Available online at www.sciencedirect.com
ScienceDirect
Computers, Environment and Urban Systems
30 (2006) 741–756
Computers, Environment and Urban Systems
www.elsevier.com/locate/compenvurbsys

Location awareness through trajectory prediction

Xiong Liu, Hassan A. Karimi *

Department of Information Science and Telecommunications, University of Pittsburgh, Pittsburgh, PA 15260, USA

Received 15 July 2004; accepted in revised form 15 July 2005

Abstract

Location-aware computing is a type of ubiquitous computing that uses user's location information as an essential parameter for providing services and application-related optimization. Location management plays an important role in location-aware computing because the provision of services requires convenient access to dynamic location and location-dependent information. Many existing location management strategies are passive since they rely on system capability to periodically record current location information. In contrast, active strategies predict user movement through trajectories and locations. Trajectory prediction provides richer location and context information and facilitates the means for adapting to future locations. In this paper, we present two models for trajectory prediction, namely probability-based model and learning-based model. We analyze these two models and conduct experiments to test their performances in location-aware systems.

Keywords: Location-aware computing; Location management; User movement; Trajectory prediction; Probabilistic methods; Machine learning algorithms

1. Introduction

The emergence of widespread mobile devices and low-cost location sensing techniques, coupled with expanding wireless coverage and more reliable wireless connections, has spurred ubiquitous mobile computing. This trend has paved the way for Geospatial Information Systems (GIS) to evolve from desktop-based systems utilized primarily by

* Corresponding author. Tel.: +1 412 624 4449; fax: +1 412 624 2788.
E-mail addresses: xliu@mail.sis.pitt.edu (X. Liu), hkarimi@mail.sis.pitt.edu (H.A. Karimi).

0198-9715/$ - see front matter
doi:10.1016/j.compenvurbsys.2006.02.007

professionals to service-oriented mobile, ubiquitous GIS accessible to and usable by all types of users. Today users carrying wirelessly connected mobile devices are able to request and obtain geographic information services as they travel through the geographic space and conduct the ordinary tasks of everyday life (Smyth, 2000).

Mobile computing can be divided, depending on information access methods, into two categories: (1) conventional mobile computing that does not have the means for obtaining and utilizing user's current location in processing activities and (2) location-aware computing that is capable of obtaining and utilizing user's current location information as one of the essential parameters for providing services and application-related optimization (Fritsch & Volz, 2003). Location-aware computing is expected to benefit customers, application developers, service providers, and networks operators (Borriello & Deshpande, 2002). For example, coupling location-aware computing and GIS has brought about Location-Based Services (LBS)—where user's current location is used to compute and process queries; a typical query processed by LBS is nearest points of interest while the user is driving on a road.

Location-aware computing and ubiquitous GIS are closely related to the research area of "context-aware computing". Dey (2001) defined context as "any information that can be used to characterize the situation of an entity" such as a person, place or object relevant to the interaction between a user and an application. In addition to the explicit input by the user, further context elements could be obtained through the use of new sensors or inference mechanisms (Zipf & Jost, 2006). Context constitutes an important aspect of LBS as both the user and location are part of a context (Jiang & Yao, 2006). For example, Zipf and Jost (2006) propose an ontology-based approach to model various types of context (e.g., means of transportation) and user parameters (e.g., demographics), and introduce several strategies to realize adaptive mobile GIS that use user and context information; Li (2006) describes an immersive Virtual Reality (VR) approach for capturing real-time context data on information transactions and individual behaviors in dynamic LBS environments. In this paper, we focus on location, which is an aspect of context. In GIS, locations are modeled in continuous or discrete georeferencing systems (Jiang & Yao, 2006). For example, the universal reference system, known as World Geodetic System 1984, is a continuous georeferencing system which models location as latitude and longitude, while a mailing address is a discrete georeference system.

Location-aware computing poses challenges regarding location and location-dependent data management, including: data stream from a poll of mobile devices; varied and intermittent wireless connections; limited bandwidth, power and device size; and location information deployment for location-dependent applications (Tan & Wolfson, 2003). Various strategies for location management have been proposed in the literature. For examples, location tracking through paging in cellular mobile networks (Yeung & Yum, 1995); updating and querying Moving Object Databases (MOD) for real-time mobile units tracking (Wolfson et al., 1999); and location sensors integration and location conflict resolution when multiple sensors provide conflicting information (Indulska & Sutton, 2003). Many existing location management strategies rely on system capability to periodically record current location of the mobile unit into a database; such strategies are called passive (Aljadhai & Znati, 2001; Liu, Bahl, & Chlamtac, 1998). In contrast, active strategies are those that predict trajectories and locations well in advance of current location. Trajectory prediction is concerned with determining future paths, and location prediction is concerned with determining future locations on predicted trajectories using estimated travel

distances. Location prediction can be regarded as a supplementary step of trajectory prediction (Karimi & Liu, 2003).

