory dimensions together. In the second scenario we consider relevant subtrajectories finding the best trajectory dimension combination for each relevant subtrajectory. Both methods are parameter-free and domain independent, and do neither perform trajectory discretization nor partition. The relevant subtrajectories extracted by both methods are neutral to classification methods, are easy to visualize, to describe, to understand, and to find them in new trajectory datasets.

The first method, called MOVELETS, discovers relevant subtrajectories for trajectory classification. MOVELETS supports multiple dimensions and considers all dimensions together. We presented an experimental evaluation of MOVELETS demonstrating the effectiveness of the proposed method in comparison with the state-of-the-art methods. Our proposal largely outperformed existing approaches for trajectory classification in almost all datasets. The main limitation of this method related to discovering relevant subtrajectories is that it uses only one distance function between trajectory elements, that encapsulates the distances of all dimensions in a single distance value. As a consequence, besides the difficulty of balancing the importance

of each dimension to build this function, it may include non-discriminant dimensions in the element distance, reducing the chance to find discriminant subtrajectories as the number of dimensions increase.

The second method, called MASTERMOVELETS, discovers relevant subtrajectories for multiple aspect trajectory classification. MASTERMOVELETS was designed to handle multiple and heterogeneous trajectory dimensions, by finding the best dimension combination for representing subtrajectories. We evaluate this method using three datasets to classify multiple aspect trajectories and compare it with methods in the literature that support multiple data dimensions with different characteristics, including space, time, and semantics. Our proposal outperformed most existing approaches by reducing the classification error from 10% to 58%. The main limitation of this method is the time complexity, that is $O(n^3 \times m^3\ log\ m \times 2^l)$, where $n$ is the number of trajectories, $m$ is the length of the longest trajectory, and $l$ the number of dimensions. This complexity is very high and thus unscalable. We also propose an adaptation of this proposal called MASTERMOVELETS-LOG to reduce the length of the longest movelet to $log_e\ m_i$, where $m_i$ is the length of each trajectory, to reduce the time complexity keeping the classification accuracy. However, the time complexity is still a problem.

According to the experimental results we recommend the use of MOVELETS in raw trajectory classification problems, mainly when trajectories with only a few dimensions and the user knows the relation among them. We recommend the use of MASTERMOVELETS in trajectory classification problems with multiple and heterogeneous dimensions, when it is difficult to know and learn the relation among dimensions. In addition, MOVELETS is faster than MASTERMOVELETS, so for longest trajectories we also recommend the MOVELETS.

It is important to highlight that so far, MOVELETS and MASTERMOVELETS are the only methods that can deal with the spatial dimension, represented by latitude and longitude, as they must be considered together. Other works extract numerical features from this dimension.

In order to solve the limitation of our methods we propose the following future work:

**Improving MASTERMOVELETS time complexity:** one of the major challenges related to the algorithm MASTERMOVELETS is reducing the time complexity, that is $O(n^3 \times m^3\ log\ m \times 2^l)$, where $n$ is the number of trajectories, $m$ is the length of the longest trajectory, and $l$ the number of dimensions. This complexity is very high and thus unscalable. To reduce the theoretical complexity of the algorithm we identified the following three issues to research: optimization methods for finding the split points in multidimensional orderlines, methods to avoid exploring all dimensions combination of each subtrajectory, and methods to reduce the number of movelet candidates to explore.

The first issue consists of reducing the complexity of the MASTERRELEVANCE algorithm that finds the dimension split points of a movelet candidate and its relevance. The problem of finding the dimension split points is quite similar to the problem of *finding the maximals of a set of vectors* (KUNG; LUCCIO; PREPARATA, 1975) and our current algorithm is an exact solution for this problem that has time complexity $O(n^2 \times l)$,

where $n$ is the number of trajectories and $l$ the number of dimensions. New exact and approximated solutions for this problem can significantly reduce the time complexity of the average-case to $O(n \times l)$ (GODFREY; SHIPLEY; GRYZ, 2005).

The second issue consists of avoiding to explore all dimension combinations of each subtrajectory. In fact, the only way to discover which of all dimension combination is the best for a movelet candidate is by exploring all of them, that requires exponential time complexity, $O(2^l)$, like the attribute selection problem in machine learning (LIU; MOTODA, 2012). A simple solution is to limit the maximum number of dimensions in a subtrajectory like we limited the maximal length of the movelets in MASTERMOVELETS-LOG. Other strategies can be used to avoid exploring all dimension combinations, like best-first search. Best first is a search algorithm which explores the space of feature combinations by expanding the most promising feature combination chosen according to its relevance. Using searching algorithms, like best-first, the time complexity of finding the best dimension combination can be reduced to $O(l^2)$.

The third issue consists of reducing the number of movelet candidates to be generated and explored. The number of movelet candidates generated for each trajectory is $O(m^2)$. One way to reduce the amount of movelet candidates generated is to use a decay rate, such as selecting half the best candidates for each subtrajectory length to explore in the next length. By applying this strategy on a trajectory, the algorithm starts exploring $m$ candidates of length 1, then it explores $m/2$ candidate of length 2 and finally it explores 1 candidate of length $log_2\ m$, which sums less than $2 \times m$ explored candidates. This strategy reduces the number of candidates for a trajectory from $O(m^2)$ to $O(m)$.

