1

Mobility data mining

Mirco Nanni[a]

1.1 Introduction

1.1.1 What is mobility data mining

The trajectories of a moving object are a powerful summary for its activity related to mobility. As seen in previous chapters, such information can be queried in order to retrieve those trajectories (and the objects that own them) that respond to some given search criteria, for instance following a predefined interesting behavior. However, when massive information is available, we might be able to move a step forward and ask that such "interesting behaviors" automatically emerge from the data. That is precisely the domain explored by mobility data mining.

Moving from queries to data mining essentially consists in adding degrees of freedom to the search process that the algorithms perform. For instance, a query might consist in searching those trajectories that at some point perform the following sequence of maneuvers: abrupt slow down, U-turn and finally accelerate. One possible corresponding data mining task, instead, might require to discover which sequences of maneuvers are performed frequently in the database of trajectories. Then, the output sequences obtained might contain also the *slow-down* $\rightarrow$ *U-turn* $\rightarrow$ *accelerate* example mentioned above. To perform this data mining process the user needs to specify the general structure of the behaviors he/she searches (sequences), what kind of elements they can contain (the set of maneuvers to consider, as well as a precise way to locate a given maneuver within a trajectory), and a criterion to select "interesting" behaviors – in our example, the user wants only behaviors that appear frequently in the data.

[a] KDD Lab, ISTI-CNR, Pisa, Italy

1.1.2 Note on terminology

In this chapter we will make frequent use of the terms "trajectory pattern". As mentioned in Chapter 1, the notion of trajectory pattern is substantially equivalent to that of "trajectory behavior", which also appeared in previous chapters of this book. The two notions originate from different communities and simply reflect different perspectives of the same subject: the data management view (where "trajectory behavior" originates) focuses more on determining *which trajectory* is associated to each behavior; the data mining view, on the contrary, is more focused on *what* are the interesting behaviors in the input trajectories.

The several forms and variants of existing analysis tasks that belong to mobility data mining cannot be easily categorized into a set of fixed classes. However, it is possible to recognize a few simple dimensions along which to locate the different analysis methods. In the following we mention one of them, that will be also used later as guideline during the presentation of analysis examples.

1.1.3 Local patterns vs. global models

The example of behavior illustrated at the beginning of this section is representative of a class of mining methods, called *local patterns* or, in most contexts, simply *patterns*. The key point of local patterns is the aim of identifying behaviors and regularities that involve only a (potentially small) subset of trajectories, and that describe only a (potentially small) part of each trajectory involved.

The complementary class of mining methods is called *global models*, or simply *models*. Their objective is to provide a general characterization of the whole dataset of trajectories, thus going towards the definition of general laws that regulate the data, rather than spotting interesting yet isolated phenomena. For instance, we will see later mining tasks aimed to define a global subdivision of all trajectories into homogeneous groups, as well as tasks aimed to discover rules able to predict the future evolution of a trajectory (i.e., the next locations it will visit).

In the rest of the chapter we will provide an overview of the problems and methods available in the mobility data mining field. For obvious reasons of space, the discussion will not cover exhaustively the available literature on the subject, and instead will propose some representative examples of the various topics. The presentation will mainly follow the distinction between local patterns and global models introduced above.

1.2 Local trajectory patterns/behaviors

The mobility data mining literature offers several examples of trajectory patterns that can be discovered from trajectory data. Despite this wide variety, most proposals actually respect two basic rules: first, a pattern is interesting (and therefore extracted) only if it is frequent, and therefore it involves (or appears in) several trajectories[1]; second, a pattern must describe the movement in space of the objects involved, and not only aspatial or highly abstracted spatial features.

While the spatial component of trajectory data is virtually always part of the patterns extracted, the temporal one (also intrinsic in trajectory data) can be treated in several different ways, and we will use this differentiation to better organize the presentation. Then, while a trajectory pattern always describes a behavior that is followed by several moving objects, we can choose whether they should do so together (i.e., during the period), at different moments yet with same timings (i.e., there can be a time shift between the moving objects), or in any way, with no constraints on time.

1.2.1 Using absolute time *or* Groups that move together

One of the basic questions that arise when analyzing moving objects trajectories is the following:

Are there groups of objects that move together for some time?

For instance, in the realm of animal monitoring such kind of patterns would help to identify possible aggregations, such as herds or simple families, as well as predator-prey relations. In human mobility, similar patterns might indicate groups of people moving together on purpose or forced by external factors, e.g. a traffic jam, where cars are forced to stay close to each other for a long time period.

Obviously, the larger are the groups and/or the longer is the period they stay together, the higher is the likelihood that the observed phenomenon is significant and not a pure coincidence. For instance, if two members of a population of zebras under monitoring happen to move

[1] While that holds for the majority of methods appeared in the mobility data mining literature, significant exceptions exist, including the extreme case of outliers detection, consisting of anomalous (and thus infrequent) patterns. For ease of presentation, outlier detection will be described later in this chapter, in the context of *global models*.

close to each other for a short time, that can be seen as a random encounter. However, if dozens of zebras are observed together for several hours, we can safely assume that they form a herd or something is happening that forces them to keep together.

The simplest form of trajectory pattern in literature that exactly answers the question posed above is the *trajectory flock*. As the name suggests, a flock is a group of moving objects that satisfy three constraints:

- a spatial proximity constraint: within the whole duration of the flock, all its members must be located within a disk of radius r – possibly a different one at each time instant, i.e. the disk moves to follow the flock;
- a minimum duration constraint: the flock duration must be at least k time units;
- a frequency constraint: the flock must contain at least m members.

Figure 1.1(a) shows an abstract example of flock, where three trajectories meet at some point (at the fifth time unit), keep close to each other for some time (four consecutive time units) and then separate (ninth time unit). If, for instance, the constraints chosen by the user are the radius r used in the figure to draw the circles, a minimum duration of four time units (or less) and a minimum size of three members, then the common movement shown in the figure will be recognized as a flock.

