Analysis of Trajectories by Preserving Structural Information

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt
von
Ahmed Jawad
aus
Chishtian, Bahawalnagar, Pakistan

Bonn, 2012

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Stefan Wrobel
2. Gutachter: Prof. Dr. Christian Bauckhage
Tag der Promotion: 10.07.2012
Erscheinungsjahr: 2012

Ahmed Jawad

University of Bonn
Department of Computer Science III

and

Fraunhofer Institute for Intelligent Analysis
and Information Systems IAIS

Schloss Birlinghoven
53754 Sankt Augustin
Germany

ahmed.jawad@iais.fraunhofer.de

Contents

Abstract

The analysis of trajectories from traffic data is an established and yet fast growing area of research in the related fields of Geo-analytics and Geographic Information Systems (GIS). It has a broad range of applications that impact lives of millions of people, e.g., in urban planning, transportation and navigation systems and localized search methods. Most of these applications share some underlying basic tasks which are related to matching, clustering and classification of trajectories. And, these tasks in turn share some underlying problems, i.e., dealing with the noisy and variable-length spatio-temporal sequences in the wild.

In our view, these problems can be handled in a better manner by exploiting the spatio-temporal relationships (or structural information) in sampled trajectory points that remain considerably unharmed during the measurement process. Although, the usage of such structural information has allowed breakthroughs in other fields related to the analysis of complex data sets [18], surprisingly, there is no existing approach in trajectory analysis that looks at this structural information in a unified way across multiple tasks. In this thesis, we build upon these observations and give a unified treatment of structural information in order to improve trajectory analysis tasks. This treatment explores for the first time that sequences, graphs, and kernels are common to machine learning and geo-analytics. This common language allows to pool the corresponding methods and knowledge to help solving the challenges raised by the ever growing amount of movement data by developing new analysis models and methods.

This is illustrated in several ways. For example, we introduce new problem settings, distance functions and a visualization scheme in the area of trajectory analysis. We also connect the broad field of kernel methods to the analysis of trajectories, and, we strengthen and revisit the link between biological sequence methods and analysis of trajectories. Finally, the results of our experiments show that — by incorporating the structural information — our methods improve over state-of-the-art in the focused tasks, i.e., map matching, clustering and traffic event detection.

Acknowledgements

Some things are just hard to explain through words. For example, the tremendous amount of courage, trust and optimism that Prof. Stefan Wrobel showed when he replied positively to an email containing my PhD application from Pakistan. Four years down the road, I admire his act even more, when I can guess better about the number of PhD applications that he receives. I am really happy that the opportunity he provided has culminated into a written thesis. He has continued to show this trust and support throughout my PhD and has contributed with invaluable suggestions in order to improve this thesis. I hope that the final product will be able to fulfil his expectations.

Dr. Kristian Kersting (or simply Kristian) has shaped and guided the development of this thesis more than any other person. He has been with me through the most difficult times of my PhD. He has shown trust and spent time over the gritty details when things were not working, and he has looked at the notation and taught me how to write a technical paper when some of them did. I must say that as a passionate and productive researcher in the field of machine learning, he has done a great job in balancing his time for the continual guidance to students in his group STREAM and in particular supervising my thesis, which is mainly aimed at bringing geo-analytics and machine learning closer. I cannot again describe in words the happiness that I feel when I think about his contributions and what we have accomplished together.

I also thank Gennady and Natalia Andrienko for their active support in the development of this thesis. They have been really nice and have continually shown me the path ahead through their discussions, suggestions and provided me with the data sets pivotal to my research.

During this PhD, I have been fortunate to be a part of a very young and energetic research group named STREAM (STatistical Relational Mining). All the members including Marion Neuman, Babak Ahmadi, Fabian Hadiji, Mirwaes Wahabzada, Zhao Xu, Martin Mladenov, Youssef El Massoudi and Novi Quadrianto (our honorary member) have contributed immensely by providing really valuable and technical feedback.

There have been many colleagues at Fraunhofer Institute who have helped me a lot; to name a few: Hans Voss, Michael May, Thomas Gärtner, Mario Boley, Katerina Vorotsu, Anja Pilz, Thomas Liebig, Christine Körner, Eike Stuckert, Ahmet Ocakli, Myriam Jourdan, Daniela Börner and Renate Henkeler.

Some of the teachers who have an taught me a lot about computer science and machine learning and therefore have a direct impact on this thesis include: Asim Karim, Sohaib Khan, Ashraf Iqbal, Arif Zaman, Amjad Luna, Umar Saif, Nabil Msutafa and Sarmad Abbasi.

No strenuous task can be accomplished without friends who lend an unconditional support under the most demanding circumstances. These are the people who have contributed the most to being who and where I am — including Aamer Zaheer, Munawar bin Abad, Saqib Ashfaq, Peer Ali Kirmani, Murtaza, Asim Sharif, Mazhar Karam Shah, Hammad Thandoo, Arman Sarwar, Zeeshan Chawala, Mamoon-ur-Rashid, Usman Bhai, Syed Sajjad Hassan, Shamoon, Abdullah Butt, Awais Karim Bajwa, Rizwan Fazal, Khizar Hayat, Wasim Kaka, Sajid, Mian Jamshed, Bilal, Junaid Naeem, Dilware khan, Shehzad Cheema, Rashid, Tariq and Shaami Rehmani and Azam Javed (Chaand). I would specially like to mention two of the dear friends who passed away, i.e., Mohammed Ahmed and Saeed Taggar.

During my whole life, I have felt blessed that I had the support of a wonderful and encouraging family. No words can describe their contribution to this thesis. Above all, the two ladies: my mother for being the one who dreamt and worked continuously for my future in a purely altruistic manner, and my wife who has patiently endured the other side of a PhD student. Hina! I love you so much. And more so for giving me the most wonderful gift of my life, our son, Aadil. I also hope that my father — the sweetest person in my life — and my other family members, especially my very supportive brother Hammad, my sisters, Shazia and Maria, uncle Maqbool and aunt Nasreen feel proud of me at finishing this thesis.

In fact, it is very hard if I had to choose one person to dedicate this thesis to. Especially, the closest people in my life; *Abbu ji* (my father) — who has supported me, loved me, taught me, and I can forget the hardships in the rest of the world when I talk to him, or *Hammad* (my brother) — who supported my studies and paved the way for this PhD, or *Hina* (my wife) — who has had the patience to go through the PhD studentship with me when I was —at best— remote, or *Aadil* (my son) — who stands as a symbol of love, happiness and hope in our lives. However, I dedicate this thesis to *Ammi Ji* (my mother) for her steadfastness, her belief in me and her acceptance of nothing but the best from me.

1. Introduction: Analysing Trajectories

> *... the unpublished maps that we make ourselves, of our city, our place, our daily world, our life; those maps of our private world we use every day; here I was happy, in that place I left my coat behind after a party, that is where I met my love; I cried there once, I was heart-sore; but felt better round the corner once I saw the hills of Fife across the Forth, things of that sort ..." — Alexander McCall Smith, Love Over Scotland*

The analysis of trajectories is an important area of research in geographical data analysis (or geo-analytics). Two of the main problems in this area are: (1) — coping with the measurement error (e.g., spatio-temporal distortion, heterogeneity, and missing values) during the sampling of trajectories, and (2) — the comparison of large and invariable length spatio-temporal sequences. In our view, these problems can be handled in a better manner by exploiting the structural information present in the sampled trajectory data. Surprisingly, this perspective has not been provided by the existing approaches. Inspired by the usage of structural information in the analysis of complex data sets [27] for machine learning, computer vision and related fields, where it has provided breakthroughs, we step ahead with this idea, and in this introduction, lay the motivations for providing a unified treatment of structural information in the analysis of trajectories. Then, we discuss the basic trajectory analysis tasks, namely, map matching, trajectory clustering and traffic event detection. During the rest of the thesis, these tasks will serve as a test bed for the usage of structural information in order to improve the analysis of trajectories. However, before starting off, lets' briefly describe the underlying motivations and contributions of this thesis.

Analysis of complex trajectory data in order to provide intelligent location base services has brought forth a change in our usage of computers to mine data. As Mitchell pointed out [41], we are beginning to analyse our reality — data recording personal activities, conversations, and movements — in space and time 'in an attempt to improve human health, guide traffic, and advance the scientific understanding of human behaviour' in general. Consider, for example, an *intelligent* traffic

monitoring system that re-distributes the traffic flow in anticipation of a congestion by recognising the early patterns of a traffic jam [34]. Obviously, such an intelligent system can potentially overcome the long standing temporal and spatial boundaries to our perception of movement [53]. Consequently, it comes as no surprise that analysing trajectories (in particular, spatio-temporal data from movement sensors) is currently receiving a lot of attention.

Successful approaches in this area, however, cannot be easily designed/developed as the datasets contain noisy and variable-length spatio-temporal sequences in the wild. In our opinion, one step towards achieving this goal is to improve some underlying *basic tasks* that are shared across multiple applications in the analysis of trajectories. To illustrate, map matching (i.e., the process of aligning raw trajectories to street network) is *one of such basic tasks* that is used in navigation, planning and transportation systems. Another important task is the comparison of trajectories (or similarity computation), which is necessary for extraction of movement patterns, clustering, and prediction based solutions. Finally, the probabilistic modelling of samples from a set of trajectories is important in order to execute membership queries for this noisy data set. For example, traffic event detection is such an application, where we build a model of trajectories relating to traffic events, e.g., jams and congestions, so that a new event can be identified.

In general, although, these basic tasks in trajectory analysis have different settings, yet a common problem in solving them is coping up with the measurement error [28]. This error makes it difficult to know the exact ground truth , and is caused, e.g.,by measuring devices, namely, GPS devices, sensors, and RFIDs that generate samples of original trajectories (curves tracked by the moving objects) recorded at different time and space intervals. It spans a combination of factors, e.g., spatial and/or temporal noise, low and non-uniform sampling rate, heterogeneity of data (noisy label information), and missing values.

To a large degree, however, the success of an algorithm in the area of trajectory analysis depends upon the capacity of this algorithm to handle the error in the data. To illustrate, a map matching algorithm that does not deal robustly with the spatio-temporal distortion will assign wrong streets to GPS points, and hence, it will result in a greater error over larger values of noise. Similarly, the performance of an event detection solution is affected by its' capacity to deal with heterogeneous data (noisy labels information about training and test events), and missing values. Finally, the alignment and comparison algorithms need to consider the spatio-temporal distortion and the non-uniform sampling rates while calculating the distances between

data points.

On the other hand, many —if not most— of the existing solutions for trajectory analysis take one of the following paths:

1. Either, they assume that the sampled data points belong to original trajectory, hence ignoring the performance compromise made in the process.

2. Or, focusing on the task at hand, they model the error using distributions and probabilistic methods that empirically deal with it in the best manner.

Surprisingly, both of these existing approaches, to a certain extent, ignore the domain invariants (i.e., the structural relationships between points) that are not disturbed a lot by different forms of noise, and remain considerably unharmed during the measurement process. Motivated to solve the real-world traffic problems in trajectory analysis, we argue that the approaches, which take the factor of structural information into consideration can produce better results. We also argue that such approaches can prove to be the next step towards improving the core analysis tasks, i.e., classification, clustering, and matching in trajectories.

In this thesis, we present these arguments, and provide a novel and unified way of using structural information in order to improve trajectory analysis tasks. In particular, we manage this by taking trajectories as walks over labelled graphs with nodes (or labels) in continuous/ discrete spaces, and edges representing differences between the adjacent labels in spatial, temporal or any other domain of interest. And, then by embedding the Geodesic distances among these labels into Euclidean space, we get Euclidean distances between all pairs of nodes that preserve structural information inside the data. Once we have these distances, we can make a connection to other Euclidean distance based approaches. For example, kernel methods, which have never been used in the area of trajectory analysis. Or, we can use these distances to apply out-of-the-box biological sequence methods that also work upon such labelled graphs. This results in novel similarity functions and visualization schemes for trajectory analysis. In the end, we apply these connections to solve three of the core tasks in trajectory analysis, i.e., map matching, clustering, and traffic event detection for trajectory data. The results of our experiments show that —by incorporating the structural information— these embeddings improve over state-of-the-art in map matching, clustering, and traffic event detection from trajectory data.

1.1. Goals and Contributions

As mentioned, an improvement in the solutions of real world trajectory analysis problems is the seed that grew into our usage of structural information into this thesis. In this thesis, we look at the structural information in trajectory data sets, and provide a method to use it in order to reach our stated objective. For applications, we focus our attention on three of the basic tasks in trajectory analysis; namely,

1. map matching [49] (a fundamental step in analysing the street network based trajectories),

2. the clustering of trajectories (for the street network based trajectories and stay-point sequences). We do this with a motivated setting of understanding the individual human mobility patterns [23],

3. and traffic event detection [29].

In order to solve these tasks, we take a generic definition of trajectories as walks over labelled graphs that can address the analysis problems in these tasks. This definition applies to most of the trajectory analysis area. Then, by embedding the Geodesic distances between labels in these trajectories into Euclidean space, we get new distances that preserve structural information inside the data and are used in combination with analytical methods to solve these tasks. The analytical methods that we use are based upon machine learning (in particular, kernel methods) and biological sequence analysis. More specifically, our contributions[1] are:

1. We improve upon state-of-the art for three above mentioned tasks (on real-world data sets).

2. We introduce novel distance functions (based upon the geo-desics of trajectories) for map matching and alignment tasks. These functions include the so-called 'spatio-temporal kernel over Euclidean distances' for map matching and 'Local similarity based alignment kernel' for clustering and classification tasks. While doing so, we connect the broad field of kernel methods and the analysis of trajectories. Moreover, by virtue of being kernelized, our work is extensible, i.e., our distance functions give an opportunity to integrate existing kernels in the field or new kernels in the future.

[1] The work in this thesis appears in the following papers [30, 34, 31, 33, 32]

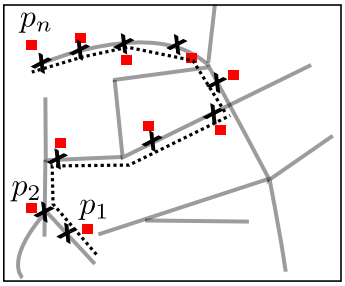


Figure 1.1.: Illustration of map matching problem. Given a graph $G = (V, E)$ denoted by gray edges and a trajectory $T = \{p_1, p_2, ..., p_n\}$ denoted by red squares, find the corresponding ground truth points for each p_i on the graph denoted as crosses.

3. In this thesis, we show that biological sequence methods with our distance functions can produce state-of-the-art results for the tasks of clustering and event detection in the trajectory datasets[2].

4. Further investigation into biological sequence methods yields 'Traffic Logos', which is a novel visualization in the traffic domain giving a compact and descriptive representation of the patterns in the data.

5. We introduce new settings and algorithms for the trajectory analysis tasks, e.g., for map matching when we take it as a multi-regression problem and then solve it through a combination of embedding and rounding approach. We also show that the profile HMMs can be used to capture the dynamics of the events in the traffic event detection which is not previously considered.

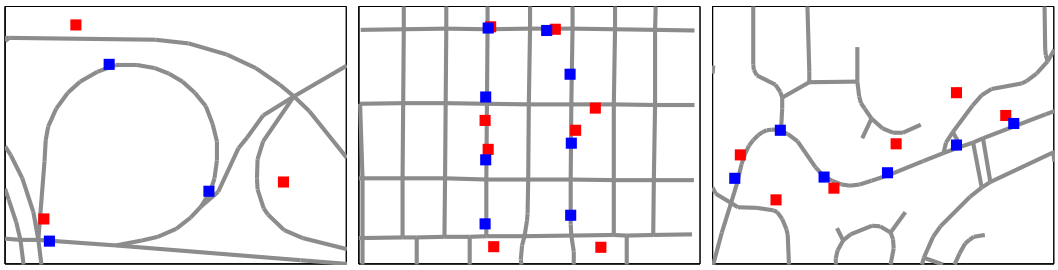
1.2. Focused Tasks in the Area of Trajectory Analysis

Having listed our contributions, we introduce the basic tasks in the area of analysis of trajectories that will serve as a test bed for the usage of structural information in the rest of this thesis.

1.2.1. Alignment of Raw Trajectories to Streets (Map Matching)

Map matching — the process of assigning the raw trajectories to street network — is a fundamental operation in many applications such as traffic analysis and location-

[2]Biological sequence methods also work on the label sequence graphs with unit weight edges and the distances similar to the ones learned by us.



(a) KMM: over a roundabout

(b) KMM: over grid structure

(c) KMM: Robustness to noise

Figure 1.2.: KMM's performance on challenging real-world situations. The red points show the input trajectory and the blue points the output path by KMM. In all three cases, KMM recovers the exact ground-truth paths. (coloured)

aware services, and, it is becoming increasingly important due to pervasiveness of measuring devices (e.g., Wifi, Mobiles, sensors inside building and vehicle navigation systems). The sampled trajectories from these devices contain noise ranging from 10 to 100 meters due to different capacity and environment issues during measurement process. Consequently, we have multiple choices of street for a sample trajectory. The process of disambiguation between these multiple paths (or alignment of sample trajectories to street network) is the core difficulty problem that we solve through the usage of structural information in raw trajectories. Fig. 1.1 describes the map matching problem with example input points and the output path on street network.

Triggered by the observation that matching a trajectory of coordinates would be easy if the observed coordinates were noise-free — the coordinates would simply constitute the solution — one may propose to treat the map matching problem as a regression problem. The task is now: estimate the noise-free function from the noisy observations. In order to solve the problem, we reduce the noise from the input trajectory by maximizing the structural similarity between the input graph and relevant part of the street map. The resulting relaxed assignment is then "rounded" into a hard assignment fulfilling the map constraints.

On synthetic and real-world trajectories, we show that our method can be used for map matching and performs well compared to probabilistic methods such as HMMs. Especially, our method performs significantly better for higher values of noise, which shows the robustness of the structural information when dealing with noise.

Fig. 1.2 shows the results of KMM performance in three challenging real life situations.

1.2.2. Trajectory Clustering

Clustering of trajectories is an important task in order to identify the groups of moving objects that exhibit similar movement behaviour. Many applications in trajectory analysis come under this scenario. For example, consider 'finding flocks of birds by identifying the clusters in their trajectories' or ' clustering trajectories of an individual human to identify her routes (user activity analysis)'. Roughly, these clusters represent frequent spatio-temporal paths that the moving objects have chosen in the data. For example, during user activity analysis, different clusters identify user's typical spatio-temporal routes from 'work to home', 'home to work' or 'work to shopping'.

However, we can do better. For example, we can cluster a user's trajectories at different abstraction levels. To illustrate, instead of clustering raw trajectories, we can do a map matching over user's trajectories and then the clustering of these map-matched trajectories will give us the street-network paths of high frequency that a user follows. Another higher level of detail can be found by identifying the stay points (or places with longer stays in trajectories) and then the trajectories can be defined in terms of daily sequences of these stay points. The cluster of such trajectories will now specify different groups of daily activities that a user follows, e.g., on work days and weekends.

In this thesis, we encode both of the abstraction scenarios for a user's trajectories into label sequence graphs. To illustrate, for the street network based trajectories, the labels (or nodes) denote corners of the streets in the street network and differences between connected labels is given by the distance between the nodes. For stay point sequences, the labels denote stay points and edges denote average temporal differences between the connected stay points of the user from data. Then we embed geodesic distances in these graphs to come up with Euclidean distances between the labels. These distances can be used in conjunction sequence similarity measures to compare and align multiple sequences. Following this, we use out-of-the-box biological sequence methods to align and cluster trajectories. Tab. 1.1 represents the results of the clustering for street network based trajectories as compared to a state-of-the-art clustering method and shows that we produce more clusters and hence reveal more patterns in the users' data. Furthermore, on the similar number of clusters found, we produce less error. Our techniques are also superior in the sense that they can be applied to a broader range of trajectory definitions as shown in the thesis.

Similarity method	Clusters	μ_{RMS}	σ_{RMS}	$\#objs(\mu \pm \sigma)$
Route Search	7	271.5	NA	261
Pairwise	7	249.24	36.4	255.6 ± 7
Traffic	5	211.78	112.43	221.2 ± 21
Sequence	6	236.58	75.74	243.6 ± 14
Alignment	≥ 8	313.98	20.13	275.1 ± 6

Table 1.1.: Comparison of clustering results between our method and a state-of-the-art algorithm with Hausdorff distance as the error measure. The error is depicted in columns 3, and 4 as the the average and standard deviation of root means squared Hausdorff distance, i.e, μ_{RMS}, and σ_{RMS} for the corresponding rows. Similairly, the mean and standard deviations for number of objects found per row are given by the last column. The first row shows the result for the state-of-the-art baseline. All other rows shows the results for our alignment-based method. As we can see both capture a similar number of trajectories and similarly good clusters. Row 5 shows that the alignment-based method can capture more patterns than the original method.

1.2.3. Traffic Event Detection

Probabilistic modelling of samples from a set of trajectories is an important analytical task in order to execute membership queries for this set. This scenario builds up during different trajectory classification schemes. For example, during traffic event detection, where we build a model of trajectories relating to traffic events[3], e.g., jams and congestions, so that a new event can be identified.

For this purpose, consider the time-series based movement data, where a sensor records movements of entities over a short window of time. For example, an optical sensor placed over a door of an office building reports an estimate of people count entered on a 30-minute basis. Or, an inductive loop sensor on a highway reports an estimate of vehicles passed on a 5 minute basis [29]. This recorded data captures periodical patterns of human activity, e.g., highways are usually busy during morning and early evening time because of traffic 'towards and from' work place. Weekdays and weekends can show periodic patterns of their own. Typically, these *periodical activity* patterns are mixed in sensor data with bursts of unusual traffic called 'events'; outliers but not noise. Example events include: traffic congestion/jams on a high way, a large meeting in an office or a concert/football game near a highway sensor. Thus, we have to separate the normal traffic activities from the traffic events.

[3] http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/

Algorithm	Training Events	True Positives	False Positives
Dodger's Base ball Game Prediction - Total events= 76			
Baseline	15	76	65
Poisson Proc.	76	75	23
HMM profile	15	74	18
Caltech Auditorium event prediction - Total events= 29			
Baseline	10	29	43
Poisson Proc.	29	24	24
HMM profile	10	25	12

Table 1.2.: Comparison of event prediction on real world data sets. In both cases, HMM profiles were able to predict almost the same number of events with a better recall (lower number of false positives) and a low training percentage of data. On average, 90 percent of original events are captured by all algorithms. However, out-of-the-box profile HMMs provide a better recall (filtering of false alarms) by capturing event persistence.

Unfortunately, there are no labels, which leads us to a problem, i.e., *the separation of normal traffic from event*. Furthermore, an event is not a single unusually high value, instead it is a chain of sensor reading having its own dynamics. For example, a traffic jam or congestion usually has a normal like curve for traffic frequencies i.e, a plot of traffic frequencies in a jam over increasing time will show that the traffic frequencies build up over time and then slowly disperse. On the other hand, a conference event in a building is identified by having bursts of traffic at two separate occasions, i.e., start and finish of the conference with a small amount of people leaving the area during the conference. In order to capture these dependencies, we need to employ a probabilistic model, which decides that the random unusual measurements of traffic frequency does not identify traffic events; instead, a traffic event is specified by a row of unusual frequencies with structural dependencies between temporal elements. In this thesis, we learn a profile hidden Markov model (a biological sequence method) from a training set of events that captures the event dynamics by encoding the structural dependencies between frequency bins from sensor readings (or, in other words, nodes of our label sequence graph). We compare our results to a state-of-the-art event detection method. Tab. 1.2 shows the results of our method on two real world data sets (for base ball game prediction at Dodger's baseball stadium and Caltech auditorium event prediction) revealing that we improve upon state-of-the-art (by giving same true positive rate and lower false positive rate) just by applying out-of-the-box profile HMMs.

Summary

In this chapter, we have briefly described the underlying motivations and contributions of this thesis. Furthermore, we have also introduced some of the fundamental tasks in the area of trajectory analysis, i.e., map matching, the clustering of trajectories and traffic event detection. We will resolve these tasks through the usage of structural information in the rest of the thesis, and they will also serve as a test bed for the usage of the structural information in order to solve analytical problems related to trajectories.

Related Publications

[1] A. Jawad and K. Kersting. Kernelized map matching. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*, pages 454–457, Seattle, Washington, USA, 2010.

[2] A. Jawad and K. Kersting. Kernelized map matching for noisy trajectories. In *Working Notes of Knowledge Discovery and Machine Learning (KDML) at LWA2010 - Learning, Knowledge & Adaptation*, pages 1–10, Kassel, Germany, 2010.

[3] A. Jawad, K. Kersting, and N. Andrienko. Biological sequence analysis meets mobility mining. In *Working Notes of Knowledge Discovery and Machine Learning (KDML) at LWA2011 - Learning, Knowledge & Adaptation*, pages 73–80, Magdeburg, Germany, 2011.

[4] A. Jawad, K. Kersting, and N. Andrienko. Building bridges between traffic and biological sequence analysis. In *International Workshop on Finding Patterns of Human Behaviors in NEtwork and MObility Data – NEMO at ECML PKDD (Europen Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases)*, pages 13–27, Athens, Greece, 2011.

[5] A. Jawad, K. Kersting, and N. Andrienko. Where traffic meets dna: Mobility mining using biological sequence analysis revisited. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*, pages 357–360, Chicago, Illinois, USA, 2011. Best Poster Award.

2. Preserving Structural Information

Many a trip continues long after movement in time and space have ceased
— John Steinbeck

In this chapter, we provide a generic mechanism to convert two fundamentally different types of trajectories, i.e., raw and symbolic trajectories into walks over underlying labelled graphs. Then, we define similarity measures over these trajectories that will preserve the structural information for analytical purposes. These similarity measures will be used to improve the analytical tasks for trajectories in later chapters. In short, this is a very important chapter that lays down the theoretical foundations of this thesis and remains a binding force throughout. However, before starting off, we briefly describe our motivation to use graphs in order to capture the notion of structure in trajectories.

The analysis of structured datasets lies at the heart of many computing endeavours, aiming to improve the averages and quality of our lives, e.g., in bio-informatics, chemi-informatics, network based traffic analysis, social networks and world wide web. Arguably, graphs present the most natural and powerful model to capture the properties of objects and their relationships for these data sets [65, 46]. Some of the manifestations of graphs in the above mentioned fields include street networks, phylogenetic trees, RNA structures, molecular structure, XML files, database schemas and web ontologies. Furthermore, their almost universally accepted usage in the above mentioned fields gives them a unique ability to serve as an analytical connection for many research fields related to structures data.

As discussed, we are motivated to improve trajectory analysis tasks by preserving the structural information. We have also discussed that trajectories can be viewed as graphs with some additional constraints (in particular, labelled graphs). However, we have not discussed yet the concrete approach that we will chose to represent trajectories as labelled graphs and the consequences of this approach. We have also not discussed that how our modelling of trajectories will helps us in reaching the end goal of our research, i.e., improvement in the focussed tasks of trajectory analysis. As we will see later, our task will become easier by building on top of the

great research work done in the fields of machine learning and data mining. In this chapter, we provide the theoretical foundations to answer these questions for the reader. We start off by connecting trajectories with structure.

2.1. Trajectories and Structure

Next to the traditional definitions of trajectory data, our approaches build upon the natural representation of structured data used in machine learning and many other fields, i.e., graphs [46, 17].

Definition 1 (Graph) *A graph G is a pair of sets, $V(G)$ and $E(G)$, corresponding to vertices and edges of the graph, i.e., $G = (V, E)$. An edge in $E(G)$ defines a relationship between the vertices of the graph, i.e., $e_{i,j} = (v_i, v_j)$ where $e_{i,j} \in E(G)$ and $v_i, v_j \in V(G)$.*

A graph G is said to be a directed graph,if an edge between two vertices v_i and v_j is not a symmetric relationship between the nodes, i.e. $(v_i, v_j) \neq (v_j, v_i)$. Essentially, this means that we are adding a direction to an edge.

An undirected graph has no directions added to its edges. Furthermore, if the edges of a graph are assigned weights then we call such a graph a weighted graph. Now, we come to the definition of neighbourhood and adjacency in a graph.

Definition 2 (Adjacency in a Graph) *Two vertices v_i and v_j are said to be adjacent (neighbours) in a graph if there is an edge between the vertices, i.e., $(v_i, v_j) \in E(G)$.*

The adjacency information of a graph is represented by the so-called adjacency matrix A. This is an $n \times n$ matrix where $A_{i,j} = 1$, if, there is an edge from v_i to v_j otherwise it is 0. The degree of a node v in a graph G is denoted by $d(v)$ and is equal to the number of adjacent nodes of v.

It is often very interesting, in a graph, to look at the information in the sequence of connected nodes, which is useful for the both analytical and application purposes. These sequences can be essentially represented by paths, walks or cycles.

Definition 3 (Walk, Path) *A walk w in a graph G is a sequence of connected nodes from $V(G)$, i.e., $w = (v_i, v_{i+1}, ..., v_j)$. A walk w is called a path p, if all the nodes in w are distinct. However, if their is a walk where $v_i = v_j$, then w is considered a cycle.*

The length of a walk is: either, the number of edges that join the successive nodes in w (in an un-weighted graph), or, it is the sum of weights for those edges (for a weighted graph).