Improving all the three issues by applying the proposed approaches the time complexity of the MASTERMOVELETS algorithm can be theoretically reduced from $O(n^3 \times m^3\ log\ m \times 2^l)$ to $O(n^2 \times m^2\ log\ m \times\ l^2)$.

**Using Embedding-Distances for Movelet Discovering:** Word embeddings is a hot topic in Machine Learning research . This approach is useful for dimensions which have many possible values, like the check-in POI type dimension. Using word embeddings we can better represent the relation of distance between POIs, instead of using the binary distance (two POIs have distance zero if they are the same POI type and one otherwise). During this thesis we tried to use an unsupervised learning approach to find the POI type embeddings without success. However, we believe that by designing a new method using supervised learning to find the POI type embeddings MASTERMOVELETS could find better movelets.

**Explore Subtrajectory Features:** the scope of this thesis was exploring element trajectory features for movelets discovering, i.e., a function that computes the distance between the subtrajectory elements (points). An additional way to compare subtrajectories is by extracting global subtrajectory features, like the traveled distance, the duration, the average

speed on a subtrajectory. We believe that by combining element and subtrajectory features we can find new subtrajectory patterns. For instance, using the spatial and time element features our algorithms can find subtrajectory patterns of moving objects that perform similar movement at similar day times, but if the movement happens in different times of the day, it would not possible to find it even if the subtrajectories duration were similar. Combining element and subtrajectory features, like the subtrajectory duration, we can move forward a new set of pattern never explored until now.

**Reducing the Number of Movelets:** the proposed methods, MOVELETS and MASTERMOVELETS, generate large number of movelets. For MOVELETS we proposed a pruning algorithm which performs pruning after movelet discovery and significantly reduce the number of movelets. However, this pruning could be made during the movelet discovery process. An interesting research work is to propose new methods for pruning movelets during movelet discovering.

**Learning Distance Measures for Movelets:** the method MOVELETS needs the definition of a distance function between trajectory elements, which can be a difficult task in domains with dimensions in different units. An interesting research topic is to automatically learn the distance function between trajectory elements considering all trajectory dimensions.

**Missing Values Evaluation:** the methods MOVELETS and MASTERMOVELETS assume that trajectory data do not have missing values. However, in real world applications of multiple aspect trajectory analysis missing values are possible. An interesting research work is to evaluate the influence of missing values in proposed methods and to propose new approaches to deal with them, such as new methods for imputing data in multiple aspect trajectories or new similarity and distance measures robust to missing values.

**Class Imbalance Evaluation:** in classification problems it is very common the classes have different frequencies and this is an open issue in data mining yet. An interesting research topic is to evaluate the influence of class imbalance to find relevant subtrajectories using MOVELETS and MASTERMOVELETS, mainly in relation to the function that measures the subtrajectory relevance.

**Ranking Method for MASTERALIGNMENT:** the proposed method for finding the best alignment between a trajectory and a subtrajectory uses a ranking method to find the position of the best alignment considering all trajectory dimensions. We used the average rank among dimensions, but other methods to combine ranking can be used to perform this task. An interesting work is to research rank composition methods to use in MASTERALIGNMENT and to evaluate them for movelet discovering.

During this thesis, other papers were published in collaboration related to multiple aspect trajectory definition (MELLO et al., 2019), multiple aspect trajectory similarity measures (PETRY et al., 2019), and data mining classification (ZALEWSKI et al., 2016; MALETZKE et al., 2017; BITENCOURT; FERRERO, 2019).

# REFERÊNCIAS

AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-based Learning Algorithms. **Machine Learning**, Springer, v. 6, n. 1, p. 37–66, 1991.

ANDRIENKO, G.; ANDRIENKO, N.; WROBEL, S. Visual Analytics Tools for Analysis of Movement Data. **ACM SIGKDD Explorations Newsletter**, ACM, v. 9, n. 2, p. 38–46, 2007.

ARVACHEH, E.; TIZHOOSH, H. Pattern Analysis usVng ZernTke Moments. In: **Proceedings of the IEEE Instrumentation and Measurement Technology Conference (IMTC)**. Ottawa, ON, Canada: IEEE, 2005. v. 2, p. 1574–1578.

ATEV, S.; MILLER, G.; PAPANIKOLOPOULOS, N. P. Clustering of Vehicle Trajectories. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 11, n. 3, p. 647–657, 2010.

BERGROTH, L.; HAKONEN, H.; RAITA, T. A Survey of Longest Common Subsequence Algorithms. In: **Proceedings of 7th International Symposium on String Processing and Information Retrieval (SPIRE)**. Washington, DC, USA: IEEE Computer Society, 2000. p. 39–48.

BERNDT, D. J.; CLIFFORD, J. Using Dynamic Time Warping to Find Patterns in Time Series. In: **Proceedigns of the Workshop on Knowledge Discovery in Databases (KDD)**. Seattle, WA, USA: AAAI Press, 1994. v. 10, n. 16, p. 359–370.

BILOGREVIC, I. et al. Predicting Users' Motivations behind Location Check-ins and Utility Implications of Privacy Protection Mechanisms. In: **Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS)**. San Diego, California, USA: The Internet Society, 2015.