Figure 1.1(b) shows an example extracted from a real dataset that contains GPS tracks of tourists in a recreational park (Dwingelderveld National Park, in Netherland). The leftmost section of the figure depicts the three trajectories that were involved in the flock, while the rightmost one shows (a zoom with) only the segments of trajectories that create the flock. As we can see, in this example a flock is a local pattern, both in the sense of involving only a small subset of trajectories (three, in our case), and in the sense of describing an interesting yet relatively small segment of the whole life of the trajectories involved.

The general concepts of *moving together* or *forming a group* are implemented by the flocks framework in the simplest way possible: the objects are required to be very close to each other during all the duration of the flock. However, a group might appear also under different conditions. One of these alternatives is to require that at each timestamp the objects form a *cluster* – thus borrowing ideas and methods from the clustering literature. A notable example are *moving clusters*, a form of pattern that

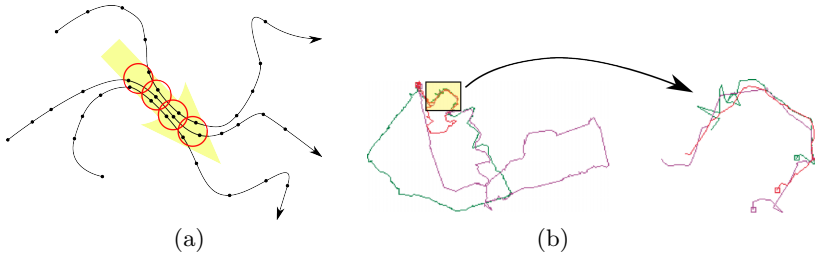

Figure 1.1 Visual representation of (a) a trajectory flock and (b) sample result on a real dataset: the left part represents the trajectories involved, the right part a zoom on the segments that form the flock

at each time stamp groups objects by means of *density-based clustering*. Such approach can be summarized in the following points:

- first, all objects that have a large number of neighbors are labeled as *core objects*; among the remaining objects, those that are neighbors of core objects are labeled as *border objects*; the remaining objects are labeled as *noise*;
- second, core objects are grouped into clusters in such a way that each pair of neighboring core objects fall in the same cluster. Essentially, clusters are computed as transitive closure of the *neighbor* relation;
- finally, border objects are assigned to the same cluster of their neighboring core objects[2], while noise is discarded.

The neighbors of an object are all the objects at a distance not larger than a threshold r, and the minimum number of neighbors required to make an object a core object is also a parameter m. Therefore, we can see that a core object and its neighbors approximately satisfy the *closeness* requirements of a flock – more exactly, these where *density* requirements. The step forward here is that multiple compact groups can be merged together if they are adjacent (see the second step), in order to form larger ones. Beside their sheer size, the groups formed through this process can also have a relatively large extension (therefore not all pairs of objects in the cluster will be close to each other, because they actually are neighbors of neighbors of neighbors...) and an arbitrary shape. In several contexts this can be useful, for instance in analyzing vehicle trajectories, since the road network simply forces large groups

[2] Notice that a border object might have two or more neighboring core objects belonging to different clusters. In this case one of them is chosen through any arbitrary criterion.

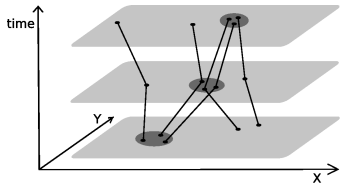

Figure 1.2 Visual example of a moving cluster over three time units

of cars to distribute along the roads (therefore creating a cluster with snake-like shape) instead of freely agglomerate around a center (which would instead yield a compact, spherical-shape cluster).

A second, interesting feature that characterizes moving clusters is the fact that the population of objects involved in the pattern can gradually change along the time: the only strict requirements are that at each timestamp a (spatial dense) cluster exists, and that when moving from a timestamp to the consecutive one the population shared by the corresponding spatial clusters is larger than a given fraction (a parameter of the method). A simple example of pattern that illustrates this point is shown in Figure 1.2: at each *time slice* a dense cluster is found, formed by three objects, and any pair of consecutive clusters share two over three objects. This way, moving clusters that last a long time, might even start from a set of objects and end into a completely different (possibly disjoint) set. In our example, only one object permanently belongs to the moving cluster. In some sense, the pattern is not strictly related to a population that generates it. The purpose of the pattern becomes to describe phenomena that happen in the population, not to find a group of individuals that do something peculiar consistently together.

One element of rigidity that affects both the patterns illustrated so far is the fact that they describe continuous portions of time. For instance, if a herd that usually moves compactly gets dispersed for a short time (for instance due to an attack by predators) and later gets compact again, both flocks and moving clusters will generally result into two different and disconnected patterns – the *before* and the *after* the temporary dispersion. One possible way to avoid this loss of information consists in allowing *gaps* in the patterns, i.e., a pattern involves a set of timestamps that are not necessarily consecutive. In the literature we can find a solution of this kind, known as *swarm patterns*. Swarms are a general form of patterns that generalize flocks and moving clusters, since any spatial clustering method can be applied at the level of single timestamp,

and then spatial clusters belonging to different timestamps are linked (in case they share an appropriate fraction of population) regardless of their temporal distance.

1.2.2 Using relative time

In some contexts, the moving objects we are examining might act in a similar way, even if they are not together. For instance, similar daily routines might lead several individuals to drive their car along the same routes, even if they leave home at very different hours of the day. Or, tourists that visit a city in different days of the year might actually visit it in the same way – for instance by visiting the same places in the same order and spending approximately the same amount of time on them – because they simply share interests and attitude. This leads to a new category of questions, which can be well represented by the following:

Are there groups of objects that perform a sequence of movements, with similar timings though possibly during completely different moments?