Graphs have many variants to capture the different flavours of structured data. For example, we can label the nodes and edges of a graph (known as labelled graphs) in order to deal with the attribute information presented in structured data. Labelled graphs are a very flexible and generic data structure having profound applications and analytical methods. In our case, we can take the help of labelled graphs to connect trajectory analysis with the existing ideas in machine learning, i.e., representing trajectories as (walks over) labelled graphs and then apply the analytical methods in machine learning over these graphs. In order to carry out these tasks, we define labelled graphs as following

Definition 4 (Labelled Graphs) *A labelled graph $G(V, E, L)$ is a graph, where the nodes and edges are assigned labels , i.e., $\Omega : V \cup E \rightarrow L$, the set of nodes V and edges E are mapped to the set of node and edge labels L. Additionally, $l(v)$ denotes the node of a label while $l(e_{i,j})$ denotes a label associated with an edge e between node i and j [52].*

Sometimes, we define constraints on the nodes and edges of a graph to model the task at hand and improve the overall complexity/ performance of the solution. One of the examples of such graphs is a Euclidean graph, which is a labelled graph $G(V, E)$ with $v \in V :\mapsto \mathcal{R}^d$ meaning that the vertices are embedded in Euclidean plane (vertex labels belonging to $\mathbf{R}^2$). An Euclidean graph is defined as follows:

Definition 5 (Euclidean Graph) *An Euclidean graph is a labelled graph in which the vertex labels represent points in the Euclidean plane, i.e., $v \in V :\mapsto \mathcal{R}^d$, and, the edges are assigned lengths equal to the Euclidean distance between those points, i.e., a straight line between corresponding vertices.*

Street network is another example of the datasets in trajectory analysis that have an Euclidean Graph representation through their poly-line structure and their vertices embedded in $\mathbb{R}^2$. To make it explicit, we can add additional vertices to every corner of the lines constituting a road segment. We define the street network graph as following:

Definition 6 (Street Network Graph) *A street network graph$G(V, E)$ (illustrated in Fig. 2.1) is a Euclidean graph where V is the collection of terminal points and*

Figure 2.1.: Street Network of Oldenburg, Germany

E is the collection of edges for street network. The vertex labels belong to Euclidean plane, i.e., $V \in \mathbb{R}^2$ and the edge labels denote the length of the straight line between those vertices.

The edges in a street network [1] can be directed (denoting a one way street) or undirected (denoting a two way street). Furthermore, the speed limits and other information can be appended to the labels of the edges, and then, a sampled trajectory from a vehicle can be considered as a walk over such graph.

However, trajectory is a more generalized notion than a sequence of vertices embedded in Euclidean plane; For example, consider a time and space tagged photo sequence describing the movement of a tourist in a city. Thinking along the same lines, a better definition of a trajectory is given by Giannotti. et. al. [21]), which views trajectory as a temporally taggged sequence of symbols. Technically speaking, every element in this sequence has a time stamp and the symbols of the sequence can belong to $\mathbf{R}^2$, or discrete alphabet representing a location category of the underlying space. In our view, even this notion restricts the relationship between vertices of a trajectory to a temporal relationship and it does not consider the graph information related to almost all trajectory scenarios. Consequently, in this thesis, we take a definition of trajectories that helps us in understanding the nature of structural

[1] For the rest of the thesis, we will use the term 'street network' instead of street network graph.

information in this data. In particular, we take the trajectory as a walk over an underlying labelled graph, where the node labels can belong to Euclidean plane or a discrete alphabet, and the edges between these labels can represent any function of the corresponding nodes mainly learned from the data, for example: average speed between two locations, average time spent or simple Euclidean distance between the nodes. Formally, we define a trajectory (for the rest of the thesis) as following:

Definition 7 (Trajectory) *A trajectory T is a specific type of a label sequence graph, i.e., walk over an underlying labelled Graph $G(V, U, L)$, where, $l(v) \in (\mathbf{R}^d$ or $\Sigma)$, i.e., a continuous plane or a discrete alphabet, and $l(e_{i,j}) = f(v_i, v_j)$.*

Essentially, what we are saying on top of the temporally tagged sequence definition is that the edges can represent any function of the corresponding nodes (the edge weights can learned or approximated from the data as well). In the coming sections, you will see that —not only— our view of seeing trajectories as labelled graphs helps us in preserving structural information —but also— it paves the way for further connections to the exciting fields of kernel methods and sequence analysis.

For a start, recall that one of the motivations of this thesis is to deal with the complex spatio-temporal nature of the data in trajectories by reducing the problems of noise and missing labels in our trajectories. If we are able to achieve this target, we can have better idea of our label positions and we can define the distances upon these labels. Intuitively, we assume that although the local positions of the labels are perturbed, however, the structural relationships (or graph distances) between these labels are considerably close to the true values. Thus, in order to learn the true distances between our labels, we can embed these geodesic distances in a Euclidean plane. By doing so, we will get metric-based distances between the labels of the trajectories, which can prove our connection to a strong machinery of kernel methods and sequence analysis (used to improve trajectory analysis tasks).

In short, our approach to preservation of structural information mainly comprises the following steps:

1. Embedding the geodesic distances in the labelled graphs (defining trajectories) into Euclidean space

2. Exploiting the output distances in the previous step to form new kernels for raw and symbolic trajectories.

3. These kernels provide us a connection the to kernel based machinery and other exciting fields like biological sequence analysis to solve the task at hand.

In the coming sections, we provide brief introductions to Embeddings, and kernel methods in order to define our approach for raw as well as symbolic trajectories.

2.2. Preserving Structure Through Embeddings

The work in this thesis builds heavily upon the idea of structural embeddings (or non-linear dimensionality reduction) [48, 24]. Although, embeddings are mostly used for the visualization of high dimensional spaces on a 2-D plane, however, they give a general framework to transform data between spaces of different dimensionality or orientations. In principle, the relationship between the two spaces can vary from a very simple to complex one, e.g., we can try to embed the points on a sphere or a manifold to a plane and vice versa, where our goal is to come up with the points in the output space having similar properties and relationships to the points in input space. Another idea, similar to that of embedding is graph matching, where we try to come up with nodes in the output graph having similar properties and relationships like the input graph nodes. Graph matching therefore is a specific version of embedding, where the input and output spaces are structured and discrete with some functional constraints on mappings. In many real world applications, we have data sets, where we have to find a match between the topology of two networks ranging from shapes, geometric information extracted from images, tertiary proteins, and street networks. Also, in many of these problems we are required to find a partial or approximate match between these two curves because of the uncertainty and corruption that exists in problems relating to the most real world data. Examples of this include:

- Map Matching where we are required to find a small curve (trajectory) inside a larger collection of curves (street network) having some spatial and structural similarity to the matched part [6] .

- Partial shape matching based on skeleton graphs where we find an object of interest inside a 3D scene or we try to find sub parts of shapes (skeletons), which are similar [13, 59, 57, 4]

In the following section, we give a short overview of the structural embedding techniques using the distance preservation as the main criterion to be preserved. We refer the interested reader to one of the elegant text books in this area, i.e., 'Non linear dimensionality reduction' by Lee. et. al. [38]. In the following, we have taken their text [38] as a basis and adapted it to our needs.

The distances in graphs cannot be perfectly preserved, however, it has been shown that structural embedding techniques preserve the local geometry and neighbourhood information in the output space. The intuition behind the distance preservation schemes is that any data set can be completely described by pairwise distances and hence, if a low dimensional representation is produced in a way that these pairwise distances are preserved then the dimensionality reduction is successful. Furthermore, the idea that a high-dimensional space, containing the data set of interest, is usually sparse with few dense regions (or clusters of data) gives rise to the practicality of preserving local geometry and neighbourhood during the process of embedding. In practise, there are a lot of distance preserving approaches for embedding. The most notable of these approaches are: principal component analysis (PCA), multidimensional scaling (MDS) and Iso-MAP [38]. The main difference between these approaches is based upon the type of distances preserved and the objective function used. In the next section, we provide a short overview of distance preservation in $\mathbf{R}^d$ (and, hence the name 'spatial distance preservation') and then go on to describe the geodesic distance preservation.

2.2.1. Preserving Spatial Distances

The term spatial distances refers to the fact that only the actual co-ordinates of the data points are used for distance calculation in embedding and the underlying structure of the data is ignored, for example, the Euclidean graph representation of trajectories. In the following subsection, we first introduce some basic definitions related to distances and metrics and then describe the Multidimensional Scaling (MDS), along with its variants, which is the most popular and classic spatial distance preservation method.

Metric Spaces, Distances and Norms

A metric space $\mathcal{M}$ is a couple $(\mathcal{Y}, d)$, where $\mathcal{Y}$ is a point set such that the notion of a distance between any two points $a, b \in \mathcal{Y}$ is defined. The distance function, denoted by $d(a, b)$ must satisfy the following properties for any points $a, b, c \in \mathcal{Y}$.

1. **Non-degeneracy.** $d(a, b) = 0 \Leftrightarrow a = b$
2. **Non-negativity.** $d(a, b) \geq 0$
3. **Symmetry.** $d(a, b) = d(b, a)$
4. **Triangle Inequality.** $d(a, b) \leq d(c, a) + d(c, b)$

The so-called dot or scalar product[2] is the most basic idea to define the notion of similarity between two vectors in metric spaces. And, for the usual Cartesian vector space $\mathbb{R}^D$, the distance functions can be derived by measuring the L_p norm of any order for the difference between two points. The L_p norm, denoted by $||x||_p$ for a point $\mathbf{x} = [x_1, ..., x_D]^T$ is defined as

$$||\mathbf{x}||_p = \left(x_1^p + x_2^p + ... + x_D^p\right)^{\frac{1}{p}} \tag{2.1}$$

Some of the most commonly distance functions based on L_p norm are city-block distance and Euclidean distances. To illustrate, for any two points $a, b \in \mathcal{Y}$

1. **City block distance.** The city block distance or the L_1 norm is given by the equation:

$$||\mathbf{a} - \mathbf{b}||_1 = \sum_{i=1}^{D} |\mathbf{a} - \mathbf{b}| \tag{2.2}$$

   Intuitively speaking, it is the sum of straight lines along the axes required to go from a to b.

2. **Euclidean distance.** The Euclidean distance or the L_2 norm is given by the equation:

$$||\mathbf{a} - \mathbf{b}||_2 = \sqrt[2]{\sum_{i=1}^{D} ||a_i - b_i||^2} \tag{2.3}$$

   Euclidean distance is the most commonly used distance measure. The idea of Euclidean distance is based upon the Pythagorean theorem in 2-D geometry, i.e., it gives the length of the straight line starting from a and ending at b.

Sometimes, for analytical purposes, the squared Euclidean distance is used and described as the Euclidean distance (we will do the same for the rest of the thesis).

For most of the embedding purposes, we use a simple embedding technique called Multidimensional Scaling (MDS) [10], based upon the idea of preserving spatial distances. In the following, we briefly describe MDS and the related invariants used in this thesis and refer to [38] for the interested reader.

[2]The dot product of two equal length n-dimensional vectors x and y is the sum of multiplications for corresponding entries and is denoted by $\langle x.y \rangle$ i.e., $\langle x.y \rangle = \sum_{i}^{n} x_i.y_i$

2.2.2. Multidimensional Scaling (MDS)

Multidimsional Scaling (MDS) refers to a family of embedding methods starting in 1930s [38]. The classical metric MDS preserves pairwise scalar products (closely related to pairwise distances) and achieves the dimensionality reduction a linear way (rather than a non-linear one).

In order to formally introduce the Multi-dimensional Scaling, we start with some notation. The observed variables are denoted by $\mathbf{y}$ and latent variables are denoted by $\mathbf{x}$. Furthermore, P refers to the dimension of latent space while D refers to the dimension of data space and our objective is to find a mapping between points lying in a Cartesian metric spaces $\mathbf{R}^D$ to another $\mathbf{R}^P$. More precisely, we want to find a D-by-P matrix W (with $W^TW = I_P$) that transforms the observed variables y into x by preserving the pairwise scalar products. i.e.

$$y = W^T x$$

For this purpose, we take a matrix S, called Gram matrix, that represents the pairwise scalar products of elements of y. i.e.

$$S = Y^TY = (WX)^T(WX) = X^TX$$

It has been shown that the latent variables can be found by computing the eigenvalue decomposition of the Gram matrix S. i.e.,

$$S = U\Lambda U^T$$

where the matrix U represents the matrix of orthogonal eigenvectors of S and Λ refers to the diagonal matrix of eigenvalues in a sorted order. The P-dimensional latent variables can now be computed by the following equation.

$$X = I_{P\times N}\Lambda^{\frac{1}{2}}U^T \tag{2.4}$$

The eigenvector decomposition of the Gram matrix S ensures that the difference between the corresponding scalar products in the two spaces is minimized [14]. In other words, Eq. 2.4 finds the optimum of the following objective function.

$$F_o = \underset{X}{\operatorname{argmin}} \sum_{i,j} \left(\langle y_i, .y_j\rangle - \langle x_i, .x_j\rangle\right)^2 \tag{2.5}$$

Eq. 2.5, also known as the classical metric MDS, has been generalized to the metric MDS in which pairwise similarities are preserved. Lets say K_x and K_y denote the similarity matrices for output X and input Y. Then the objective function F_o using metric MDS can be described as follows:

$$F_o = \underset{X}{\operatorname{argmin}} \sum_{i,j} \left(K_x(i,j) - K_y(i,j)\right)^2 \tag{2.6}$$

Although spatial distance based approaches,e.g., classical MDS. work well on simple data sets however one has to be careful when the observed data has an underlying structure, e.g., trajectories on a street network because the data points are constrained to live on the underlying network and the Euclidean distances between data points do not represent this information. In such cases, Fig. 2.2 describes how to embed a trajectory in a one dimensional space by preserving Euclidean or geodesic distances.

2.2.3. Geodesic Distances

The fundamental idea behind embedding is that the data lies over a manifold (a curved and sparse subspace) in a high dimensional space. The term geodesic distances is used for the distances that respect this manifold during similarity calculation. Formally, the geodesic distance, denoted by $\Delta(p_i, p_j)$ between two points p_i and p_j for a manifold X, in a metric space $(\mathcal{X}, d)$ space, is defined as following:

Definition 8 (Geo-Desic Distances) *It is the shortest distance between p_i and p_j realized through a sequence of points in X, i.e., $(p_i, p_{k_1}, ..., p_{k_l}, ..., p_{k_n}) \in \mathcal{X}$ such that for any two consecutive points in this sequence the manifold distance $\Delta(p_{k_l}, p_{k_{l+1}})$ equals the metric distance $d(p_i, p_j)$.*

In order to explain the concept,we give some relevant examples: To illustrate, consider a single trajectory $T = \{p_1, p_2, ..., p_n\} \in \mathbf{R}^2$ and the latent variable $X = \{x_1, x_2, ..., x_n\} \in \mathbf{R}$.

$$\mathcal{R} \to T \in \mathcal{R}^2 : x \mapsto \mathbf{m}(x) = [m_1(x), m_2(x)] \tag{2.7}$$

The the arc length g from point p_i to p_j is computed through the integral

$$g = \int_{p_i}^{p_j} dg = \int_{x_i}^{x_j} ||\mathbf{J}_x \mathbf{m}(x)|| dx \tag{2.8}$$

where $\mathbf{J}_x$ denotes the Jacobian matrix (or first order partial derivatives) of the manifold points w.r.t. x. In simple words, the geodesic distance can be calculated by summing the lengths of straight lines lying on the trajectory between two points.

A raw trajectory is essentially a one dimensional manifold (a very easy prey to distance preserving embedding). In the case of graphs (a manifold where local neighbourhood is defined by multiple points) the situations gets more complex. In this case, several different paths go from one point to the other and one particular path is considered a sub-manifold of the observed data and the integral in Eq.2.8 has to be minimized over all such paths. Note that under the assumption of dense graph, the geodesic distance in a graph is approximated well by the shortest path distance[3] [38], and is defined as the shortest distance between two nodes, i.e., $v_i, v_j \in V(G)$, required to reach from v_i to v_j by considering the distances of all paths from v_i to v_j. This distance is usually known as the rail-road distance as well, as rails are constrained to live on the underlying network. Furthermore, the great circle distance is a geodesic distance as well, as, it is computed upon the curves drawn on the surface of the earth instead of the three dimensional spatial plane of our existence. Fig. 2.2 describes the consequences of preserving structural information in trajectories through Euclidean distances versus geodesic distance based embeddings. Although, one may propose that the structural information between the trajectory points can be preserved by computing the Euclidean distances in $\mathbf{R}^2$, however, these distances (e.g., red line between p_2 and p_n) do not respect the underlying manifold (or street network over which T exists). Consequently, such distances cannot be used for trajectory analytical tasks in a structure preserving way. The idea is further illustrated through the help of triangular distances (red triangle) between p_2, p_4 and p_n. The individual aspects of the triangle give the Euclidean distance between corresponding points. Roughly speaking, the street network distance between p_2 is approximately thrice the length of the aspect between p_2 and p_4. This miscalculation can produce error for the analytical tasks built upon this information. However, there is a solution to this problem, i.e., to use the graph distances between the trajectory points. Intuitively speaking, as there is only one path between two point of the trajectory, therefore, if we want to embed a raw trajectory onto a one dimensional plane then the trajectory will be unrolled. This issue can possibly be addressed by going along the trajectory curve instead of computing the distance in $\mathbf{R^2}$. The results

[3]Following the well-known Iso-Map [38], we use the term geodesic distance instead of the shortest path distance when we assume that our graph is a curved sub-space (manifold) lying in a larger space, and our aim is to estimate the pair-wise Euclidean distances between the nodes of this graph by unfolding it through embeddings. We refer to [38] for more details.

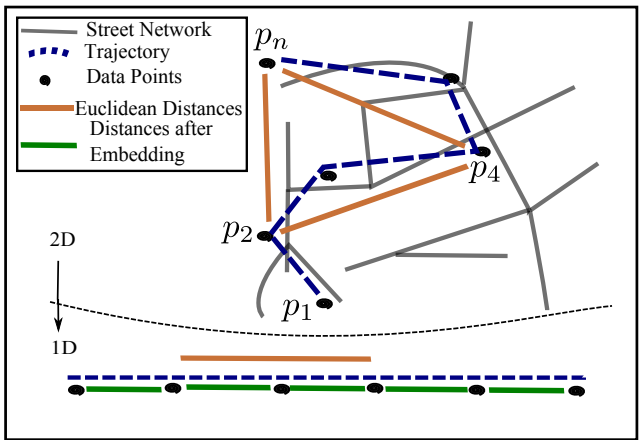


Figure 2.2.: The simplest case for geodesic distance based embedding for a sampled trajectory over the street network (gray edges). The trajectory T (blue dashed line) is a continuous curve in $\mathbf{R}^2$ and it is sampled through the noisy points $T = \{p_1, p_2, ..., p_n\}$ (in black circles). The purpose of the graph is to show that the structural distances between input points cannot be represented by the Euclidean distances in $\mathbf{R}^2$, because, they do not respect the underlying street network. The idea is further illustrated through the help of triangular distances (red triangle) between p_2, p_4 and p_n in Fig.2.2, where the individual edges of the triangle give the Euclidean distance between the corresponding points. Roughly speaking, the street network distance between p_2 and p_n is approximately thrice the length of the edges between p_2 and p_4. This issue can possibly be addressed by going along the trajectory curve instead of computing the distance in $\mathbf{R^2}$. The results of distance preservation based embedding in a 1-D plane with geodesic distances are shown.

of distance preservation based embedding in a 1-D plane with geodesic distances is shown. Notice that the pairwise Euclidean distances cannot be easily obtained for a non chain like graph (i.e., a manifold, where there are multiple paths between two nodes). One way to resolve this situation is given by the Iso-Map [38], that embeds the shortest paths in such graphs using MDS in order to come up with pairwise Euclidean Distances. The same technique will be used later in this chapter to compute distances between more complex trajectories than the above one.

Embedding approaches, i.e., MDS and other, are called 'kernelized embedding' [24] when they use kernel matrices as the similarity criteria, i.e., K_x and K_y. We proceed by explaining kernel matrices and kernel methods, in general, in next section.

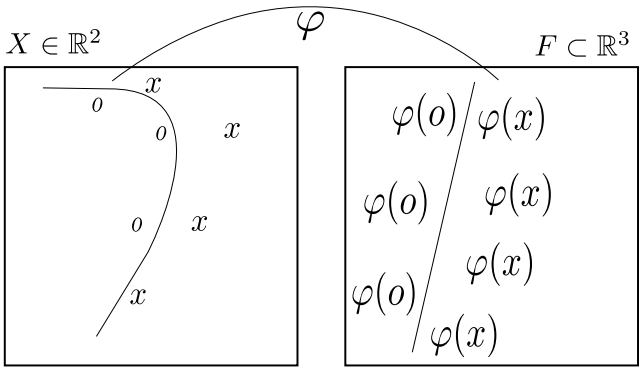


Figure 2.3.: Illustration of the main idea underlying kernel methods. The function φ embeds the data into a feature space F where the complex pattern now appears linear. The kernel computes dot products in F.

2.3. Kernels for Structured Data

Kernel based distance functions and learning methods are among the most popular machine learning techniques. The reasons for their popularity are their tendency towards easier geometric interpretation, solid mathematical background and strong empirical performance in a vast majority of machine learning tasks. The basic idea behind a kernel is to improvise a mapping for the data points such that the distance between two data points can be calculated as an inner product in the mapped space. Thinking along these lines, a large volume of work is devoted to graph kernels and kernels for structured data that make it possible to apply the vector based learning schemes to structured data. In the remaining part of this section, we give a brief introduction to kernels and refer to one of the many introductory text books on kernel methods [60] (used as a reference text in this section) for interested readers.

The word 'kernel' or 'kernel function' (denoted by k) is used in machine learning as a similarity criteria more suitable to the non-vectorial data sets (text, images, protein sequences, graphs, and trajectories). The function k maps each pair of points in the dataset to a dot product in a high dimensional feature space, cf. Fig. 2.3. The matrix constructed from applying kernel function to the all pairs of data points is called 'kernel matrix' of Gram matrix. Through the help of these dot products (kernels), we search for linear relationships in this high dimensional space, which correspond to some complex relationships in the input space. In this way, we can find complex relationships among the data points easily. The following scenario illustrates this concept in a gentle manner:

Consider a two-dimensional input $X \subseteq R^2$ together with a mapping φ, which

maps an input $x = (x_1, x_2)$ to a three-dimensional feature space $F \subseteq R^3$:

$$\varphi : x = (x_1, x_2) \mapsto \varphi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Consider $x, z \in X$. Once we have mapped them to F (a high dimenisonal feature space), we can carry out the kernel $k(x, z)$ as the dot product in F as follows:

$$\begin{aligned} k(x,z) &= < (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) > \\ &= (x_1^2z_1^2, x_2^2z_2^2, x_1x_2z_1z_2) \\ &= (x_1z_1 + x_2z_2)^2 = < x, z >^2 \end{aligned} \tag{2.9}$$

Notice that a straight line in F is translated to a polynomials in R^2, hence the linear relationships in F are not linear in R^2. We summarize the key aspects of kernel methods below:

1. Data items are embedded into a high dimensional space F where the kernel $k(x, z)$ is computed as a dot product between mappings.

2. Linear relations are searched among the mappings of data items in feature space through these dot products (kernels).

3. These relationships are translated back to input space as complex relationships and kernels in this way help finding the complex relationships easier.

4. Kernels provide great flexibility due to this property of inner product representation. We can apply kernel methods to any data set (irrespective of type, e.g graphs, trajectories, strings, and images), which can be mapped to a high-dimensional space where dot products can be computed.

5. A kernel is a valid kernel if there exists a feature space **F** where it can be computed as a dot product between the mappings of input vectors, i.e., $k(x, y) = \langle \varphi(x), \varphi(x) \rangle$. In terms of proving the validity of the kernel, this means one of the two choices:

   (i) To manually construct a mapping φ for the input elements, give a formulation of the kernel $k(x, y)$ and show that $k(x, y)$ is computable as a dot product between $\varphi(x)$ and $\varphi(y)$.

   (ii) Infact, one does not need to construct an explicit feature map φ at all. One only needs to show that $k(x, y)$ corresponds to $\langle \varphi(x), \varphi(y) \rangle$ for some

φ. This raises a very interesting question, i.e., Is it possible to show that $k(x, y) = \langle \varphi(x), \varphi(x) \rangle$ without knowing the feature map? The answer to this situation is provided by Mercer's theorem saying that one only needs to show that the Gram Matrix (the matrix computing the kernel values for input elements) is semi-positive definite. This technique is knows as the kernel trick and illustrated in more detail in many of the text books related to kernel methods, for example, [60].

6. **Kernel Construction Properties**. Kernels are extendible and can be combined through linear transformations, i.e multiplication, addition and a linear combination of them. Here, we provide a few properties to construct a valid kernel $k(x, y)$ from two valid kernels $k_1(x, y)$and $k_2(x, y)$.

   a) $k(x, y) = c.k_1(x, y)$ where $c > 0$

   b) $k(x, y) = k_1(x, y) + k_2(x, y)$

   c) $k(x, y) = k_1(x, y) \cdot k_2(x, y)$

   d) $k(x, y) = f(x)k_1(x, y)f(y)$ where $f(.)$ is any function

   e) $k(x, y) = e^{k_1(x,y)}$

   f) $k(x, y) = k_1(\varphi(x), \varphi(y))$ where φ maps input to R^M and k_1 is a valid kernel in R^M.

As mentioned above, kernel methods can be applied to any data set including graphs [25, 54]. Because we are more interested in continuous and discrete trajectories, which can be considered as label sequences over graphs therefore we are naturally interested in the 2nd category. The study for Kernel methods in structured and graph data started in 2002. Graph kernels can be used to measure the similarity between graphs for different matching problems, e.g., isomorphism, and auto morphism and can be broadly divided into kernels between two graphs and constructing kernels (kernels between the nodes of a graph) [65, 17, 19]. On similar lines, kernels for labelled graphs are of two types, i.e., kernels between two labelled graphs [64] or kernels between two label sequences (see [58] and Chap. 7 of [52]) of a particular graph. Having defined the basics of structure preservation, we provide our approach to preserving structure in trajectories by defining kernels for trajectories. These kernels are based upon the labelled graph definition of trajectories and they emphasize upon the notion of preserving the structural relationships in trajectory points.

2.4. Our Approach to Preserving Structure in Trajectories

Similarity criteria lie at the heart of most analytical tasks, and, therefore — our approach to preserving structure in raw trajectories is based upon defining the similarity criteria for trajectories as walks over labelled graphs. Building upon that, we capture the structural information in the two types of common trajectories, i.e., raw and symbolic trajectories with the help of a simple idea, i.e., take the geodesic distances among points in this label sequence graph and then embed them in Euclidean space to come up with the distances that respect the properties of a distance measure. These distances can then be used in conjunction with the existing kernel machinery to solve trajectory analysis tasks. Although the idea is simple enough, yet, the technical details of the two kernels are different because of the nature and applications considered. In the following sections, we give motivations and a description of our approach to preserving structure in raw and symbolic trajectories along with the kernels mentioned above.

2.4.1. Raw Trajectories

Our approach to the problem of structure preservation in raw trajectories is to view them as Euclidean graphs (with possibly additional labels for vertices and edges). This helps us in capturing the spatial and temporal aspects of the trajectory and also connecting it to the analytical methods developed in graph theory and machine learning. Actually, it is easy to see how trajectories can be represented by Euclidean graphs, because, the spatial aspects of a trajectory are already captured through $\mathbb{R}^2$ co-ordinates of the graph's vertices. Furthermore, the temporal aspects of a trajectory can be captured by assigning weights to edges according to the time spent upon them or time stamping the vertices of the graph in the order of the trajectory.

Recall from the previous chapters, that we have chosen map matching, i.e., assignment of raw trajectories to street network as an example problem for structure preservation in raw trajectories. In the start of this chapter, we have defined street network as an Euclidean graph. Therefore, map matching, can be viewed as a problem of

> "*matching, i.e., finding similarity between two Euclidean graphs*".

Indeed, Euclidean graphs can actually be found in many learning problems related to Geo-analytics such as map matching [6], shape analysis [13, 26, 57], protein structure analysis, and time series similarity analysis. However, —to the best of our

knowledge—, the problem of matching Euclidean graphs, i.e., "finding the similarity between two Euclidean graphs" has not been considered yet. Instead, the problem has been approximated by casting it into a traditional "embedding" respectively "graph matching" problem and in turn dropping important information. Furthermore, many of the graph matching problems are modelled as a *quadratic assignment problem*, which is NP-Complete [45], and most of the emphasis in solving graph matching problem is put upon finding the approximate solution to *the quadratic assignment problem*. However, due to the inherent problems in solving quadratic assignment problem, machine learning has started to look at kernel methods and other approaches to graph matching. Specifically, *graph matching* is typically formulated as a problem where we only seek a node-to-node matching between the input and output graphs. Euclidean graph matching is different as the nodes of the input graph (trajectory) can be mapped to any point lying on the edges of the output graph (street network graph). Embedding [38], on the other hand, is a generic problem: find a set of points in a (typically) low-dimensional space having similar properties and relationships as the points in the original input space. So far, however, it has only been considered for matching graphs in the traditional sense discussed above, see e.g. [38, 24, 47]. Hence, this approach suffers from the same issues as the standard graph matching approach. Nevertheless, they employ kernel methods. In the next section, we will show a kernel, which preserves the structure for raw trajectories and can be used with the help of above mentioned approaches to solve the problem of structure preservation in raw trajectories.