BITENCOURT, P. B.; FERRERO, C. A. Predição de Risco de Evasão de Alunos Usando Métodos de Aprendizado de Máquina em Cursos Técnicos. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. Brasília, DF: SBC, 2019. v. 8, n. 1, p. 149–158.

BOGORNY, V. et al. CONSTAnT - A Conceptual Data Model for Semantic Trajectories of Moving Objects. **Transactions in GIS**, v. 18, n. 1, p. 66–88, 2014. ISSN 1467–9671.

BOLBOL, A. et al. Inferring Hybrid Transportation Modes from Sparse GPS Data Using a Moving Window SVM Classification. **Computers, Environment and Urban Systems**, Elsevier, v. 36, n. 6, p. 526–537, 2012.

CAI, G.; LEE, K.; LEE, I. Discovering Common Semantic Trajectories from Geo-tagged Social Media. In: **Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE)**. Morioka, Japan: Springer, 2016. p. 320–332.

CHEN, L.; ÖZSU, M. T.; ORIA, V. Robust and Fast Similarity Search for Moving Object Trajectories. In: **Proceedings of the ACM International Conference on Management of Data (SIGMOD)**. Baltimore, Maryland: ACM, 2005. p. 491–502.

CHEN, Y.-W.; LIN, C.-J. Combining SVMs with Various Feature Selection Strategies. In: GUYON, I. et al. (Ed.). **Feature Extraction: Foundations and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 315–324.

CHO, E.; MYERS, S. A.; LESKOVEC, J. Friendship and Mobility: User Movement in Location-based Social Networks. In: **Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. San Diego, California, USA: ACM, 2011. p. 1082–1090.

CORTES, C.; VAPNIK, V. Support-Vector Networks. **Machine Learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

DODGE, S.; WEIBEL, R.; FOROOTAN, E. Revealing the Physics of Movement: Comparing the Similarity of Movement Characteristics of Different Types of Moving Objects. **Computers, Environment and Urban Systems**, Elsevier, v. 33, n. 6, p. 419–434, 2009.

DODGE, S.; WEIBEL, R.; LAUBE, P. Exploring Movement-similarity Analysis of Moving Objects. **SIGSPATIAL Special**, ACM, v. 1, n. 3, p. 11–16, 2009.

ENDO, Y. et al. Classifying Spatial Trajectories Using Representation Learning. **International Journal of Data Science and Analytics**, Springer, v. 2, n. 3-4, p. 107–117, 2016.

ETEMAD, M.; JÚNIOR, A. S.; MATWIN, S. Predicting Transportation Modes of GPS Trajectories using Feature Engineering and Noise Removal. In: **Proceedings of 31st Canadian Conference on Artificial Intelligence, Canadian**. Toronto, ON, Canada: Springer, 2018. p. 259–264.

FENG, Z.; ZHU, Y. A Survey on Trajectory Data Mining: Techniques and Applications. **IEEE Access**, v. 4, p. 2056–2067, 2016.

FERRERO, C. A. **MasterMovelets Code**. 2019. https://github.com/anfer86/dmkd_masterMovelets_results. [accessed 16-July-2019].

FERRERO, C. A.; ALVARES, L. O.; BOGORNY, V. Multiple Aspect Trajectory Data Analysis: Research Challenges and Opportunities. In: **XVII Brazilian Symposium on Geoinformatics (Geoinfo)**. Campos do Jordão, SP, Brazil: MCTIC/INPE, 2016. p. 1–12.

FERRERO, C. A. et al. Movelets: Exploring Relevant Subtrajectories for Robust Trajectory Classification. In: ACM. **Proceedings of the 33rd ACM Symposium On Applied Computing (SAC), Knowledge Extraction from Geographical Data (KEGeoD)**. Pau, France, 2018. p. 1–8.

FERRERO, C. A. et al. MasterMovelets: Discovering Heterogeneous Movelets for Multiple Aspect Trajectory Classification. **Data Mining and Knowledge Discovery**, p. 1–32, 2020. (Accepted for publication).

FURTADO, A. S. et al. Unveiling Movement Uncertainty for Robust Trajectory Similarity Analysis. **International Journal of Geographical Information Science**, Taylor & Francis, v. 32, n. 1, p. 140–168, 2018.

FURTADO, A. S. et al. Multidimensional Similarity Measuring for Semantic Trajectories. **Transactions in GIS**, Wiley Online Library, v. 20, n. 2, p. 280–298, 2016.

GAO, P. et al. Quantifying Animal Trajectories Using Spatial Aggregation and Sequence Analysis: A Case Study of Differentiating Trajectories of Multiple Species. **Geographical Analysis**, v. 48, n. 3, p. 275–291, 2016.

GAO, Q. et al. Identifying Human Mobility via Trajectory Embeddings. In: **Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)**. Melbourne, Australia: AAAI Press, 2017. p. 1689–1695.

GARCÍA, S. et al. Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power. **Information Sciences**, Elsevier, v. 180, n. 10, p. 2044–2064, 2010.

GODFREY, P.; SHIPLEY, R.; GRYZ, J. Maximal Vector Computation in Large Data Sets. In: **Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)**. Trondheim, Norway: VLDB Endowment, 2005. p. 229–240.