Patterns like flocks and moving clusters can provide some answers to the question, but usually it is a small set, since it is limited to movements that happen synchronously among all objects involved. The question posed above involves a much weaker constraint on the temporal dimension of the problem, and therefore might allow many more answers. In the following we will present one example of pattern that goes in this direction and extracts spatio-temporal behaviors that are followed by several objects, but allowing any arbitrary time shift between them.

T-Patterns (abbreviation of *Trajectory patterns*) are defined as sequences of spatial locations with typical transition times, such as the following two:

$$\text{Railway Station} \xrightarrow{15min} \text{Museum} \xrightarrow{2h15min} \text{Castle Square}$$
$$\text{Railway Station} \xrightarrow{10min} \text{Middle Bridge} \xrightarrow{10min} \text{Campus}$$

For instance, the first pattern might represent the typical behavior of tourists that rapidly reach a museum from the railway station and spend there about two hours before getting to the adjacent square. The second pattern, instead, might be related to students that reach the university campus from the station by passing through the mandatory passage on the central bridge over the river. A graphical example is also provided in Figure 1.3(left).

The two key points that characterize T-patterns are the following: first, they do not specify any particular route among two consecutive regions described: instead, a typical travel time is specified, which approximates the (similar) travel time of each individual trajectory represented by the pattern. In the gap between two consecutive regions a trajectory might even have stopped in other regions not described in the pattern; second, the individual trajectories aggregated in a pattern need not to be simultaneous, since the only requirement to join the pattern is to visit the same sequence of places with similar transition times, even if they start at different absolute times.

T-patterns are parametric on three main parameters: the set of spatial regions to be used to form patterns, i.e., the spatial extension of "Railway Station" and any other place considered relevant for the analysis[3]; the minimum support threshold, corresponding to the minimum size of the population that contributes to form the pattern (the parameter m for flocks); and a time tolerance threshold τ, that determines the way transition times are aggregated: transition times that differ less than τ will be considered *compatible*, and therefore can be joined to form a common typical transition time.

Figure 1.3(right) depicts an example of T-pattern on vehicle data describing the movements of a fleet of trucks. The pattern shows that there exists a consistent flow of vehicles from region A to region B, and then back to region C, close to the origin. Also, the time taken to move from region A to region B (t_1 in the figure) is around ten times larger then the transition time from B to C. That might suggest, for instance, that the first part of the pattern describes a set of deliveries performed by the trucks, while the second part describes the fast return to the base.

1.2.3 Not using time

In many cases it is interesting to understand if there are typical routes followed by significant portions of the population, i.e.:

Are there groups of objects that perform a common route (or segment of route), regardless of when and how fast they move?

That means, for instance, that we are interested in what path an individual follows, but not the hour of the day he/she does it, nor the

[3] Actually, the algorithmic tool provided in literature to extract T-patterns also contains heuristics to automatically define such regions, but in general the domain expert might want to do it maually in order to better exploit its knowledge or to better focus the analysis, or both.

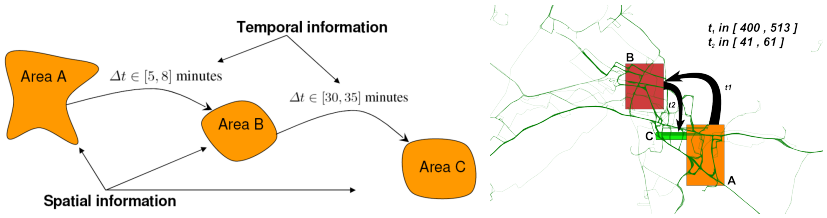

Figure 1.3 Visual representation of a T-pattern (left) and sample result on a real dataset (right)

transportation means adopted: cars, bicycles, pedestrians and people on the bus might follow the same path yet at very different speeds, resulting in different relative times. Also notice that we are interested here in routes that might be just a small part of a longer trip of the individual.

The mobility data mining literature provides a few definitions of patterns that can answer to the question given above. In particular, we will briefly summarize one of the earliest proposals appeared, at that time named generically as *spatio-temporal sequential pattern* (in contrast, the trend in recent times is to assign elaborate and sonorous names to any new form of pattern or model).

The basic idea, also depicted in Figure 1.4, consists of two steps[4]: first, segments of trajectories are grouped based on their distance and direction, in such a way that each group is well described by a single representative segment (see the two thick segments in the figure); second, consecutive segments are joined to form the pattern. Frequent sequences are then outputted as sequences of rectangles such that their width quantifies the average distance between each segment and the points in the trajectory it covers. Figure 1.4 depicts a simple pattern of this kind, formed of two segments and corresponding rectangles. In particular, it is possible to see how the second part of the pattern is tighter than the first one, i.e., the trajectory segments it represents are more compact.

1.3 Global trajectory models

A common need in data analysis at large is to understand which are the laws and rules that drive the behavior of the objects under monitoring.

[4] The original proposal of this pattern considers a single, long input trajectory. However, the same concepts can be easily extended to multiple trajectories.

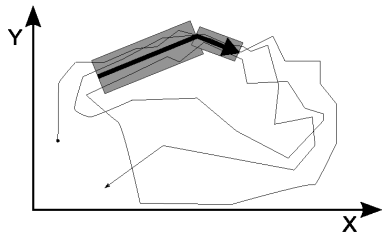

Figure 1.4 Visual representation of a spatio-temporal sequential pattern

In the context of mobility data mining we refer to such laws and rules as (global) trajectory models, and in this area we can recognize three important representative classes of problems: dividing trajectories into homogeneous groups; learning rules to label any arbitrary trajectory with some tag, to be chosen among a set of predefined classes; predicting where an arbitrary trajectory will move next. In the following we will introduce and discuss each of them.

