Predicting Future Locations Using Clusters' Centroids

Sigal Elnekave
Deustche Telekom AG Labs
Ben-Gurion University
elnekave@bgu.ac.il

Mark Last
Deustche Telekom AG Labs
Ben-Gurion University
mlast@bgu.ac.il

Oded Maimon
Tel Aviv University
maimon@eng.tau.ac.il

ABSTRACT
As technology advances we encounter more available data on moving objects, thus increasing our ability to mine spatio-temporal data. We can use this data for learning moving objects behavior and for predicting their locations at future times according to the extracted movement patterns.

In this paper we cluster trajectories of a mobile object and utilize the accepted cluster centroids as the object's movement patterns. We use the obtained movement patterns for predicting the object location at specific future times. We evaluate our prediction results using precision and recall measures. We also remove exceptional data points from the moving patterns by optimizing the value of an exceptions threshold.

Categories and Subject Descriptors
I.5.3 **[Pattern Recognition]:** Clustering- algorithms.

General Terms
Algorithms, Performance, Experimentation.

Keywords
Spatio-temporal data mining, Moving objects, Prediction, Clustering.

1. INTRODUCTION
With technological progress, more data is available on the location of moving objects at different times, either via GPS technologies, mobile computer logs, or wireless communication devices. This creates an appropriate basis for developing efficient new methods for mining moving objects.

Spatio-temporal data can be used for many different purposes. The discovery of patterns in spatio-temporal data, for example, can greatly influence different fields like animal migration analysis, weather forecasting, and mobile marketing. Clustering spatio-temporal data can also help in social groups' discovery, which is used in tasks like shared data allocation, targeted advertising, and personalization of content and services. Spatio-temporal prediction can be used to improve resource utilization in wireless networks and to introduce a variety of innovative location-based services for mobile users.

The goal of this work is to predict locations of mobile objects at future times. We use a compact representation of a spatio-temporal trajectory and we cluster periodical trajectories of each object using a similarity measure that allows the discovery of recurring trajectory patterns. We remove exceptional data-points from the cluster's centroids and use the centroids as movement patterns. Then we suggest an algorithm for predicting mobile object's locations at future times according to its discovered movement patterns. Finally we evaluate the proposed algorithm by conducting experiments on a real-world collection of spatio-temporal data. We predict locations at future times by the proposed algorithm, and evaluate the precision and recall of the results. We also examine different exception thresholds in order to optimize the prediction results by tuning this parameter.

2. RELATED WORK
Krumm and Horvitz [5] describe a method called Predestination that uses a history of a driver's destinations, along with data about driving behaviors, to predict where a driver is going as a trip progresses. Driving behaviors include types of destinations, driving efficiency, and trip times. Four different probabilistic cues were considered and combined in a mathematically principled way to create a probability grid of likely destinations. The authors introduced an open-world model of destinations that helps the algorithm to work well in spite of a paucity of training data at the beginning of the training period by considering the likelihood of users visiting previously unobserved locations based on trends in the data and on the background properties of locations. The best performance on 3,667 different driving trips gave an error of two kilometers at the trip's halfway point. This technique, however, predicts the driver's destination, while we try to predict his/her location at any time during his/her trip.

Other methods for prediction in spatio-temporal databases assume that objects move according to linear functions, though in practice individual objects may follow different motion patterns, Tao et al. [6] introduce a general framework for monitoring and indexing moving objects, where first, each object computes individually the function that accurately captures its movement and then a server indexes the object locations at a coarse level and processes queries using a filter-refinement mechanism. A novel recursive motion function is suggested that supports a broad class of non-linear motion patterns. The function does not presume any a-priori movement but can postulate the particular motion of each object by examining its locations at recent timestamps. An indexing scheme is suggested that facilitates the processing of predictive queries without false misses. This prediction technique is based on recognizing a movement function, but finding such

ACMGIS'07, November 7–9, 2007, Seattle, WA.

function can be unrealistic in some cases. Therefore we will try a different approach.

In this paper we define an algorithm for location prediction in spatio-temporal data according to movement patterns that were obtained by clustering object trajectories that are cheaper to mine due to their compact representation.

3. SPECIFIC METHODS

3.1 Representing trajectories

A periodic spatio-temporal trajectory is a series of data-points traversed by a given moving object during a specific period of time (e.g., one day). Since we assume that a moving object behaves according to some periodic spatio-temporal pattern, we have to determine the duration of a single period. Thus, in the experimental part of this paper, we assume that a moving object repeats its trajectories on a daily basis, meaning that each trajectory describes an object movement during one day. In a general case, each object should be examined for its periodic behavior in order to determine the duration of its recurring movement. The training data window is a sequence of periods used to learn the object's periodic behavior based on its recorded trajectories (e.g., daily trajectories recorded during one month).