Trajectory prediction is part of location management as it helps improve the provision of mobile services in many aspects such as query processing, caching and data dissemination. More importantly, it enhances location awareness. Current location-aware systems utilize location information to determine such services as nearest features of interest from a given location. The general assumption in such systems is that the area centered at the current location of the user is where the services are needed. Although this assumption is valid and used as the basis of many computing strategies, it constrains location awareness to only current location information. However, trajectory prediction extends location awareness to include future locations. Knowing future locations, the system is able to infer activities well in advance and provide much richer context information for preparing services, especially those with time-critical constraints. For example, knowing the precise location of a driver five minutes in advance it is possible to check traffic information and provide dynamic routing in case of traffic jam, or to check driving conditions and provide safety alerts in case of poor driving conditions. These value-added services will be very helpful for mobile users in unfamiliar or unsafe driving environments.

Trajectory prediction is related to time geography and travel demand analysis in transportation planning (Miller, 2003; Miller & Storm, 1996). Time geography studies how people move through social spaces in their daily activities using space–time activity data. Travel demand analysis is based on four components: (1) trip generation from specified origins, (2) trip allocation from a given origin among destinations, (3) modal split for trips between origin–destination pairs, and (4) route choice within each mode. While time geography and travel demand analysis are mainly used for public transportation planning and transportation policy development, trajectory prediction is concerned with real-time prediction of future trajectories for personalized and time-critical tasks. Due to the differences between the requirements in travel demand analysis and those in trajectory prediction, conventional travel demand analysis models are not directly applicable to trajectory prediction, thus requiring new techniques.

The rest of the paper is organized as follows: In Section 2, trajectory modeling is discussed. In Section 3, the computing infrastructure and trajectory data storage issues are described. In Section 4, the trajectory prediction process is outlined. In Sections 5 and 6, the probability-based prediction and the learning-based prediction are presented in details. In Section 7, experimentations and the performances of the two prediction approaches are discussed. Finally, in Section 8, conclusions are given.

2. Trajectory modeling

Trajectory modeling is an abstraction of user movement required for trajectory prediction. Depending on how a mobile unit's location is represented, there are two trajectory modeling methods: cell-based trajectory modeling and network-based trajectory modeling.

In cell-based modeling, the geographic area is divided into regular-shaped cells, usually determined by the architecture of the cellular network, and the mobile unit's location is determined as the cell within which it is contained (Das & Sen, 1999; Kyriakakos, Hadjiefthymiades, Frangiadakis, & Merakos, 2003; Shah & Nahrstedt, 2002). The trajectory is expressed as a series of cells making the path of the mobile unit, and the problem of trajectory prediction is to determine the next set of cells the mobile unit will follow.

Cell-based modeling has played an important role in mobile computing to improve performance such as Quality of Service (QoS) routing in mobile ad hoc networks (Shah & Nahrstedt, 2002). However, this method does not consider the geometry and topology of the actual path, and cannot precisely locate a mobile user because the radius of a typical cell is 150 m and upward (a cell may have a radius of more than 30,000 m) (Zhao, 2000). Also, it can only calculate the cell change probability based on the boundary through which the user leaves the current cell, but cannot precisely estimate the travel distance and travel time to a destination. Another problem with the cell-based modeling method is that a cell may contain several roads while the provision of a service, e.g., routing, may require the exact road on which the user is driving. Due to these limitations, the cell-based modeling method is not suitable for certain applications where on-road services are demanded.

Network-based modeling determines the mobile unit's location as a point constrained to a network instead of a cell using such geopositioning technologies as the Global Positioning System (GPS) (Karimi & Liu, 2003). In outdoor environments, the network is a road network where it contains actual roads obtained from digital maps; vertices represent intersections and edges represent road segments. In addition, frequently traveled paths can easily be incorporated into a road network database. GPS has become the dominant geopositioning technology, and it is anticipated that there will be ample GPS data generated by mobile users over time. Time-tagged GPS data present a close estimation, depending on GPS errors, of the user's actual trajectories. An accuracy of better than 30 m is expected from stand-alone GPS, while differential GPS would provide an accuracy of a few meters. Due to GPS errors, map-matching is used to locate the whereabouts of the user on the road. For example, Map Matched GPS (MMGPS) is an application that corrects raw GPS data using history of previous position estimates and road geometry (Blewitt & Taylor, 2002). Furthermore, odometer-derived MMGPS (OMMGPS) (Taylor et al., 2006) uses height information obtained from Digital Terrain Models (DTM) to improve the accuracy of GPS positions with poor satellite geometry. In network-based modeling, a trajectory is represented as a series of interconnected road segments or intersections (Karimi & Liu, 2003). The advantage of network-based modeling is that it utilizes both geometry and topology of actual roads and paths to predict trajectories. The geometry (shape) and topology (connectivity) of actual roads represent possibilities and constraints for user movement. Also, speed limits of actual roads influence user's driving speed and thus provide a reasonable estimation of travel speed.

3. Computing infrastructure

Of the different computing infrastructures possible for mobile computing environments, the one with mobile units and fixed base stations is assumed in this paper. The mobiles are linked to the base stations via wireless communications, and each mobile unit is equipped with a GPS receiver. The infrastructure also contains different databases for trajectory prediction. Available at the base stations are the road network database containing all the roads within a given area, a map-matched trajectory database containing each mobile's historical trajectories, and other useful information such as user registration profiles. Available at the mobile unit is a dynamic trajectory database containing a trajectory collected during a travel.

We call previously completed trajectories *long-term historical trajectories* and real-time trajectories *short-term historical trajectories*. A short-term historical trajectory is connected

to the future trajectory via the user's current location. The collection of all users' long-term historical trajectories is useful for discovering group movement patterns or knowledge. Short-term historical trajectories capture user's activities immediately before current location, and provide a real-time context for predicting future trajectories.