1.3.1 Trajectory clustering

In data mining, clustering is defined as the task of creating groups of objects that are similar to each other, while keeping separated those that are much different. In most cases, the final result of clustering is a partitioning of the input objects into groups, called *clusters*, which means that all objects are assigned to a cluster, and clusters are mutually disjoint. However, exceptions to this general definition exists and are relatively common.

While the data mining literature is extremely rich of clustering methods for simple data types, such as numerical vectors or tuples of a relational database, moving to the realm of trajectory makes it difficult to directly apply them. The problem is, trajectories are complex objects, and many traditional clustering methods are tightly bound to the simple and standard data type they were developed for. In most cases, to use them we need to adapt the existing methods or even to re-implement their basic ideas in a completely new, trajectory-oriented way. We will see next some solutions that try to reuse as much as possible existing methods and frameworks; then, we will discuss a few clustering methods that were tailored around trajectory data since the beginning.

Generic methods with trajectory distances. Several clustering methods

in the data mining literature are actually clustering schemata that can be applied to any data type, provided that a notion of similarity or distance between objects is given. For this reason, they are commonly referred to as *distance-based* methods. The key point is that such methods do not look at the inner structure of data, and simply try to create groups that exhibit small distances between its members. All the knowledge about the structure of the data and their semantics is encapsulated in the distance function provided, which summarizes this knowledge through single numerical values – the distances between pairs of objects; the algorithm itself, then, combines such summaries to form groups by following some specific strategy.

To give an idea of the range of alternative clustering schemata available in literature, we mentioned three very common ones: *k-means*, *hierarchical clustering*, *density-based clustering*.

K-means tries to partition all input objects into k clusters, where k is a parameter given by the user. The method starts from a random partitioning and then performs several iterations to progressively refine it. During an iteration, k-means first computes a centroid for each cluster, i.e., a representative object that lies in the perfect center of the cluster[5]), then re-assigns each object to the centroid that is closest to it. Such iterative process stops when convergence (perfect or approximate) is reached.

Hierarchical clustering methods try to organize objects in a multi-level structure of clusters and sub-clusters. The idea is that under tight proximity requirements, several small and *specific* clusters might be obtained, while loosening the requirements some clusters might be merged together into larger and more *general* ones. For instance, *agglomerative* methods start from a set of extremely small clusters – one singleton for each input object – and iteratively selects and merge together the pair of clusters that are most similar. At each iteration, then, the number of clusters decreases of one unit, and the process ends when only one huge cluster is obtained, containing all objects. The final output will be a data structure called *dendogram*, represented as a tree where each singleton cluster is a leaf, and each cluster is a node having as children the two sub-clusters that originated it through merging.

Finally, *density-based clustering*, as already introduced in Section 1.2.1,

[5] Notice that such object is a new one, computed from those in the cluster. Therefore, some level of understanding of the data structure is needed, here. When that is not possible, usually a variant is applied, called *k-medoid* that selects the most central object of the cluster as representative.

is aimed to form maximal, crowded (i.e., dense) groups of objects, thus not limiting the cluster extension or its shape and, in some cases, putting together also couples of very dissimilar objects.

How to choose the appropriate clustering method? While no strict rule can exist, a general hint consists in paying attention to some basic characteristics of the data and the expected characteristics of the output. For instance, if we expect that our data should form compact clusters of spherical shapes (i.e., they should agglomerate around some centers of attraction), then k-means is a good candidate, especially if the dataset is large – k-means is known to be very efficient. However, the user should know the number k of clusters to be found in the data, or at least some reasonable guess. That can be avoided with hierarchical, agglomerative algorithms, since the dendograms they produce synthesize the results that can be obtained for all possible values of k, from 1 to N (the number of input objects). The choice of the most appealing k can be postponed after the computation, and be supported by an examination of the dendogram. However, hierarchical clustering is usually expensive (efficient variants exist, yet introducing other factors to be evaluated), so it is not a good option with large datasets. Finally, density-based methods apparently do not suffer of any of the issues mentioned above, and are also more robust to noisy data, yet the resulting clusters will usually have an arbitrary shape and size – a feature that might be unacceptable in some contexts, and extremely useful in others.

Depending on the analysis task that the user wants to perform, once the clustering schema to be adopted has been selected, he/she needs to choose the most appropriate similarity function, i.e., the numerical measure that quantifies how much two trajectories look similar. The range of possible choices is virtually unlimited. The examples that can be found in the literature include the following, approximately sorted in increasing order of complexity:

- spatial starts, ends, or both: two trajectories are compared based only on their starting points (the origin of the trip), or the ending point (the final destination of the trip), or a combination of them. The distance between the trajectories, then, reduces to the spatial distance between two points. When both starts and ends are considered, the sum or average of their respective distances is computed. The output of a clustering based on these distances will generally put together

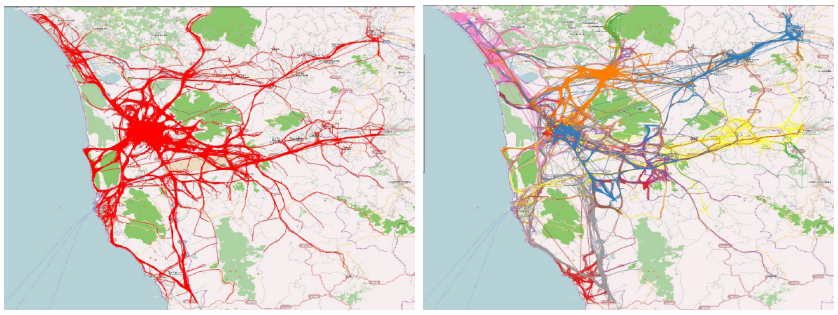

Figure 1.5 Sample trajectory clustering on a real dataset, obtained using a density-based clustering schema, and a spatial route distance function

trajectories that start or end in similar places, regardless of when they do start/end and what happens in the rest of the trajectory;

- spatial route: in this case, the spatial shape of the trajectory is considered, and two trajectories that follow a similar path (though possibly at different times and with different speeds) from start to end, will result in a low distance.
- spatio-temporal route: in this case, also the time is considered, therefore two trajectories will be similar when they approximately move together throughout their life.