As a part of our suggested preprocessing technique [2] we represent a trajectory as a list of minimal bounding boxes. A minimal bounding box (MBB) represents an interval bounded by limits of time and location. Figure 1 demonstrates an object's trajectory and its MBB-based representation for a given period.

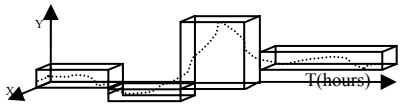

Figure 1. Object's trajectory

Incoming data-points update the current MBB in the order of their arrival times. Therefore, the minimal time bound of the first MBB is the time of the earliest data-point in the dataset and the maximal time bound of each MBB is stretched until the time or the space distance between the maximal and the minimal locations of this MBB reaches some pre-defined segmentation thresholds. When one of these thresholds is exceeded, a new minimal bounding box is initiated with the time of the subsequent data-point as its minimal time bound. The larger the threshold is, the higher is the summarization level of the trajectories, meaning that we increase the processing rate of the next mining stages (shorter running times) but also decrease their precision.

3.2 Defining a similarity measure

We define similarity between two trajectories as the sum of the similarities between each two overlapping MBBs, divided by the amounts of MBBs in each of the compared trajectories.

In [2] we empirically compared two similarity measures between two MBBs. The first similarity measure is called "minimal distances" [1]. It defines the distance between the trajectory MBBs as a lower bound of the original distance between the raw data, which is an essential property for guaranteeing correctness of results for most mining tasks.

The second similarity measure is the "data-amount-based" similarity that was shown to outperform the "minimal distances" similarity measure in [2]. We multiply the minimal-distances measure by the distance between the amounts of data points of the two compared MBBs (data#D). Since each MBB summarizes some data points, the more data points are included in both of the compared MBBs, the stronger support we have for their similarity. Our "data-amount-based" distance is calculated as:

$$d(MBB(T_i), MBB(T_j)) = \min D(MBB(T_i), MBB(T_j)) \cdot |t_m - t_n| \cdot \text{data\#} D(MBB(T_i), MBB(T_j)) \quad (1)$$

Where the distance between the amounts of data points in two MBBs is calculated by:

$$\text{data\#} D(MBB(T_i), MBB(T_j)) = |MBB(T_i).data\ \# - MBB(T_j).data\ \#| \quad (2)$$

Figure 2 describes the minimal distance and the times of overlap t_m and t_n:

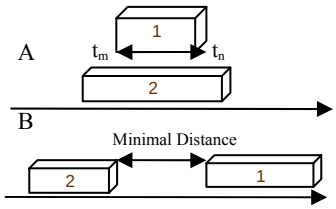

Figure 2: A. Times of overlapping between two MBBs; B. minimal distance between two MBBs

3.3 Finding movement patterns by clustering

A trajectories cluster contains similar periodic trajectories. Trajectories in the same cluster contain as much similar MBBs as possible (close in space and time). The centroid of a trajectories cluster represents a group of similar trajectories, meaning that this cluster's centroid can represent a movement pattern of a given object. Since in order to run generic clustering algorithms on the trajectories data, the algorithm needs to handle an input that consists of bound intervals (trajectories) instead of numeric vectors, we developed in [3] a spatio-temporal version of the K-Means algorithm for clustering trajectories using the data-amount-based similarity measure defined above. This version handles interval-bounded data represented by a variable amount of attributes. It uses a new centroid structure and a new centroid updating method, which are defined below.

We adapt the incremental clustering approach [3] in order to benefit from the difference between the clustering for the first training data window (e.g. trajectories during the first month of data collection), when no previous data is available, and clustering for the subsequent windows, where using previous clustering centroids can help performing a more efficient incremental clustering process. Less updates are needed assuming that the movement behavior of the same object stays relatively stable.

3.4 Representing a cluster centroid

The centroid of a trajectory cluster should represent all trajectories that belong to that cluster in some summarized manner. We represent a cluster centroid as a summarized trajectory, or in other words as a set of MBBs. Each MBB is an interval that holds information about the upper and lower bounds

in each one of the d location dimensions (in our case $d = 2$) , lower and upper time bounds, and the amount of data-points that are summarized by the MBB.