4. Trajectory prediction process

In this paper, we introduce a network-based modeling method for real-time trajectory prediction in outdoor environments. We denote user's current location at time t_k as L_k (where k is a time epoch), and the duration between the current time t_k and a later time t_{k+1} as the prediction period T. Given the current location (L_k, t_k) and a prediction period T, the problem of trajectory prediction is to predict a trajectory or path starting from current location to a future location at time $t_k + T$ or t_{k+1}. All locations are map-matched against the road network and predicted trajectories are represented as a series of interconnected road segments, or intersections.

The trajectory prediction process is accomplished through the following steps: (1) estimate the geographic range or area within which user's future activities will be performed, (2) identify a list of candidate trajectories based on the topology of the estimated geographic range, and (3) select the most likely trajectory from the list of candidate trajectories. The details of these steps are discussed in Sections 4.1–4.3.

4.1. Dynamic window

Considering the geographic range of a service area (e.g., the Pittsburgh metropolitan), it is inefficient to retrieve the entire road network for trajectory prediction since only a small portion of this area may be where user's activities are performed. We call this user's activities range, i.e., the area that contains future activities, *dynamic window*. The center of dynamic window is at user's current location and its shape can be regular, such as a circle, or irregular. For computational efficiency, we focus on circular dynamic windows. Its radius is determined by user's travel speed and the prediction period T. While T is predetermined by the system, user's travel speed is an uncertain factor. When traveling on a highway, the posted speed limit on the highway could be assumed as the speed of the user, however when traveling on a local road, the speed could be estimated at less than the posted speed limit as it is affected by traffic lights and traffic conditions. For simplicity, we assume that the user's speed is the same as the posted speed limit of the road on which the user is driving, $\mathrm{SpeedLimit}(t_k)$. Therefore, the radius R of dynamic window can be determined by

$$R = \mathrm{SpeedLimit}(t_k) \cdot T \tag{1}$$

For example, if the user is driving on a 25 miles/h road and the prediction period time is 30 min, then the size of a dynamic window is 12.5 miles.

Fig. 1 shows a road network (solid lines) and a dynamic window (dashed circle). A road network is modeled as a graph $G = \langle V, E \rangle$ where V is the set of all intersections with size $|V|$ and E is the set of all road segments with size $|E|$. Each road segment e may have such attributes as speed limit, length and road width. In practice, each road segment in the network has a direction, a one-way road or a two-way road. If direction is not considered, G is a non-directed (symmetric) graph. When direction is taken into account, G is a directed

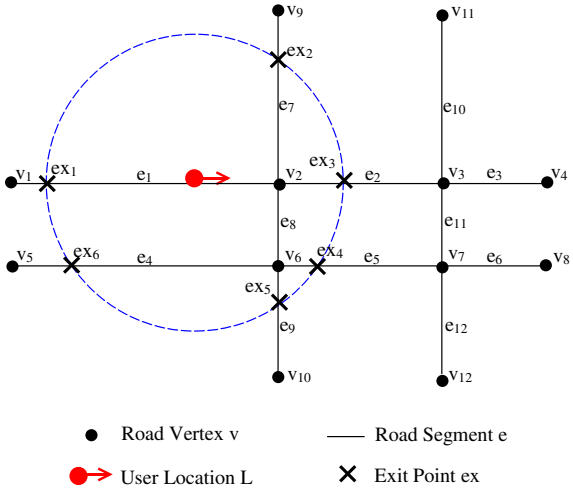

Fig. 1. Road network and dynamic window.

(non-symmetric) graph. In this paper, we assume non-directed road networks throughout. Because dynamic window is a clipping process, it clips the road network into a smaller portion. Only the road network contained in dynamic window is used for trajectory prediction. We consider all the vertices inside dynamic window and their immediate (neighbor) vertices as the vertices of the sub-network. The sub-network is a new graph $G' = \langle V', E' \rangle$ where V' is the set of all retrieved vertices with size $|V'|$ and E' is the set of all retrieved edges with size $|E'|$.

Dynamic window represents the largest geographic area within which user's activities are expected during the prediction period T. We define the intersections between dynamic widow and the road segments of the sub-network as *exit points* because they represent all possible exit locations where the user may leave the window (see Fig. 1). Exit points may be considered as destinations of predicted trajectories. The number of exit points can be inferred from the topological information in G'. Suppose the number of vertices inside a dynamic window is N, we define a new graph $G'' = \langle V'', E'' \rangle$ where V'' is the set of vertices (N) inside the window and E'' is the set of edges between those N vertices. For each vertex u inside the window, we know the number of edges linked to it or the number of neighbor vertices (num_neighbors(u)) by looking up the row of u in the adjacency matrix of G'. If we sum up the number of edges linked to each vertex in G'' as $\sum_{u(G'')}$num_neighbors(u), we get a duplicated count of those edges in G'', which is exactly twice the number of edges ($2|E''|$). Therefore, if we subtract the duplicated count of edges in G'' from $\sum_{u(G'')}$num_neighbors(u), we get the number of edges that intersect with the window, which is also the number of exit points (num_Exp):

$$\text{num\_Exp} = \sum_{u(G'')} \text{num\_neighbors}(u) - 2|E''| \tag{2}$$

For the example shown in Fig. 1, v_2 has four edges (e_1, e_2, e_8, e_7) linked to it and four neighbor vertices (v_1, v_3, v_6, v_9). Similarly, v_6 has four edges (e_8, e_4, e_5, e_9) and four neighbor vertices (v_2, v_5, v_7, v_{10}). G'' is a graph including only v_2 and v_6, and there is only one edge e_8 in G''. Therefore, the number of exit points is $(4 + 4) - 2 = 6$, as shown in Fig. 1.