Obviously, the selection of the clustering schema and the selection of the distance function might also be performed in the opposite order. Indeed, in some cases the choice of the distance to adopt is relatively easy or even enforced by the specific application, in which case the selection of the distance is performed first.

Figure 1.5(right) shows an example of result obtained by a specific combination of schema and distance, namely a density-based clustering algorithm using the spatial route distance described above. Different clusters are plotted with different colors. The dataset used in the example contains trajectories of vehicles in Tuscany, Italy, also plotted on the left of the figure.

Trajectory-oriented clustering methods. A complementary approach to clustering, as opposed to the distance-based solutions described so far, consists in algorithms that try to better exploit the nature and inner structure of trajectory data. From a technical point of view, that

usually translates to deeply re-adapt some existing solution in order to accommodate the characteristics of trajectory data.

One important family of solutions makes use of standard probabilistic modeling tools. A very early example was provided by *mixture models-based clustering* of trajectories. The basic idea is not dissimilar from k-means: we assume that the data actually forms a set of k groups, and each group can be summarized by means of a representative object. The difference is that now the representative is a probability distribution of trajectories that fits well with the trajectories in its cluster. The key point in this approach, obviously, is exactly how a probability distribution of trajectories can be defined (and fitted on a dataset). In short, the solution adopted computes a central representative trajectory plus a random Gaussian noise around it. The closeness of a trajectory from the cluster center, then, is simply computed as its likelihood, i.e., the probability that it was generated from the central trajectory adding some Gaussian noise. Another well-known statistical tool often adopted when dealing with trajectories are Hidden Markov Models (HMMs). The basic approach, here, consists in modeling a trajectory as a sequence of transitions between spatial areas. Then, a cluster of trajectories is modeled by means of a Markov model (i.e. the set of transition probabilities between all possible pairs of regions) that better fits the trajectories. Moreover, the precise position that a trajectory is expected to occupy within each spatial region is also modeled through a probability distribution. The clustering problem, then, consists in finding a set of HMMs (the clusters), such that each of them fits well with a subset of the trajectories.

Other examples of trajectory-oriented clustering methods can arise by adding novel dimensions to the clustering problem. For instance, in the literature it was investigated the problem of finding clusters by means of a distance-based clustering method (a density-based one, more exactly, though a similar process might be easily replicated for other approaches) when it is not known in advance the time interval to consider for clustering. For instance, we might expect that rush hours in urban traffic data exhibit cluster structures that are better defined than what happens in random periods of the day. The problem, then, becomes to find both the optimal time interval (rush hours were just a guess to be confirmed) and the corresponding optimal cluster structure. The solution proposed, named *time-focused trajectory clustering*, adopts a trajectory distance computed as the average spatial distance between the trajectories within a given time interval, which is a parameter of the distance. Then, for each time interval T, the algorithm can be run focusing on the

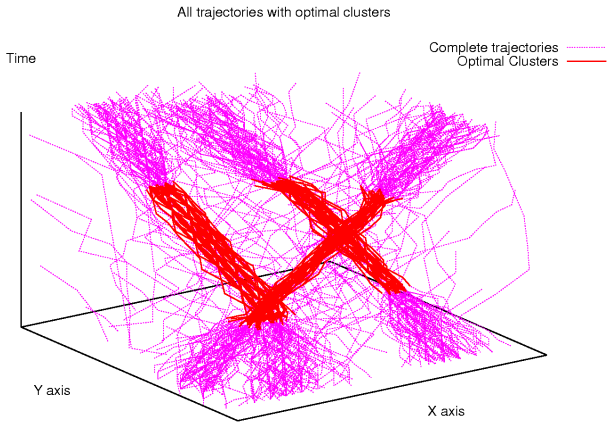

Figure 1.6 Three-dimensional depiction of sample result obtained with time-focused trajectory clustering on a dataset of synthetic trajectories.

trajectory segments laying within T. The quality of the resulting clusters is evaluated in terms of their density, and an heuristics is provided to explore only a reasonable subset of the possible values of T. A sample result of the process is given in Figure 1.6, that depicts a set of trajectories forming three clusters (plus some noise) and shows the optimal time interval (that where the clusters are clearest) as dark trajectory segments.

1.3.2 Trajectory classification

Clustering is also known as unsupervised classification, since the objective is to find a way to put objects into groups without any prior knowledge of which groups might exist, and what their objects look like. In several contexts such knowledge is available, more exactly in the form of a set of predefined *classes* and a set of objects that are already labeled with the class they belong to – the so called *training set*. The problem, here, becomes to find rules to classify new objects in a way that is coherent with the prior knowledge, i.e. they fit well with the training set. For instance, we might have access to a set of vehicle trajectories that were manually labeled with the vehicle type (car, truck, motorbike), and we would like to find a way to automatically label an other, much larger set of new trajectories.

The simplest solution to the problem is the so called *k-nearest neighbors (kNN)* approach: instead of inferring any classification rule, it directly compares each new trajectory t against the training set, and finds the k labeled trajectories that are closest to t. The most popular label among the neighbors is then assigned also to t. The assumption is that the more similar are two trajectories, the more likely they belong to the same class. Obviously, everything revolves around a proper choice for the similarity measure applied, which should be as much coherent as possible with the classification problem at hand. As an example, we can expect that a similarity function which takes into consideration the acceleration of objects will recognize well the vehicle type – the lighter the vehicle, the easier is to reach high accelerations. On the contrary, a measure based only on the places visited might perform worse.