2.4.2. Geodesic Distance based RBF Kernel

In order to define a valid kernel over trajectories, taken as label sequences over Euclidean graphs, we first need to define a mapping of the individual elements to a feature space where the inner product between the elements gives us the kernel between the trajectories. However, as our intended application to kernel definition is the reduction of noise during map matching, therefore, we define a hyper parameter for the kernel, σ, which tells that how much a trajectory deviates from the original path. This hyper parameter is analogous to standard deviation between spatial points, however, our space of reference is the length of path between sampled points. Such a mapping would also require a notion of total distance travelled between two points. Obviously, this distance can be computed by taking the sum of distances between all successive points needed to reach from one input point to the other. In order to do that, we define an explicit feature map for the trajectory points x_i to

$\mathbf{R}^+$.

$$\phi : x = (x_1, x_2) \mapsto \mathbf{R}^+ = \sum_{k=1}^{i-1} (x_k - x_{k+1})^2$$

In other words, our mapping function maps a point in a trajectory on positive real line such that the first point of the trajectory represents 0 and all other points of the trajectory are assigned numbers equal to their shortest paths from the starting point in trajectory in its Euclidean graph representation. Notice that this representation maps each of the trajectory points on a straight line where we can compute the kernel. An intuitive distance measure between these mappings can be the Mahalanobis distance, (denoted by g), between the mappings of the corresponding points with the given sigma I.e.,

$$g(\phi(x_i), \phi(x_j); \sigma) = (\phi(x_i) - \phi(x_j))^2 / 2\sigma^2) = \sum_{i \leq k < j} (x_k - x_k + 1)^2 / \sigma^2 \tag{2.10}$$

Notice that this distance is slightly different than the Euclidean distance between the corresponding points in the sense that it chooses to go through the points already travelled by summing up the distances between the successive points lying in between the inputs. By taking this approach, this distance measure incorporates the path length and sigma incorporates the distortion occurred during the travel of the trajectory. Additionally, the Mahalanobis distance between the mappings is a discrete version of the geodesic distance equation provided in Eq. 2.8 from Sec. 2.2.3 with a constant factor $1/\sigma^2$, i.e.,

$$g(\phi(x_i), \phi(x_j); \sigma) = 1/\sigma^2 \cdot \int_{x_i}^{x_j} ||\mathbf{J}_x \phi(x)|| dx \tag{2.11}$$

Notice that because of our mapping the common distances in the two mapping will be cancelled out. Now, we take this geodesic distance as the core distance of the RBF kernel to give the final equation of geodesic distance kernel.

$$K(x_i, x_j) = e^{-g(\phi(x_i), \phi(x_j); \sigma)} = e^{-\sum_{i \leq k < j} (x_k - x_k + 1)^2 / 2\sigma^2} \tag{2.12}$$

The purpose of making this kernel function is to connect map matching with the broad kernel machinery. As shown in next chapter, the geodesic distance kernel can be successfully applied to give competitive results for map matching.

Validity of 'Kernel for Raw Trajectories'

Recall that in the standard "kernel trick", one does not need to construct an explicit feature map φ at all. One only needs to show that $k(x, y)$ corresponds to $\langle \varphi(x), \varphi(x) \rangle$ for some φ. An equivalent to this statement is to show that the Gram Matrix (the matrix computing the kernel values for input elements) is semi-positive definite, i.e., it fulfils the Mercer's conditions in [60]. Thinking along the same lines, many of the kernels for structured data do not provide a feature map for the input elements, instead, they show that a function of the input elements satisfies the Mercer's conditions as outlined above.

Our technique of giving a valid kernel, however, is different as we are lucky enough to find an feature map that maps the input points to $\mathbf{R}^+$. What remains to show is that this kernel is the calculation of a dot product in the mapped feature space. Here, because we are using an RBF kernel over the feature maps, therefore we only have to show that RBF kernel is a valid kernel over $\mathbf{R}^+$. This has been done earlier by taking the Taylor's expansion of exp function over the two inputs and then computing the dot product between these expansions in the infinite dimensional space. Knowing this, we can prove it through kernel construction property f in Sec.2.3 [60]. Or, alternatively, we can prove it in a simple way as following:

$$e^{-((\phi(x)-\phi(y))^2/2\sigma^2)} = e^{(-\phi(y)^2/2\sigma^2)} \cdot e^{(\phi(x)\phi(y)/\sigma^2)} \cdot e^{(-\phi(y)^2/2\sigma^2)}$$

where, the kernel construction properties, c and e from Sec.2.3 are used.

2.4.3. Symbolic Trajectories

In many traffic applications, our goal is to carry out a high-level analysis of raw trajectories. In order to reach this goal, we discretize the raw trajectories into the so-called symbolic trajectories. These symbolic trajectories can then be fed to graph and sequential approaches for further analysis. In this section, we first define a method to discretize raw traffic into sequences of traffic symbols (from a chosen alphabet) to proceed towards symbolic trajectory analysis. Furthermore, in order to use out-of-the-box methods for sequence analysis, we need a (symbol) similarity score Δ in order to compute the similarity score between traffic sequences and align them. And, then in the end, we give a geodesic distance based embedding that can be used with alignment kernel to compute the similarity between two symbolic trajectories.

We start by defining a so called 'translation method', that bundles together the

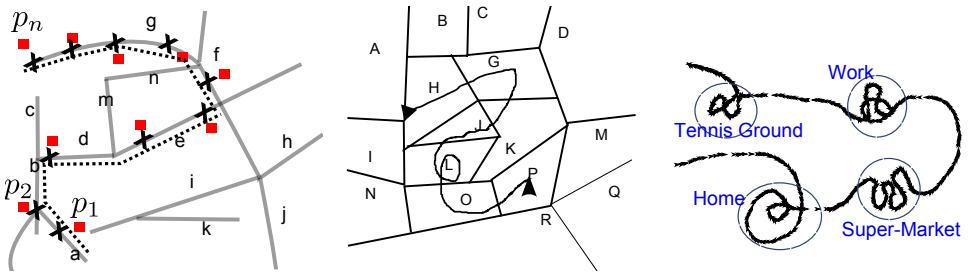
ingredients required to perform a symbolic (or high level) analysis of raw trajectories with sequence analysis approaches. Essentially, it is composed of three basic steps, i.e., choosing an alphabet, discretization and defining a symbol similarity score for alignment. More formally, let $\mathcal{X}$ be some raw traffic data. We now convert $\mathcal{X}$ into a set $\mathcal{S}$ of traffic sequences using a translation method $\mathcal{M}$ i.e $\mathcal{M} = (\mathcal{A}_\mathcal{M}, \Delta_\mathcal{M}, \mathcal{F})$ where $\mathcal{A}_\mathcal{M} = \{a_1, a_2, ..., a_l\}$ is an alphabet (set of symbols the sequences are composed of) and, a traffic sequence $T_\mathcal{M}$ for $\mathcal{M}$ is a walk over a labelled graph where the nodes of the graph are symbols in our alphabet and the edges are a function of two connected nodes, e.g., time spent, speed, and spatial distance. that is $T_\mathcal{M} = \{a_{t_1}, f(a_{t_1}, a_{t_2}), a_{t_2}, ..., f(a_{t_{n-1}}, a_{t_n}), a_{t_n}, \}$. Furthermore, $\mathcal{F}$ denotes a discretization function, which maps raw traffic data $\mathcal{X}$ to the set of traffic sequences $\mathcal{S}$ according to $\mathcal{M}$, that is $\mathcal{F}(\mathcal{X}) \leftarrow \mathcal{S}_\mathcal{M}$. Finally, $\Delta_\mathcal{M}$ is an l-by-l matrix of pair-wise similarities between symbols in $\mathcal{A}_\mathcal{M}$. The similarities in $\Delta_\mathcal{M}$ will be learned from the labelled graph representation of trajectories in Sec.2.4.4. Here, we start by describing the details of discretization function $\mathcal{F}$ and the alphabet (the set of nodes for our labelled graph).

Discretization Function

The discretization function $\mathcal{F}$ is a function, which maps the raw traffic data to trajectories, and — in general, $\mathcal{F}$ is application dependent. Examples include map matching, i.e., the process of assigning raw trajectories to street segments, see e.g. [30], region based division of Euclidean space in T-Pattern mining [22], frequency bins from sensor readings, and stay point extraction from user trajectories [68], among others, as illustrated in Fig. 2.4. Let us now touch upon the alphabet and similarity score used in more detail.

The Alphabet

The alphabet $\mathcal{A}_\mathcal{M}$ is a set of symbols i.e $\mathcal{A}_\mathcal{M} = \{a_1, a_2, ..., a_l\}$ with $|\mathcal{A}_\mathcal{M}| = l$. Every symbol $a \in \mathcal{A}$ corresponds to a set of traffic objects. Therefore, it is natural to assume that for any two symbols $a_i, a_j \in \mathcal{A}_\mathcal{M}, a_i \cap a_j = \emptyset$, that is a_i and a_j correspond to disjoint/non-overlapping sets of traffic objects. Note that the symbols usually represent spatial and unary objects like regions of a city or streets in a street network, however they can also represent non-spatial entities of interest like frequency bins for sensor readings or categories of streets, e.g., highway, and link road. The condition that symbols in $\mathcal{A}_\mathcal{M}$ represent disjoint sets is sufficient for



(a) Map Matching (b) Regional alphabet (c) Stay points based alphabet

Figure 2.4.: Discretization function $\mathcal{F}$ and traffic sequence extraction from raw data. x-denotes a traffic sequence extracted after discretization. (a) Map matching, i.e., assignment of raw trajectory points to street network. 'Gray lines' denote street network; 'Red squares'– raw trajectory points; 'black crosses' – corresponding ground truth and alphabetical symbols denote street names. The output of map matching is a sequence of street segments, i.e., $X = abdefg$. (b) Region based division of underlying space for T-Pattern mining. 'labelled Polygonal regions' denote regions and directional blue line denotes raw trajectory. The output of the discretization function $\mathcal{F}$ is a sequence of regions, i.e., $X = HGJLOP$. (c) Raw GPS trajectory is shown in black and blue circles denote the stay points of the users. $\mathcal{F}$ produces a sequence of stay points (or activities), i.e., $X =$ 'Home, Shopping, Work, Sports' after discretization.

sequence comparison methods and yet intuitive and powerful enough to capture a very broad range of traffic applications. Indeed, we loose information but gain a more condensed and often more easy-to-grasp view on the data. Specifically, the benefits of using the symbols are:

1. Symbols lists are user friendly like street names or regions of a city.

2. Raw traffic data is huge and cumbersome to query and analyse; 'symbol lists' summarize the data for faster computation.

3. (Disjoint) symbols force the user to define a suitable abstraction level and perform analysis according to her interests; e.g. Fig. 4.6 shows 'Traffic Logos' for stay point based alphabet after a 99% compression and still describes almost all semantic information present in Fig. 4.10 i.e an analysis carried over a 100k GPS readings. Tab. 2.1 describes the examples for translation of traffic problems into sequential problems.

Once, we have described the alphabet, we describe how to learn the similarity matrix from the labelled graph representation of trajectories.

2.4.4. Geodesic Distance based Alignment Kernel

The similarity matrix $\Delta_{\mathcal{M}}$ describes the similarity between symbols in $\mathcal{A}_{\mathcal{M}}$. In the context of computational biology, $\Delta_{\mathcal{M}}$ is driven by the following insight: two molecules have higher similarity if they can be converted through chemical reactions readily and vice versa. Therefore, standard matrices have been developed. For traffic applications, the situation is different. There is a multitude of traffic data sets, all with their own characteristics and invariants. Hence, it is unlikely that there is a single good similarity matrix. Instead, it depends upon the application at hand.

Hence, we now propose a 'data driven' approach to devise a similarity matrix Δ. Recall that we are representing our trajectory as a walk over the underlying labelled graph, i.e., $T_{\mathcal{M}} = \{a_{t_1}, f(a_{t_1}, a_{t_2}), a_{t_2}, ..., f(a_{t_{n-1}}, a_{t_n}), a_{t_n}, \}$, and we want to embed the geodesic distances in these graphs in order to come up with valid distances that can be used with kernels. In some cases, the geodesic distances of our interest in the underlying labelled graph can be given, for example, we can chose shortest path distances for the model where the input alphabet consists of streets from a street network and the application of interest is 'trajectory clustering'. For cases, where we do not have such domain knowledge available, we estimate these distances by fixing a function of interest for the pairwise nodes and then compute geodesic distances from it. Consider, for example, the case of temporally tagged sequences [21]. In this case, we can take the average temporal difference between the corresponding nodes as the weight of the edge between them. To illustrate, we turn a sequence into a graph in the following way. Each unique symbol in the sequence is a node. Then if two symbols are consecutive in the sequence, there is an edge between the corresponding nodes in the graph, which is weighted by a function of the two corresponding nodes (in case of multiple edges, we can take the average weight). Finally, if there are multiple sequences, we simply average all resulting distance matrices. Now, we calculate the shortest path distances between all nodes in the graph. Unfortunately, it may very well happen (in particular for rather small data sets) that there are pairs of symbols, which never co-occur in a traffic sequence. In turn, the *average temporal difference* distance cannot be computed. For example, in the dataset we used for the *analysis of user activities*, the user never does sports and shopping in a sequence together. In this case, we assign an infinity value. In other words, we just ensure that the two symbols are maximally dissimilar. We

Application	Alphabet	Edge Labels for IsoMap
User Activity Extraction	Stay points	Median Distances
'Popular' route finding	Streets in street network	Street Length
Analysis of Highway Usage	Street Category for trajectories	Average Temporal Difference
Traffic Event Detection	bins from sensor readings	Distance between bins

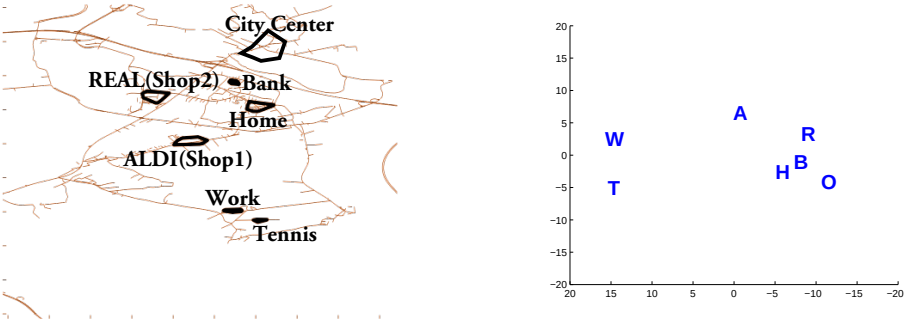
Table 2.1.: Alphabets along with edge labels for IsoMap.

Algorithm 1: Embedding Geodesic Distance for Symbol Similarity Δ

Input : $\mathcal{A}_\mathcal{M}, D = \{T^1_\mathcal{M}, T^2_\mathcal{M}, ..., T^m_\mathcal{M}\}$ - Set of Traffic Sequences
Output: $\Delta_\mathcal{M}$ - Symbol Similarity Matrix

1. Define two Matrices, Frequency and Cumulative temporal difference
2.**foreach** $T^i_\mathcal{M} \in D$ **do**
  foreach *consecutive pair* $(a, b) \in T^i_\mathcal{M}$ **do**
    Get The Function of pair e.g., temporal difference
    Add temporal difference and frequency calculations
3. Get Average of temporal differences by dividing it with Frequency
4. Set missing differences to infinity
5. Compute Shortest Paths from Difference Matrix
6. Embed in $\mathbf{R^2}$ using IsoMap [38], and output differences

note that now we are in a very similar situation as the well-known IsoMap approach for computing low-dimensional Euclidean embedding [61]. Simply following it, i.e., we embed the weighted graph into Euclidean space $\mathbf{R^2}$ resulting in distances d_{ij} using *multi-dimensional scaling* [10]. This new distance respects well the intrinsic geometry of the data manifold described by the weighted graph. Tab. 2.1 gives the example edge labels for the underlying labelled graph used to compute the geodesic distances. This scheme has a nice probabilistic interpretation. In essence, every trajectory lies on the same labelled graph but we do not know the weight on the edges of these labelled graph. Therefore, we assume that the observed weight (e.g., time, or speed) is actually a sample of the original weight. Considering that these sampled weights on the edges of our graph are i.i.d under a normal distribution, we take the Maximum likelihood estimation for these weights, which is the average. Kindly, note that our goal is to compute the geodesic distance between these nodes, which is well approximated by the observed weights in the nodes and shortest paths in the overall graph. Similarly one may be tempted to compute the shortest path in every trajectory between two nodes and take this as a sample for the combined geodesic distance. In our view, this will not yield good results because in a shortest path (for one trajectory) there are multiple edges (each of which is i.i.d.), therefore



(a) The original map in real world with labelled stay points at their spatial coordiantes

(b) Stay points after geodesic distance based embedding with temporal differences

Figure 2.5.: (a). Spatial map in the true world. Convex hulls of labelled stay points (polygons over gray-edged street network). The main stay points are Home, Office, Bank, Tennis, Shopping-1(ALDI), Shopping-2(REAL) and City Center. (b). Temporal space projection for the similarities learned from labelled graph representation. Notice that the original map is very close to a rotation of the embedding. This makes sense as the time spent in reaching from one place to the other in real world is roughly proportional to the spatial distances.

if we keep out concatenating these independent probabilities, our confidence about the overall geodesic distance will get lower. Therefore, we calculate the geodesic distance only between the connected nodes as the average of the observed weights for connected nodes (which is really frequent and can be calculated according to a high confidence) and in the end, the shortest path (an approximation for the geodesic distance) for the graph observed by such distances. In order to come up with the Euclidean space based distances, we unfold all these geodesic distances by embedding them.

One of the example of these weights can be temporal difference between two nodes, (i.e. average of shortest temporal differences between the symbols in sequences). This is essentially triggered by Buchin *et al.*'s temporal differences between trajectory points' [8]. An embedding based upon such distances will be like oprojecting the original trajectories into a temporal space. We show the results in Fig. 2.5. Notice that after rotation of the axis, our embedding scheme yields a temporal space based map, which quite closed to spatial space map. This makes sense as the time spent in reaching from one place to the other in real world is roughly proportional to the spatial distances. Finally, we turn the Euclidean distances into similarities by using RBF kernels [30], i.e., $\Delta_{ij} = \exp(-d_{ij})$.

The similarity criteria defined above along with the discretization methods introduced provides a mechanism to run black box sequence analysis methods over raw traffic data. For instance, we can align a set of sequences. However, we can do even better. For instance, standard alignment assumes that the time lapsed between two consecutive symbols is constant. This is not true for most traffic data. To accommodate for variable-size steps in time, that means to balance between duration of a time step and the Euclidean distance between the two corresponding symbols, we add a penalty term to the Euclidean distance between them. Specifically, let π^* denote the alignment between two traffic sequences s and s' of length m and n respectively. Furthermore, let $d(s_i, s'_j)$ denote the distance after embedding between symbols in s and s' at position i and j respectively. Now, we define a similarity based on d:

$$d'(s_i, s'_j) = d(s_i, s'_j) + \lambda \cdot (t_i - t'_j)^2 \tag{2.13}$$

where $\lambda \in [0, 1]$ denotes the regularizer for variable-size time steps. Its value is application dependent. In case of a gap, we simply fix the gap penalty as a constant, i.e., $d'(s_i, -) = d'(-, s'_j) = c$

Now, we simply use an alignment algorithm on the fashion [11] of to compute π^* and $\theta(\pi^*)$ using the similarity $\Delta' = \exp(-d')$. Moreover, we can naturally turn the score of the alignment into a similarity score among pairs of whole sequences by normalizing it in the following way.

$$K_{s,s'} = \frac{\theta(\pi^\star_{s,s'})}{sqrt(\theta(\pi^\star_{s,s}), \theta(\pi^\star_{s',s'}))} \tag{2.14}$$

Now, we finally have everything together to employ the out-of-the-box sequence analysis tools for the analysis of symbolic trajectories.

Validity of 'Kernel for Symbolic Trajectories'

Note that our approach to define kernels for trajectories is not based upon giving formulations for new kernels, instead, we provide mappings for the inputs and use them with existing kernels in the machine learning. One has be careful in such a construction that whether the existing kernel is valid upon the feature space provided by these mappings. In the case of global alignment kernel, this means that we need to be sure that global alignment kernel is valid over a Euclidean symbol space and a constant gap penalty. The discussion on the validity of such kernels is provided by many authors, e.g., [11, 12].

Summary

In this chapter, we have outlined a detailed method in order to preserve structural information in trajectories. Specifically, it comprises techniques to convert two different types of trajectories, i.e., raw and symbolic trajectories into walks over labelled graphs. And, then we have described distance measures that preserve the geodesic distances (over spatial, temporal, or any other domain of interest) in these graphs (or trajectories). These distances are then connected to the well known kernel functions, i.e., RBF kernels and alignment kernels, which will be used to improve the analytical tasks for trajectories in later chapters.

3. Preserving Structure in Raw Trajectories

'Enlightening! I really started looking at birds in a different way, [for example, considering] the issue of space from the perspective of a bird — a projectile, a trajectory.— David Rubin

The distance function capturing the structural information in raw trajectories is presented in Chap 2. In this chapter, we show how to use this distance function to solve real world trajectory problems; in particular, our application of interest, i.e., map matching. In the next section, we will start by reviewing map matching and how structure preservation can help us in solving this problem.

Our approach to the problem of structure preservation in raw trajectories is to view them as Euclidean graphs (with possibly additional labels for vertices and edges). This helps us in capturing the spatial and temporal aspects of the trajectory and also connecting it to the analytical methods developed in graph theory and machine learning. Actually, it is easy to see how trajectories can be represented by Euclidean graphs, because, the spatial aspects of a trajectory are already captured through $\mathbb{R}^2$ co-ordinates of the graph's vertices. Furthermore, the temporal aspects of a trajectory can be captured by assigning weights to edges according to the time spent upon them or time stamping the vertices of the graph in the order of the trajectory.

Recall from the previous chapters, that we have chosen map matching, i.e., assignment of raw trajectories to street network as an example problem for structure preservation in raw trajectories. In the start of this chapter, we have defined street network as an Euclidean graph. Therefore, map matching, can be viewed as a problem of

"*matching, i.e., finding similarity between two Euclidean graphs*".

Map matching is a fundamental step for analysis of traffic behaviour based upon some network and is becoming increasingly important due to pervasiveness of measuring devices (e.g., Wifi, Mobiles, sensors inside building and vehicle navigation

systems). The sample trajectories from these devices contain noise ranging from 10 to 100 meters due to different capacity and environment issues during measurement process. Consequently, we have multiple choices of street for a sample trajectory. The process of disambiguation between these multiple paths (or alignment of sample trajectories to street network) is the core difficulty problem that we solve through structure preservation in raw trajectories.

Triggered by the observation that matching a trajectory of co-ordinates would be easy if the observed co-ordinates were noise-free — the co-ordinates would simply constitute the solution — one may propose to treat the map matching problem as a regression problem. That is, we treat a trajectory as a function t for which we observe a sequence of noisy values, the co-ordinates $t(i) + \epsilon$ at inputs $i = 1, 2, 3, \ldots, k$, the temporal order of co-ordinates. The task is now: estimate the noise-free function t from the noise observations. In contrast to most traditional regression tasks, however, outputs are structured due to the physical constraints in the world and in turn there are non-linear dependencies among co-ordinates. Although kernel methods are powerful tools for modelling non-linear dependencies, and hence seems to be relevant for map matching, most kernel (regression) models focus on the prediction of a single output. Although generalizations to multiple outputs can often be achieved by training independent models for each one or tying parameters across dimensions, this fails to account for output correlations [67]. Consequently, we propose a different approach, namely to "embed" the output of f, i.e., the co-ordinates of trajectory into the same space as the trajectory and the network and in turn reducing the noise while still capturing the multi-output, non-linear dependencies present in trajectories. Specifically, ignoring the map constraints, we first embed the trajectory and hence reducing noise.

Next to regression, our approach to map matching problem is built upon the ideas of graph matching and embedding described in Chapter 2. While many of the traditional problems related to matching are solved by preprocessing data to find nodes of the query object curve and then applying graph matching solutions to find a match between the query object and the data set, we assume a different setting where the nodes of query object can be matched to any of the points lying on the lines of polygonal curves in the dataset. Many of these approaches employ kernel methods which have a demonstrated ability to deal with sparseness and noise in data. Surprisingly, kernel methods and embeddings have not been used for map matching before. In order to capture the structural dependencies in the input data, we view map matching as a problem of Embedding one Euclidean graph in to the other such

that the similarity between nodes in the input graph is preserved in the output graph. Although, Euclidean graphs can be found in many machine learning problems such as map matching [6], shape matching, i.e., analysis and retrieval [13, 57, 59], protein structure analysis, time series similarity analysis, among others. However, —to the best of our knowledge—, the problem of matching Euclidean graphs, i.e., "finding similarity between two Euclidean graphs" has not been considered yet. Instead, the problem has been approximated by casting it into a traditional "embedding" respectively "graph matching" problem and in turn dropping important information. Specifically, *graph matching* is typically formulated as a problem where we only seek a node-to-node matching between the input and output graphs. Euclidean graph matching is different as the nodes of the input graph (trajectory) can be mapped to any point lying on the edges of the output graph (street network graph). Embedding [38], on the other hand, is a generic problem: find a set of points in a (typically) low-dimensional space having similar properties and relationships as the points in the original input space. So far, however, it has only been considered for matching graphs in the traditional sense discussed above, see e.g. [38, 24, 48, 9]. Hence, this approach suffers from the same issues as the standard graph matching approach.

Our approach to the problem constitutes two parts, a discrete part similar to graph matching where we have to find a walk on the output graph similar to input graph and a continuous part similar to embedding, where we want to find a correspondence between the nodes of the input graph and the points lying on the edges of output graph. In short, we want to embed a Euclidean graph structure into another euclidean graph. This approach is best suited to partial matching problems or scenarios where noise exists in query data, which is the case for most real world problems. The advantages of our approach are illustrated through results on map matching which is a partial matching problem, because, we are required to find the match of a GPS trace to a sub graph of street network and the nodes of trace are actually points on the edges of this street network. Moreover our approach is based on kernels, which provide a lot more flexibility and robustness in solving the problems at hand.

In this chapter, we provide an algorithm for map matching based upon the idea of structural preservation in raw trajectories. Clearly, the idea of aligning sampled trajectories in Euclidean plane to a Euclidean graph requires some transformation from an unconstrained space to a constrained one. On top of that, this transformation shall be such that the original noise is reduced. Thinking along these lines, we

devise an intuitive optimization scheme named KMM-Discrete which carries out the noise reduction and alignment alternatively in an EM manner until the algorithm is converged (for details see Appendix). Intuitively, KMM-discrete is a nice optimization scheme but its results are not satisfactory. Here, we make the observation that the limitations of KMM-Discrete lie in the simultaneous resolution of two fundamental problems in map matching, i.e., noise reduction and alignment. To validate our conjecture, we provide a second algorithm, KMM which separates out the noise reduction and alignment in two separate steps. KMM produces better results by dealing with the shortcomings of KMM-Discrete. The details of our experiments are provided in the results section. Consequently, we show the benefits and flexibility of structural preservation for alignment of raw trajectories to street networks. Furthermore,we make a comment that the success of a kernel based scheme depends upon the strength of the optimization solutions that encapsulates it. At the end of the chapter, we show the effect of parameter selection on our optimization scheme and also show that our scheme produces compatible results to the state-of-the-art.

However, lets' first start by a formal description of the problem.

3.1. Euclidean Graph Matching

We proceed with some notation and problem description. $G(V, E)$ denotes a trajectory (input graph) where vertices are indexed by time t, i.e., $V = \{v_t\}_{t=1}^{|V|}, v_t \in \mathbb{R}^2$, while $G^{'}(V^{'}, E^{'})$ denotes street network graph (output graph) where $V^{'} = \{v_i^{'}\}_{i=1}^{|V^{'}|}, v_{i^{'}} \in \mathbb{R}^2$ are the set of nodes for street segments and $E^{'} = \{e_{i,j}^{'}\}_{i,j=1}^{|V^{'}|}, e_{i,j}^{'} \in \{\theta v_i + (1-\theta)v_j, \theta \in [0,1]\}$, if $e_{i,j}^{'}$ corresponds to a street segment otherwise it is empty. Notice that $e_{i,j}^{'}$ is defined as a convex combination of vertices, i.e., a straight line between the nodes. In general, a street segment is defined as a poly-line however we can always divide it into a set of straight lines considering each corner of lines as a vertex to match our representation. Φ denotes the vector of mappings between a trajectory G and street network $G^{'}$. Unlike traditional graph matching, where the nodes of one graph are only matched to the nodes of another graph, we need a mapping Φ which matches the nodes of our input graph (trajectory) to any location over street segments in $E^{'}$: i.e., $\Phi_t = \Phi : v_t \mapsto e^{'} \times [0,1]$ where the interval $[0,1]$ specifies a value of θ pointing the exact location of mapping over a street segment in E'.

The truth value of vertex assignments to street segments can be stored in a binary matrix Δ, i.e., $\Delta \in \{0,1\}^{|V| \times |E^{'}|}$ subject to $\Delta^{\top}\mathbf{1} = \mathbf{1}$. Here $\mathbf{1}$ denotes a column

vector of all ones. The last constraint enforces that each trajectory point can only be mapped to one street edge at a time. The Euclidean graph matching problem can now be defined as follows:

"Find a correspondence Φ between vertices of trajectory G and locations on street network G' such that the two matched sets, V and Φ, look most similar according to an objective criteria F_o."