4.2. Candidate trajectories

The topology information in G' is used to generate candidate trajectories. We use a graph search tree to identify candidate trajectories in a sub-network G', where the root is user's current location L, the intermediate nodes are vertices in G', the leaf nodes are exit points, and the links between the nodes are road segments. To reduce the search space, we assume the following rules:

1. User does not make a u-turn at an intersection.
2. User does not visit an intersection that has already been visited during a trip.

Fig. 2 depicts a search tree constructed for the dynamic window in Fig. 1. Clearly, certain branches are pruned by the rules of the search tree.

Fig. 3 illustrates the *TrajSearch* algorithm that identifies candidate trajectories. The algorithm takes the sub-network G', the exit points set ExpSet and the current location L. It first initializes the vertices in G' to an unprocessed status. Then it calls a Depth First Search (DFS) algorithm to detect candidate trajectories. DFS first takes the nearest vertex u_0 of L based on the moving direction of the user. It then changes the status of u_0 to "processed" and finds all the children (neighbor vertices) of it. Then the children become parents by calling DFS (see Line 6 in DFS), the recursive process continues until one exit point is detected (see Line 5 in DFS). DFS then changes the status of the input vertex to "stopped", meaning one trajectory has been detected. There is one loop in the algorithm which goes through all neighbor vertices of a particular vertex u. All other operations performed within the loop, such as changing status, have $O(1)$ time complexity. The loop through all neighbors of all vertices in G' takes $O\left(\sum_{u(G')}\text{num\_neighbors}(u)\right) = O(2|E'|)$. Therefore, the overall time complexity is $O(|E'|)$, where $|E'|$ is number of edges in G'.

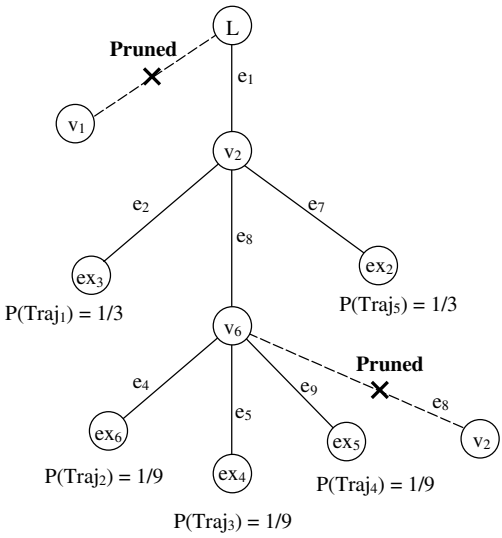

Fig. 2. Trajectory search tree.

```
TrajSearch(G′, L, ExpSet)          //G′ = <V′, E′>
                                   //L is current location
                                   //ExpSet is the set of exit points
1. for each u in V′ do
2.     status[u] = "unprocessed"
3. create list Vetx                //Vetx stores vertices of a trajectory
4. create list Traj                //Traj stores all trajectories
5. i = 0                           //i stores the number of trajectories
6. u0 = the nearest vertex of L
7. DFS(u0)                         //begin depth-first search from L
8. return Traj

DFS(u)
1. status[u]= "processed"
2. add u to Vetx
3. for each neighbor vertex v of u do
4.     e = the edge between u and v
5.     if status [v] != "processed" and
       e does not intersect with any exit point in ExpSet then
6.         DFS(v)
7. add Vetx to Traj[i] and increment i by one
8. status[u] = "stopped"
9. remove u from Vetx
10. return
```

Fig. 3. TrajSearch algorithm.

4.3. Trajectory prediction

After all candidate trajectories are identified, the task of trajectory prediction is to select the most likely one that a user will take. For the trajectory search tree shown in Fig. 2, when the user reaches the nearest intersection of current location, which is v_2, the system is faced with the problem of deciding which road segment the user will take: e_2, e_8 or e_7. If the system decides that e_8 is the road that the user will take, then the system is faced with the second intersection v_6 and again needs to decide which road segment the user will take: e_4, e_5 or e_9. The process of deciding which road segment the user will take at an intersection continues until an exit point or leaf node is reached.

To resolve uncertainty at intersections, there is a need to decision models that consider all road choices and choose one. Developing such decision models is one main problem in trajectory prediction. With an appropriate decision model, a trajectory using the trajectory search tree, such as the one shown in Fig. 2, can be determined. We present two decision models: a probability-based model and a learning-based model. Both models use long-term and short-term historical trajectories for prediction. The difference is that the probability-based model adopts probabilistic information for prediction while the learning-based model characterizes the features of long-term and short-term trajectories and is based on machine learning algorithms such as classification for prediction. The details of the two models are discussed in Sections 5 and 6, respectively.