The same idea is also applied in several sampling-based solutions to the clustering problem: when the dataset is too large to process, one approach consists in randomly sampling a small subset of trajectories, and computing clusters on them. Then, all others trajectories are assigned to the cluster (i.e., classified) with a kNN approach or by comparing them against the centroid of each cluster.

Approaching the problem from a different viewpoint, each class involved in the classification problem could be modeled through a probabilistic model that is fitted to the available trajectories in the class. Then, each new trajectory can be assigned to the class whose model most likely generated it. Similarly to what we have seen with clustering, hidden Markov models are a common choice to do it. As compared to clustering, the problem is now simplified, since the association trajectories $\leftrightarrow$ classes is known apriori. Behind the probabilistic framework they operate in, HMMs essentially aggregate trajectories based on their *overall* shape, again assuming that similar trajectories have better chances of belonging to the same class.

The final way to classify trajectories we will see, is based on a traditional two-steps approach: first extract a set of discriminative features by a preliminary analysis of the trajectories; then, use such features – that can be expressed as a database tuple or a vector – to train any existent standard classification model for vector/relational data.

The first step requires to understand which characteristics of the trajectories appear to better predict which class each trajectory belongs to. One straightforward approach might consist in calculating a predefined set of measures expected to be informative enough for the task. For in-

stance, aggregates such as average speed of the trajectory, its length, duration, average acceleration and diameter of the covered region might be used. Other, more sophisticated, solutions might instead try to extract finer aspects of the movement, tuned to calculate only the most useful ones. A proposal of this kind can be find in literature with the name *TraClass*, which heavily relies on a trajectory clustering step.

TraClass is based on a fundamental observation: in several (if not most) cases, the features that best discriminate trajectory classes are related to a small part of the overall trajectory. All approaches mentioned so far, on the contrary, uniquely consider overall characteristics – that includes HMM-based solutions, since each model must fit whole trajectories. Single, short-duration events hidden in the long life of a trajectory might then be lost in the process. TraClass tries to fill in the gap by extracting a set of trajectory behaviors (which, we recall, look for local behaviors rather than overall descriptions of full trajectories). The basic tool adopted is trajectory segmentation and the clustering of such segments to form movement patterns.

TraClass works at two levels: regions and trajectory segments. At the first one, it extracts higher-level features based on the regions of space that trajectories visited, without using movement patterns; at the second one, lower-level trajectory-based features are computed, using movement patterns. The extraction phase is made more effective by evaluating the discriminative power of the regions and patterns under construction. For instance, a frequent movement that is performed by trajectories of all classes will be not useful for classification (knowing that a trajectory contains such pattern does not help in guessing the right class to associate to it); on the contrary, a slightly less frequent pattern that is mostly followed by trajectories of a single class is a very promising feature. In the proposed framework, trajectory partitioning makes discriminative parts of trajectories identifiable, and the two types of patterns collaborate to better characterize trajectories.

Once a vector of features has been computed for each trajectory, we can choose any generic, vector-based classification algorithm. One representative (and easy to grasp) example are *decision trees*. The resulting classification model has the structure of a tree, whose internal nodes represent tests on the features of the object to classify, and the leaves indicate the class to associate to the objects. Figure 1.7 shows a fictitious example based on TraClass features, with two classes: positive (P) and negative (N). When a new trajectory needs to be classified, the test on the root (the top circle) is performed on it. In the example, if the trajec-

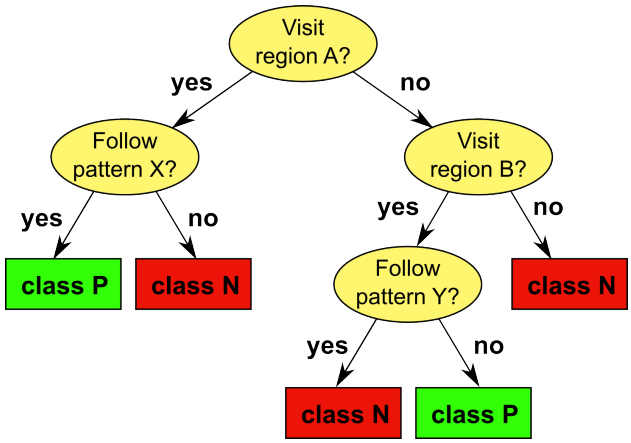

Figure 1.7 Sample decision tree on regions and patterns.

tory actually visits region A, then we move to the left child of the root and continue the evaluation from there, otherwise we move to the right child. In the first case, we have now to test whether the trajectory follows pattern X: in case of positive answer, the trajectory is labeled with "class P", otherwise with "class N". The classification process proceeds in a similar way when different outcomes are obtained, always starting from the root and descending through a path till a leaf is reached, which provides the label prediction. Another way to read a decision tree is as a set of decision rules, one for each path from root to leaf, such as "If (Visit region A) AND (Follow patter X) THEN Class P".

1.3.3 Trajectory location prediction

Trajectory classification can be seen as the problem of predicting a categorical variable related to a trajectory. However, prediction is most naturally related to the temporal evolution of variables. Since the basic aspect of objects in the context of trajectory is their location, predicting their future position appears to be a problem of primary interest.

The modeling tools that are able to model the sequential evolution of the objects they describe, are good candidates for a predictive usage. Indeed, once a trajectory has been associated to the most likely model (for instance by choosing one of the k HMMs combined in a mixture-model, as described for the clustering problem), such model can be *run* to

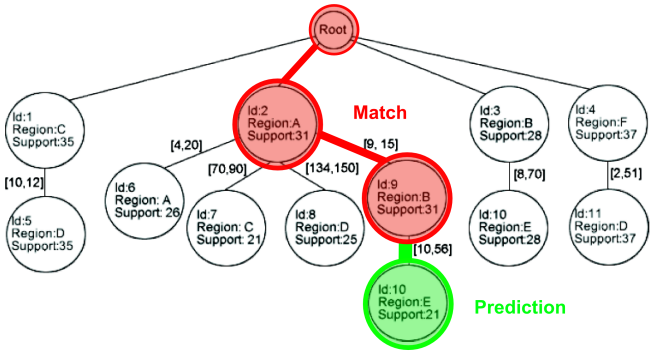

Figure 1.8 Sample Prediction tree produced by WhereNext.

simulate the most likely next steps. In most cases we can apply the same remarks discussed earlier in this section for classification: if the model is based on an overall summary of the behavior of a set of trajectories, most likely it will not be able to capture local events, even though their appearance is highly correlated with a future behavior – in our case, the next location.