The problem is solved through finding an assignment of V to points lying on the edge set E' that minimizes the criteria F_o. The function F_o is of special interest and should have a form which preserves the relationships among input points while translating them to output space. We further assume that F_o is a decomposable function of a summation of basis functions, denoted by f_{Φ_i,Φ_j}. Thus to minimize F_o, we need to minimize the individual entries of summation. We also introduce the assignment matrix Δ in F_o. The assignment matrix Δ, enforces the Euclidean graph (or street network) constraints on the output, i.e.,

$$\begin{aligned} F_o &= \underset{\mathbf{\Phi}}{\operatorname{argmin}} \sum_{l}^{|E^{'}|} \sum_{i,j}^{|V|} \Delta_{i,l} \cdot f_{\Phi_i,\Phi_j} \\ s.t.\ \ \Delta &\in \{0,1\}^{|V|\times|E^{'}|}, \Delta^{\top}\mathbf{1} = \mathbf{1} \end{aligned} \tag{3.1}$$

Eq. (3.1) describes the map matching problem as a so-called integer linear program (ILP) which are generally known to be NP-hard except for a few classes of them. Such mathematical programs have a discrete and a continuous part. In our case, the discrete part chooses the correct combination of trajectory point ($v \in V$) versus street segment ($e' \in E'$). The total number of combinations is $V^{E'}$, which easily gets intractable even for a modest number of trajectory points and street network segments. Eq. (3.1) describes F_o as a summation of entries f_{Φ_i,Φ_j}. Now we come to details of an individual entry f_{Φ_i,Φ_j}. For this purpose, we define two Kernel matrices K_G and $K_{G'}$, where $K_{G_{i,j}}$ is the kernel function between trajectory vertices v_i and v_j: i.e., $K_{G_{i,j}} = k_G(v_i, v_j)$ and $K_{G'_{i,j}}$ is the kernel function between the mappings Φ_i, Φ_j of the vertices v_i and v_j, i.e., $K_{G'_{i,j}} = k_{G'}(\Phi_i, \Phi_j)$. The widths of the Kernels K_G and $K_{G'}$ are denoted by σ_G and $\sigma_{G'}$. Now we define f_{Φ_i,Φ_j} as the 'difference of kernels' function.

$$f_{\Phi_i,\Phi_j} = (k_G(v_i, v_j) - k_{G'}(\Phi_i, \Phi_j))^2 \tag{3.2}$$

Now we can substitute the value of f_{Φ_i,Φ_j} in Eq. (3.1).

$$F_o = \underset{\mathbf{\Phi}}{\operatorname{argmin}} \sum_{i,j}^{|V|} \left(k_G(v_i, v_j) - k_{G'}(\Phi_i, \Phi_j)\right)^2 \\ s.t.\ \ \Delta \in \{0,1\}^{|V|\times|E^{'}|}, \Delta^\top \mathbf{1} = \mathbf{1} \tag{3.3}$$

Eq. (3.3) comes from an embedding technique known as Multidimensional scaling described in Eq. (2.6). We further note that Eq. (3.3) is like a regression equation with multiple outputs where we want to preserve the correlation among inputs during our structured prediction process and it can be used for embeddings across different spaces and structures, however our case is a special case where the input and output spaces are the same. To encode our prior knowledge that the solution of the embedding lies in the spatial neighbourhood perturbed by Gaussian noise, we add a regularization term which fuses the input and output space into one. We propose a kernel function $k_{GG'}$ between the respective points of our input and output graphs and define it as

$$k_{GG'}(\lambda, \sigma_N, i) = e^{-((v_i - \Phi_i)^2 - \lambda\sigma_N^2)^2/2\sigma_N^2} \tag{3.4}$$

where λ is a stiffness parameter of the kernel function while σ_N, the kernel width, is the estimated standard deviation of noise in trajectory. We finalize our objective function F_o as following:

$$F_o = \underset{\mathbf{\Phi}}{\operatorname{argmin}} \sum_{ij} (K_{G_{i,j}} - K_{G'_{i,j}})^2 - \sum_i k_{GG'}(\lambda, \sigma_N, i) \\ s.t.\ \ \Delta \in \{0,1\}^{|V|\times|E^{'}|}, \Delta^\top \mathbf{1} = \mathbf{1} \tag{3.5}$$

Eq. (3.5) is the objective function which we are using in kernelized map matching. However alternative embedding approaches can also be used in principal.

Geodesic distance Kernels — $K_{G_{i,j}}$ and $K_{G'_{i,j}}$

The kernels $K_{G_{i,j}}$ and $K_{G'_{i,j}}$, that we have used are described in Chapter 2 in detail. To summarize: for two points p_i and p_j in a trajectory where $j > i$, the sum of successive euclidean distances is denoted by $\delta_{i,j}$, and is defined as:

$$\delta_{i,j} = \sum_{i \leqslant m < j} |p_m, p_{m+1}|_2$$

The kernel $K_{i,j}$ is simply an RBF kernel with $\delta_{i,j}$ as the core part instead of the direct euclidean distance between i and j.

$$K_G(i,j) = e^{-\delta_{i,j}/\sigma_k} \tag{3.6}$$

The only difference with an RBF kernel is that we use 'the sum of euclidean distances for all successive pairs of points between i and j' instead of using 'euclidean distance between i and j' directly. This allows us to capture the spatial and temporal correlation among trajectory points in our kernel matrices.

3.2. Kernelized Map Matching (KMM)

Recall from the introduction that map matching would be easy if the observed co-ordinates were noise-free. In this case the observed co-ordinates would simply constitute the solution. In general, we cannot expect to reduce the noise completely. Consequently, we propose a two steps approach:
(1) To reduce the noise, embed the trajectory from $\mathbb{R}^2$ into $\mathbb{R}^2$ using kernel methods to capture the multi-output, non-linear dependencies present in trajectories.
(2) To account for remaining noise, "round" the embedding into an hard matching. We will now explain each of the steps in turn.

Relaxation is a standard technique to reduce the complexity of ILP problems. In relaxation, we drop the discrete part of the problem to make it a standard linear programming problem where the optimization can be carried out in polynomial time. The result of the optimization is then rounded into a hard assignment fulfilling the discrete constraints to get an approximate solution. In our case, discrete constraints amount to street network constraints and relaxation means ignoring these constraints. Hence, we provide a two step framework for the solution of Eq. (3.1)

- Optimize the relaxed objective function to approximate the trajectory path.

- Provide a rounding scheme for assignment of step a output to street network.

We proceed by describing the details of these two steps in turn.

Optimization Step

Eq. (3.7) describes the unconstrained F_o

$$F_o = \underset{\mathbf{\Psi}}{\operatorname{argmin}} \sum_{i,j}^{|V|} f_{\Psi_i,\Psi_j} \tag{3.7}$$

Notice that we have changed the output vector Φ to intermediate output Ψ. The mapping Ψ is different from Φ as it maps a a vertex v_t to $\mathbb{R}^2$: i.e., $\Psi_t = \Psi : v_t \mapsto \mathbb{R}^2$. Finally, we change the value of f_{Ψ_i,Ψ_j} to the summation of basis functions in MDS and add the regularization term to it.

$$F_o = \underset{\mathbf{\Psi}}{\operatorname{argmin}} \sum_{ij} (K_{G_{i,j}} - K_{G'_{i,j}})^2 - \sum_{i} k_{GG'}(\lambda, \sigma_N, i) \tag{3.8}$$

The solution of the optimization will be an approximation of the path used by trajectory instead of the original path. Afterwards we can convert this approximate path into the street network path through a rounding step. We can take the derivative of F_o in Eq. (3.5) w.r.t. Ψ and can find the answer. We use 'scaled conjugate gradient' algorithm for optimization [55]. We implement the optimization step in sliding window style of width k_w, because we think that the global structure does not affect the local position of output points, furthermore it makes the solution real time and efficient.

Rounding Step

After the noise is reduced by the optimization, we have to assign the points to street network. For assignment purposes, we apply a rounding scheme on the result of continuous optimization. The simplest rounding scheme can be a nearest neighbour based one. However, we provide a more sensible scheme, which is based upon the following assumptions:

1. **Assumption 1**: For nearby points, the Euclidean distance between points resembles shortest street network graph distance;
2. **Assumption 2**: Most of the map matching algorithms use a radius ϵ to restrict the assignment possibilities.

For assigning a point Ψ_i to the street network, we pick all edges inside ϵ-neighbourhood and find nearest neighbours of Ψ_i on these edges. The resulting points are our candidates for the solution Φ_i. We denote the set of candidate solutions for Φ_i as C_{Φ_i}.

According to our assumption 1, Ψ_i should be assigned to a point $c \in C_{\Phi_i}$ which minimizes the difference between euclidean distance and shortest path distance between the solution Φ_i and Φ_{i-1}. For this purpose, we take an RBF kernel K_Ω between graph distance (denoted by $\Omega(x, y)$) and Euclidean distance (denoted by $d(a, b)$) as following:

$$K_{\Omega_i} = e^{-(d(\Psi_i, \Psi_{i-1}) - \Omega(c \in C_{\Phi_i}, \Phi_{i-1}))^2} \tag{3.9}$$

K_Ω stipulates the assumption 1 for two consecutive points. However to avoid K_Ω output going away from input point, we add a regularizer term which is also an RBF kernel between a Ψ_i and the candidate in question. Now for all elements $c \in C_{\Phi_i}$ we calculate the following term

$$-e^{-(c \in C_{\Phi_i} - \Psi_i)^2} \cdot K_{\Omega_i} \tag{3.10}$$

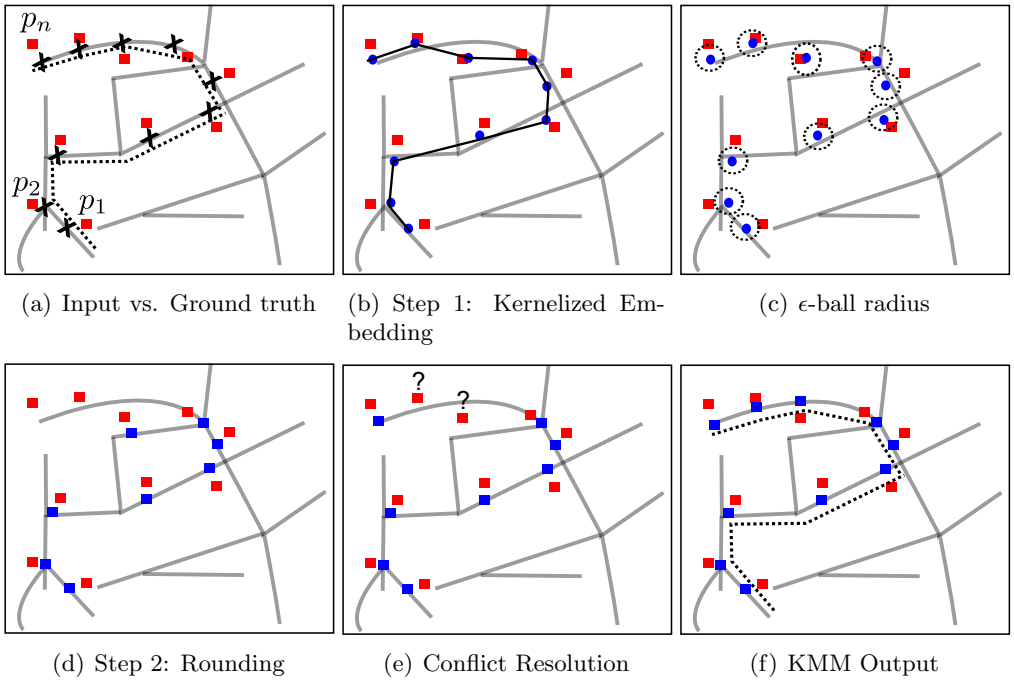
The candidate point which gives the minimum value for this term is chosen as solution Φ_i. In most of the cases, the comparison term between candidates, i.e., regularized K_{Ω_i} produces the right result however it is possible that after assignment the graph distance between Φ_i and Φ_{i-1} is far greater than the Euclidean distance between them, which introduces a conflict and violates our observation 1. To check these we take a constant α and multiply euclidean distance by it. After assignment, if the graph distance $\Omega(\Phi_i, \Phi_{i-1})$ is greater than $\alpha \times d(\Psi_i, \Psi_{i-1})$ (α-condition); we employ a resolution scheme, inspired by [44]. We discard Φ_i and Φ_{i-1} and consider Φ_{i-2} as our previous point instead of Φ_i. After the assignment Φ_{i+1}, we again see whether the condition 1 is violated or not, if it happens again, we discard Φ_{i+1} and Φ_{i-2} and go ahead in the same fashion. We continue doing so until α-condition is not violated any more after an assignment of a point denoted by $\Phi_{\hat{i}}$. Once we find such a point $\Phi_{\hat{i}}$, we can again resume the standard comparison. However, before resuming the comparison, we assign all the discarded points on the shortest path between $\Phi_{\hat{i}}$ and the corresponding previous point.

A conceptual work-flow of KMM steps together with explanations for each step are provided in Fig. 3.1.

3.3. Experimental Evaluation

In this section we report the results from a series of experiments which we conducted in order to empirically investigate the following questions:

(Q1) Can kernel methods be used for map matching?



(a) Input vs. Ground truth (b) Step 1: Kernelized Embedding (c) ϵ-ball radius

(d) Step 2: Rounding (e) Conflict Resolution (f) KMM Output

Figure 3.1.: Step-by-step illustration of Kernelized Map Matching. (a) Map matching input with ground truth values. Red squares denote the input Trajectory; Gray edges, the street network graph; dashed lines, the ground truth path and crosses the ground truth points. (b) Approximation of ground-truth path by kernelized embedding in a relaxed setting. Blue circles denote the optimization output Ψ. (c) Imposition of $\epsilon-$ball radius to reduce the number of candidate matches for rounding. Dashed-circles denote $\epsilon-$balls. (d) Hard assignment of Ψ to street network graph in the rounding step on the basis of RBF kernels Blue squares denote the assignment. (e) A conflict and its resolution. The graph distance is greater than $\alpha\times$ Euclidean distance, both points are discarded to get a vote from neighbours. (f) The final output points and path computed by KMM. (Best viewed in color)

(Q2) If so, how do they perform compared to state-of-the-art methods?

(Q3) Is kernelized embedding indeed reducing the noise?

(Q4) Does the rounding step in KMM contribute to the error reduction?

To this aim, we implemented KMM in scientific Python running on a standard Intel-Quadcore 2GHz computer.

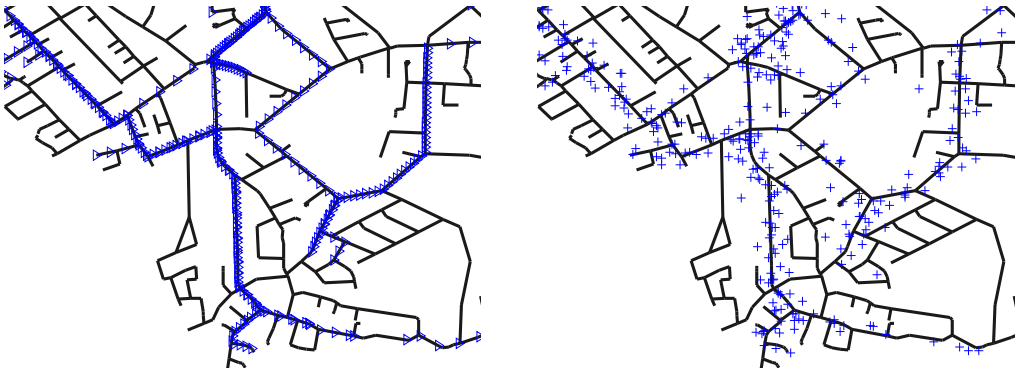
Overall, we decided for two experimental setups. Our first experimental setup evaluates and compares KMM's accuracy performance on a real-world dataset re-

Algorithm 2: KMM

Input : G,G', K_G,$K_{G'}$,$K_{GG'}$,K_Ω,λ,k_w, α,σ_N
Output: $\Phi(V)$ - The Output Path P

// G, G'- Euclidean Graphs
// $K_G, K_{G'}, K_{GG'}, K_\Omega$ -Graph Kernels
// λ -Regularizer
// k_w- sliding window width
// α- constant for Euclidean versus Graph distance validation
for $i \leftarrow 1 : |V| - k_w$ **do**
  foreach *sliding window* **do**
    compute $K_G, K_{G'}$
    $\Psi \leftarrow$ Optimize F_o w.r.t. Ψ
$i \leftarrow 1$, $i_{prev} \leftarrow 1$, $prev_{dist} \leftarrow 0$
while $i < |V|$ **do**
  $i \leftarrow i + 1$
  $i_{prev} \leftarrow max(1, i_{prev})$
  $min_{obj} \leftarrow \infty$
  if $i_{prev} = i - 1$ **then**
    $prev_{dist} \leftarrow 0$
    $con_{dist} \leftarrow d(\Psi_i, \Psi_{i-1})$
  else
    $con_{dist} \leftarrow d(\Psi_i, \Psi_{i-1}) + prev_{dist} + d(\Psi_{i_{prev}}, \Psi_{i_{prev}+1})$
  for $c \in C_{\Phi_i}$ **do**
    $obj_{val} \leftarrow -e^{-(c-\Psi_i)^2}.K_{\Omega_i}$
    if $obj_{val} < min_{obj}$ **then**
      $min_{obj} \leftarrow obj_{val}$
      $\Phi_i \leftarrow c$
  if $\Omega(\Phi_i, \Phi_{i_{prev}}) > \alpha \times d(\Psi_i, \Psi_{i_{prev}})$ **then**
    $i_{prev} \leftarrow i_{prev} - 1$
    $prev_{dist} \leftarrow con_{dist}$
  else
    Assign all points between i and i_{prev} to shortest path between Φ_i and $\Phi_{i_{prev}}$
Output path P by connecting consecutive Φ

cently used in [44] to evaluate an hidden Markov model based map matching approach and hence addresses **Q1**, **Q2**. To address **Q4** we compare the performance of our rounding step with a randomized rounding step. The second set-up investigates **Q3** by comparing the result of KMM's embedding step to baseline "closest point on edge" using synthetically generated dataset.



(a) Ground Truth (b) Map Matching input

Figure 3.2.: Synthetic Data Generation. (Left)- Ground truth data. Trajectory are shown in blue, where the direction of '¿' sign gives the travel direction. (Right). Same data after addition of Gaussian noise is given to map matching as input.

3.3.1. Traffic Data Generation

There are some very nice data generators available for trajectory data generation. IBM's data generator and Thomas Brinkhoff's data generator are very good in the sense that they can generate trajectories with changing speeds and sampling time variation according to some underlying road network. There is still very few real data available without licensing constraints. It has been empirically shown that Noise in data generated from navigation systems is normally distributed and average of noise (σ_N) ranges from 10-100 meters. [44] has discussed the different types and amount of noise in movement data and has shown a nice Gaussian curve of noise from real world data. Keeping in mind that the noise in the real world trajectories is normally distributed, we generated the ground truth data from the generators and added noise to it. We have compared our results to a simple baseline algorithm which assigns a data point to the nearest neighbour in the road network. We used Thomas Brinkhoff's traffic data generator for generating 100 trajectories of average length 50. We got 10,00 trajectories as we added σ_N from 10-100 meters by a 10 meters step size.

3.3.2. Comparison of KMM vs. Base Line

Fig. 3.3 shows the scatter plots of the root mean squared (RMS) error of KMM and of the baseline, closest point on edge, for different window sizes (K) on the Oldenburg dataset for different noise levels $(20, 30, 40, \ldots, 100)$. For better comparison with the

equilibrium, we also show the linear regression line of the values. As one can see KMM performs much better with increasing noise levels. The regression lines have consistently a smaller slope than the "equilibrium" line. The turning point is around noise level 25. Fig. 3.4 shows the importance of K-neighbourhood with respect to σ_N and it also shows a graph of error for different K and values of noise Vs. baseline.

3.3.3. Real World Dataset

In our first experiment, we compared KMM's performance to the performance of Krumm and Newson's recent hidden Markov model based approach [44]. To measure performance, we used the Route Mismatch Fraction measure already used by Newson and Krumm. Route Mismatch Fraction (RMF) is the total length of false positive and false negative route divided by length of original route [1]:

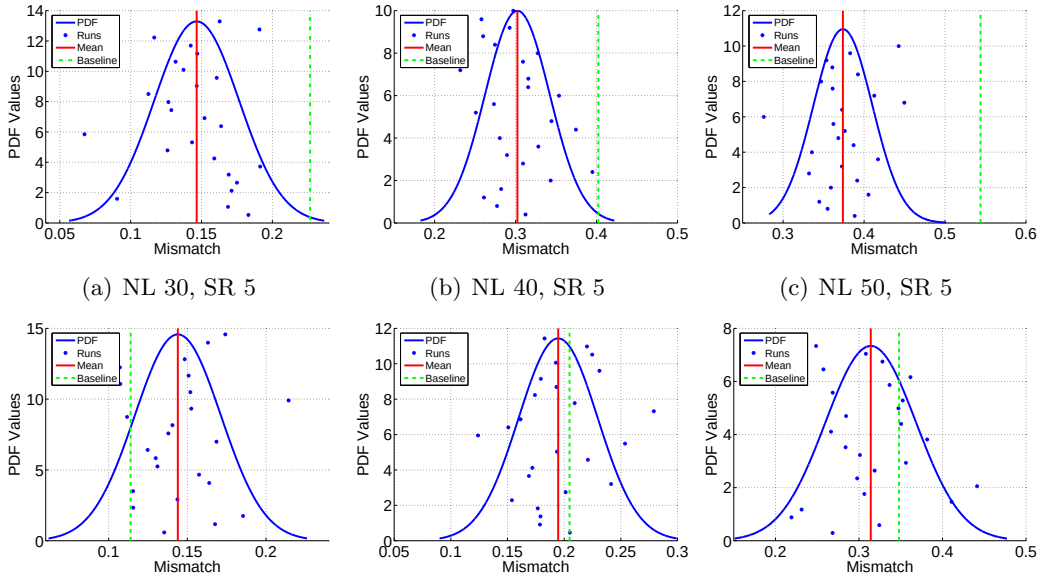
$$
\begin{aligned}
&d_+ = \text{length of erroneously added route}\\
&d_- = \text{length of erroneously subtracted route}\\
&d_o = \text{length of original route}\\
&RMF = \frac{(d_+ + d_-)}{d_o}
\end{aligned}
$$

We used Krumm and Newson's dataset, which is described by them as following: "It consists of a 50-mile route in Seattle sampled at 1 Hz, giving one trajectory of 7531 time stamped latitude/longitude pairs along with manually matched ground-truth path. The street network comprises around 150,000 road segments" [44]. presented results for different sampling intervals and noise degradations of this data. We take six base cases where HMM model performance is good, i.e., 5,10 seconds sampling intervals vs. 30,40,50 meters noise. Because we want to have a statistical comparison with HMM, we perform 25 experiments for comparison with each base value in the following way. For one sampling interval, say 10, we choose 5 different starting points from the initial 5 points of the trajectory and then sampled at the given rate. Following this procedure, we prepared experimental datasets for each sample by adding 5 instances of noise for one standard deviation (e.g 30), i.e.,

$$25 \text{ datasets/combination} = 5 \text{ samples} \times 5 \text{ noise instances.}$$

Fig. 3.3 summarizes the experimental results showing the RMF errors. As one can see, in 5 out of 6 cases KMM estimated a lower route mismatch fraction. The statistical significances of the results are shown in Table 3.1. In 4 out of 5 cases where we

[1] definition as provided by [44]



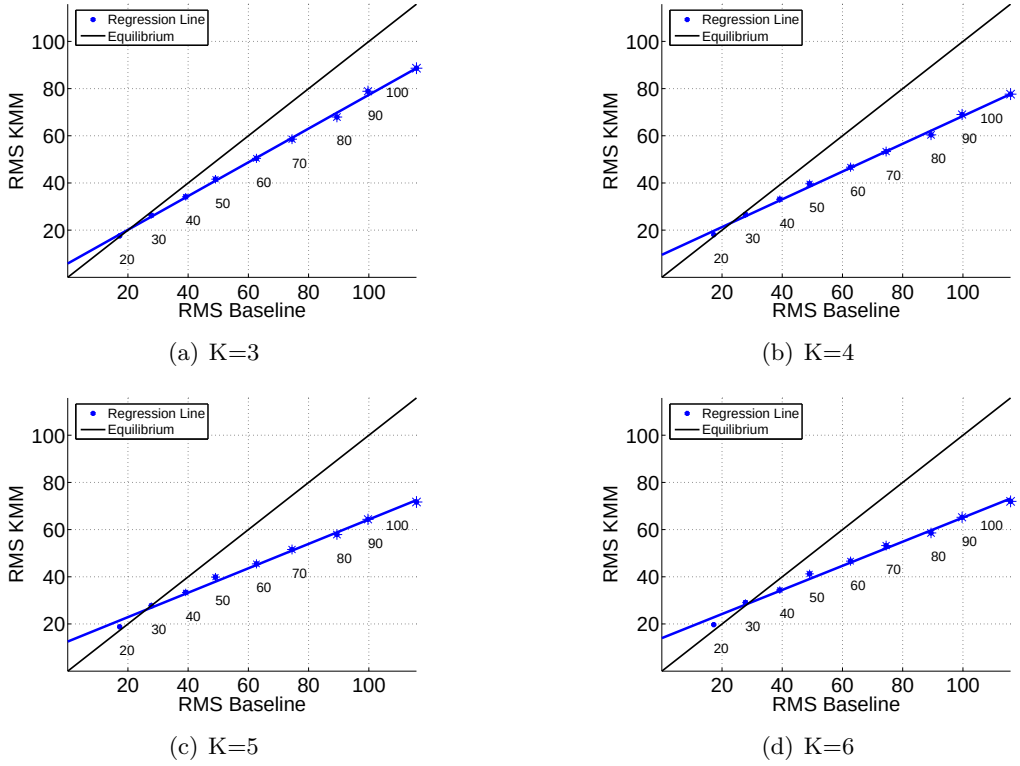
(a) NL 30, SR 5 (b) NL 40, SR 5 (c) NL 50, SR 5

(d) NL 30, SR 10 (e) NL 40, SR 10 (f) NL 50, SR 10

Figure 3.3.: Route Mismatch Fraction of KMM vs. HMM for different noise level (NL, in meters) and sampling rates (SR, in seconds). We ran 150 experiments, i.e., 25 experiments/per NL-SR setting. One blue dot denotes the route mismatch fraction for an experiment, blue graphs the estimated normal distributions, red lines the means, and dashed green lines the route mismatch fraction for an HMM as reported by Newson and Krumm. As one can see, in 5 out of 6 cases KMM estimates a lower route mismatch fraction. In 4 out of 5 cases, the differences in mean values are significant (t-test, $p = 0.05$). Averaged over all experiments, a Wilcoxon test identifies KMM to be significantly better. (Best viewed in color)

are better, the differences in mean values are significant (t-test, $p = 0.05$). Averaged over all experiments, a Wilcoxon test identifies KMM to be significantly better.
To address **Q4**, we compared the performance of our rounding step with a randomized rounding step, i.e., random assignment of embedding output Ψ to a street network point in ϵ-ball, over the real-world dataset described above. Table 3.2 summarizes the results. As one can see our rounding step always performs better, and in most cases outperforms the randomized rounding step by a margin of 4-10 percent in error.

In the next section, we analyse the results of KMM by comparing it with a naive baseline, i.e., nearest neighbour approach, a state-of-the-art HMM and also study



(a) K=3

(b) K=4

(c) K=5

(d) K=6

Figure 3.4.: Scatter plots of the root mean squared(RMS) error of KMM and of the baseline, closest point on edge, for different window sizes (K) on the Oldenburg dataset for different noise levels $(20, 30, 40, \ldots, 100)$ as denoted by the numbers associated with the dots. For better comparison with the equilibrium (solid straight line), we also show the linear regression line of the values. As one can see KMM performs much better with increasing noise levels. The regression lines have consistently a smaller slope than the "equilibrium" line. The turning point is around noise level 25. (Best viewed in color)

the effect of parameter selection over the solution. To summarize, the results clearly answer questions **Q1**, **Q2** and **Q4** affirmatively.

3.3.4. Synthetic Datasets

In order to investigate how well the embedding step reduces noise, we generate ground truth points with the help of synthetic data. To generate the data, we used Thomas Brinkhoff's data generator [7]. It allows to generate trajectories according to some underlying road network for different speed and sampling time variation setting. Additionally, we assumed normal noise on the generated observations. it

Statistics for sampling rate=5			
Noise	σ_{error}	student t	Wilcoxon signed rank
30	0.029	1	0.00012
40	0.03992	1	0.0027
50	0.036	1	0.00122
Statistics for sampling rate=10			
Noise	σ_{error}	student t	Wilcoxon signed rank
30	0.027	1	0.0004
40	0.034	0	0.047
50	0.054	1	0.00941

Table 3.1.: Statistics table for comparison with HMM

Noise Ratio	KMM Error	ERR SR=5	ERR SR=10
30	0.1463	0.1783	0.1480
40	0.3022	0.3221	0.2350
50	0.3739	0.4671	0.3979

Table 3.2.: Comparison of average KMM error with average Randomized Rounding (ERR) error over different Noise Ratio and Sampling Rates (SR)

is well known navigation systems produce noise that is normally distributed with average of noise (σ_N) ranging from 10-100 meters. For instance, Krumm and Newson [44] discussed the different types and amount of noise in real-world movement data and have shown that they can be described well by a Gaussian shape.

Specifically, we generated 100 trajectories of average length 50 from the street network of Oldenburg, Germany. Then, we added noise σ_N for $\sigma_N = 20, 30, \ldots, 100$. This resulted in an overall set of 900 trajectories. As baseline for comparison we used a "project to the closed point in the street network" approach. To get a fair comparison, we also used the "project to the closed point in the street network" as "rounding" method for KMM. We report on the root means squared difference in meters achieved by the two methods.

The results are summarized in Fig. 3.4. As one can see, in all cases the embedding indeed reduced noise considerably. Moreover, it performs better with increasing noise levels as the regression lines have consistently a smaller slope than the "equilibrium" line. The turning point is around noise level 25. This clearly answers question **Q3** affirmatively.

To summarize, the results of our experiments indicate that kernel methods can indeed be used for map matching and achieve results comparable to state-of-the-art

techniques.

3.3.5. Parameter Selection

KMM has following input parameters, namely, σ_n, standard deviation of noise; σ_G and $\sigma_G^{'}$, i.e., kernel widths for input and output graphs; λ, stiffness parameter for regularization term of F_o; α, the rounding step parameter and k_w, width of sliding window for objective function F_o. We discuss selection process for these parameter as following:

σ_N is required as an input to F_o in the regularization term. For the experimental results, our method for estimating σ_N is carried out on a holdout sample of 20 ground truth points and the approach is same as [44]. Alternative strategies include estimators from other map matching solutions or empirical standards for given application. λ is the stiffness parameter of regularization term. A reasonable range for choice of λ is between 0.5 and 1. Clearly, $\lambda > 1$ make the standard deviation of Ψ more than the trajectory while $\lambda < 0.5$ is too restrictive on output. In our experiments, have chosen λ=0.6. k_w is the sliding window width for execution of optimization step. Fig. 3.4 shows the results for $k_w = \{3, 4, .., 6\}$ over a synthetic data set. We have chosen $k_w = \{4, 5\}]$ for most of our experiments. According to our observation, $k_w > 6$ proves a bit time consuming during optimization while $k_w < 3$ is not sufficient to capture the local geometry. α- is the parameter 'governing relationship between Graph and Euclidean distance'. We set it to 1.5 which means that Graph distance should not be greater than 1.5×Euclidean distance. A reasonable range is $1 \leq \alpha \leq 2$. In the following text, we provide a visual analysis of parameter selection for the optimization function F_o to guide the user in the right direction.