5. Probability-based model

The probability-based model calculates the probability of taking each road segment at each intersection. The probability parameters are stored in a conditional probability

matrix for each intersection. For an intersection v with n road segments, the probability matrix is defined as

$$M(v,t_k)=\begin{bmatrix} P(R_1|R_1) & P(R_2|R_1) & \cdots & P(R_n|R_1) \\ P(R_1|R_2) & P(R_2|R_2) & \ldots & P(R_n|R_2) \\ \cdots & \cdots & \cdots & \cdots \\ P(R_1|R_n) & P(R_2|R_n) & \cdots & P(R_n|R_n) \end{bmatrix} \tag{3}$$

where t_k is current time, and $R_1, R_2, \ldots, R_n$ are road segments that share v. $P(R_1|R_1)$ is the probability of taking R_1 when the user is currently on R_1, and $P(R_2|R_1)$ is the probability of taking R_2 when the user is currently on R_1, and so on. Long-term historical trajectories are used to infer the number of times the user has traveled on each road segment and the trajectory choice at each intersection. The data is then used to calculate the parameters of the probability matrix. Therefore, $M(v,t_k)$ can be expressed as

$$M(v,t_k)=\begin{bmatrix} N_{1,1}/N_1 & N_{2,1}/N_1 & \cdots & N_{n,1}/N_1 \\ N_{1,2}/N_2 & N_{2,2}/N_2 & \cdots & N_{n,2}/N_2 \\ \cdots & \cdots & \cdots & \cdots \\ N_{1,n}/N_n & N_{2,n}/N_n & \cdots & N_{n,n}/N_n \end{bmatrix} \tag{4}$$

where N_i is the number of times the user has traveled on road segment R_i. $N_{j,i}$ is the number of times the user has taken road segment R_j when on road segment R_i. In other words, the following should hold:

$$\sum_{j=1}^{n} N_{j,i} = N_i \tag{5}$$

We call intersection v where a road choice needs to be predicted as a *decision point*. When a user is on a road segment R_i $(1 \leqslant i \leqslant n)$ connected to the decision point, the road segments R_j $(j \neq i)$ become road choices. We call R_j class labels and their probabilities $P(R_j|R_i)$ class priors. The probability-based model chooses the class label R_k with the largest prior $P(R_k|R_i)$. Therefore, the decision function of a class label $\hat{y}$ is

$$\hat{y} = \max_k (P(R_k|R_i)) \quad (1 \leqslant k \leqslant n) \tag{6}$$

With the decision model and a trajectory search tree, we are able to predict the user's road choice intersection by intersection until an exit point is reached. For the trajectory search tree shown in Fig. 2, the model must be applied twice at decision points v_2 and v_6 in order to predict a complete trajectory within the dynamic window.

6. Learning-based model

The probability-based model utilizes long-term historical trajectories for probability calculation and trajectory prediction. However, the prediction does not consider context information such as time of the day, travel distance and travel direction. In contrast, the learning-based model incorporates context information in trajectory prediction. One advantage of the learning-based model is that mobile contexts can be quantitatively measured and mapped into a feature space for prediction.

6.1. Context feature characterization

Mobile contexts vary significantly and may change dynamically during usage. We focus on mobile contexts that capture common characteristics of historical trajectories. Common trajectory features include distance, movement direction, time, and travel time (Schlieder & Werner, 2003). After feature characterization, both long-term and short-term historical trajectories are mapped into data points in a N-dimensional feature space, where N is the number of features. As in the probability-based model, the road segments R_i $(1 \leqslant i \leqslant n)$ connected to the decision point P are called class labels. For all the long-term historical trajectories that pass through P, the class label is known. While for short-term historical trajectories, the label is unknown and needs to be predicted. The unknown label represents the user's road choice, which is immediately connected to the short-term historical trajectory via the decision point P.

6.2. Multi-way classification

A machine learning algorithm, called multi-way classification, is used to predict the class label of a short-term trajectory. The idea is to divide the feature space into different portions, or classes, using long-term historical trajectory data points with known labels. Then a short-term historical trajectory data point, with an unknown label, will automatically be assigned a label depending on which portion of the space the point is located. The algorithm decomposes the classification problem into several binary classification tasks, and then combines the results of the sub-tasks into a hypothesized solution to the original problem. The number of binary classification tasks depends on the number of class label an intersection or decision point is assigned. We use logistic regression model for the binary classification task.

6.2.1. Logistic regression model

For binary classification, the logistic regression model uses discriminant functions

$$g_1(x) = g(w^{\mathrm{T}}x) \tag{7}$$

and

$$g_0(x) = 1 - g(w^{\mathrm{T}}x) \tag{8}$$

where x is the input with dimension d, w is the array of weights, and $g(z) = 1/(1 + \mathrm{e}^{-z})$ is a logistic function.