In literature it can be found an approach called *WhereNext*, that works in a way not too dissimilar from the one followed by TraClass for the classification problem. Basically, WhereNext extracts T-patterns (see Section 1.2.2) from a training dataset of trajectories and combine them into a tree structure similar to a prefix-tree. In particular, each root-to-node path corresponds to a T-pattern, and root-to-leaf paths correspond to maximal patterns. Figure 1.8 shows a sample prediction tree, condensing 12 patterns, 7 of which are maximal (one per leaf).

When a new trajectory is presented, its most recent segment is compared against the regions represented in the tree, looking for the best match among the root-to-node paths. For instance, Figure 1.8 depicts the case where the last part of the trajectory visits region A followed by region B after a delay between 9 and 15 time units. The match is depicted by the red sequence. Then, the model finds that the matched sequence is a prefix of a longer pattern, and so it suggests as likely continuation region E (marked in green in the figure), to be reached after a delay between 10 and 56 time units.

WhereNext has a few characteristic features that distinguish it from most alternative approaches: first, the next location prediction is equipped also with a temporal delay; second, if no good match is found between

new trajectories and prediction tree, no prediction is provided, while most standard approaches always output a suggestion, even if it has an extremely low confidence; finally, the location prediction occurs in terms of regions and not single spatial points, although the center of the region can be returned if a single point is required by the specific application.

1.3.4 Trajectory outliers

The general objective of clustering is to fit each object in data into some category (and discovering the categories is part of the problem). However, sometimes the analyst is exactly interested in those objects that deviate from the rest of the dataset, and therefore cannot really fit any category. Such objects are called *outliers*.

Finding an outlier object means to discover some feature or pattern that holds for the object, and yet is anomalous or at least very rare in the dataset. In this sense, the problem can be properly seen as a (infrequent) pattern discovery task. The reason for discussing it now is that most outlier detection methods in literature actually adopt some clustering procedure, and identify outliers as those objects that are (or would be) left out of any cluster. Here we provide two examples.

A basic method for discovering trajectory outliers consists in adopting a density-based clustering perspective, and therefore compute the number of neighbors of each trajectory over a reasonably large neighborhood. Then, the trajectories that have too few neighbors are classified as outliers. As density-based clustering, the method is parametric on the distance measure adopted, and therefore, in principle, any distance between trajectories can be applied. Alternatively, from each trajectory a set of predefined representative features can be extracted, such as average speed, initial position, and similar, and then apply any standard distance over vector data.

In Section 1.3.2 the *TraClass* trajectory classification method was presented, which has the characteristic of working over trajectory segments (obtained by properly cutting original trajectories) rather than whole trajectories. By clustering such segments, relevant sub-trajectory patterns were extracted and later used for classification purposes. Following the same idea, outliers can be found within trajectory segments, therefore focusing on single parts of trajectory that behave in a anomalous way. In particular, each trajectory segment is compared against the rep-

resentative segment of each cluster, and if no representative segment fits well enough, the input trajectory segment is classified as outlier.

1.4 Conclusion

We conclude this chapter with a few notes on the topics presented and some of the open questions in mobility data mining research.

1.4.1 Summary

Mobility data mining, as many other instantiations of the general data mining paradigm into specific contexts, brings with itself the general categorization of problems and methods it inherited from standard data mining. In particular, the three main categories – frequent patterns, clustering and classification – appear again. However, some specificities of trajectory data emerged and stimulated the development of new approaches. In particular, the complexity of the data, joining temporal and spatial information, greatly increases the search space of most interesting problems, such as finding patterns or discovering discriminative spatio-temporal features for classification or prediction problems.

1.4.2 Open questions

One aspect of mobility data mining that the reader might have guessed by reading this chapter is the fact that it still lacks an overall, comprehensive and clear theoretical framework. Such a framework should be able to accommodate existing problems and solutions proposed in literature, as well as clarify the relations between them. Some examples of efforts in this direction exist in literature, and we also reported a few of them – for instance, the relation between local trajectory patterns and global trajectory classification models, and their ability to grasp different, complementary kinds of discriminatory features of trajectory data; or the relations between some of the various forms of trajectory pattern. However, such cases are rather isolated, and at the present, providing an integrated view of methods and issues is still a largely unexplored part of the research field.

Another important point in mobility data mining is the fact that several data sources might provide information about the same mobility phenomena coming from different viewpoints. Each data source usually

has distinctive characteristics, strong points and limitations, and their integration might help in overcoming the limits of each of them. For instance, vehicle GPS data are usually very detailed in space (i.e., spatial uncertainty is small) and time (frequency of data acquisition is relatively high), yet it is inherently limited to the vehicles that are involved in the data collection process; instead, mobile phone service providers are able to collect information about mobility of all their customers, and through the collaboration of a few providers it is possible to cover the activity of very large portions of the real population. One example are *call detail records (CDRs)*, that describe the cell towers that served each call performed by each phone, together with its timestamp. CDRs allow to build mobility trajectories for each customer served. However, such trajectories are very sparse (one point corresponds to a call, which are usually not so frequent) and spatially rough (a point actually represents the whole area served by the cell tower). Activities that try to combine these two data sources are appearing nowadays, with the aim of improving the representativity of GPS data through the extremely high penetration of the (spatially and temporally poor) CDR data.