Instead of the traditional centroid structure of a numeric vector and its common cluster updating method that calculates a cluster's centroid as a vector of averages of the items in the cluster, we represent clusters as MBBs, which requires using a bounding technique for updating clusters, since averaging bounds will lead to invalid bounds, that are not the MBB real bounds. [2]

3.5 Identifying exceptions

Our purpose is to maintain reliable and generic movement patterns, without recording exceptional movements that do not occur frequently within the clustered period. Since centroids are the movement patterns in our case, we need to remove exceptional movements from the centroids, or more precisely, we need to remove exceptional MBBs from the centroids. We can detect an exceptional MBB by its "data amount" property that records the amount of data points that are summarized within that MBB during the training window. If the object is frequently found in this location at this time, the "data amount" of the MBB will be a large number, but if the object rarely reaches this location at this time, the "data amount" of the MBB will be a small number, so we can use this property for recognizing sparse MBBs, or exceptional MBBs. The algorithm for removing exceptional MBBs is as follows:

```
Input: original cluster centroids (C), an exceptions-threshold
Output: updated cluster centroids (C)
Removing exceptions from centroids:
For each c in C        --For each cluster centroid
  For each MBB in c    --For each MBB in the centroid
    If MBB.data-amount ≤ exceptions-threshold
      c.removeMBB(MBB) --Removes MBB from centroid
```

Our experiments below are aimed at finding the optimal value of the exception threshold, which is the maximal data-amount where an MBB is considered exceptional.

Our exception bound is calculated according to the following decimal scaling:

$$\text{bound} = \frac{\text{amount} - \text{of} - \text{datapoints}}{k \cdot 10^{|\text{amount-of-datapoints}| - p}} \quad (1)$$

where k is the clusters amount and p is a tuning parameter. In the experiments section below we evaluate four different values of p: 0, 1, 2, 3. Using the decimal scaling we decrease the number of data points to its first p digits, and divide it by the clusters amount (k) since more clusters lead to less data points in each MBB.

3.6 Prediction according to cluster centroids

After clustering mobile object's trajectories within a time period, and removing exceptional MBBs from the obtained centroids, we can treat the clusters centroids as the movement patterns of each mobile object. We can also predict the mobile object's locations at specific times in future periods according to the recognized moving patterns.

In our case study below, where each trajectory records information of one day, and we cluster information of about one month, we get centroids that represent a period of one day (the date is ignored). We predict the object's location at a future time by searching in each centroid for an MBB that contains the future time within its time bounds, and returning its x and y maximal and minimal bounds. If no such MBB exists, we will return the MBB which is closest in time to the search time. If more than one MBB matches the search time, there are several options: (1)The algorithm returns the coordinates of all matching MBBs. (2)The algorithm returns the coordinates of the matching MBB with the highest amount of summarized datapoints. In this paper we will operate according to the first option, which ensures that no potential MBB is ignored in the prediction stage. The algorithm is as follows:

```
Input: search time (T), object's cluster centroids (C)
Output: a list of results (R)
Predicting mobile object's future location:
For each c in C                    --For each centroid of the clusters
  i =0; MBB=c.getMBB(i)
  while  MBB.maxT< T       --Proceed to a relevant MBB
        MBB=c.getMBB(i);  i=i+1
  while MBB.minT≤T&MBB.maxT≥T  --Add relevant MBBs
        R.addMBB(MBB);  i=i+1;  Found = true
  If MBB.minT>T and not Found   --Add MBB with nearest
        If i>0                        time when there is no match
            Prev = c.getMBB(i-1)     -- Set the previous MBB
        Else Prev = c.getLastMBB()--Before 0:00 comes 23:59
        If MBB.minT-T > T-Prev.maxT
            R.addMBB(MBB) -- If current MBB is closer to T
        Else R.addMBB(Prev)--If previous MBB is closer to T
```

Using experiments on a real dataset we will evaluate the recall and the precision of this technique.

4. EXPERIMENTAL RESULTS

For empirically evaluating the proposed algorithm for predicting location of moving objects in future periods and for optimizing the exceptions bound that is used for removing exceptional data points from the cluster centroids, we used INFATI, a real-world collection of spatio-temporal data, described in [4] which contains information about 11 cars and their locations within three weeks. Each run was repeated with five different cluster amounts (k) and with four different exception bounds as explained next. The algorithm learned the movement of a mobile object from a training set that contained all dates of the data collection, except for the last two dates that were used for testing. After learning patterns from the training set we used our location prediction algorithm to predict the objects future location at each tenth timestamp in the test set, since predicting the location at each timestamp could be redundant. We evaluate the prediction results with recall and precision measures. We also measured the range

of the prediction which is the maximal distance between two points in the predicted MBB.