GRÜNWALD, P. Model Selection Based on Minimum Description Length. **Journal of Mathematical Psychology**, Elsevier, v. 44, n. 1, p. 133–152, 2000.

HILLS, J. et al. Classification of Time Series by Shapelet Transformation. **Data Mining and Knowledge Discovery**, v. 28, n. 4, p. 851–881, 2014.

HOLT, G. A. ten; REINDERS, M. J.; HENDRIKS, E. Multi-dimensional Dynamic Time Warping for Gesture Recognition. In: **Proceedings of the 13th Annual Conference of the Advanced School for Computing and Imaging**. Heijen, the Netherlands: Academic Press, 2007. v. 300, p. 1–8.

HUANG, Z.; LENG, J. Analysis of Hu's Moment Invariants on Image Scaling and Rotation. In: IEEE. **Proceedings of the 2nd International Conference on Computer Engineering and Technology (ICCET)**. Chengdu, China, 2010. v. 7, p. 476–480.

KALYVAS, C.; TZOURAMANIS, T. A Survey of Skyline Query Processing. **arXiv preprint arXiv:1704.01788**, 2017.

KUNG, H.-T.; LUCCIO, F.; PREPARATA, F. P. On Finding the Maxima of a Set of Vectors. **Journal of the ACM (JACM)**, ACM, v. 22, n. 4, p. 469–476, 1975.

LEE, J.-G.; HAN, J.; LI, X. Trajectory Outlier Detection: A Partition-and-Detect Framework. In: **Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE)**. Washington, DC, USA: IEEE Computer Society, 2008. p. 140–149.

LEE, J.-G. et al. *TraClass*: Trajectory Classification using Hierarchical Region-based and Trajectory-based Clustering. In: **Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)**. Auckland, New Zealand: VLDB Endowment, 2008. p. 1081–1094.

LEE, J.-G. et al. Mining Discriminative Patterns for Classifying Trajectories on Road Networks. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 23, n. 5, p. 713–726, 2011.

LEHMANN, A. L.; ALVARES, L. O.; BOGORNY, V. SMSM: a Similarity Measure for Trajectory Stops and Moves. **International Journal of Geographical Information Science**, Taylor & Francis, p. 1–26, 2019.

LI, X. Using Complexity Measures of Movement for Automatically Detecting Movement Types of Unknown GPS Trajectories. **American Journal of Geographic Information System**, Scientific & Academic Publishing, v. 3, n. 2, p. 63–74, 2014.

LIN, J. et al. Experiencing SAX: a Novel Symbolic Representation of Time Series. **Data Mining and knowledge discovery**, Springer, v. 15, n. 2, p. 107–144, 2007.

LINES, J.; BAGNALL, A. Alternative Quality Measures for Time Series Shapelets. In: **Proc. of the 13th International Conference on Intelligent Data Engineering and Automated Learning**. Berlin, Heidelberg: Springer-Verlag, 2012. p. 475–483.

LIU, H.; MOTODA, H. **Feature Selection for Knowledge Discovery and Data Mining**. Norwell, Massachusetts, USA: Kluwer Academic Publishers, 2012. v. 454.

LIU, H.; SCHNEIDER, M. Similarity Measurement of Moving Object Trajectories. In: **Proceedings of the 3rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS), International Workshop on GeoStreaming (IWGS)**. New York, NY, USA: ACM, 2012. p. 19–22.

LV, M.; CHEN, L.; CHEN, G. Mining User Similarity Based on Routine Activities. **Information Sciences**, v. 236, p. 17–32, 2013.

MACDONALD, A.; ELLEN, J. Multi-level Resolution Features for Classification of Transportation Trajectories. In: **Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA)**. Miami, Florida, USA: IEEE, 2015. p. 713–718.

MALETZKE, A. G. et al. Medical time series classification using global and local feature extraction strategies. **Journal on Health Informatics**, p. 1–10, 2017. **Accepted for publication**.

MARLER, R. T.; ARORA, J. S. Survey of Multi-Objective Optimization Methods for Engineering. **Structural and Multidisciplinary Optimization**, Springer, v. 26, n. 6, p. 369–395, 2004.

MAZIMPAKA, J. D.; TIMPF, S. Trajectory Data Mining: A Review of Methods and Applications. **Journal of Spatial Information Science**, v. 2016, n. 13, p. 61–99, 2016.

MELLO, R. d. S. et al. MASTER: A Multiple Aspect View on Trajectories. **Transactions in GIS**, Wiley Online Library, v. 23, p. 805–822, 2019.

MIKOLOV, T. et al. Efficient Estimation of Word Representations in Vector Space. **arXiv preprint arXiv:1301.3781**, 2013.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

MUEEN, A.; KEOGH, E.; YOUNG, N. Logical-Shapelets: An Expressive Primitive for Time Series Classification. In: **Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)**. San Diego, California, USA: ACM, 2011. p. 1154–1162.

PATEL, D. et al. Incorporating Duration Information for Trajectory Classification. In: **Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE)**. Washington, DC, USA: IEEE, 2012. p. 1132–1143.

PETRY, L. M. et al. Towards Semantic-Aware Multiple-Aspect Trajectory Similarity Measuring. **Transactions in GIS**, Wiley Online Library, v. 23, p. 960–975, 2019.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

RANU, S. et al. Indexing and Matching Trajectories under Inconsistent Sampling Rates. In: **Proceedings of the IEEE 31st International Conference on Data Engineering (ICDE)**. Seoul, South Korea: IEEE, 2015. p. 999–1010.