Analysis of Optimization Parameters

The optimization function F_o has five parameters: k_w, $\sigma_{G'}$, σ_G, σ_N, and λ. These parameters can be learned from the equation along with the $G^{'}$. However from an analytical perspective it is very important to study them so that the user has a fair idea about their initialization for learning purposes or if he wants to give them as a direct input. Please note that in all of the following figures, the $G^{'}$ or output trajectory is represented by a series of red lines connected by dots which are individual $G_i^{'}$, while G_i or input trajectory is represented by a series of blue lines with input points blue + signs. The street network is shown in gray edges.

1. **Regularization**

λ is the regularizee on the width $\sigma_{GG'}$. It is quite clear that a large value of λ results in rather flat surface for the derivative of $K_{GG'}$ over the solution surface. This allows the first term of the objective function to dictate the solution more and optimize very quickly. On the other hand a smaller value of λ keeps the resulting points so close to G that the first term is unable to optimize in a good way. Therefore we want to have a value of λ which keeps the solution in a reasonable radius near G and is able to optimize in conjunction with first term. We clarify our point of view in Fig. 3.5 on page 58, we also show that keeping λ with in the above mentioned constraints is not so difficult. Notice that we keep the values of $\sigma_{G'}$, σ_G, and σ_N constant to study the impact of λ only. We show through visual analysis that the change in objective function optimization w.r.t. λ is smooth and the range of λ for good results in not very small. In our experiments the limit for λ value is between 0.5 and 1.5.

2. **Undesirable Results**

The absolute value and relative ratio of kernel widths $\sigma_{G'}$ and σ_G is very important for producing the right results. Primarily, the undesirable results fall into two categories

a) **(Over Fitting)** Over fitting in this context happens when we get a copy of the input trajectory as output or an amplification of the noise. Fig. 3.7 illustrates this phenomena over real world trajectories along with selected parameters. Essentially, If $\sigma_{G'} > \sigma_G$, the it means that smaller distances in the input trajectory will counted as larger distances for output, and, this will amplify the noise in input trajectory. Another important consideration is the absolute values of kernel widths. Choosing a small absolute value for kernel width, e.g., 100 meters will ignore the over all structure of the trajectory and produce a copy of the input trajectory.

b) **(Under Fitting)** Under-fitting happens when we get a smooth curve like version of the input trajectory that does not respect the underlying street network structure. Fig.3.6 illustrates this phenomena over real world trajectories along with selected parameters. If $\sigma_G \gg \sigma_{G'}$ then it means that any large distance in the input trajectory will be mapped to too smaller a distance in the output trajectory. Effectively, this will results into a curve like output trajectory, i.e., more contracted and straightened version of the input trajectory than we want. On the similar lines, a large absolute value for kernel widths, e.g.,1000 meters will ignore the local

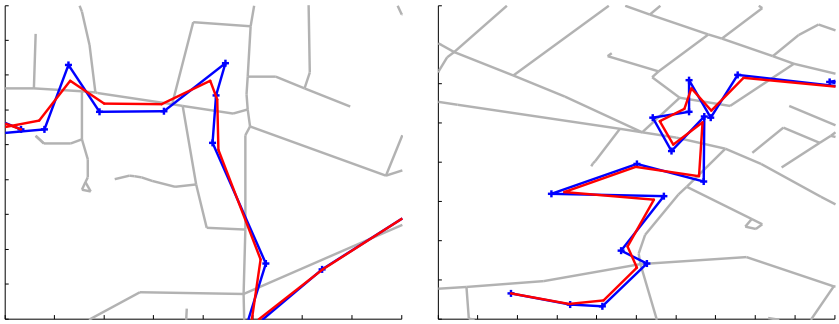
influence of the street network graph structure on the output trajectory.

3. **The Right Direction for Parameter Selection**

   We have shown through experiments that $\sigma_{G'}$ and σ_G are correlated. This restricts our degree of freedom from choosing σ_G independently to choosing a proportionality constant. We tried to fit a curve in our experiments to the values of c and it comes out that $1 < c < 1 + \sigma_{G'}^{-k_w}/10$, this happens due to repetitive addition of noise in shortest path kernel of G. The main concern then remains choosing $\sigma_{G'}$ which depends on the user's choice about how quickly he wants his kernels to diminish. For vehicle navigation system we guessed that [200..500] meters is a good range for choosing $\sigma_{G'}$ which roughly means that a driver's current location on road network is learned from his location on the road network within previous 300-400 meters or so. In a way, we are determining how far we should look when seeking structural information,as, the kernels diminishes quickly once the difference in points is greater than one standard deviation. We only observe that the constant c increases as the value of $\sigma_{G'}$ increase.

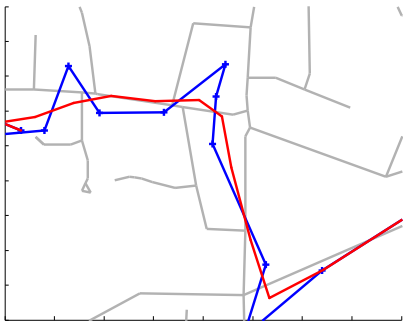
   Fig. 3.8 describes the right direction for parameter selection according to tehse observations. In all the figures, the ratio $\frac{\sigma_{G'}}{\sigma_G} \lesssim 1$ which is in the right direction and produces good optimization results in first two rows. For bottom row, the kernel width $\sigma_{G'} = 1000$ is too large and can produce undesirable results. We have already described that $\sigma_{G'}$ and σ_G are correlated which restricts our degree of freedom for choosing σ_G independently of $\sigma_{G'}$. The main concern that remains is the value of $\sigma_{G'}$ which determines how quickly the kernel will diminish in optimization. Consequently, a large value for $\sigma_{G'}$ will result in ignorance of local trajectory structure. Similarly, small value will result in a narrow focus of local structure ignoring the graph constraints in street network.

4. **Sliding Window Width for KMM** We have already stated that $k_w > 6$ proves a bit time consuming during optimization while $k_w < 3$ is not sufficient to capture the local geometry. In this section, we provide experimental results to show the effect of k-neighbourhood over the optimization step. Actually, the value of k_w is most related to σ_N, i.e., the local noise in the regularization term, where a smaller k is better for smaller values of noise and a large k is good for higher values of noise. Figs. 3.9, 3.10, 3.11 show the effect of $k_w = \{4, 5, 6\}$ w.r.t. noise and kernel widths. It is clear from visual analysis as well that for lower values of noise, a smaller k is better and large k is good for higher

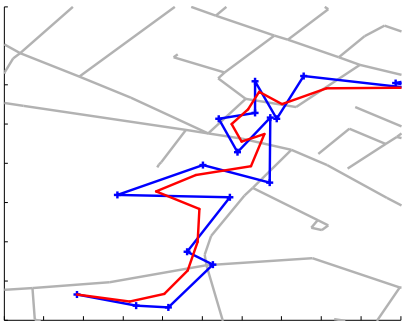


(a) Trajectory 1 , ($\lambda = 0.1$)

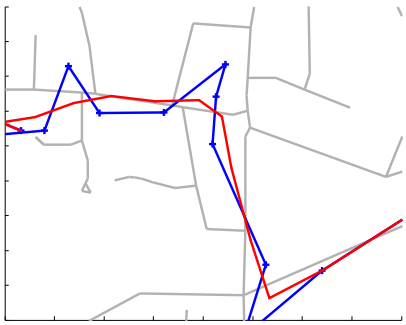
(b) Trajectory 2 , ($\lambda = 0.1$)



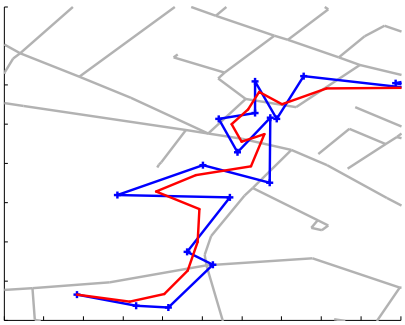
(c) Trajectory 1 , ($\lambda = 1$)



(d) Trajectory 2 , ($\lambda = 1$)

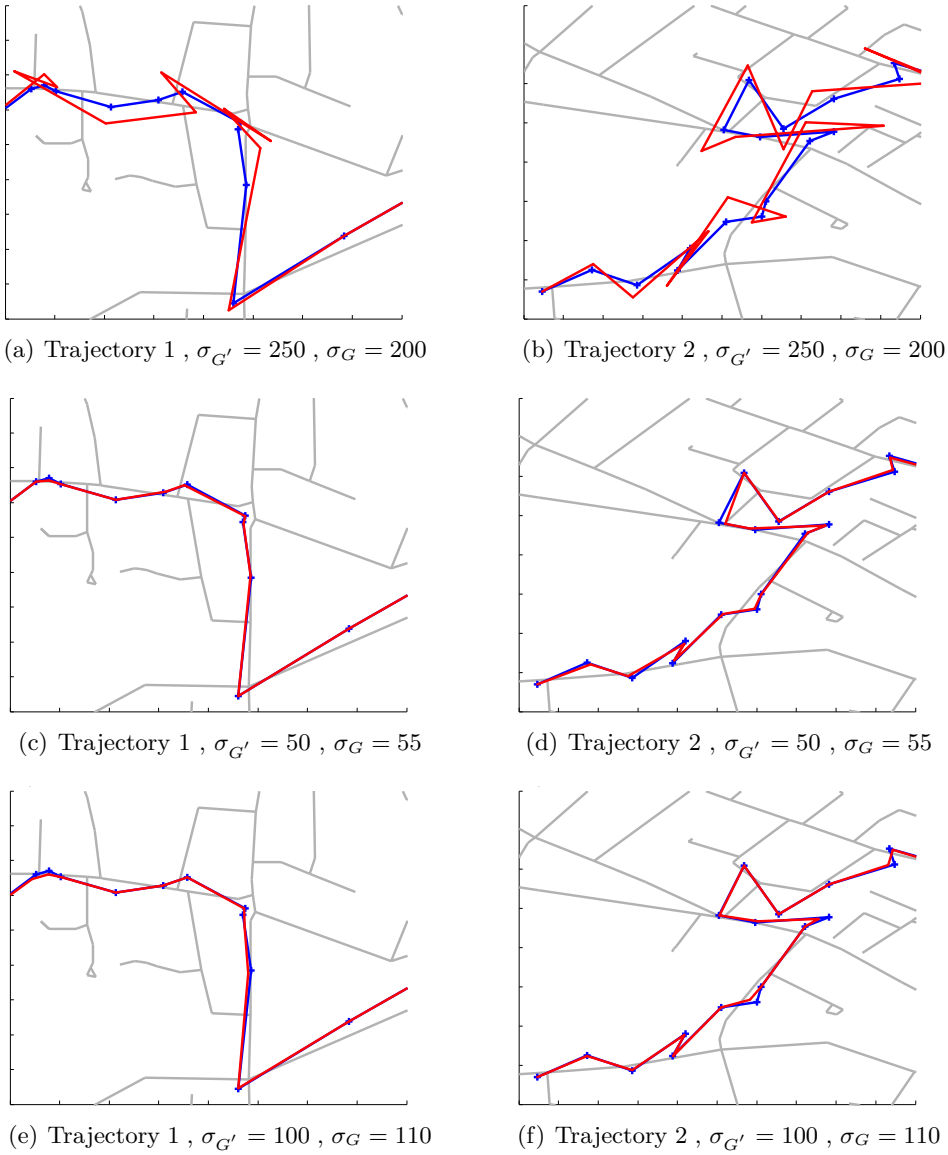


(e) Trajectory 1 , ($\lambda = 2$)



(f) Trajectory 2 , ($\lambda = 2$)

Figure 3.5.: **Regularization:**. The effect of λ on the optimization solution is illustrated, where $G^{'}$ or output trajectory is represented by a series of red lines connected by dots which are individual $G_i^{'}$, and, G_i or input trajectory is represented by a series of blue lines with input points blue + signs. The underlying street network is shown in gray edges. (Top) $\lambda = 0.1$, a small value restricting the solution in a close neighbourhood. (Middle) $\lambda = 1$, allowing the embedding term to dictate the solution. (Below) $\lambda = 2$, same as $\lambda = 1$, however, larger values can decrease the effect of regularization.



(a) Trajectory 1 , $\sigma_{G'} = 250$, $\sigma_G = 200$

(b) Trajectory 2 , $\sigma_{G'} = 250$, $\sigma_G = 200$

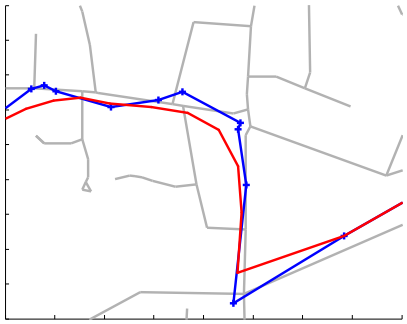
(c) Trajectory 1 , $\sigma_{G'} = 50$, $\sigma_G = 55$

(d) Trajectory 2 , $\sigma_{G'} = 50$, $\sigma_G = 55$

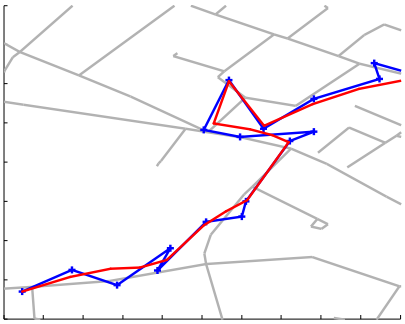
(e) Trajectory 1 , $\sigma_{G'} = 100$, $\sigma_G = 110$

(f) Trajectory 2 , $\sigma_{G'} = 100$, $\sigma_G = 110$

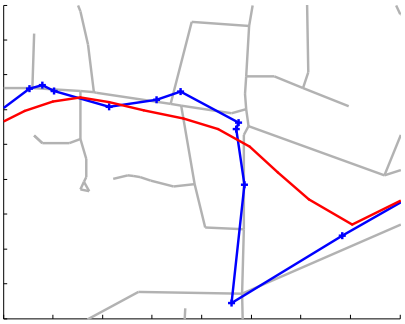
Figure 3.6.: **Over Fitting:** The phenomena of over fitting is illustrated where G' or output trajectory is represented by a series of red lines connected by dots which are individual G'_i, and, G_i or input trajectory is represented by a series of blue lines with input points blue + signs. The underlying street network is shown in gray edges. (a—b) The ratio of kernel widths, i.e., $\frac{\sigma_{G'}}{\sigma_G}$ is > 1, which results in larger distances in the output trajectory as equivalent to smaller distances in the input trajectory (by the MDS formula). Thus, X amplifies the noise of Y. (c—h) The ratio of kernel widths, i.e., $\frac{\sigma_{G'}}{\sigma_G}$ is < 1, however, the absolute value of kernel widths is too small to reduce the noise and hence overt-fits the original trajectory.



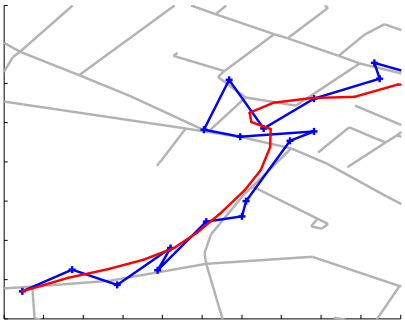
(a) Trajectory 1, $\sigma_{G'} = 100$, $\sigma_G = 200$)



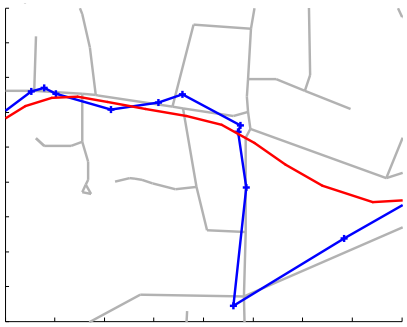
(b) Trajectory 1, $\sigma_{G'} = 100$, $\sigma_G = 200$)



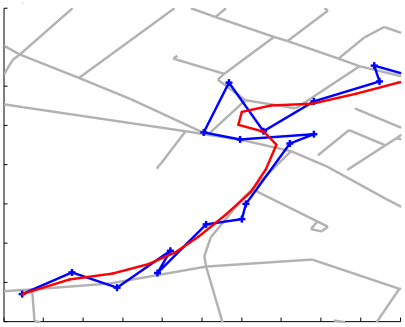
(c) Trajectory 1, $\sigma_{G'} = 250$, $\sigma_G = 500$)



(d) Trajectory 2, $\sigma_{G'} = 250$, $\sigma_G = 500$)

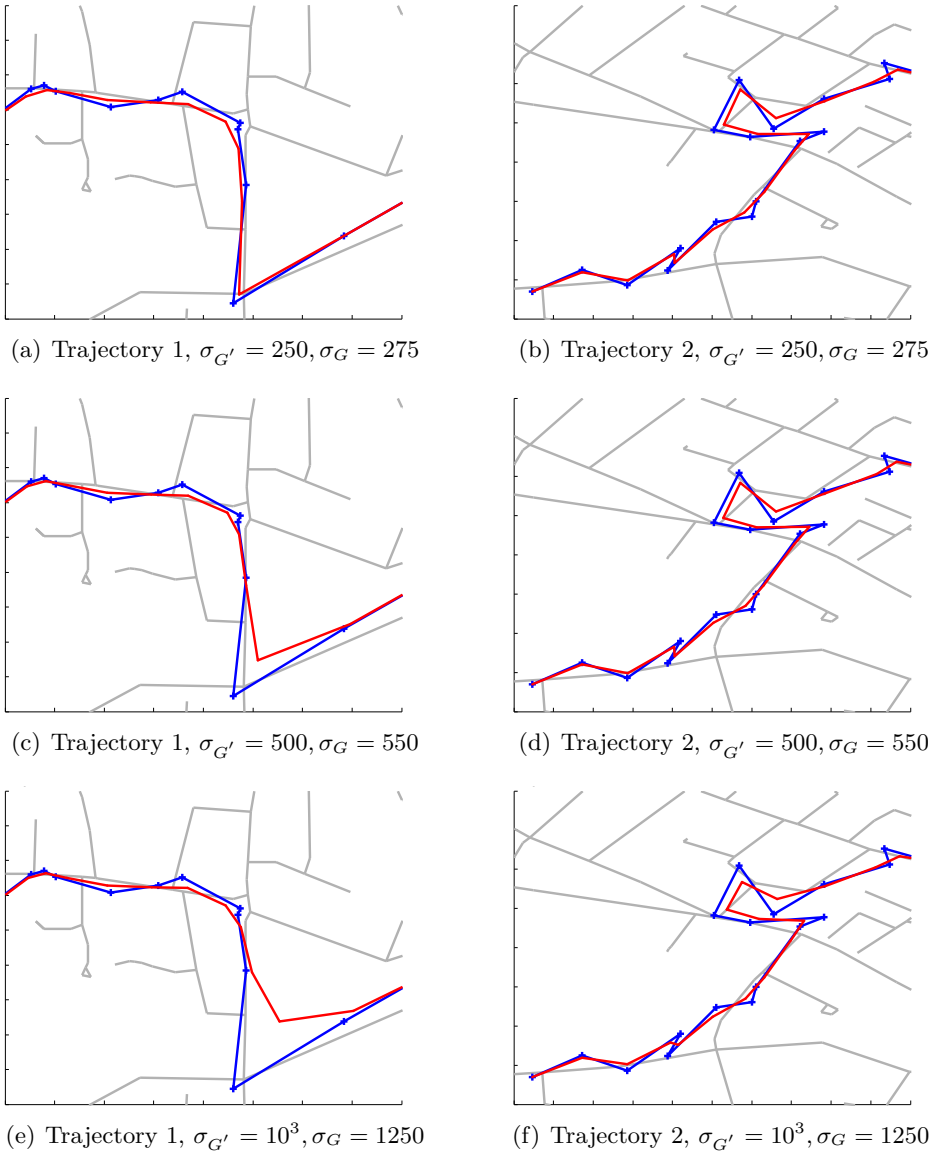


(e) Trajectory 1, $\sigma_{G'} = 500$, $\sigma_G = 10^3$



(f) Trajectory 2, $\sigma_{G'} = 500$, $\sigma_G = 10^3$

Figure 3.7.: **Under Fitting:** with $\sigma_G \gg \sigma_{G'}$. In all the figures, $\frac{\sigma_G}{\sigma_{G'}} = 2$, therefore the resulting trajectories have smaller distances between individual points, resulting in a under fitting, i.e., smoothing of out trajectory with a disregard to street network structure. Additionally, a large absolute value of kernel widths (or neighbourhood of influence) can cause under-fitting as well. (Top)- small neighbourhood of influence, i.e., $\sigma_{G'} = 100$ meters with $\sigma_G \gg \sigma_{G'}$, resulting in a non -smooth contraction. (Middle)- Under fitting: because $\sigma_G \gg \sigma_{G'}$ though absolute kernel widths are chosen in a better manner, i.e., $\sigma_{G'} = 250$ meters. (Below). Increasing the absolute kernel width with $\sigma_G \gg \sigma_{G'}$ produce further undesirable results.



(a) Trajectory 1, $\sigma_{G'} = 250, \sigma_G = 275$

(b) Trajectory 2, $\sigma_{G'} = 250, \sigma_G = 275$

(c) Trajectory 1, $\sigma_{G'} = 500, \sigma_G = 550$

(d) Trajectory 2, $\sigma_{G'} = 500, \sigma_G = 550$

(e) Trajectory 1, $\sigma_{G'} = 10^3, \sigma_G = 1250$

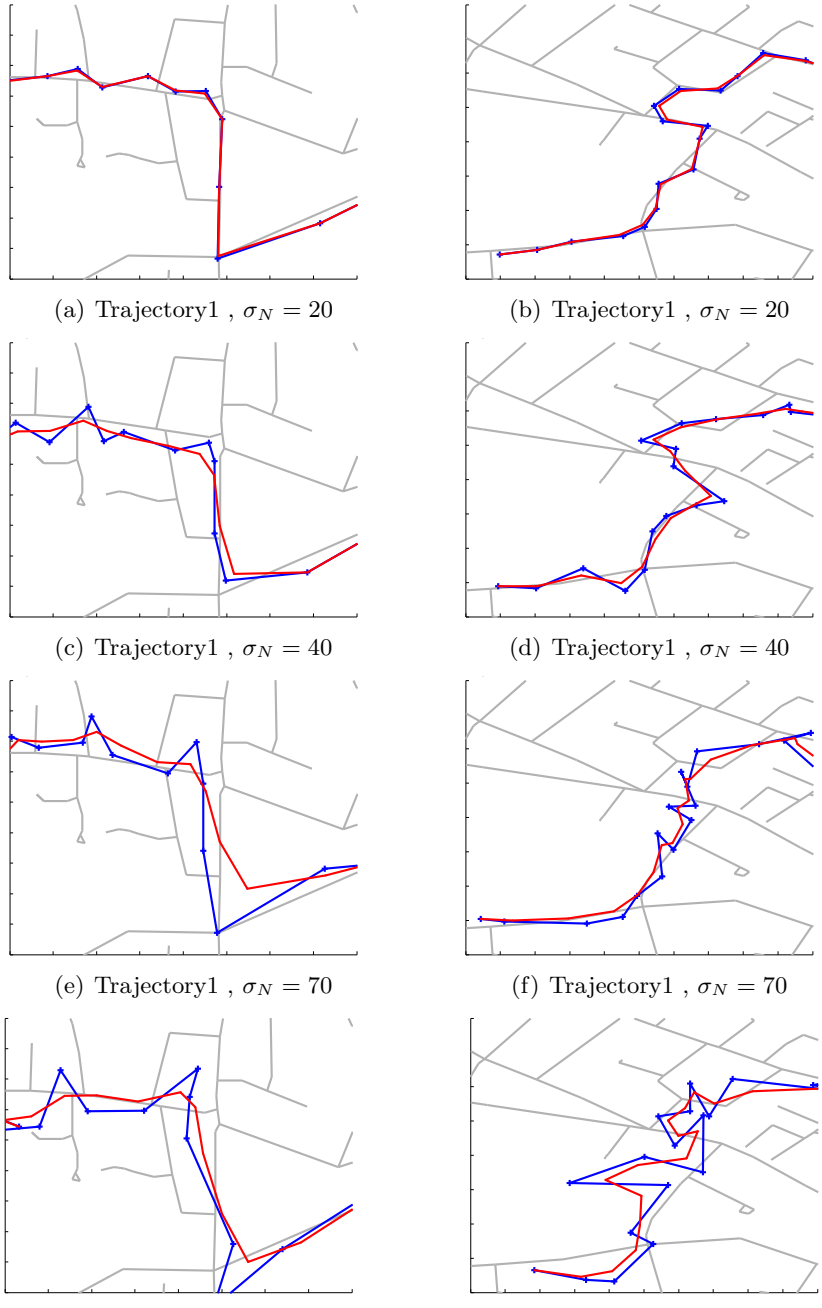
(f) Trajectory 2, $\sigma_{G'} = 10^3, \sigma_G = 1250$

Figure 3.8.: **In The Right Direction:** (Top and Middle) The relative ratio and absolute values of kernel widths are balanced. (Bottom) The ratio $\frac{\sigma_{G'}}{\sigma_G} \lesssim 1$ which is in the right direction but the kernel width $\sigma_{G'} = 1000$ which is too large and can produce undesirable results. Explanation: $\sigma_{G'}$ and σ_G are correlated which restricts our degree of freedom for choosing σ_G independently of $\sigma_{G'}$. The main concern that remains is the value of $\sigma_{G'}$ which determines how quickly the kernel will diminish in optimization. Consequently, a large value for $\sigma_{G'}$ will result in ignorance of local trajectory structure. Similarly, small value will result in a narrow focus of local structure ignoring the graph constraints in street network. In our view, 250 -500 meters is the right choice for $\sigma_{G'}$ w.r.t. how far we should look when seeking structural information.

values of noise. In our experiments, we found that k=3 is a good value for $10 \leq \sigma_N \leq 40$ meters. Like wise k=4 is a good value for $40 \leq \sigma_N \leq 70$ meters while k=5,6 are better values for higher values of noise.

Summary

In this chapter, we have demonstrated that structural information can be used to improve a fundamental task related to the analysis of raw trajectories, i.e., map matching. In a nutshell, we view map matching as a regression problem where the noisy observations constitute the sampled trajectory and the noise free result is the ground truth path over a street network. Then, we have provided an embedding scheme that preserves the structural relationships and reduces the noise during the embedding step. At the core of the embedding step lies the 'geodesic distance based RBF kernel' (outlined in Chap.3), which is used to capture the structural information during embedding step. After the embedding step, a rounding step is provided in order to assign the resulting points to the underlying street network. The results of our method show that Kernelized Map Matching (KMM) is significantly better than state-of-the-art.



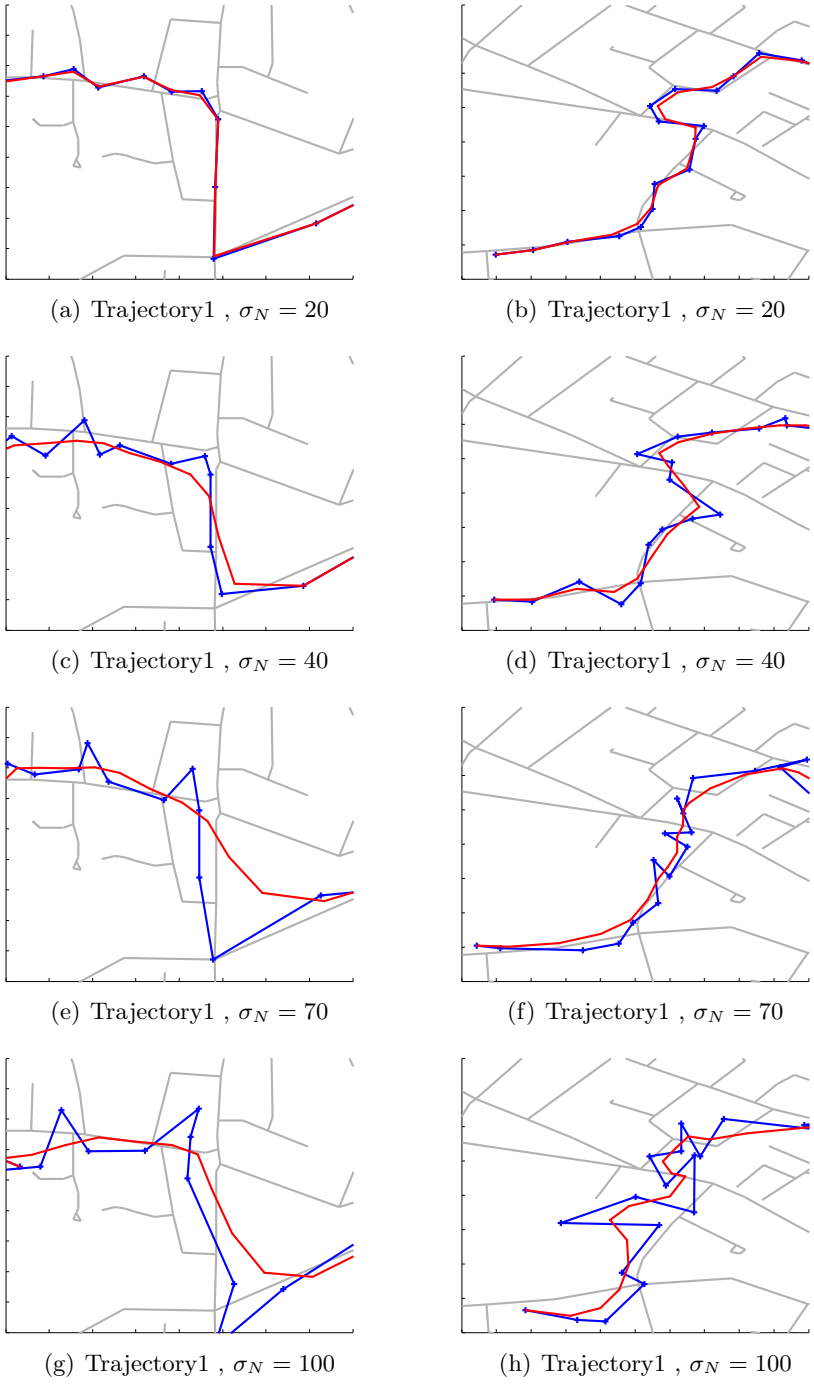
(a) Trajectory1 , $\sigma_N = 20$ (b) Trajectory1 , $\sigma_N = 20$

(c) Trajectory1 , $\sigma_N = 40$ (d) Trajectory1 , $\sigma_N = 40$

(e) Trajectory1 , $\sigma_N = 70$ (f) Trajectory1 , $\sigma_N = 70$

(g) Trajectory1 , $\sigma_N = 100$ (h) Trajectory1 , $\sigma_N = 100$

Figure 3.9.: **Sliding Window Width=4**. (c–g) $k_w = 4$ is a better choice for medium–high range of noise. However, there is one result where we can do better, i.e.,(h) where $\sigma_N = 100$ and $k_w = 4$ seems to be unable to contract the trajectory in the right way.