Finally, so far, our discussion has always implicitly assumed that the trajectory data were analyzed off-line and in a centralized setting, i.e., by first collecting all data in a single database and then analyzing them. However, mobility data are usually massive and arrive as a continuous stream from the data source(s). Massiveness and streaming nature of data leads, at a large scale, to make it impossible to collect them in a centralized database, and therefore analysis methods need to be developed that exploit appropriate technologies, such as distributed databases (a paradigm where data are distributed along several data centers, to be queried to obtain the data needed for each specific analysis or computation step), distributed computation (several nodes with computation powers collaborate to analyze data) and streaming-oriented computation (essentially aimed to perform computation by looking at the input data only once).

1.5 Bibliographic Notes

As mentioned at the beginning of the chapter, the literature on mobility data mining is rather extensive – especially for such a young field – and heterogeneous. Attempting an exhaustive discussion of existing problems and proposals would require much more space and would be beyond our

purposes as well. In the following, we will provide an essential list of bibliographic references for the reader, including those describing the methods cited in the chapter and a few pointers for further readings.

The original definition of flock patterns required that the group of objects meet at a single time instant and have the same direction of movement. Successive variants introduced the temporal duration constraint, also adopted in this chapter, starting from Gudmundsson et al. (2004). Moving clusters were defined by Kalnis et al. (2005), provided with a few heuristics for incrementally computing the interesting patterns, while spatio-temporal sequential patterns appeared in Cao et al. (2005).

T-patterns were introduced by Giannotti et al. (2007), and later were exploited in building WhereNext – a location prediction method by Monreale et al. (2009) – as well as in several application works.

One rich source for a library of trajecory distances – to be used within generic clustering algorithms – is provided by Pelekis et al. (2007). Several references exist for standard (distance-based) clustering schema that can be applied to trajectory data, including basic introductions to data mining such as Tan et al. (2005).

Model-based approaches to trajectory clustering can be found in several, isolated papers, especially on specific application domains (video surveillance, animal tracking, etc.). The mixture-models trajectory clustering described in this chapter was first introduced in Gaffney and Smyth (1999), later extended to include time shifts. Hidden Markov Models-based approaches can be found, for instance, in Mlich and Chmelar (2008).

Time-focused clustering, an extension of density-based clustering for trajectories, was presented in Nanni and Pedreschi (2006).

The TraClass framework for trajectory classification was introduced in Lee et al. (2008a), mainly based on previous works of the same authors on trajectory segmentation and clustering. The same principles were then applied to the outlier detection problem, as described in Lee et al. (2008b).

Finally, a few sources already exist for deeper exploring the subject of data mining on trajectory data, including the book by Giannotti and Pedreschi (2008), which contains a chapter on spatio-temporal data mining, and the book chapter on spatio-temporal clustering by Kisilevich et al. (2010).

References

Cao, H., Mamoulis, N., and Cheung, D.W. 2005. Mining Frequent Spatio-temporal Sequential Patterns. Pages 82–89 of: *Proceedings of the 5th International Conference on Data Mining*. IEEE Computer Society Press.

Gaffney, S., and Smyth, P. 1999. Trajectory Clustering with Mixture of Regression Models. Pages 63–72 of: *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*. ACM Press.

Giannotti, F., Nanni, M., Pinelli, F., and Pedreschi, D. 2007. Trajectory Pattern Mining. Pages 330–339 of: *Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining*. ACM Press.

Giannotti, Fosca, and Pedreschi, Dino (eds). 2008. *Mobility, Data Mining and Privacy - Geographic Knowledge Discovery*. Springer.

Gudmundsson, J., van Kreveld, M. J., and Speckmann, B. 2004. Efficient detection of motion patterns in spatio-temporal data sets. Pages 250–257 of: *Proceedings of the 12th International Workshop on Geographic Information Systems*. ACM Press.

Kalnis, P., Mamoulis, N., and Bakiras, S. 2005. On Discovering Moving Clusters in Spatio-temporal Data. Pages 364–381 of: *Proceedings of the 9th International Symposium on Spatial and Temporal Databases*. LNCS 3633. Springer.

Kisilevich, S., Mansmann, F., Nanni, M., and Rinzivillo, S. 2010. Spatio-temporal clustering. Pages 855–874 of: Maimon, O., and Rokach, L. (eds), *Data Mining and Knowledge Discovery Handbook*, second edn. Springer.

Lee, J.-G., Han, J., Li, X., and Gonzalez, H. 2008a. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, **1**(1), 1081–1094.

Lee, Jae-Gil, Han, Jiawei, and Li, Xiaolei. 2008b. Trajectory Outlier Detection: A Partition-and-Detect Framework. Pages 140–149 of: *Proceedings of the 24th International Conference on Data Engineering*. IEEE Computer Society Press.

Mlich, J., and Chmelar, P. 2008. Trajectory classification based on Hidden Markov Models. Pages 101–105 of: *Proceedings of the 18th International Conference on Computer Graphics and Vision*.

Monreale, A., Pinelli, F., Trasarti, R., and Giannotti, F. 2009. WhereNext: a location predictor on trajectory pattern mining. Pages 637–646 of: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM Press.

Nanni, M., and Pedreschi, D. 2006. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, **27**(3), 267–289.

Pelekis, N., Kopanakis, I., Marketos, G., Ntoutsi, I., Andrienko, G., and Theodoridis, Y. 2007. Similarity Search in Trajectory Databases. Pages 129–140 of: *Proceedings of the 14th International Symposium on Temporal Representation and Reasoning.* IEEE Computer Society.

Tan, P.-N., Steinbach, M., and Kumar, V. 2005. *Introduction to Data Mining.* Addison-Wesley.