The amount of K-Means iterations was set to 20. We evaluate five different values of *k*: 4, 5, 6, 7, 8, and the summarization bounds for x and y coordinates were set to:

$$bound = \frac{\text{var}(D)}{(\max(D) - \min(D))} \quad (2)$$

where *D* refers to the data values in each dimension. [2]

Our exception bound is calculated according to formula (1).

We evaluate four different values of *p*: 0, 1, 2, and 3.

After running experiments, where our independent variable is exception bound and our dependent variables (tested separately) are precision, recall, prediction range, and runtime, we can evaluate our algorithm for predicting future locations using centroids. We calculate recall as the percentage of successful predictions where future locations are found within the predicted MBBs. We refer to precision as 1 if the future location is found within the predicted MBB, otherwise we calculate it as 1 divided by the minimal distance between the future location and the predicted MBB. If several MBBs are predicted we consider the maximal precision between them. We calculate the predicted range of successful predictions as the maximal distance between two points (the diagonal) in the predicted MBB. In case where several MBBs are predicted we consider the maximal diagonal over all predicted MBBs (we do not add distances since predicted MBBs may overlap). According to figure 3, the greater the exception threshold is, the more MBBs are removed and so the prediction algorithm needs to check less MBBs and therefore its running time decreases. In figure 4 we can see that recall and precision only decrease when *p*=3 in the exception bound, so the exception bound where *p*=2 is the optimum, since it removes the greatest amount of MBBs without decreasing the precision and recall. We can also see that our location prediction algorithm provides good results of 89.6% precision and 89.5% recall. We also measured the prediction average range, meaning the size of MBBs that were predicted when searching for one point. We found that this range is 13.1 in X coordinate and 27.4 in Y coordinate. In figure 5 we can see that the optimal exception bound (*p*=2) brought to the removal of 58% of the MBBs that originally constructed the clustering centroids.

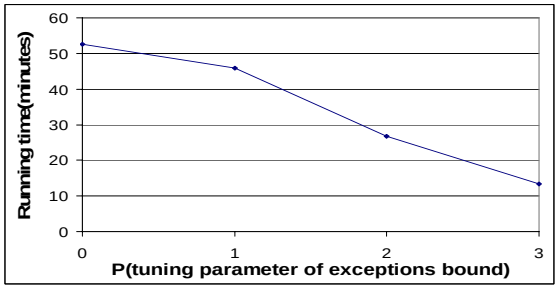

Figure 3: Running time vs. Exception bound

Figure 4: Recall and precision vs. Exception bound

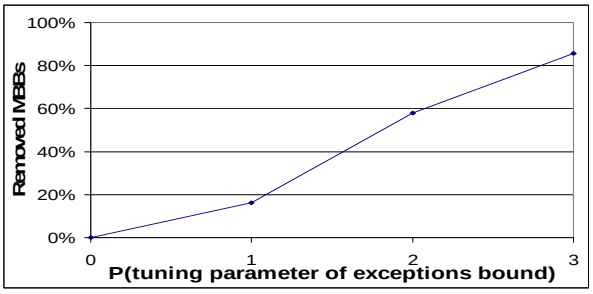

Figure 5: Amount of removed MBBs vs. Exceptions bound

5. CONCLUSIONS

In this paper, we used a novel method for clustering trajectories into periodic movement patterns, as a basis for a location prediction algorithm. The algorithm was shown to operate well with 89.6% average precision and 89.5% average recall.

We also found the optimal value of an exception threshold, which is the maximal amount of data points where an MBB is considered exceptional and is consequently removed from the cluster centroid before the prediction process starts.

Further work is needed for improving prediction accuracy by predicting different locations (MBBs) with different likelihoods according to the amount of data points within each predicted MBBs.

6. REFERENCES

[1] Anagnostopoulos A., Vlachos M., Hadjieleftheriou M., Keogh E.,Yu P.s. "Global Distance-Based Segmentation of Trajectories". *KDD'06*, 2006

[2] Elnekave S., Last M., Maimon O. " A Compact Representation of Spatio-Temporal Data*". To be published in SSTDM07*, 2007.

[3] Elnekave S., Last M., Maimon O. "Incremental Clustering of Mobile Objects*". STDM07*, IEEE, 2007.

[4] Jensen C. S., Lahrmann H., Pakalnis S., and Runge J. "The INFATI Data". *TimeCenter Technical Report*. 2004

[5] Krumm J. and Horvitz E., "Predestination: Inferring Destinations from Partial Trajectories", *UbiComp 2006*.

[6] Tao Y, Faloutsos C, Papadias D, Liu B. "Prediction and indexing of moving objects with unknown motion patterns". *SIGMOD.* ACM Press, 2004. 611-622