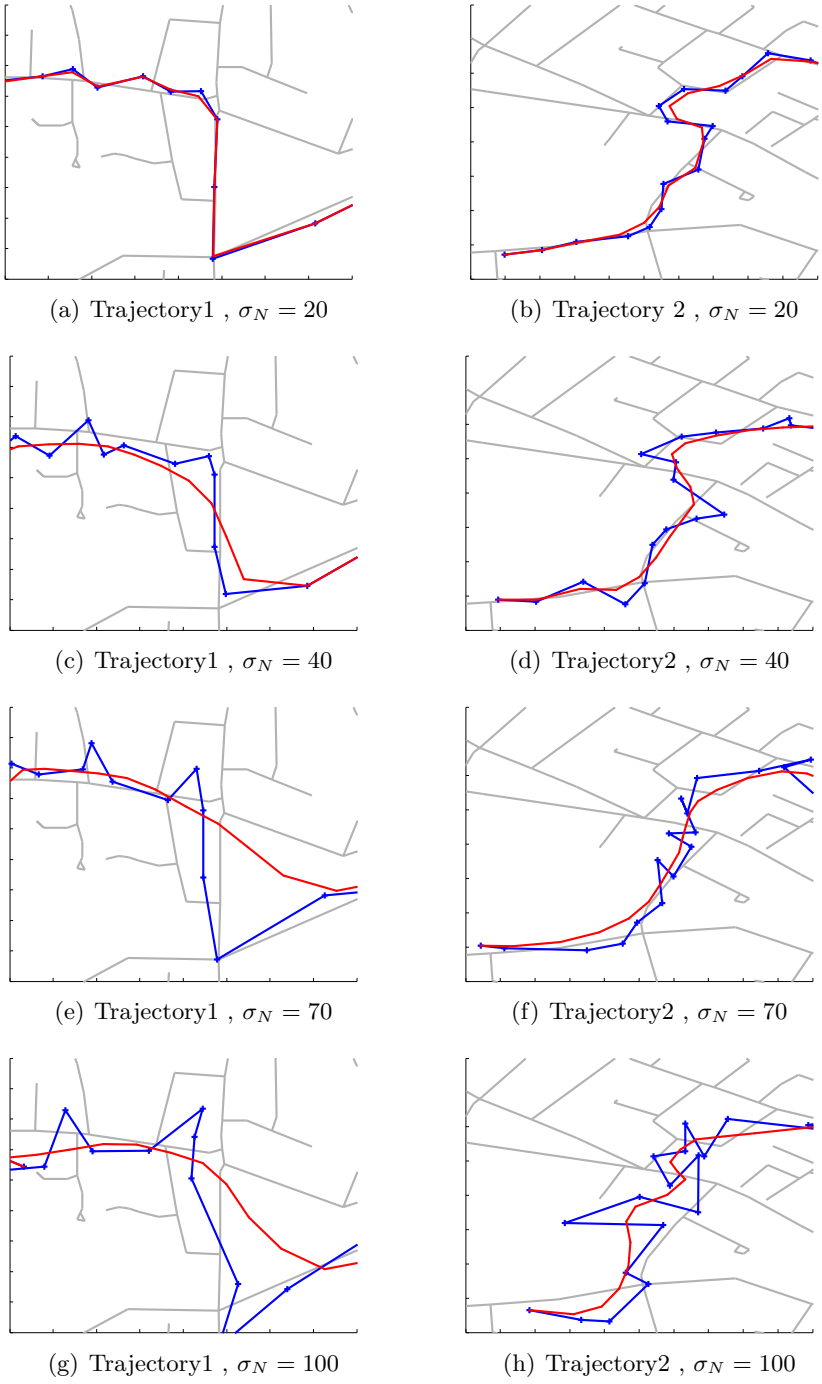
(a) Trajectory1 , $\sigma_N = 20$ (b) Trajectory1 , $\sigma_N = 20$

(c) Trajectory1 , $\sigma_N = 40$ (d) Trajectory1 , $\sigma_N = 40$

(e) Trajectory1 , $\sigma_N = 70$ (f) Trajectory1 , $\sigma_N = 70$

(g) Trajectory1 , $\sigma_N = 100$ (h) Trajectory1 , $\sigma_N = 100$

Figure 3.10.: **Sliding Window Width=5**. (a–f) $k_w = 5$ tends to under fit for low-medium ranges of noise as compared to $k_w = 4$. (g–h) $k_w = 5$ is a better choice for higher values of noise as compared to $k_w = 4$.



(a) Trajectory1 , $\sigma_N = 20$ (b) Trajectory 2 , $\sigma_N = 20$

(c) Trajectory1 , $\sigma_N = 40$ (d) Trajectory2 , $\sigma_N = 40$

(e) Trajectory1 , $\sigma_N = 70$ (f) Trajectory2 , $\sigma_N = 70$

(g) Trajectory1 , $\sigma_N = 100$ (h) Trajectory2 , $\sigma_N = 100$

Figure 3.11.: **Sliding Window Width=6**. (a–g) $k_w = 6$ tends to under fit for low-high ranges of noise as compared to $k_w = 4, 5$. (h) $k_w = 6$ is a better choice as compared to $k_w = 4, 5$.

4. Preserving Structure in Symbolic Trajectories

"In silence and movement you can show the reflection of people." — Marcel Marceau

The distance function capturing the structural information in symbolic trajectories is presented in Chap. 2, i.e., geodesic distance based embeddings with alignment. In this chapter, we show how to use this distance function in combination with out-of-the-box (biological) sequence analysis to solve real world problems for symbolic trajectories; in particular, our focused tasks,i.e., *Trajectory clustering* and *Traffic event detection*. let's start off by motivating the analysis of symbolic trajectories along with the two tasks mentioned above.

In many geo-analytics and traffic applications, our goal is to carry out a high-level analysis of raw trajectories. In order to reach this goal, we discretize the raw trajectories into the so-called symbolic trajectories. These symbolic trajectories can then be fed to graph and sequential approaches for further analysis. Recall that, we have given a general mechanism to convert raw trajectories into symbolic trajectories and an alignment based structure preserving kernel over these trajectories in Chap.3. In this chapter, we take two important tasks for symbolic trajectory analysis, i.e., trajectory clustering, and traffic event detection and apply biological sequence analysis in combination with our alignment kernel to solve them. In the following text, we will motivate and introduce the above mentioned tasks for symbolic trajectory analysis. Then, we will also describe the reasons which led to the choice of biological sequence analysis in order to solve these real world problems in symbolic trajectories. Afterwards, we will illustrate the usage of biological sequence analysis over a small real world running example before continuing with the description and experimental evaluation of our methods. Lets' start by introducing one of the focused, i.e., trajectory clustering under the motivated settings of user activity analysis.

Clustering of trajectories is an important task in order to identify the groups of moving objects that exhibit similar movement behaviour. Many applications in trajectory analysis come under this scenario. For example, consider 'finding flocks of birds by identifying the clusters in their trajectories' or ' clustering trajectories of an individual human to identify her routes (user activity analysis)'. *User activity analysis* essentially abstracts a person's movement from raw GPS data to places of interest and then mines relationships among these places [39, 68]. Imagine we want to extract a person's activities and mine her daily routines based on these activities, i.e., we ask the following questions: What are interesting and/or frequent stops (stay points) in a person's travel routines like Banks, Restaurants, Supermarkets, Gyms, Home, Office place? What route does she usually choose for travelling between two stay points? What time does she usually choose to travel from home-to-office and back? What are the shopping routines and weekend routines? And, what time does she usually choose to do sports? Going one step further, one is actually interested in combinations of primitive routines that a user does such as Home- Office- Sports- Home, Home- Office- Shopping- Home, or Home- Bank- Shopping- City centre- Home. Such clusters of activities in a user's movement over the day are clearly interesting in order to identify the user's most likely activity sequences and profiling of multiple users on the basis of their activities.

The other important analytical task we investigate in the present chapter is that of *event detection*[1]. Consider, e.g., 'time-series based movement data where a sensor records movements of entities over a short window of time. For example, an optical sensor placed over a door of an office building reports the number of people entered after every 30-minute intervals. Or, an inductive loop sensor on a highway reports the numbers of vehicle passed on a 5 minute basis [29]. This recorded data captures periodical patterns of human activity, e.g., highways are usually busy during morning and early evening time because of traffic 'towards and from' work place. Weekdays and weekends can show periodic patterns of their own. Typically, these *periodical activity* patterns are mixed in sensor data with bursts of unusual traffic called 'events'; outliers but not noise. Example events include: traffic congestion/jams on a high way, a large meeting in an office or a concert/football game near a highway sensor. Thus, we have to separate the normal traffic activities from traffic events. Unfortunately, there are no labels which leads us to a problem i.e., *separation of normal traffic from event*. Furthermore, an event is not a single unusually high value, instead it is a chain of sensor reading having its own dynamics. As Smyth *et*

[1] http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/

al. [29] nicely described it: in order to separate normal traffic from events, "we need to define a model of uncertainty (how unusual is the measurement?), and additionally incorporate a notion of event persistence, i.e., the idea that a single, somewhat unusual measurement may not signify anything but several in a row could indicate the presence of an event." In this chapter, we will employ a hidden Markov model to capture *event persistence*, i.e., encoding the dependencies between consecutive readings.

Dozens of approaches have been developed for interesting tasks such as trajectory clustering — extraction of routines from trajectory data — and 'traffic event detection' — identify *unusual* bursts of traffic frequencies to detect traffic jams, accidents or gathering at meeting places. We have selected these tasks as they illustrate well the two specific issues we have to deal with when working with traffic data:

1. Trajectory data contains noise and missing label information.

2. Trajectory data is composed of sequences in time and space of different lengths

To address both issues, the biological sequence view helps a lot. Actually, we argue that because we abstract the raw data into sufficiently small alphabets using standard discretization techniques, we can instantly solve **(a)** and in turn **(b)**. Why? sequence analysis will do the rest for us. It was designed to deal with large numbers of variable length sequences. Actually, there is a rich toolbox of *Biological Sequence Analysis* for all sorts of data analysis tasks including clustering, classification, visualization, and probabilistic modelling of data, among others. In a sense, biological sequence analysis is a field which has a commonality to traffic data. Because, biology was (and is still) facing the problem of sequence analysis in the wild in order to understand the immense amount of data produced by for instance the Human Genome Project. On this quest, many powerful methods have been developed, often based on principles of probabilistic models. Consider for instance the classical task of aligning to sequences, say HAL and HL. Intuitively, two sequences in an alignment are of same length and similar symbols are matched per position. Getting the sequences to the same length is realized by introducing gaps between consecutive symbols. In our example this could result in H-H,A-gap,L-L; for a more complicated example please see Fig. 4.1. Indeed, many more correspondences between biological sequence methods and analysis of trajectories tasks exists, and a biologist would typically use multiple tools and views to solve a task at hand. Examples include conserved regions (consecutive areas of high support), consensus sequences (most probable sequence after alignment) and 'sequence logo' i.e., entropy at different sequence positions. All

of this carries over to the traffic domain as illustrated in Fig. 4.1.

Indeed, whereas traffic sequences are continuous in time and space, biological sequences are composed of discrete symbols over discrete time. In turn, one may argue that we cannot make use of biological sequences techniques for traffic data. In this chapter, we make the somewhat surprising claim that this is not the case. Actually, we demonstrate that standard discretization techniques for traffic data together with biological sequences analysis can result in state-of-the-art performance. Specifically, we

1. Revisit and strengthen the link [35] between *analysis of trajectories* and *biological sequence analysis*.

2. We use geodesic distance based alignment kernel in combination with 'black box' biological techniques, namely sequence alignments and profile hidden Markov models, and demonstrate that state-of-the-art performance can be achieved for two important analytical tasks: *trajectory clustering* and *traffic event detection*.

3. Finally, again by exploiting the link established, we introduce 'Traffic Logos', a novel visualization technique that provides a compact yet descriptive view on the *information content* of traffic sequences.

However, in order to apply biological sequence analysis to trajectory data, we have to be a little bit more careful. As the similarity scores used for biological sequences do not carry over to the traffic domain, the invariances in both domains are completely different. Consequently, we have to come up with our own similarity scores. This is not an obvious step in many — if not most — tasks. For this purpose, we can use the geodesic distanced based alignment kernel (introduced in Chap.3). In short, using well-known discretization techniques such as stay-point detection and map matching, we can turn most — if not all — traffic sequences into a "biological" sequence. Then, we apply the rich toolbox for biological sequence analysis to traffic data. For instance, by just looking at complex traffic data through the biological glasses of sequence logos we get a novel, easy-to-grasp visualization of the data, called "Traffic Logos". Sequence alignment can be used for activity analysis, and profile hidden Markov models are well suited for capturing event persistence during event detection. Actually, our empirical evaluation on three real-world data sets demonstrates that exploiting the link between traffic and DNA can result in state-of-the-art performance.

We proceed as follows. We start off by reviewing biological sequence analysis in terms of traffic symbols. Then, we show how to actually convert traffic data to symbol sequences. Afterwards, we demonstrate that the link established can achieve state-of-the-art performance within our analytical tasks. Before concluding, we briefly touch upon related work.

4.1. Sequence Analysis for Trajectories

Analysis of trajectories is a fascinating area of geo-analytics and geographical information systems that impacts the lives of millions of people every day. Another well-known scientific field that impacts lives of millions is biological sequence analysis. It has experienced an incredible evolution in the recent past, especially since the Human Genome project. Similarly, both face a similar challenge, namely the identification of relevant patterns in massive sequential information. Indeed, whereas biological sequence analysis has mainly focused on sequences of (few) symbols, analysis of symbolic trajectories often focus on sequences of continuous values. Thus, one may argue that building bridges between them is insurmountable. In this following example, we show that this is actually not the case.

Assume that we have turned a traffic data that is continuous in space and time into a sequence of traffic symbols; we will show how to do this in the next section. We start off by explaining traffic sequence alignment and then continue to profile hidden Markov models, conserved regions and visualization techniques. Before we proceed, let us introduce some notations and definitions required. An alphabet Σ with $|\Sigma| = l$ is a set of symbols in (traffic/biological) sequences. Consider for example a set of labelled stay points from a user's GPS traces: Home, Work, Shopping, Tennis, Friends, Bank and City-Center. That is $\Sigma = \{H, W, S, T, F, B, C\}$. Let S denote the set of all sequences in a dataset. A single sequence $s \in S$ is a sequence of symbols in Σ, say $s = HWTHFH$. Say now that we take actually one week of daily raw GPS traces of a user:

$$S = \left\{ \begin{array}{l} s_{mon} : \text{“}HWTHFH\text{”} \\ s_{tue} : \text{“}HWSH\text{”} \\ s_{wed} : \text{“}HWH\text{”} \\ s_{thu} : \text{“}HWSFH\text{”} \\ s_{fri} : \text{“}HWH\text{”} \\ s_{sat} : \text{“}HBSCH\text{”} \end{array} \right\}. \tag{4.1}$$

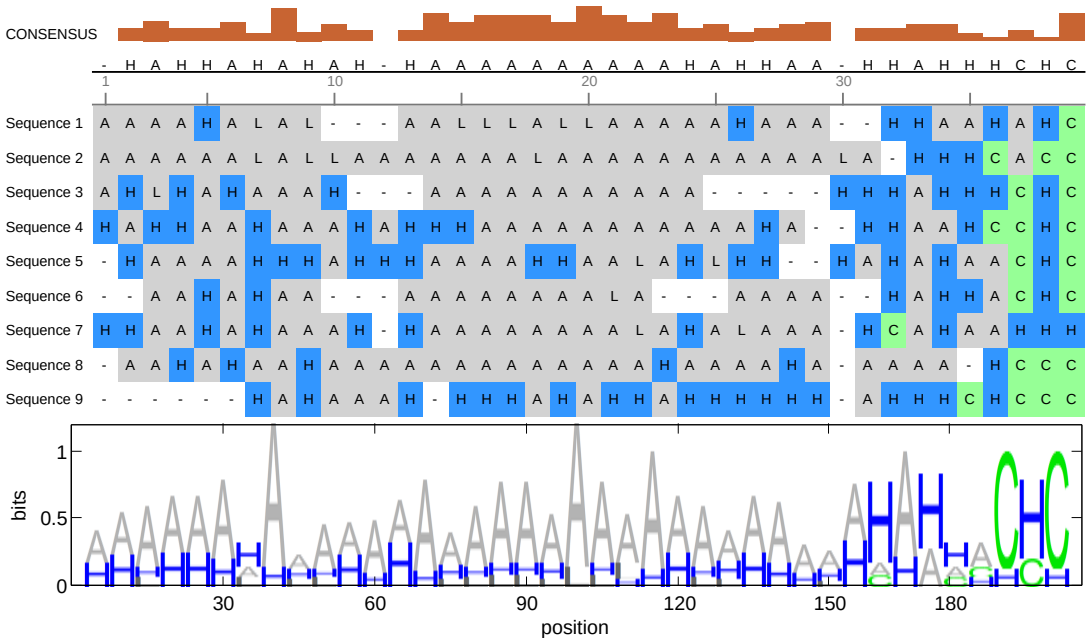


Figure 4.1.: Traffic data through the glasses of a biologist: traffic frequencies over time for 9 baseball games at 'Dodger's stadium'. L denotes low, A average, and H high densities; C denotes congestions. (Top) Consensus sequence and 3 conserved regions. The histograms show the amount of symbols' conservation. As one can see, in the first region, traffic is average but high near the start of the games. During the game, the traffic is average. When the games is over, at 180 minutes, we have congestions and high traffic densities. (Middle) The multiple sequence alignment that yield the conserved regions. (Below) Traffic logo shows the 'information content' and in turn reveals trends.

Given two sequences s_1 and s_2 of length n and m respectively, an alignment π_{s_1,s_2} with $|\pi| = p \leq n + m$ defines a correspondence between the elements of s_1 and s_2 and additional gaps (if required). Gaps are essentially null elements (sometimes we will also denoted them using '-' or $\emptyset$) meaning that we do not match an element of the one sequence with an element of the other sequence. More formally

$$\pi_{s_1,s_2} = \begin{cases} (\pi_{s_1}(1), ..., \pi_{s_1}(p)) \in \{s_1 \times \emptyset\} \\ (\pi_{s_2}(1), ..., \pi_{s_2}(p)) \in \{s_2 \times \emptyset\} \end{cases} \quad \text{s.t. } |p| < n + m$$

The set of all possible alignments between two sequences s_1, s_2 is denoted as Π_{s_1,s_2}. An alignment problem (sometimes called global alignment) is to find the alignment

Biological Sequence Method	Corresponding Tasks in Trajectory Analysis
Pairwise sequence alignment: Dynamic programming based similarity criteria for variable length sequences	- Similarity criteria between traffic sequences - Flexible Pattern matching [63] - compression of traffic sequences [12]
Multiple Sequence alignment (MSA): Makes all input sequences of the same length by guessing the missing details	- Performing vector like operations - Core step for visualization and probabilistic analysis
Conserved regions: Consecutive areas of high support in data.	- T-Pattern mining [22] - Tagging and compression of sequences [12] - Forming hypothesis about functionality
Consensus sequence/ Profile HMMs: Representative model of a sequence family	- Probabilistic modelling of relevant traffic sequences [29]
Sequence Logos: Information theoretic visualization scheme	- Dense visualization for 'information content' in mobility patterns

Table 4.1.: **Biological sequence methods with potential applications for traffic data**

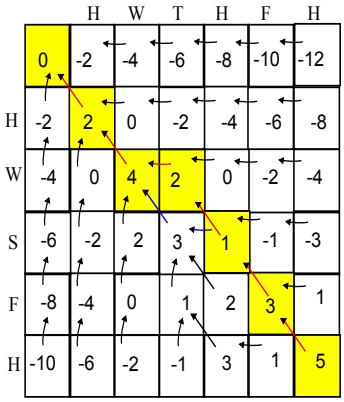
$\pi^\star$ which maximizes a score θ between s_1 and s_2 resulting from the matrix of pairwise similarity scores Δ between symbols:

$$\pi^\star_{s_1,s_2} = \arg \max_{\pi \in \ \Pi_{s_1,s_2}} \theta(\pi) \tag{4.2}$$

To illustrate, we take two sequence from our running example, namely $s_{mon} = HWTHFH$ and $s_{thu} = HWSFH$ with

$$\Delta = \begin{cases} \pi_{s_i,s_j}(i) = +2 \leftarrow \text{ match, i.e. } s_i = s_j \\ \pi_{s_i,s_j}(i) = -1 \leftarrow \text{ mistmatch, i.e. } s_i \neq s_j \\ \pi_{s_i,s_j}(i) = -2 \leftarrow \text{ gap penality, i.e. } s_i = \emptyset \text{ or } s_j = \emptyset \end{cases} \tag{4.3}$$

On a more technical level, an alignment π is actually a path in a dynamic programming matrix where the score of each individual cell is the maximum of three scores:



(a)

Figure 4.2.: Dynamic programming matrix and Viterbi algorithm for pairwise Sequence Alignment. Directional arrows show the cell contributing to the score of current cell. Red arrows and yellow squares describe the optimal path. Blue arrows describe an alternative path with maximal score.

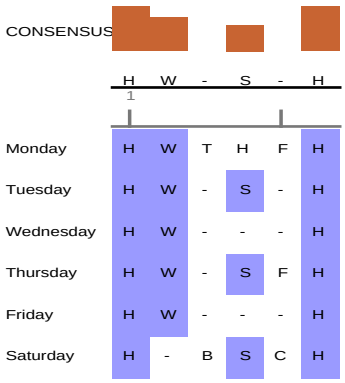
(i) diagonal above + match/mismatch between corresponding symbols; (ii) horizontal above + gap; or (iii) vertical left + gap. The well-known Viterbi algorithm finds the path that maximizes the score $\theta(\pi)$ in $O(mn)$ time and outputs the alignment $\pi^\star$ along with similarity score. Fig. 4.2 illustrates applying Viterbi on our running example. It results in the global alignment between s_1 and s_2:

$$\pi^\star_{s_1,s_2} = \begin{matrix} H & W & T & H & F & H \\ H & W & - & S & F & H \end{matrix} \tag{4.4}$$

with $\theta(\pi^\star) = (2 + 2 - 2 - 1 + 2 + 2) = 5$.

More important than aligning two sequences is multiple sequence alignment (MSA), i.e., to find the best alignment between multiple sequences under similarity matrix Δ, i.e., given k sequences, $s_1, s_2, \ldots, s_k \in S$. This is a fundamental step in many analysis methods for biological sequences as it helps in learning profile HMMs, conserved regions, and visualization, among others. Essentially it builds on top of the pairwise alignment we just described. The exact details, however, are not important for this chapter, and instead we present only a result for our running example in Fig. 4.3. For more details we refer to [15].

Given a multiple sequence alignment, one can compute the so-called 'consensus sequence'. Fig. 4.3 shows the consensus sequence HW-S-H of our runnning example. It describes the most frequent symbols at every position of the alignment (note that



(a)

Figure 4.3.: Multiple sequence alignment showing distribution of symbols. (Top) Three conserved regions: (1) from home to work at the start of the day, (2) shopping (after work), and (3) back home at the end of the day. Conserved regions filter infrequent symbols like T and F. Histograms show the amount of symbol's conservation.

a gap is treated as a symbol here). Intuitively, it say the user goes from home to work in the morning. Then, she goes shopping before coming back home again. An alternative view is provided by so-called 'conserved regions'. Instead of looking at the most likely symbols only, it provides us with the regions (consecutive symbols) with high support in a multiple alignment; the sequence of bars on top of Fig. 4.3. A conserved region indicates that this region is performing a specific functionality for the set of aligned sequences. In our case, H-W is more frequent than S. In other words, she does not shop every day. However, being at home in the morning and the evening is more likely than going to work due to the weekends where our user is not working. Traffic logos provide a richer and more precise description of traffic sequences than would consensus sequences. They are visualizations which show the variability of symbols at specific positions. Specifically, they comprise stacks of symbols placed next to each other. The over all height of one stack denotes the sequence conservation for one particular position in the multiple sequence alignment, whereas the height of a symbol in a stack shows the entropy of that traffic symbol at the particular position. Fig. 4.1 is a traffic logo for a highway traffic densities during baseball games in a nearby stadium.

However, even logos ignore correlations over time and instead treat the positions independently of each other. To overcome this, one typically uses so-called profile hidden Markov models (HMMs). A profile HMM turns a multiple alignment into a

probabilistic model over the edit operations (match, add, delete) for the alignment. In turn, if we want to check how likely a new sequence s matches a set F of sequences, we compute how likely s is according to F's profile HMM; e.g., a traffic sensor reading can be compared to the profile of an accident or a traffic jam to check whether it is an accident or a traffic jam or none of them. Again, we refer to [15] for the details.

Our intention here is to investigate the usefulness of biological sequence analysis method for traffic data. Specifically, we investigated the following questions:

- **(Q1)** Is it possible to solve Traffic problems with the help of out-of-the-box biological sequence analysis methods?
- **(Q2)** If so, how do they perform compared to state-of-the-art methods?
- **(Q3)** Can we gain interesting insights into traffic data with the help of biological sequence methods?

To answer **Q1**, we choose two tasks being investigated by geo-analytics community, namely *traffic event detection* and *trajectory clustering*. We apply out of the box biological sequence methods in combination with out alignment kernels , and compare our results with state-of-the-art approaches to answer **Q2**. In order to provide an answer to **Q3**, we show visualizations for the analysis performed using traffic logos.

4.2. Trajectory Clustering (for User Activity Analysis)

We followed two complementary approaches to analysing user routines at different abstraction levels. In the first approach, we extracted daily sequences of the user's stay points, clustered them using alignments, and analysed the resulting clusters using traffic logos. In the second one, we dig deeper and analysed the user's routines using 'map-matched' trajectories. This helps in grouping functionally relevant trajectories and in turn in identifying specific routes over the street network. Specifically, we used DBscan [16] using the the pair-wise alignment score and then visualized the resulting clusters. The second approach also helps us in comparing the performance to state-of-the-art methods (as we will show). Both approaches were applied to the same dataset of 112k recorded position within 363 trajectories.

4.2.1. Stay Points Discretization

A stay point is typically defined as a ball of radius r such that a trajectory stays inside for at least time t. Our goal, however, is to extract more frequent stay points

and prune less important ones. For this purpose, first we marked GPS positions from raw trajectories, which stayed within a radius $r = 100$ meters for time $t = 10$ minutes. Then we clustered these marked points with the help of DBscan [16] to find area which are more dense among these marked positions. In the end, we took the convex hull of each cluster to get the shape of a stay point. To check whether the user is inside a stay point, we took a threshold distance from the boundary (w.g. 30 meters) to deal with noise in GPS.

Our next step is stay point labelling. To do this, we first looked at the temporal distributions of the stay points in order to label the most important stay points, in our case 'home' and 'work'. Stay points where the user stayed during the daytime (9:00 Am–6:00 Pm) for ≥ 6 hours during weekdays were labelled as work and during night as home. The rest of the stay points are labelled with the help of Google maps (e.g., restaurants, post office, bank, shopping markets and Tennis courts). Finally, we pruned stay points that correspond to very short stays with less significance, e.g., gas stations. To extract traffic sequences out of the trajectories, we took every daily trajectory and extract the points where user stayed for significantly long time based upon the temporal difference between points. If these points were within a threshold (e.g., 30 meters) of a labelled stay point boundary, we added it as a suffix to our traffic sequence and continue till the end of the user's day. The extracted stay points along with their labellings are shown in Fig. 4.4; for the sake of keeping privacy, we are omitting the latitudes and longitudes. For the sake of visualization of activity sequences based upon these stay points, we cluster and align these sequences with the biological sequence analysis and visualize them using a biological routine called 'sequence logos'. The process is define below.