Let $\mu_i = p(y_i = 1|x_i, w) = g(z_i) = g(w^{\mathrm{T}}x)$. Then the likelihood of outputs is

$$L(D, w) = \prod_{i=1}^{n} P(y = y_i|x_i, w) = \prod_{i=1}^{n} \mu_i^{y_i}(1 - \mu_i)^{1-y_i} \tag{9}$$

The log-likelihood is

$$l(D, w) = \log \prod_{i=1}^{n} \mu_i^{y_i}(1 - \mu_i)^{1-y_i} = \sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i) \tag{10}$$

Logistic regression model tries to find weights that maximize the likelihood of outputs. The derivatives of the log-likelihood is

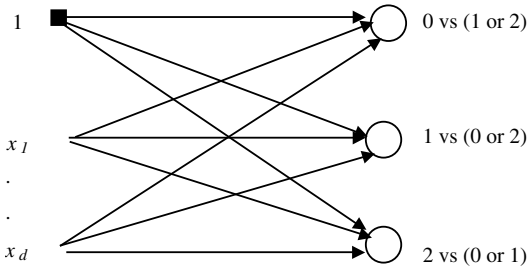

Fig. 4. Three-way classification.

$$-\frac{\partial}{\partial w_j} l(D, w) = \sum_{i=1}^{n} -x_{i,j}(y_i - g(z_i)) \tag{11}$$

where D is the set of examples and y_i is the true class label.

Let the derivatives be zeros, we get a non-linear function of w_j. The gradient solution is

$$w^{(k)} \leftarrow w^{(k-1)} + \alpha(k) \sum_{i=1}^{n} [y_i - f(w^{(k-1)}, x_i)] x_i \tag{12}$$

where $\alpha(k)$ is learning rate.

The decision function of predicted class label $\hat{y}$ is

$$\hat{y} = \begin{cases} 1, & g_1(x) \geqslant g_0(x) \\ 0, & g_1(x) < g_0(x) \end{cases} \tag{13}$$

6.2.2. Multi-way classification

In the multi-way classification algorithm, a binary classification using logistic regression model is performed on each pair of classes. Fig. 4 depicts the situation for a three-way classification, where the three class labels are denoted as "0", "1" and "2". Class "0" is distinguished from a combination of classes "1" and "2", class "1" is distinguished from a combination of classes "0" and "2", and class "2" is distinguished from a combination of classes "0" and "1".

This three-way classification generates three decision boundaries. Then the data point for the short-term trajectory is mapped into the feature space and assigned a class label. With the multi-way classification algorithm and a trajectory search tree, we are able to predict the user's road choice intersection by intersection until an exit point is reached. For the trajectory search tree shown in Fig. 2, the model must be applied twice for the decision point v_2 and the decision point v_6 in order to predict a trajectory.

7. Experimentation

In this section, we present an experimentation that was conducted to test the performances of the two prediction models. The focus of the experimentation was to measure the prediction accuracy at one decision point. The accuracy of predicting a complete trajectory within a dynamic window using a trajectory search tree can be calculated based on the accuracy at each decision point, however, its discussion is beyond the scope of this paper.

7.1. Trajectory dataset

Synthetic trajectory data sets are used for this experimentation. User's movement can be off-road or random walk. However, many studies have argued that mobile users, especially drivers, very often follow a network and use a fastest or shortest path to their destination (Brinkhoff, 2002). Fig. 5 shows a portion of the road network of downtown Pittsburgh area used in this experimentation. A total of 4624 fastest routes between every possible pair of intersections were computed. Table 1 shows a list of sample trajectories, where each trajectory is represented as a sequence of intersections or nodes.

Suppose the user is moving in the direction from "20" to "21", and "21" is the decision point. To predict the user's road choice at "21", only those trajectories that pass through "21" are selected. There are 552 such trajectories, as shown in Fig. 5. Because three road segments are connected to "21", the user will have three road choices. We denote the road segment $21 \rightarrow 25$ as class "0", the road segment $21 \rightarrow 45$ as class "1", and the road segment $21 \rightarrow 23$ as class "2". Then the problem becomes: Which road segment, "0", "1" or "2", will the user take?

7.2. Trajectory feature characterization

For a decision point P with n road segments, we characterize three features: distance L, movement direction θ and time-of-day T. Distance here refers to the trajectory length between the start point S_i and the decision point P. For movement direction, we introduce a *reference vector* which is a direction from P to the East and a *direction vector* from S_i to P. A polar coordinate is used for measuring movement direction which is the angle between the reference vector and the direction vector, see Fig. 6(a). Fig. 6(b) shows an example of calculating movement direction for the trajectory $0 \rightarrow 1 \rightarrow 22 \rightarrow 20 \rightarrow 21$, where the start point is "0" and the decision point is "21". The time-of-day feature refers

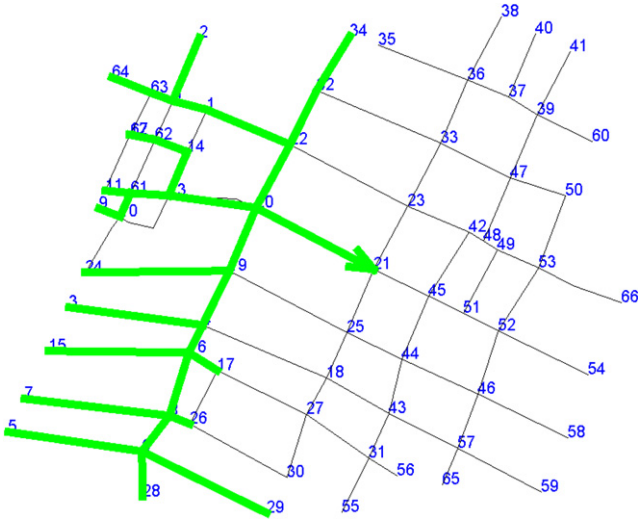

Fig. 5. Trajectories that pass through intersection "21".