REUMERS, S. et al. Semantic Annotation of Global Positioning System Traces: Activity Type Inference. **Transportation Research Record: Journal of the Transportation Research Board**, Transportation Research Board of the National Academies, n. 2383, p. 35–43, 2013.

RISSANEN, J. Modeling by Shortest Data Description. **Automatica**, Elsevier, v. 14, n. 5, p. 465–471, 1978.

ROSSI, L.; MUSOLESI, M. It's the Way you Check-in: Identifying Users in Location-Based Social Networks. In: ACM. **Proceedings of the 2nd ACM Conference on Online Social Networks**. Dublin, Ireland, 2014. p. 215–226.

SANTOS, I. J. **TRACTS: Um Método para Classificação de Trajetórias de Objetos Móveis Usando Séries Temporais**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brasil., 2011.

SCHUSTER, M.; PALIWAL, K. K. Bidirectional Recurrent Neural Networks. **IEEE Transactions on Signal Processing**, IEEE, v. 45, n. 11, p. 2673–2681, 1997.

SHARMA, L. K. et al. Nearest Neighbour Classification for Trajectory Data. In: **Proceedings of the International Conference on Advances in Information and Communication Technologies (ICT)**. Kochi, Kerala, India: Springer, 2010. p. 180–185.

SHOKOOHI-YEKTA, M. et al. Generalizing dtw to the multi-dimensional case requires an adaptive approach. **Data Mining and Knowledge Discovery**, Springer, v. 31, n. 1, p. 1–31, 2017.

SILVA, C. L. da; PETRY, L. M.; BOGORNY, V. A Survey and Comparison of Trajectory Classification Methods. In: **Proceedings of the 8th Brazilian Conference on Intelligent Systems (BRACIS)**. Salvador, BA, Brazil: IEEE, 2019. p. 788–793.

SILVA, E. L. d.; MENEZES, E. M. **Metodologia da Pesquisa e Elaboração de Dissertação**. 4. ed. Florianópolis, SC, Brazil: UFSC, 2001.

SOKOLOVA, M.; LAPALME, G. A Systematic Analysis of Performance Measures for Classification Tasks. **Information Processing & Management**, Elsevier, v. 45, n. 4, p. 427–437, 2009.

SOLEYMANI, A. et al. Integrating Cross-scale Analysis in the Spatial and Temporal Domains for Classification of Behavioral Movement. **Journal of Spatial Information Science**, v. 2014, n. 8, p. 1–25, 2014.

SOLEYMANI, A. et al. Developing and Integrating Advanced Movement Features Improves Automated Classification of Ciliate Species. **PloS one**, Public Library of Science, v. 10, n. 12, p. e0145345, 2015.

SPACCAPIETRA, S. et al. A Conceptual View on Trajectories. **Data and Knowledge Engineering**, Elsevier Science Publishers B. V., Amsterdam, Netherlands, v. 65, n. 1, p. 126–146, abr. 2008.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. 1. ed. Boston, Massachusetts, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

VARLAMIS, I. Evolutionary data sampling for user movement classification. In: **Proceedings of the Congress on Evolutionary Computation (CEC)**. Sendai, Japan: IEEE, 2015. p. 730–737.

VELDHUIZEN, D. A. V.; LAMONT, G. B. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. **Evolutionary computation**, MIT Press, v. 8, n. 2, p. 125–147, 2000.

VLACHOS, M.; KOLLIOS, G.; GUNOPULOS, D. Discovering similar multidimensional trajectories. In: **Proceedings of the 18th International Conference on Data Engineering**. San Jose, CA, USA: IEEE, 2002. p. 673–684. ISSN 1063-6382.

VRIES, G. de; SOMEREN, M. van. Clustering vessel trajectories with alignment kernels under trajectory compression. **Machine Learning and Knowledge Discovery in Databases**, Springer, p. 296–311, 2010.

WITTEN, I. H. et al. **Data Mining: Practical Machine Learning Tools and Techniques**. Cambridge, Massachusetts, USA: Morgan Kaufmann, 2016.

XIAO, X. et al. Inferring Social Ties Between Uers with Human Location History. **Journal of Ambient Intelligence and Humanized Computing**, Springer Berlin Heidelberg, v. 5, n. 1, p. 3–19, 2014.

XIAO, Z. et al. Identifying Different Transportation Modes from Trajectory Data Using Tree-Based Ensemble Classifiers. **ISPRS International Journal of Geo-Information**, Multidisciplinary Digital Publishing Institute, v. 6, n. 2, p. 57, 2017.

YANG, D. et al. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, v. 45, n. 1, p. 129–142, 2015.

YE, L.; KEOGH, E. J. Time Series Shapelets: A Novel Technique that Allows Accurate, Interpretable and Fast Classification. **Data Mining and Knowledge Discovery**, v. 22, n. 1-2, p. 149–182, 2011.

YING, J. J.-C. et al. Semantic Trajectory Mining for Location Prediction. In: **Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)**. Chicago, Illinois: ACM, 2011. p. 34–43.