4.2.2. Visualization of Clustered Activity Sequences

After the extraction of stay points, we built the similarity matrix using the geodesic distance based alignment kernel. These stay points served as the symbols for activities in our traffic sequences. We calculate distance matrix between daily activity sequences from user with pairwise sequence alignment. Then, the sequences were clustered based using DBscan [16]. The sequences of each clustering were additionally aligned and we produced traffic logos for them shown in Fig. 4.5, Fig. 4.6 (a-d) and Fig. 4.7 (a-b). For the sake of visualization of time information, we labelled the time of stay points in every sequence with M-Morning (before 9:00 Am), D-Day (9:00 am to 6:00 pm) and E-Evening (after 6:00 pm). In the traffic logos we used different colours to indicate these time labels, namely green for morning, yellow for

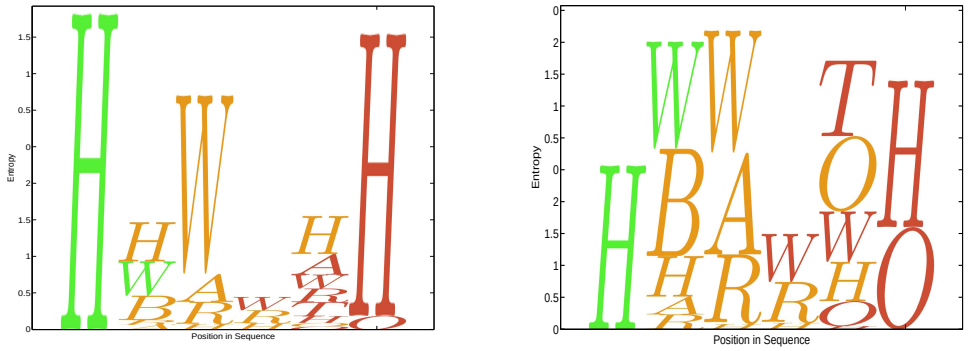
daytime, and red for evening. Furthermore, we have used two different formulas for the hight calculation of each individual symbol in the logos, i.e., (a) relative entropy of the symbol and (b)relative entropy divided by base frequency. (b) is also derived from biological sequence analysis where the importance of a symbol at a a particular position is visualized by changing the entropy calculation (diving the entropy by overall base ferquency). As a result, symbols which are rare in the whole sequence and predominantly appear at a certain position are more pronounced. The same logic applies to traffic sequences.

As one can see, traffic logos show a very dense and illustrative view of clusters for user's daily activities. Fig. 4.5 describes the logos for the whole data set: (a). Traffic logos demonstrate the mixed patterns of activity for the whole data set. For example, staying at home in the morning and evening is more certain than going to work (presumably, due to non work days at the weekend. Similarly occasional shopping in evening (red 'A' and 'R' symbols) is alternatively done with still less frequent Tennis in the evening (red 'T' symbol). Occasional swings in the normal routine, i.e., early and late work going routine (red and green 'W's) can also be picked up. Furthermore, a yellow 'B' followed by 'A' and 'R' hints at a weekend routine of going to ATM machines in banks and then shopping. These activity sequences will be further segmented to enhance the visibility of each pattern. (b). The height of a symbol is divided by its over all base frequency to know its relevance to a certain position in the sequence, i.e., if a position only occurs at a specific sequence most of the times, then its height will be increased. Consequently, this gives us an activity-versus-position binding which in the context of analysing preferred order of user routines can be very useful. For example, going to bank is mainly done as first activity on the weekend during day time, Tennis comes out only as an evening hobby and 'O' (friends and city center) is mainly carried out in the night. Furthermore, going to shopping is quite a routine in the evening and rare in daytime, therefore, its affect is accentuated in day time and nullified in evening. Notice that 'Traffic Logos' a give a compact representation of user activity patterns after *approx* 99% compression, i.e., almost all of the semantic information present in the detailed activity analysis (see Fig. 4.10) of the raw data can be described through logos.

Fig. 4.6 describes the cluster wise logos after alignement of sequences in each cluster, e.g., (a) Cluster 1 is the most frequent cluster in the data comprising around 40% of user's routine days. This cluster is composed of one accentuated pattern of Home (Morning)—Work (Day Time)—Home (Evening) with a occasional deviations from the routine like Early or Late Office going and leaving routine. A small 'O' at

(a) Street Network Map (b) Labelled Stay Points

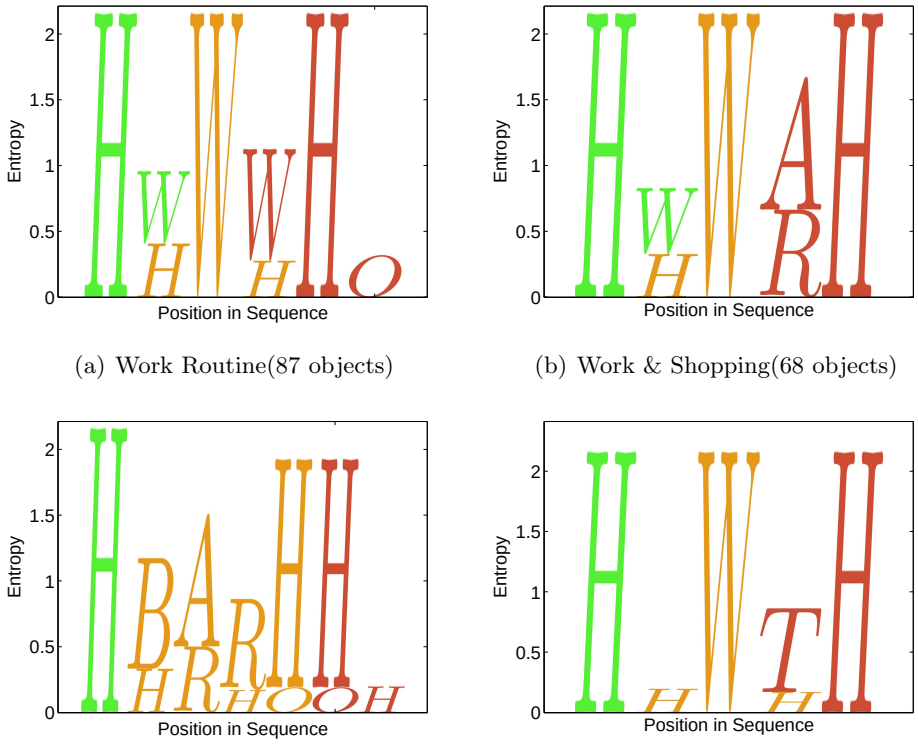
Figure 4.4.: Convex hulls of labelled stay points (blue polygons over gray-edged street network). The main stay points are Home, Office, Bank, Tennis, Shopping-1 (ALDI), Shopping-2 (REAL) and City Center.



(a) Sequence logos for the whole data set (b) Sequence logos for the whole data set

Figure 4.5.: Traffic logos for stay-point based sequences. x-axis denotes sequence positions and y-axis denotes 'information' present in each column. The symbols in the figure denote labels of activities based on stay points i.e: H denotes staying at Home; W–Working; A–shopping at ALDI; R–shopping at Real; B–getting cash from Bank, T–playing Tennis and O denotes Other activities for leisure (i.e., city center roaming and visiting friends). Colours of symbols denote the time of day, i.e., green denotes Morning (before 9.am); yellow–daytime (9am-6pm) and red — evening (after 6pm). The height of a symbol denotes the certainty of an activity at the given day time and position in the data.

the end presents occasional tendency to go 'Out' in the night. It becomes readily clear that this cluster encodes the daily routine of staying at Home in the morning with a higher certainty and then going to office early or staying at home with a very



(a) Work Routine(87 objects)

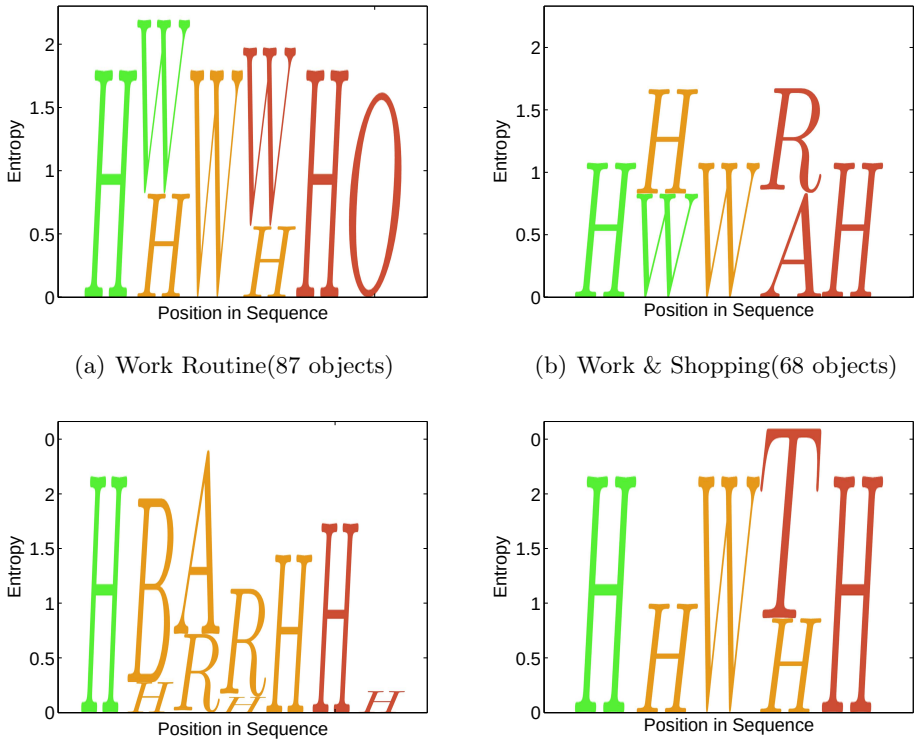
(b) Work & Shopping(68 objects)

(c) Weekend Routine(34 objects)

(d) Work & Sports(26 objects)

Figure 4.6.: Traffic Logos for the after segmentation of daily activity sequences. Every segment (or cluster) describes one of the possible routines that user follows in her daily life. The symbols in the figure denote labels of activities based on stay points i.e: H denotes staying at Home; W–Working; A–shopping at ALDI; R–shopping at Real; B–getting cash from Bank, T–playing Tennis and O denotes Other activities for leisure (i.e., city center roaming and visiting friends). Colours of symbols denote the time of day, i.e., green denotes Morning (before 9.am); yellow–daytime (9am-6pm) and blue-evening (after 6pm).

small possibility. In the daytime, the user goes to work with a very higher certainty and comes back around 6pm with a small possibility of staying at work. The small 'O' at the end of the logo describes a small possibility of going for other leisure activities (city center roaming or visiting friends). (b). Cluster 2 comprises around 25% of user's routine days. This cluster is composed of one accentuated pattern of Home (Morning)—Work (Day Time)—Shopping (Evening)—Home (Evening). This is a quite similar daily routine to (a). There is, however, an important difference. The user shops at either of two shopping centres (ALDI, and REAL) after work. (c)



(a) Work Routine(87 objects)

(b) Work & Shopping(68 objects)

(c) Weekend Routine(34 objects)

(d) Work & Sports(26 objects)

Figure 4.7.: Enhanced traffic logos for visualizing the importance of a user activity w.r.t. a particular position in her activity routines. X-axis denotes sequence positions and Y-axis denotes 'information' present in each column. Every cluster describes one of the possible routines that user follows in her daily life. The symbols in the figure denote labels of activities based on stay points i.e: H denotes staying at Home; W–Working; A–shopping at ALDI; R–shopping at Real; B–getting cash from Bank, T–playing Tennis and O denotes Other activities for leisure (i.e., city center roaming and visiting friends). Colours of symbols denote the time of day, i.e., green denotes Morning (before 9.am); yellow–daytime (9am-6pm) and blue-evening (after 6pm).

describes a cluster which is possibly weekend-routine since there is not high W(ork) symbol at all. So on the weekend, the user stays at home in the morning and then gets cash from bank with a small probability. Afterwards the user shops from ALDI-then-Real or only REAL during the day time. Then she comes back home and stays. However, with a small possibility, instead of coming back to home after shopping, she chooses to do describe an occasional tendency (yellow and red) 'O's to do *Other*

leisure activity like city center roaming or a visiting a friend at weekends (later in the day or evening). (d). A small percentage of user days are composed of tennis playing hobby along with regular routine, i.e., Home (Morning) — Work (Day)— Tennis(evening) — Home (evening)

Fig. 4.7 describes the enhanced traffic logos for visualizing the importance of a user activity w.r.t. a particular position in her activity routines. Roughly speaking, this figure gives an activity-versus-position binding which in the context of analysing user routines can be very useful. For example, the rare position specific routines of going to bank early in the day on weekends, visiting friends in the evening and playing tennis in the evening after some work days is pronounced in comparison to Fig.4.6 and gives an idea about the preferences and order of specific activities in the data.This is an affirmative answer to questions **(Q1)** and **Q3)**.

4.2.3. Map Matching Discretization

To investigate whether our methods can perform comparably with state-of-the-art methods, we focused on clustering trajectories [51]. Whereas the state-of-the-art method clusters raw trajectories, we used the sequences of map matched street segments to cluster our data set. This helps us in analysing specific map routes that user selects during her travels. For map matching we followed [30]. After clustering on the map matched level, we projected the labelled trajectories back into Euclidean space and visualized them over the street network in Fig. 4.10. We show the different clusters in descending order according of their sizes, namely $\{119, 78, 28, 11, 7, 7, 6, 6\}$. The clusters were labelled with already extracted stay points, cf. Fig. 4.4, for user activity. As one can see, the largest cluster is the daily travel from 'Home to Work'. The runner-up largest cluster is a direct route back home from office. However, as the user may also take alternative routes back home from office, for instance through shopping center ALDI (clusters 3) or REAL (cluster 4), the back home cluster 2 is not as large as the home to work cluster. Cluster 5-8 indicate travelling to Work-to-Tennis, casual shopping from ALDI, weekend shopping results, and city center roaming. So again, we find meaningful clusters. However, are they also as good as state-of-the-art?

4.2.4. Experimental Evaluation

To see this, i.e., for a quantitative comparison, we computed the Hausdorff distance for both clustering solutions as well as the error. The error term RMS_{STD} is a

measure of clustering compactness [40] which gives *sum of average within cluster variances*. The distance measure used, Hausdorff distance or $H(A, B)$, measures the degree of mismatch between two trajectories A and B. More formally, let $h(A, B)$ define the maximum Euclidean distance of a point in trajectory A to the nearest point in trajectory B i.e., $h(A, B) = \max_{a \in A} \min_{b \in B} d(a, b)$. Now, we define $h(B, A)$ in a similar fashion and in turn *Hausdorff distance* b/w A and B:

$$H(A, B) = \max(h(A, B), h(B, A)) \tag{4.5}$$

Intuitively, if the Hausdorff distance is d, then every point of A must be within a distance d of some point of B and vice versa.
For clustering, our alignment-based method used again *DBscan [16]*. As state-of-the-art baseline, we used *OPTICS [3]* route similarity search, see [51]. The data set used is explained in detail in [1, 2]. As both clustering methods are density based and filter outliers, this comparison is fair. Using K-means as baseline for example is not a fair option as it does not filter outliers. In other words, it will produce a-priori much higher error. The parameters of both density based algorithms are ϵ (minimum similarity threshold to consider two points as neighbours) and min_n (minimum number of neighbours needed to define a point as a 'core point'). We used a grid search to determine the best parameters for the same number of clusters as found by the baseline, namely 7.

Our method produced (using a grid search on $\epsilon \in [0.5 : 0.01 : 1.0]$ and $min_n = [2 : 1 : 6]$) an error of $RMS_{STD} = 197$ with parameter settings $\epsilon = 0.65$ and $min_n = 5$ as compared to $RMS_{STD} = 271$ with $\epsilon = 1KM$ and $min_n = 5$ for the baseline. However, the number of trajectories clustered by our method was 232 whereas the baseline clustered 261 trajectories. The remaining ones of the in total 363 trajectories were filtered out. This, however, is only giving a 'point estimate' of our performance. To see the general picture, i.e., performance over the range of grid search, we provide the performance averaged per number of clusters As one can see, *pairwise traffic sequence alignment* is able to capture similar number of objects with a better mean of error than the baseline by filtering out the noise in a better way. We believe that this happens because of high gap costs in the alignment computation. They penalize to group together trajectories with dissimilar sub-parts. Consequently, more compact clusters are found. Moreover, it finds a similar number of clusters as the baseline, namely $7 - 8$. Fig.4.10 shows the 8-distinct patterns found by our algorithm. Here, the clusters in Figs. 4.10 (a-g) were also

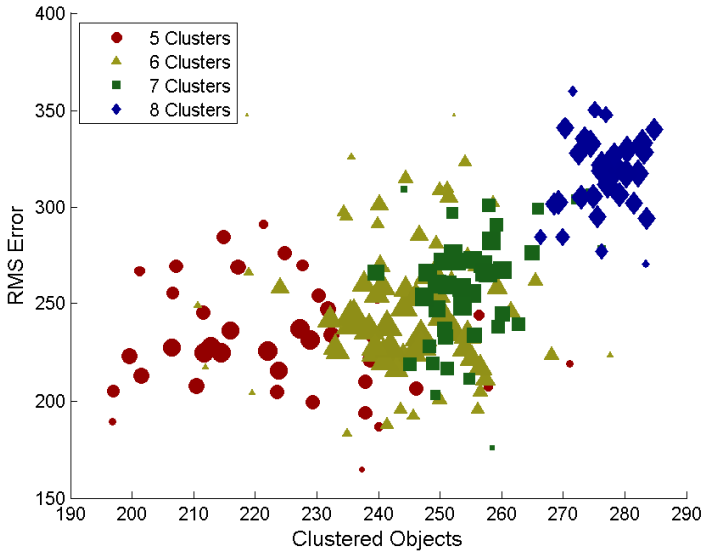


Figure 4.8.: Scatter plot of clustering experiments for number of clustered objects (higher value is desirable) versus RMS error (higher value not desirable). Every marker describes the outcome of one experiment and different markers (in type and color) are used for different 'number of clusters' found (see legend). Moreover, the size of a marker describes the density of corresponding cluster numbers in the vicinity. This scatter plot helps us in identifying the right number of clusters present in the data set. As evident from the picture, 5 clusters has a high variance in both dimensions, therefore, it is insufficient to capture the diversity in data. Consequently, more appropriate values for the clustering are 6 and 7 where 7 is even better with more clustered objects and less variance in error than 6. However, one can also try to come up with more more patterns in data at the cost of a higher error (the case of 8 clusters)

found by the baseline. The additional cluster shown in Fig.4.10 (h) is Home to City Center. Moreover, we contacted the owners of the dataset and they agreed with the possibility of clusters found.

This is clearly an affirmative answer to question **(Q1)- (Q3)**.

4.3. Traffic event Detection

To further investigate the performance of the biology view on traffic analysis, we considered a classification setting, namely to detect traffic events from sensor data as already described in the introduction. Specifically, we were interested in (a) event

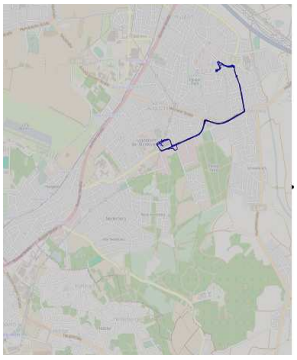
(a) Home to Office

(b) Office to home

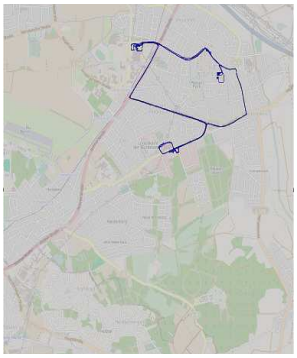
(c) Office to home Via ALDI

(d) Office to home Via REAL

(e) Office to Tennis

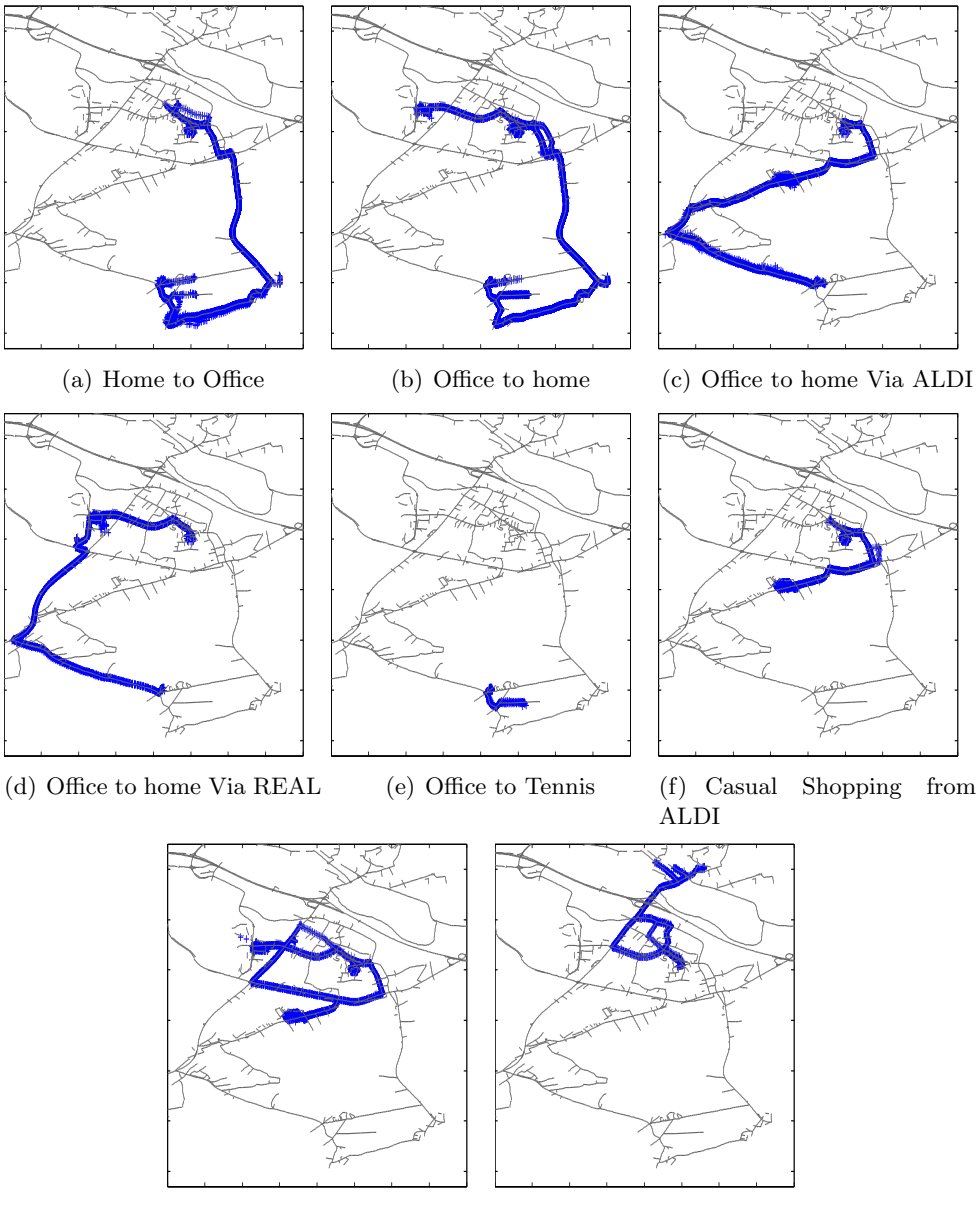


(f) Casual Shopping from ALDI



(g) Weekend Shopping Routine

Figure 4.9.: Clusters of raw trajectories (sets of blue points) from baseline solution [51] over shaded street network. The similarity criteria used is route similarity search along with clustering algorithm known as OPTICS [3] (a density based clustering algorithm). As depicted, base line solution outputs 7 clusters where each cluster describes a specific route in user's routine including office travel, shopping routine, sports and weekend routine. The semantics of the clustering are extracted with the help of labelled stay points in the stay point discretization process. Finally, intra-cluster distance is low as all trajectories in a cluster are compact and follow same route.



(a) Home to Office (b) Office to home (c) Office to home Via ALDI

(d) Office to home Via REAL (e) Office to Tennis (f) Casual Shopping from ALDI

(g) Weekend Shopping Routine (h) Home to City Centre

Figure 4.10.: Clusters of raw trajectories (blue points) based on map-matched street segments from street nework (gray edges). The trajectories are projected back into the original space. Furthermore, the similarity criteria used is pairwise sequence alignment along with clustering algorithm known as DBscan [16] (a density based clustering algorithm) (with $\epsilon = 0.58$ and $min_n = 3$). The clustering process captures 262 objects —described in order of cluster labels from 'a' to 'h', i.e., $\{119, 78, 28, 11, 7, 7, 6, 6\}$. As depicted, these clusters indicate office travel, shopping routine, sports, weekend routine and city center roaming. Finally, (following the base line solution) the intra-cluster distance is low as all trajectories in a cluster are compact and follow same route.

Similarity method	Clusters	μ_{RMS}	σ_{RMS}	$\#objs(\mu \pm \sigma)$	$\epsilon(\mu \pm \sigma)$	$min_n(\mu \pm \sigma)$
Route Search	7	271.5	NA	261	NA	NA
Pairwise	7	249.24	36.4	255.6 ± 7	0.63 ± 0.01	3 ± 2
Traffic	5	211.78	112.43	221.2 ± 21	0.64 ± 0.07	4 ± 1
Sequence	6	236.58	75.74	243.6 ± 14	0.60 ± 0.04	4 ± 1
Alignment	≥ 8	313.98	20.13	275.1 ± 6	0.53 ± 0.04	3 ± 1

Table 4.2.: Comparison of clustering results using the Hausdorff distance to compute errors. The error is depicted in columns 3, and 4 as the the average and standard deviation of root means squared Hausdorff distance, i.e, μ_{RMS}, and σ_{RMS} for the corresponding rows. Similairly, the mean and standard deviations for number of objects found per row are given by the fifth column. Last two columns show the means and deviations for parameters of the density based algorithms, i.e., ϵ, and min_n. The first row shows the result for the state-of-the-art baseline. As we can see, both capture a similar number of trajectories and similarly good clusters, see also Fig. 4.10 (a-g). Row 3,4 have a lower error but also capture a smaller number of trajectories because of small ϵ and large min_n. Row 5 shows that the alignment-based method can capture more patterns than the original method, cf. Fig. 4.10(h), at the cost of a higher error with large ϵ and small min_n.

persistence and (b) separation of normal traffic from event data. Since events are very rare and usually composed of less than 1% of the dataset, the main step to achieve (b) is subtracting the mean. However, after this normalization choosing a threshold value for classification will not work well since it does not consider **a**. Consequently, it would produce a lot of false positives (i.e., noise or unusually high readings due to some temporary phenomena). Fig 4.11 (a—d) illustrates this point with the help of sensor readings acquired over a publicly available event based data set. The plot of all readings is composed of two different trends in auditorium entrance, i.e., set of blue (non-event) points close to the axis and a normal-like curve for blue and red points. On further inspection, it is revealed that these two trends relate to entrances on work and non-work days. Consequently, we have built two different models for the two different trends (b)Readings for non-event workdays, i.e., normal movement data with low readings for night/morning times and normal like curve for noon. Unusual spikes are mixture of noise and unreported events. (c) Readings for non-event weekends, i.e., very low movement in and out out of the auditorium. (d) Normalized date after subtracting the corresponding means from weekends and work-days' readings. This helps us in separating out the data in two classes as most 'event days' have high value after normalization. However, a

threshold over the normalized values cannot be used as a criteria for classification because it will generate a lot of false positives (due to mixing of blue dots). In order to improve the accuracy of classification, we need to build a probabilistic model that captures the idea of 'event persistence', i.e., structurally correlated high values of readings.

To address **a**, we propose to learn a profile HMM from the event data. This is a sensible idea for two reasons. **(1)** profile HMMs capture event dynamics probabilistically and in turn can be used for soft comparison between event and non-event sensor readings. **(2)** Most of the events are different in length, e.g., congestion at the end of a concert can last 30-45 minutes. Comparing variable length sequences is one of the strength of profile HMM. As our experimental results show, they actually perform better (in terms of false positive rates) than state-of-the-art approaches.

4.3.1. Experimental Evaluation

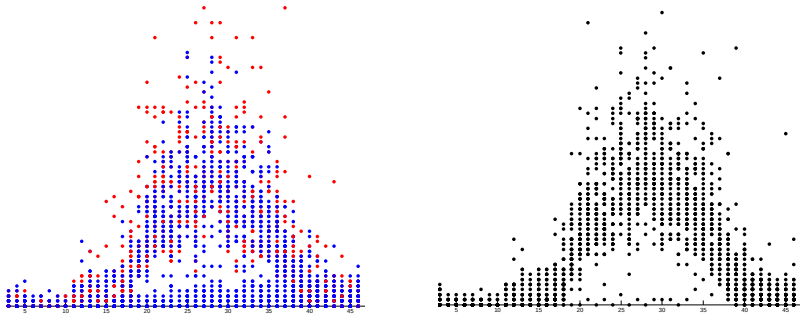
We considered two real world data sets[2] also used by [29]. The data sets' description (adopted from [29]) is as following :

The first data set is referred to as Caltech auditorium entrance data. It comprises 3 months of count data for entrances at the front door of the Cal-IT2 institute. The data is accumulated on a 30 minute basis by optical detectors that register the number of entrances through the main doors. The goal here is to predict the presence of a conference on a particular day in the building.

The second data set is referred as the dodgers baseball game data. This loop sensor data was collected for a free-way in Los Angeles. It is close enough to the stadium to see unusual traffic after a Dodgers game, but not so close that the signal for the extra traffic is overly obvious. The observations were taken over 25 weeks with 288 time slices per day in 5 minute counts. Here, the goal is to predict the presence of a baseball game at Dodgers stadium on a particular time.