Table 1
Sample simulated trajectory data

Trajectory ID	Nodes							
1	0	1	22	20	21	45	51	
2	4	18	25	21	45	51	52	
3	9	10	61	13	20	21	45	51
4	46	52	51	45	21	23	33	
5	66	53	52	51	45	21	25	18

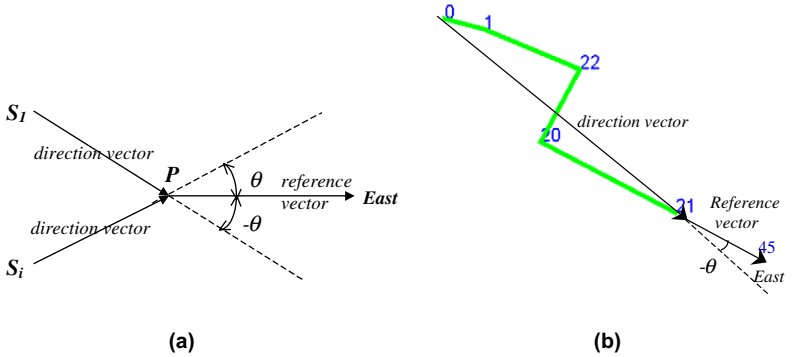

Fig. 6. Movement direction calculation.

to the time when mobile users pass through the decision point. It is generated as a Gaussian distribution of time.

L, θ and T are vectors of attribute values or inputs with different scales. They have to be normalized before they are passed to prediction algorithms. The following equations are used to calculate normalized inputs, L', θ', and T', where mean( ) is the mean of input and std( ) is the standard deviation of input:

$$L' = \frac{L - \text{mean}(L)}{\text{std}(L)} \tag{14}$$

$$\theta' = \frac{\theta - \text{mean}(\theta)}{\text{std}(\theta)} \tag{15}$$

$$T' = \frac{T - \text{mean}(T)}{\text{std}(T)} \tag{16}$$

We divide the normalized trajectories into a training set with 407 trajectories and a test set with 145 trajectories. The training set contains long-term historical trajectories whose class labels are known to the prediction models, while the test set contains short-term historical trajectories whose class labels need to be predicted by the models. Fig. 7 shows the 3D view of the training set.

7.3. Analysis of results

We applied the probability-based model and the multi-way logistic regression classification on the training set, and computed the misclassification errors for both the training set

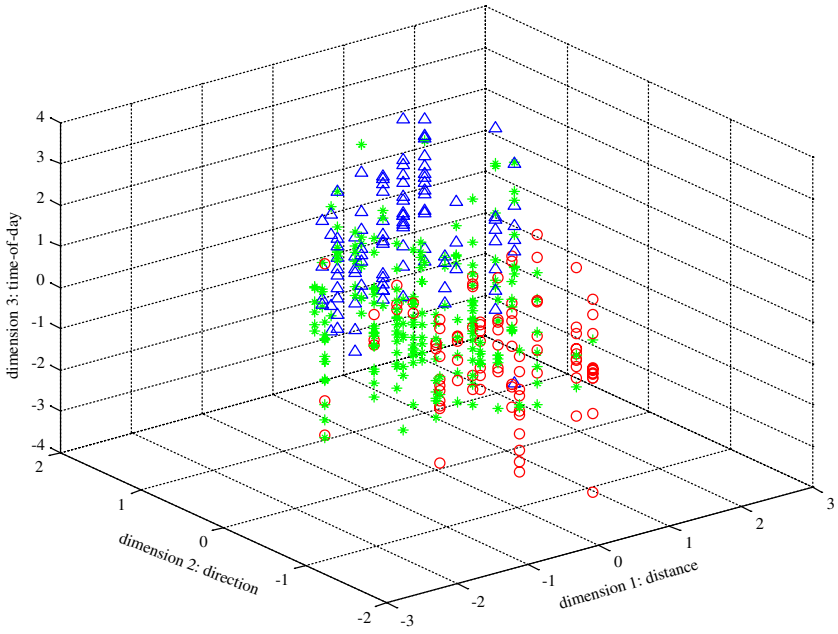

Fig. 7. Visualization of training set, where red "○" is class 0, green "∗" is class 1 and blue "△" is class 2. (For interpretation of color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Experimentation results

Technique	Training error	Test error
Random prediction	0.6682	0.6731
Probability-based prediction	0.4865	0.5517
Multi-way logistic regression	0.3145	0.3241

and test set. Also, we calculated misclassification errors of random prediction for comparison (see Table 2). It can be seen that both the probability-based model and the multi-way logistic regression have much better prediction performance than random prediction. Also, it is shown that the multi-way logistic regression based on feature characterization performs better than the probability-based model which does not consider trajectory features.

It should be noted that the results are dependent on the trajectory dataset used. There may exist such cases where the probability-based model outperforms the multi-way classification. However, the probability-based model only uses probabilistic information for prediction without considering context information. On the contrary, the learning-based model, such as the multi-way classification, is able to include any context information through feature characterization and selection, and proves to be a much more flexible prediction strategy.