YING, J. J.-C. et al. Mining User Similarity from Semantic Trajectories. In: **Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks (LBSN)**. New York, NY, USA: ACM, 2010. p. 19–26.

ZALEWSKI, W. et al. Exploring Shapelet Transformation for Time Series Classification in Decision Trees. **Knowledge Based Systems**, Elsevier Science Publishers B. V., Amsterdam, Netherlands, v. 112, n. C, p. 80–91, 2016.

ZHENG, Y. Trajectory Data Mining: An Overview. **ACM Trans. Intell. Syst. Technol.**, ACM, New York, NY, USA, v. 6, n. 3, p. 29:1–29:41, 2015. ISSN 2157-6904.

ZHENG, Y. et al. Understanding Transportation Modes based on GPS Data for Web Applications. **ACM Transactions on the Web (TWEB)**, ACM, v. 4, n. 1, p. 1, 2010.

ZHONG, Y. et al. You are Where you Go: Inferring Demographic Attributes from Location Check-ins. In: **Proceedings of the 18th ACM International Conference on Web Search and Data Mining**. Shanghai, China: ACM, 2015. p. 295–304.

ZHOU, F. et al. Trajectory-User Linking via Variational AutoEncoder. In: **Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)**. Stockholm, Sweden: AAAI Press, 2018. p. 3212–3218.

# 7 APPENDIX 1

## 7.1 COMPUTING THE DISTANCE BETEWEEN TRAJECTORY ELEMENTS

To exemplify how we compute the distance between two trajectory elements, i.e., two trajectory points with $l$ dimensions, we present a running example. Let us consider a trajectory $T$ containing 5 elements ($T.length = 5$) and a trajectory $T_1$ of a set $\mathbf{T}$, containing 4 elements ($T_1.length = 4$). In this example, for the sake of simplicity, we only consider the dimension space. Figure 7.1 shows $T$ and $T_1$, where each point indicates the trajectory element dimensions, $x$ and $y$.

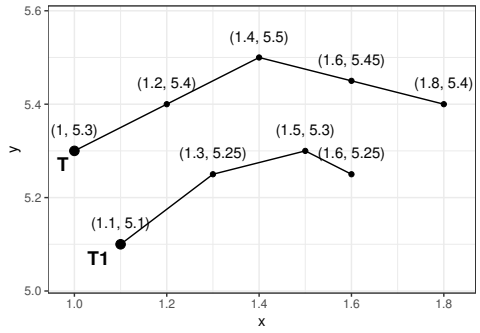


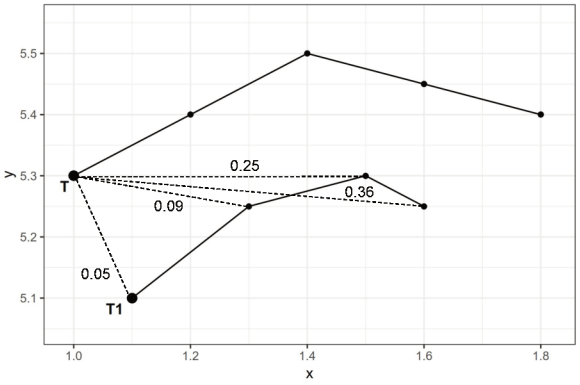

Figura 7.1 – Example of two trajectories $T$ and $T_1$.

For calculating the part of the array $A_1$ containing the element distances between $T$ and $T_1$, we calculate the distance between each pair of trajectory elements $T[j]$ and $T_1[k]$. Figures 7.2(a-e) show the distance values between each pair of trajectory elements for $j$-th element in $T$ and Figure 7.2(f) shows $A_1[1,..,..]$ is a 2-dimensional array with 5 rows and 4 columns, where $A_1[1,j,k]$ is the distance between the $j$-th trajectory element of $T$ and the $k$-th trajectory element of $T_1$.

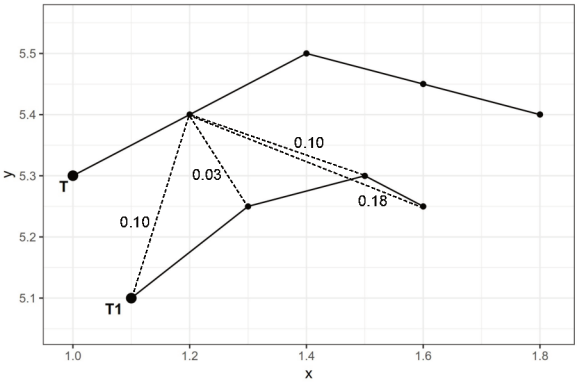
Suppose that each element distance computation is performed as the Euclidean Distance of the dimension space, i.e., $dist_e(e_i,e_j) = \sqrt{(e_i.x - e_j.x)^2 + (e_i.y - e_j.y)^2}$. In Figure 7.2(a) the distance between the first element of each trajectory, $T[1]$ and $T_1[1]$, is 0.05 because besides the Euclidean Distance in the dimension space is $\simeq 0.22$ ($\sqrt{(1-1.1)^2 + (5.3-5.1)^2} \simeq 0.22$) the square distance is $\simeq 0.05$, that is equivalent to directly perform $(1-1.1)^2 + (5.3-5.1)^2 \simeq 0.05$. The distances between $T[1]$ and all elements in $T_1$ are 0.05, 0.09, 0.25, and 0.36, corresponding to the first row in Figure 7.2(f).