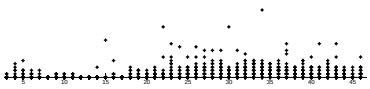
For comparison, we selected two baseline approaches. In baseline 1, we take a small portion of event data as training set, i.e., max (20%,10 events). Then we mixed it with the same proportions from non-event data. Now, we divided the training data into two groups, i.e., *weekend* and *weekdays*. We do so because both of these groups have different trends of traffic. We normalized both groups of training data by subtracting their corresponding means. Finally, with the help of cross validation, we chose a threshold value to classify events that captures all events (irrespective of the number of false positives captured). In baseline 2, we took the state-of-the art

[2] http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/

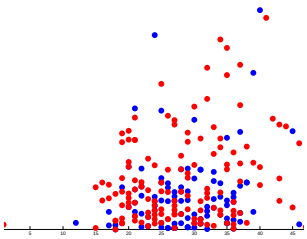


(a) Sensor readings

(b) Workdays



(c) Weekends



(d) Normalized

Figure 4.11.: Sensor readings from Caltech Auditorium Entrance. X-axis denotes time and y-axis denotes sensor readings. 'Red dots' denote readings for event days and 'blue/gray dots' — non-event days. (a)All readings. The plot of all readings is composed of two different trends in auditorium entrance, i.e., set of blue (non-event) points close to the axis and a normal-like curve for blue and red points. On further inspection, it is revealed that these two trends relate to entrances on work and non-work days. Consequently, we have built two different models for the two different trends (b)Readings for non-event workdays, i.e., normal movement data with low readings for night/morning times and normal like curve for noon. Unusual spikes are mixture of noise and unreported events. (c) Readings for non-event weekends, i.e., very low movement in and out out of the auditorium. (d) Normalized date after subtracting the corresponding means from weekends and work-days' readings.

event detection algorithm by Ihler *et al.* [29], which uses adaptive Poisson processes to identify events.

Our profile HMM approach was built upon the output of baseline 1. As there are missing observations, the sequences are of different length. Therefore, we aligned all of the event sequences together and set the length of the profile HMM (a parameter

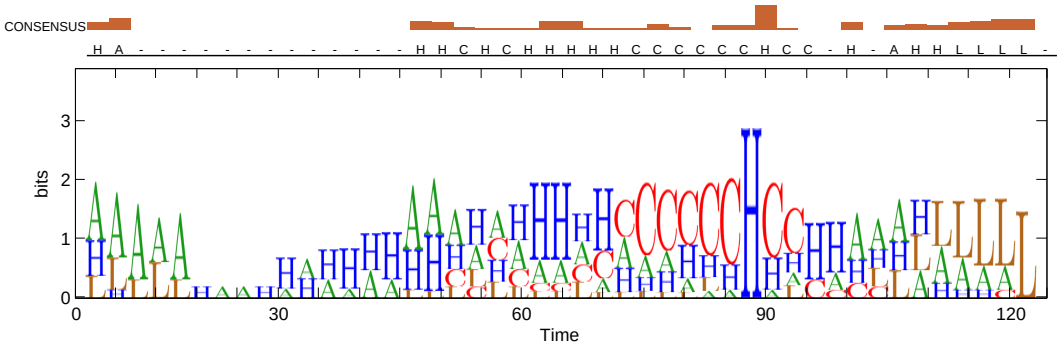


Figure 4.12.: Traffic Logos for Baseballs games in *training set* for '+-1 hour' of game endings. Symbols denote frequency of traffic, i.e., L-Low, A-Avg, H-High and C-Congestion. (Top)5-conserved regions. (From left to right) region 1 describes 'low to average' traffic during play. Regions 2–4 collectively describe traffic frequency for game endings. These regions show high density traffic with real congestion starting approx. 15 minutes after game endings. This trend declines approx. 45 minutes after the games where conserved region region no.5 starts showing a tendency towards normal traffic, i.e., average and low density.

Algorithm	Training Events	True Positives	False Positives
Dodger's Base ball Game Prediction - Total events= 76			
Baseline	15	76	65
Poisson Proc.	76	75	23
HMM profile	15	74	18
Caltech Auditorium event prediction - Total events= 29			
Baseline	10	29	43
Poisson Proc.	29	24	24
HMM profile	10	25	12

Table 4.3.: Comparison of event prediction on real world data sets. In both cases, HMM profiles were able to predict almost the same number of events with a better recall (lower number of false positives) and a low training percentage of data. On average, 90 percent of original events are captured by all algorithms. However, out-of-the-box profile HMMs provide a better recall (filtering of false alarms) by capturing event persistence.

of it) according to the average length of events in the training data. At classification time, we got a complete sequence of sensor readings. To compare it with the much smaller event profile, we used a sliding window equal in length of the event profiles, i.e., profile HMM. Windows with a higher score than a threshold value ρ (chosen through cross validation) were marked as event traffic. Tab. 4.3 shows the results.

Note that all algorithms capture a high percentage of true positives. However, one can clearly see that just by using out-of-the-box biological techniques, we can get comparable (in one case even significantly lower) false positive rate than the state-of-the-art technique. Having a better recall (low number of false positives) is important in cases when there is a cost attached to a false positive. Consider, e.g., sending a traffic inspector to control a traffic jam when there is no jam. The traffic logos shown in Fig. 4.12 validate that the profile HMMs capture plausible event patterns.

Taking all of our event detection results together, they clearly provide an affirmative answer to questions **(Q1)-(Q3)**.

Running Time

We have performed our experiments on Intel Core(TM)2 Duo E6850 processor with 3.3GHz frequency and 3.24 GB RAM. *Approx* 2 GB memory at maximum is available for the process. The operating system is windows XP and language interpreter is JAVA. For sequences of 'street segments travelled', the similarity matrix between '363 trajectories with 16616 street segments' takes around 8 hours to compute and rest of the analysis takes less than a minute; however, since shortest path graph computations and map matching processes are involved along with dynamic programming, therefore, we expect it to be a time-consuming process. In contrast, for stay point based activity sequences, the time for analysis, i.e., stay point detection, multiple alignment, clustering and traffic logos is very fast as it takes less than 10 minutes to finish. For traffic event detection, the whole process takes *approx* 10 minute to finish for 51840 sequences, i.e., 288 sequences/day (of length 40) for 6 months.

Summary

In this chapter, we have demonstrated that structural information can be used to improve two fundamental tasks related to the analysis of symbolic trajectories, i.e., *trajectory clustering* and *traffic event detection*. In a nutshell, the process is described as following: we first take the sampled raw trajectories and discretize them using the standard discretization measures, e.g., map matching and stay point detection. The discretization process converts these trajectories into sequences which can be analysed through the tool box of biological sequence analysis methods. The key distinction, here, however, is that we use the geodesic distance based alignment ker-

nel (outlined in Chap.3) as an input to the biological sequence analysis tool box. In the end, we show that the biological sequence analysis in combination with geodesic distance based alignment kernel shows state-of-the-art performance for the above mentioned tasks.

5. Conclusions

"Nothing is more revealing than movement." Martha Graham.

The analysis of trajectories has a broad range of applications, for example, in urban planning, transportation systems, navigation and localized search methods. Most of these applications share some underlying basic tasks. One of such basic tasks is map matching —the process of aligning raw trajectories to street network — which is used in navigation, planning and transportation systems. Another task is the comparison of trajectories which is necessary for localized search methods, user similarity search, and —in general— finding similar patterns. Finally, probabilistic modelling of trajectories is required for classification/prediction related tasks. For example, the traffic event detection is such an application where we build a model of trajectories relating to traffic events, e.g., jams and congestions, so that a new event can be identified.

While these tasks have different settings, a common and yet difficult problem in solving them is coping up with the measurement error in the sampled trajectories. This error occurs due to multiple issues , e.g., spatial and/or temporal distortion, low and non-uniform sampling rate, heterogeneity of data (noisy label information) and missing values. In order to cope with these problems, we look at the invariants in the data, i.e., the spatio/temporal structures in trajectories that remain considerably unharmed during the measurement process. In particular, we take a trajectory as a walk over the underlying labelled graph, and embed the information in these graphs into the Euclidean space in order to preserve the structural relationships between sampled points. These distances then prove our connection to the fields of kernel methods and biological sequence analysis which are used to improve the trajectory analysis tasks.

One of the above mentioned connections,i.e., kernel methods is manifested in Map Matching. Roughly speaking, the existing approaches for map matching can be categorized into four groups: geometric, topological,probabilistic, and other advanced techniques. Surprisingly, the kernel methods such as support vector machine and Gaussian processes [50] had not received attention yet although they are very popu-

lar in the machine learning community due to their solid mathematical foundation, tendency toward easy geometric interpretation, and strong empirical performance in a wide variety of domains. In a nutshell, kernel methods first process a dataset into a kernel matrix that roughly expresses the idea of two data points are"equivalent as far as some function f of the data can tell". By representing the data in terms of a kernel matrix, the data can be of various types, and also heterogeneous types such as trees and graphs. This makes kernel approaches very flexible. In a second step, a variety of kernel algorithms that have been developed can be used to analyse the data, using only the information contained in the kernel matrix. Kernels are also readily extendible, therefore, every kernel matrix provides an opportunity to integrate its knowledge with the existing kernels in the field. An investigation of using kernel methods for map matching motivated by their well-known strength was the seed that grew into our proposal for kernelized map matching.

In this thesis, we have shown that kernelized learning schemes can be used to reduce the problems associated with noise and sampling rate in map matching. To illustrate, we have proposed a simple objective function based on minimizing the difference between kernels in the output and input space. Our algorithm KMM has very promising results. The significant contributions of this work can be listed as the addition of kernel methods to the tools of Map Matching and the learning of road networks from trajectories instead of matching it. We also investigate the possible kernels in Map-Matching scenario, and come up with Geodesic distance kernel which successfully encodes the priors of problem.

Furthermore, we have strengthened and revisited the link between 'biological sequence analysis' and 'analysis of trajectories'. Up to now, both fields had met, however, there was no concrete method to incorporate the invariances of traffic domain into biological domain. The main step to achieve this goal consisted of using Geodesic distance based embedding in combination with discretization methods to map the continuous in time and space traffic data in symbols over time. We have exploited our methods to show that the state-of-the-art performance can be achieved using off-the-shelf biological sequence analysis tools. Specifically, we demonstrated that sequence alignment can be used for activity analysis, and the profile hidden Markov models are well suited for capturing event persistence during event detection. In particular, the link allowed us to introduce a novel visualization scheme for traffic data, called Traffic Logos. They provide a condensed, yet illustrative picture of the patterns in traffic sequence data.

5.1. Lessons Learned

The analysis of trajectories (spatio-temporal data) and its related areas, for example, mobility mining are application intensive, and, researchers/practitioners in these area invariably grapple with the issues of following nature:

1. the large scale structured data sets,

2. the spatio-temporal nature of the data set and the interplay of spatial and temporal dimensions,

3. the complexity of the underlying graphs upon which trajectories live,

4. preserving the privacy of users, and, the inability to get user datasets due to privacy issues

In this thesis, we have made some advances towards (2), and (3). on the other hand, recently machine learning has made fascinating advances to solve (1), i.e., the scaling problem in structured datasets through, e.g., hashing [56], matrix factorization [62], stochastic gradient descent, and lifted information retrieval [36]. Therefore, it is very important for both fields to communicate and learn from each other's experience. Furthermore, researchers in machine learning shall also remain attentive to the rest of the problems mentioned and make tangible advances in these directions.

Another problem that we faced during the research was the nature of two different fields. To illustrate, Geo-analytics research is application intensive, and, one Geo-analytics research problem is usually composed of solving a real life situation, e.g., 'tourist route finding' with different settings, i.e., time, cost, and path constraints. On the other hand, machine learning research mainly focuses on the methods and techniques to solve one fixed problem setting that can be used across a lot of applications, e.g., structured prediction, clustering, embedding, policy learning and graph matching with quadratic constraints. In the above mentioned application, changing from path constraints (a problem of graphical nature) to cost and time constraints will change the settings of the problem and one has to apply a different machine learning algorithm. Therefore, a machine learning researcher trying to analyse the trajectories has to be careful that his approach covers all of the different situations that can arise in the given scenario.

5.2. Future Work

In this thesis, we have tried to connect machine learning and Geo-analytics by providing basic distances that are intuitive to understand and can be used as a black box in a lot of applications. However, we advise the reader to look *in depth* in terms of the problem constraints for the Geo-analytics application being considered and how to combine these distances for a given model to solve Geo-analytics problem. As, in our case, we also needed to come up with specific regularizations and optimization techniques along with kernels to get state of the art performance.

In the following text, we will outline some of the new directions and future work in three of the major areas that we have explored for connecting machine learning and Geo-analytics.

In Map Matching

The link established between map matching and kernels provides many interesting avenues for future work; we have only scratched the surface. Indeed, one should study alternative kernels and map features, cost functions and hyper-parameter optimization criteria. For instance, so-called Fisher kernels are kernels derived from hidden Markov models. In turn, we may utilize any HMM based map matching approach. Testing KMM within a real-world system tracking system is another interesting avenue. Proving the hardness of map matching problems along with guarantees on approximation are interesting venues of future work. Finally, KMM directly generalize to the case of 3D trajectories.

In Clustering and Event Detection

The link between sequence analysis and traffic provides many attractive avenues for future works . First of all, one should investigate the usage of label sequence graph kernels and random walk kernels [18] for traffic data. Another direction could be exploring the benefits of out-of-the-box biological sequence techniques for other applications in the area of trajectory analysis. We are currently working on generating more complex profiles and diaries of user's activities. One should also start comparing such profiles against each other to get user similarity.

In Structural Information

Geo-analytics represent many opportunities to learn from recent advances in the analysis of structured data in machine learning [9]. Surprisingly, the usage of structural information, i.e., Geodesic distances, structural embeddings, graph distances and assignment problems [45] for graph matching has not been considered in Geo-analytics before. One of the reasons for this could be the implicit complexity of graphs in Geo-analytics, e.g., Euclidean graphs. On the other hand, machine learning has gained a great momentum to solve the structured data problems on a large scale. So far we have only explored classical machine learning techniques like HMMs, kernels, embeddings and sequence analysis. And, a more in depth analysis of recent machine learning progress, e.g., structured SVMs [5], graph kernels [20], stochastic gradient based approaches [43], spectral and latent learning [37], and lifted information retrieval [42] is needed to benefit Geo-analytics from machine learning in a comprehensive manner. Another of the very interesting problems in Geo-analytics is that of recommendation and Geo-Community extraction. As more and more evidence pours out that social contacts depend upon geographic proximity and travel routines [23], an inverse hypothesis is also shaping up, i.e., users having similar travel routines and location proximity are more prone to be future friends [66]. Thus, an investigation of structural information present in the tripartite graphs of users, locations and travel sequences is an interesting line of future work. In general, there should be more traffic through the connections made, and Geo-analytics researchers shall remain aware of advances in machine learning made for solving structured data analysis.

Bibliography

[1] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti. Interactive visual clustering of large collections of trajectories. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 3 –10, 2009.

[2] G. Andrienko, N. Andrienko, and S. Wrobel. Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations Newsletter (Special Issue on Visual Analytics)*, 9(2):38–46, 2007.

[3] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of ACM International Conference on Management of Data (SIGMOD)*, pages 49–60, 1999.

[4] X. Bai and L. J. Latecki. Path similarity skeleton graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(7):1282–1292, 2008.

[5] G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.

[6] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, pages 853–864, 2005.

[7] T. Brinkhoff. Generating network-based moving objects. In *Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 253–255, 2000.

[8] K. Buchin, M. Buchin, M. van Kreveld, and J. Luo. Finding long and similar parts of trajectories. In *Proceedings of ACM International Conference on Advances in Geographic Information Systems (ACM GIS)*, pages 296–305, 2009.

[9] R. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(6):1048–1058, 2009.

[10] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling.* Chapman and Hall/CRC, 2nd edition, 2000.

[11] M. Cuturi, J. P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 413–416, april 2007.

[12] G. de Vries and M. van Someren. Clustering vessel trajectories with alignment kernels under trajectory compression. In *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 296–311, 2010.

[13] F. Demirci, A. Shokoufandeh, and S. J. Dickinson. Skeletal shape abstraction from examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(5):944–952, 2009.

[14] R. O. Duda, D. G. Stork, and P. E. Hart. *Pattern classification.* Wiley, New York; Chichester, 2nd edition, 2000.

[15] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis.* Cambridge University Press, 1998.

[16] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 226–231, 1996.

[17] T. Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[18] T. Gärtner. *Kernels for structured data.* World Scientific, Hackensack, N.J., 2008.

[19] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of Conference on Learning Theory (COLT)*, pages 129–143, 2003.

[20] T. Gärtner, T. Horváth, Q. V. Le, A. J. Smola, and S. Wrobel. Kernel methods for graphs. In *Mining Graph Data*, pages 253–282. John Wiley and Sons, Inc, 2006.

[21] F. Giannotti, M. Nanni, D. Pedreschi, and F. Pinelli. Mining sequences with temporal annotations. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, pages 593–597, 2006.

[22] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *Proceedings of ACM International Conference on Knowledge discovery and data mining (SIGKDD)*, pages 330–339, 2007.

[23] M. C. Gonzalez, C. A. Hidalgo, and A. L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[24] Y. Guo, J. Gao, and P. W. Kwan. Twin kernel embedding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(8):1490–1495, 2008.

[25] D. Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.

[26] L. He, C. Han, and W. Wee. Object recognition and recovery by skeleton graph matching. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, pages 993–996, 2006.

[27] T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 158–167, 2004.

[28] D. W. Hubbard. *How to Measure Anything: Finding the Value of Intangibles in Business*. Wiley, 2 edition, 2010.

[29] A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying poisson processes. In *Proceedings of ACM International Conference on Knowledge discovery and data mining (SIGKDD)*, pages 207–216, 2006.

[30] A. Jawad and K. Kersting. Kernelized map matching. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*, pages 454–457, 2010.

[31] A. Jawad and K. Kersting. Kernelized map matching for noisy trajectories. In *Working Notes of Knowledge Discovery and Machine Learning (KDML) at LWA2010 - Learning, Knowledge & Adaptation*, pages 1–10, 2010.

[32] A. Jawad, K. Kersting, and N. Andrienko. Biological sequence analysis meets mobility mining. In *Working Notes of Knowledge Discovery and Machine Learning (KDML) at LWA2011 - Learning, Knowledge & Adaptation*, pages 73–80, 2011.

[33] A. Jawad, K. Kersting, and N. Andrienko. Building bridges between traffic and biological sequence analysis. In *International Workshop on Finding Patterns of Human Behaviors in NEtwork and MObility Data – NEMO at ECML PKDD (Europen Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases)*, pages 13–27. 2011.

[34] A. Jawad, K. Kersting, and N. Andrienko. Where traffic meets dna: Mobility mining using biological sequence analysis revisited. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*, pages 357–360, 2011. **Best Poster Award**.

[35] C. H. Joh, T. A. Arentze, and H. J. Timmermans. Multidimensional sequence alignment methods for activity-travel pattern analysis: A comparison of dynamic programming and genetic algorithms. *Geographical Analysis*, 33(3):247–270, 2001.

[36] K. Kersting, Y. E. Massaoudi, B. Ahmadi, and F. Hadiji. Informed lifting for message–passing. In *Proceedings of the 10th Conference on Artificial Intelligence (AAAI)*, pages 232–237, 2010.

[37] N. D. Lawrence. Spectral dimensionality reduction via maximum entropy. *Journal of Machine Learning Research (JMLR)*, 15:51–59, 2011.

[38] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, New York; London, 2007.

[39] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *International Journal of Robotics Research (IJRR)*, 26(1):119–134, 2007.

[40] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. Understanding of internal clustering validation measures. In *IEEE International Conference on Data Mining (ICDM)*, pages 911–916, 2010.

[41] T. Mitchell. Mining our reality. *Science*, 326(5960):1644–1645, 2009.

[42] M. Mladenov, B. Ahmadi, and K. Kersting. Lifted linear programming. In *15th International Conference on Artificial Intelligence and Statistics (AISTATS 2012)*, 2012. Volume 22 of JMLR: W&CP 22.

[43] S. Natarajan, T. Khot, K. Kersting, B. Gutmann, and J. Shavlik. Gradient–based boosting for statistical relational learning: The relational dependency network case. *Machine Learning Journal*, 86(1):25—56, 2012.

[44] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of ACM International Conference on Advances in Geographic Information Systems (ACM GIS)*, pages 336–343, 2009.

[45] P. M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. In *Proceedings of DIMACS Workshop on Quadratic Assignment Problems*, pages 1–42, 1994.

[46] S. Pemmaraju and S. Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press, 2003.

[47] N. Quadrianto, A. J. Smola, L. Song, and T. Tuytelaars. Kernelized sorting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(10):1809–1821, 2010.

[48] N. Quadrianto, L. Song, and A. J. Smola. Kernelized sorting. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1289–1296, 2009.

[49] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007.

[50] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning*. MIT Press, 2005.

[51] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko. Visually driven analysis of movement data by progressive clustering. *Information Visualization*, 9(3-4):225–239, 2008.

[52] B. Schlkopf, K. Tsuda, and J. P. Vert. *Kernel methods in computational biology*. MIT Press, MIT Massacheusets, Cambridge, 2004.

[53] A. Schmidt, M. Langheinrich, and K. Kersting. Perception beyond the here and now. *IEEE Computer*, 44(2):86–88, 2011.

[54] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[55] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

[56] Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, and S. Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research (JMLR)*, 10:2615–2637, Dec. 2009.

[57] H. Sundar, D. Silver, N. Gagvani, and S. J. Dickinson. Skeleton based shape matching and retrieval. In *Proceedings of Shape Modeling International (SMI)*, pages 130–139, 2003.

[58] E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research (JMLR)*, 4:773–818, 2003.

[59] J. Tangelder and R. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471, 2008.

[60] J. S. Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[61] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 5500(390):2319–2323, 2000.

[62] C. Thurau, K. Kersting, M. Wahabzada, and C. Bauckhage. Descriptive matrix factorization for sustainability adopting the principle of opposites. *Data Mining and Knowledge Discovery*, 24(2):325–354, 2012.

[63] M. R. Vieira, P. Bakalov, and V. J. Tsotras. Querying trajectories using flexible patterns. In *Proceedings of International Conference on Extending Database Technology (EDBT)*, pages 406–417, 2010.

[64] S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 113–130, 2003.

[65] S. V. N. Vishwanathan, K. M. Borgwardt, I. R. Kondor, and N. N. Schraudolph. Graph kernels. *Journal of Machine Learning Research (JMLR)*, 99:1201–1242, August 2010.

[66] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabási. Human mobility, social ties, and link prediction. In *Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1100–1108, 2011.

[67] J. Weston, O. Chapelle, A. Elisseeff, B. Scholkopf, and V. Vapnik. Kernel dependency estimation. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 873–880, 2002.

[68] Y. Zheng, L. Zhang, X. Xie, and W. Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of International Conference on World Wide Web (WWW)*, pages 791–800, 2009.

A. Discrete Variant of Kernelized Map Matching

The problem describe in Eq. (3.1) can be clearly seen as a combination of discrete and continuous optimization problem . We try to solve the problem in Alternate steps for discrete and continuous parts. Solving the discrete part is about finding the right assignment of V over P and continuous optimization part for the solution contains moving these points along the edges to find best solution for F_o.

A.0.1. The Discrete Part

In the beginning we try to solve the discrete part by considering a subset E'_s of Edges in E' according to some intelligent or random initialization scheme. Once we have fixed this subset E'_s, we start treating every edge as a straight line. we try to solve the continuous optimization part by finding the best position of points along these lines. Our approach here is to assume that the solution of this optimization problem gives the best solution over all possible subsets of E' unless it violates some basic constraint of the problem. Notice that optimizing along lines is not same as optimization along edges. If the solution contains a point which lies outside the edge describing this line then we may no longer use this solution because one constraint of the problem (the output points should lie along the edges) is violated. we discard all such associations between points in V and edges of E'_s. The we search for new association of V in the spatial neighbourhood and apply continuous optimization again. We continue to do so until we find a solution where no association between points and edges is violating the constraint. Our approach is summarized as follows

1. initialize assigning points to output graph
2. fix assignments and optimize along lines
3. If the points remain on edges, keep them otherwise assign them to nearest edge
4. Repeat steps 1-2, until we can't find improvement

Algorithm 3: KMM-Discrete

Input : G, G', K_G,$K_{G'}$,$\Psi_{G'}$, λ_1, λ_2, t_o
Output: $\Phi(V)$ - The required mapping between V and G'

// G, G'- Euclidean Graphs
// $K_G, K_{G'}$ -Graph Kernels
// λ_1, λ_2 -Regularizers
// t_o- Threshold of convergence
forall the v *in* $V(G)$ **do**
  $\phi(v)$= FindClosest(v, G')
$K_G \leftarrow$ Rbf(G) // compute K_G applying Rbf kernel on G
repeat
  $\Psi_{G'} \leftarrow$ShortestPath$(\Phi(V), G')$ // shortest paths among $\Phi(V)$
  $K_{G'} \leftarrow e^{\Psi_{G'}}$ // get $K_{G'}$ from distance measure $\Psi_{G'}$
  $r \leftarrow$Parametrize$(\Phi(V))$ // Parametrize $\Phi(V)$ into spherical coordinates to fix points on straight lines
  $C_{o1} \leftarrow C_o(r)$ $r' \leftarrow \frac{\triangle C_o}{\partial r}$// compute derivative of Objective function w.r.t r
  $r = r - \alpha r'$ // compute new value of r
  $C_{o2} \leftarrow C_o(r)$
  Update(α)
  $t \leftarrow C_{o2} - C_{o1}$ // compute threshold from previous two values of objective function
until $t \geq t_o$;
$\Phi(V) \leftarrow$ DeParametrize(r) // Output deparametrized r

Note that our approach of finding the subset of edges E'_s is similar to WalkSAT and other greedy solutions for finding approximate solutions to SAT problem.

A.0.2. The Continuous Part

Once the scheme for solving the discrete part of C_o is devised. We can dig into the continuous part. Here we take two kernels $K_{G'}$ and K_G, respectively between the output points and input points and try to come up with a solution which minimizes the difference between these two kernels. We are using the matching scheme described in "twin kernel embedding" to match these two kernels. The kernel between input points is an RBF kernel while the kernel between output points is an RBF kernel over shortest path graph distance. Our decision to take kernel over shortest path distances in output graph instead of taking direct euclidean coordinates changes

matters a bit. Lets have a look at the optimization equation and its derivation first.

$$\frac{\triangle C_o}{\partial P_i} = \{-K_G + 2 * \lambda_1 . K_{G'}\} . \frac{\partial K_{G'}}{\partial P_i} + \lambda_2 . \frac{\partial tr(V.P_2)}{\partial P_i} \tag{A.1}$$

We denote the shortest path distance measure matrix over G' by $\Psi_{G'}$. We know $K_{G'_{ij}} = e^{-\Psi_{G'_{ij}}}$, Now

$$\frac{\partial K_{G'_{ij}}}{\partial P_i} = -e^{-\Psi_{G'_{ij}}} . \frac{\partial \Psi_{G'_{ij}}}{\partial P_i} \tag{A.2}$$

Notice that $\Psi_{G'_{ij}}$ is composed of a sum of euclidean distances over the edges comprising shortest path from P_i to P_j, because we are computing the derivative w.r.t P_i, only the edge part counts which lies between P_i and the same edge node lying on the the shortest path from P_i to P_j. We denote this node by $P_{i'}$. The Derivative of euclidean distance between P_i and $P_{i'}$ is straight forward. Once we are able to calculate the derivative of individual terms of $K_{G'_{ij}}$ w.r.t P_i, the derivative of C_o can be caluclated by matrix chain rule as follows

$$\frac{\triangle C_o}{\partial P_i} = Tr\{(\frac{\partial C_o}{\partial K_{G'}})^T . \frac{\partial K_{G'}}{\partial P_i}\} \tag{A.3}$$

Comparison of KMM Discrete Versus Base line

Figure A.1 shows the comparison of results of baseline algorithm with KMM-Discrete. Y-axis shows the mean squared Error of Baseline minus mean squared Error of KMM-Discrete in meters. Although with a smaller value of noise, the base line performs better than KMM-Discrete however as noise grows KMM-Discrete is able to improve 5-10 meters per point in the trajectory. We still think that our results with the Discrete version are not satisfactory and we need to work more on the optimization part of the algorithm. The reason seems to be the inability of our algorithm to coup for route error. Notice that although the noise is Gaussian, the route error tends to be exponential [44].

A.0.3. KMM-Discrete Limitation:

The results from KMM-Discrete show that it fails to achieve a significant advantage over Baseline, instead in a few cases it under performs the very basic algorithm suggested. While we need to work more on the optimization side of the algorithm, we have a few ideas why this is happening.

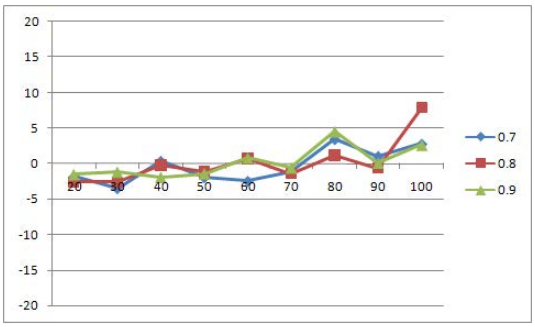


Figure A.1.: KMM-Discrete Vs.Base line over different values of λ

- One problem is the discrete nature of the search space where many local minima may exist having distant values from Global minima.
- Another problem is that the route error tends to be very large and the task of KMM-Discrete is to reduce this route error through the objective function. It is quite possible that after initialization, the shortest path between two neighbouring points is quite large because they lie on parallel roads. In this case the algorithm will try to travel through this whole path to approach near the neighbouring point but we have a regularization term in the objective function which is adding a squared penalty as the point tries to go far from a certain radius of the input location. Therefore it becomes a constant struggle between the regularization term and the first term: The route error used in the shortest path induced kernels reduces if the point tries to travel through the shortest path but on the other hand penalty term is adding a squared error. To overcome this deficiency, we might need to have a very small lambda as well but in this case the points are not restricted to stay within a close radius of the input point and therefore the solution can lie far from the original points.