8. Conclusions

Location-aware computing introduces new challenges and opportunities for location management in mobile computing. Many existing location management strategies in location-aware computing rely on system capability to periodically record current location information. A major shortcoming of this strategy is that it constraints location awareness to current location, hence impeding the possibility of inferring future activities. Trajectory prediction offers richer location and context information and facilitates adaptation to future locations. In this paper, we presented a process for trajectory prediction and discussed two prediction models: probability-based and learning-based. Both models use long-term and short-term historical trajectories for prediction. The major difference between the two models is that the probability-based model adopts probabilistic information for prediction while the learning-based model characterizes trajectory features and adopts machine learning algorithms for prediction. To realize the benefits of these models, an experiment was conducted. The results of this experiment reveal that both models perform much better than random prediction. Furthermore, the results of the experiment indicated that the learning-based model is more flexible than the probability-based model because it supports more context information which is needed for better prediction.

References

Aljadhai, A. R., & Znati, T. (2001). Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications, 19*, 1915–1930.

Blewitt, G., & Taylor, G. (2002). Mapping dilution of precision (MDOP) and map matched GPS. *International Journal of Geographical Information Science, 16*(1), 55–67.

Borriello, G., & Deshpande N. (2002). Location-aware computing: Creating innovative and profitable applications and services. *Intel Developer Update Magazine*.

Brinkhoff, T. (2002). A framework for generating network-based moving objects. *GeoInformatica, 6*(2), 153–180.

Das, S. K., & Sen, S. K. (1999). Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. *The Computer Journal, 42*(6), 473–486.

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing, 5*, 4–7.

Fritsch, D., & Volz, S. (2003). NEXUS – the mobile GIS-environment. In *Joint first workshop on mobile future and symposium on trends in communications* (pp. 1–4).

Indulska, J., & Sutton, P. (2003). Location management in pervasive systems. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers* (Vol. 21, pp. 143–151).

Jiang, B., & Yao, X. (2006). Location-based services and GIS in perspective. *Computers, Environment and Urban Systems*, this issue, doi:10.1016/j.compenvurbsys.2006.02.003.

Karimi, H. A., & Liu, X. (2003). A predictive location model for location-based services. In *Proceedings of the 11th ACM international symposium on advances in geographic information systems* (pp. 126–133). New Orleans, LA, USA.

Kyriakakos, M., Hadjiefthymiades, S., Frangiadakis, N., & Merakos, L. (2003). Multi-user driven path prediction algorithm for mobile computing. In *14th International workshop on database and expert systems applications (DEXA'03)* (pp. 191–195).

Li, C. (2006). User preferences, information transactions and location-based services: A study of urban pedestrian wayfinding. *Computers, Environment and Urban Systems*, this issue, doi:10.1016/j.compenvurbsys.2006.02.008.

Liu, T., Bahl, P., & Chlamtac, I. (1998). Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. Wireless access broadband networks [Special issue]. *Proceedings of the IEEE Journal on Special Areas in Communications, 16*(6), 922–936.

Miller, H. J. (2003). What about people in geographic information science? *Computers, Environment and Urban Systems, 27*(5), 447–453.

Miller, H. J., & Storm, J. D. (1996). Geographic information system design for network equilibrium-based travel demand models. *Transportation Research Part C: Emerging Technologies, 4*(6), 373–389.

Schlieder, C., & Werner, A. (2003). Interpretation of intentional behavior in spatial partonomies. In C. Freksa, W. Brauer, C. Habel, & K. F. Wender (Eds.), *Spatial cognition III: Routes and navigation, human memory and learning, spatial representation and spatial learning* (pp. 401–414). Berlin: Springer.

Shah, S. H. & Nahrstedt, K. (2002). Predictive location-based QoS routing in mobile ad hoc networks. In *Proceedings of IEEE International Conference on Communications (ICC 2002)* (pp. 1022–1027), New York.

Smyth, S. (2000). Mobile geographic information services: Turning GIS inside out. Microsoft Report.

Tan, K. L., & Wolfson, O. (2003). Guest editors' introduction to MONET special issue on mobile and wireless data management. *Mobile Networks and Applications, 8*(4), 315–316.

Taylor, G., Brunsdon, C., Li, J., Olden, A., Steup, D., & Winter, M. (2006). GPS accuracy estimation using map matching techniques: applied to vehicle positioning and odometer calibration. *Computers, Environment and Urban Systems*, this issue, doi:10.1016/j.compenvurbsys.2006.02.006.

Wolfson, O. et al. (1999). Updating and querying databases that track mobile units. Mobile data management and applications [Special issue]. *Distributed and Parallel Databases Journal, 7*(3), 257–287.

Yeung, K., & Yum, T. S. (1995). Cell group decoupling analysis of a channel borrowing based dynamic channel assignment strategy in linear radio systems. *IEEE Transactions on Communications, 43*(2–4), 1289–1292.

Zhao, Y. (2000). Mobile phone location determination and its impact on intelligent transportation systems. *IEEE Transaction on Intelligent Transportation Systems, 1*(1), 55–64.

Zipf, A., & Jost, M. (2006). Implementing adaptive mobile GI services based on ontologies: Examples from pedestrian navigation support. *Computers, Environment and Urban Systems*, this issue, doi:10.1016/j.compenvurbsys.2006.02.005.