## 7.2 COMPUTING THE DISTANCE BETWEEN SUBTRAJECTORIES

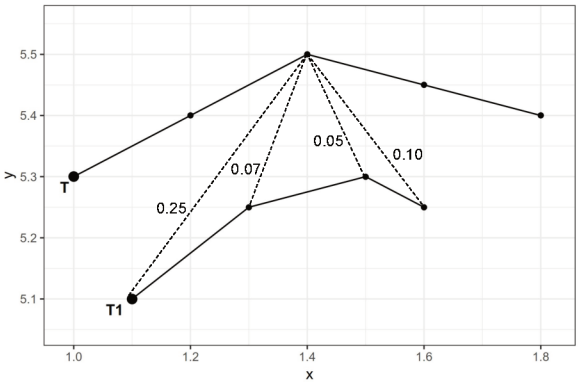
Following the previous running example, Figures 7.3 and 7.4 show the subtrajectory distance calculation between $T$ and $T_1$ for subtrajectory lengths 2 and 3, respectively. Figure 7.3 demonstrates how the subtrajectory distance calculation happens for subtrajectory length 2, $w = 2$. As mentioned, the algorithm calculates $A_w$ based on $A_{w-1}$ and $A_1$. *Note* that for $w = 2$



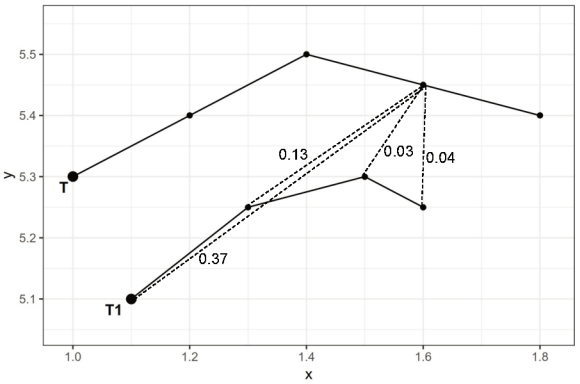

(a) Distances for the 1st element in T, j = 1.



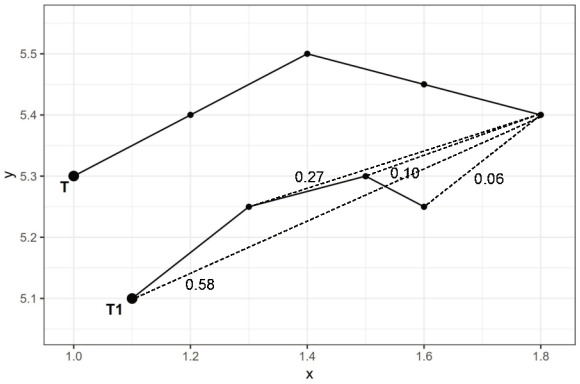

(b) Distances for the 2nd element in T, j = 2.




(c) Distances for the 3rd element in T, j = 3.




(d) Distances for the 4th element in T, j = 4.




(e) Distances for the 5th element in T, j = 5.

| $A_1[1,\ldots,.]$ | | *k* | | | |
|---|---|---|---|---|---|
| | | *1* | *2* | *3* | *4* |
| *j* | *1* | 0.05 | 0.09 | 0.25 | 0.36 |
| | *2* | 0.10 | 0.03 | 0.10 | 0.18 |
| | *3* | 0.25 | 0.07 | 0.05 | 0.10 |
| | *4* | 0.37 | 0.13 | 0.03 | 0.04 |
| | *5* | 0.58 | 0.27 | 0.10 | 0.06 |

(f) Two-dimensional array of distances between T and T1.

Figura 7.2 – Running example of computing element distances between two trajectories.

the array $A_{w-1}$ is in fact $A_1$. So, Figures 7.3(a-c) show the arrays $A_{w-1}[1,..,..]$, $A_1[1,..,..]$, and $A_w[1,..,..]$, respectively, where the arrays of Figures 7.3(a) and (b) are the same, because of $A_{w-1}$ is $A_1$ for $w = 2$. In this example, we calculate the distances of $A_2$ based on the array $A_1$.

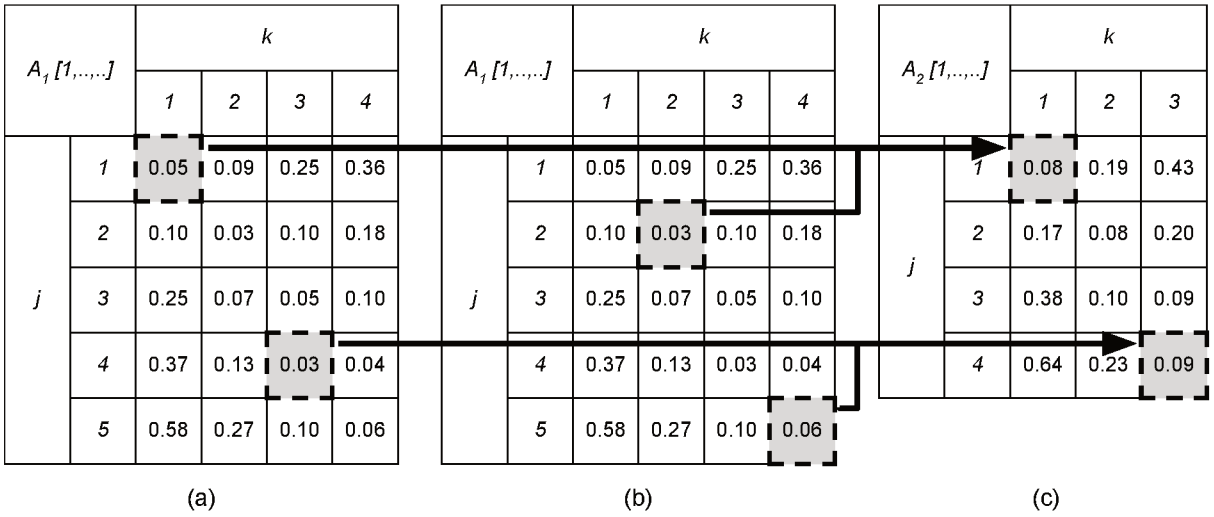



Figura 7.3 – Running example of subtrajectory distance calculation for $w = 2$.

The first distance calculation is for the subtrajectory of length 2 in $T$ starting at position $j = 1$ and the subtrajectory in $T_1$ starting at position $k = 1$ of the same length. This distance value is calculated by the sum of the subtrajectory distance $A_{w-1}[1,1,1] = 0.05$ in Figure 7.3(a) and the element distance $A_1[1,2,2] = 0.03$ in Figure 7.3(b), and the result is stored in $A_w[1,1,1] = 0.08$, according to Figure 7.3(c). Following, the last calculation is for the subtrajectory of $T$ starting at position $j = 4$ and the subtrajectory in $T_1$ starting at position $k = 3$, which is computed as the sum of the subtrajectory distance $A_{w-1}[1,4,3] = 0.03$ in Figure 7.3(a) and the element distance $A_1[1,5,5] = 0.06$ in Figure 7.3(b), that is stored in $A_w[1,4,3] = 0.09$, as in Figure 7.3(c).

Figure 7.4 demonstrates the subtrajectory distance calculation for subtrajectory length 3, $w = 3$. In this example, we calculate the distances of $A_3$ based on $A_2$ and $A_1$. In Figure 7.4 the first distance calculation is for the subtrajectory of length $w = 3$ of $T$ starting at position $j = 1$ and the subtrajectory in $T_1$ starting at position $k = 1$. This distance value is the sum of the subtrajectory distance $A_2[1,1,1] = 0.08$ in Figure 7.4(a) and the element distance $A_1[1,3,3] = 0.05$ in Figure 7.4(b). The result is stored in $A_3[1,1,1] = 0.13$, as in Figure 7.4(c). Finally, the last calculation is for the subtrajectory in $T$ starting at position $j = 3$ and the subtrajectory in $T_1$ starting at position $k = 2$, which is computed as the sum of the subtrajectory distance $A_2[1,3,2] = 0.10$ and the element distance $A_1[1,5,5] = 0.06$, that is stored in $A_3[1,3,2] = 0.16$.

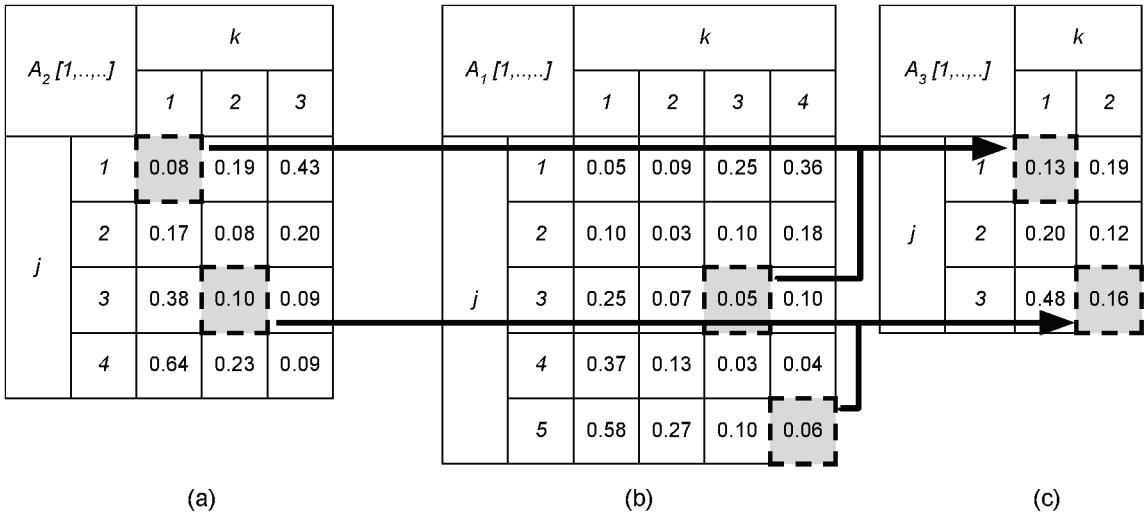



Figura 7.4 – Running example of subtrajectory distance calculation for $w = 3$.