Predicting User's Movement with a Combination of Self-Organizing Map and Markov Model

Sang-Jun Han and Sung-Bae Cho

Department of Computer Science
Yonsei University
134 Shinchon-dong, Seodaemun-ku, Seoul 120-749, Korea
{sjhan, sbcho}@sclab.yonsei.ac.kr

Abstract. In the development of location-based services, various location-sensing techniques and experimental/commercial services have been used. We propose a novel method of predicting the user's future movements in order to develop advanced location-based services. The user's movement trajectory is modeled using a combination of recurrent self-organizing maps (RSOM) and the Markov model. Future movement is predicted based on past movement trajectories. To verify the proposed method, a GPS dataset was collected on the Yonsei University campus. The results were promising enough to confirm that the application works flexibly even in ambiguous situations.

1 Introduction

Location-based services (LBS) have been a hot topic in the field of wireless and mobile communication devices. One reason for this is because mobile device users want to be able to access information and services specific to their location. As location-sensing and wireless network technologies have developed, various kinds of LBS have emerged. In the field of context-awareness and artificial intelligence, researchers have attempted to develop novel smart location-based applications (see the Related Works section). Prediction of future movement is one key aspect of the next generation of LBS. Current LBS applications attempt to meet the user's present needs. But if the application can also predict where the user will be, it will be able to provide services the user may need in the future, as well as the services they need at present.

In previous research on movement prediction, the method of modeling the transitions between locations was used (Figure 1 (b)). The Markov model was used to represent the transitions between locations and future movement was predicted based on the highest probability transition from the current location [1]. However, this model is inflexible because it takes into account only the current location or place of the user. For example, if the transition from place A to place B has the highest probability and the transition to place C has the second highest probability, the application will always say "you will go to place C" to the user, even if this is untrue. That is to say, the model cannot cope with ambiguous situations.

S. Kollias et al. (Eds.): ICANN 2006, Part II, LNCS 4132, pp. 884 – 893, 2006.

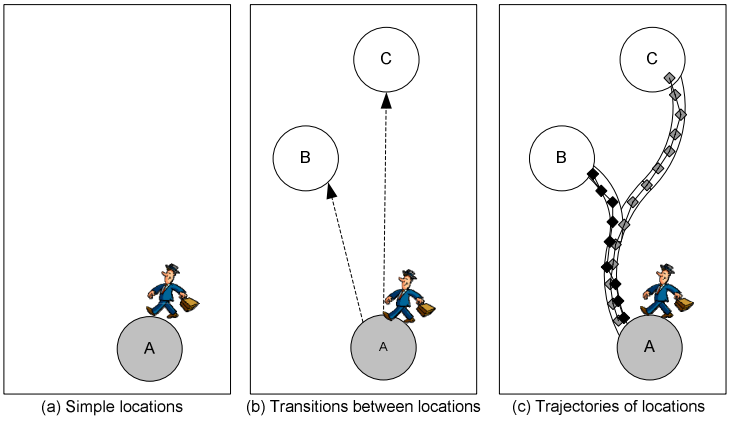

Fig. 1. Comparison of location systems

To achieve more intelligent and flexible predictions, we propose a trajectory-based approach (Figure 1(c)). The main idea is to model the trajectories of locations for movement prediction so that the predictions are based on past trajectories, not the current location. A trajectory-based approach enables the system to distinguish whether the user will head for place B or for place C. It can then react adaptively to the user's destination and future movement can be predicted more flexibly and accurately than when using the transition-based approach. In addition, the system will act differently according to the route taken by the user. For example, different services can be offered to users when they travel along the highway and when they travel along residential roads.

2 Related Works

Many commercial location-based services are already widely used. Wireless service providers offer customer-based plans which assign different rates to calls made from home or from the office [2]. Major credit card companies have created wireless ATM-locator services. AT&T provides `find people nearby' services which allow users to locate friends and family members.

A location-aware event planner designed by Z. Pousman *et al.* integrates a friend finder application which displays locations on a given campus map [3]. The user can organize social events in contextually-enhanced ways. The system also includes privacy management functionality which enables the user to manage visibility to others. Location-based games like `Can You See Me Now' of the University of Nottingham and `Human Pacman' of the National University of Singapore provide novel gaming environments which are enhanced by physical locations [4][5].

Many researchers have attempted to go beyond present-day location systems by extracting high-level information from raw location data. D. Ashbrook *et al.* proposed a method for predicting future movements which used a modification of the k-means

clustering algorithm and the Markov chain model [6]. D.J. Patterson *et al.* proposed a method to be used in the current transportation mode which used a dynamic Bayesian network model [7]. Domain knowledge was incorporated into the Bayesian network model and the parameters of their network were learned using an expectation-maximization (EM) algorithm. In their experimental results, the Bayesian network model outperformed both the decision tree and the Bayesian network model without any domain knowledge. Sto(ry)chastics by F. Sparacino estimated the type of museum visitor for user adaptive storytelling in museums [8]. The visitor's location can be tracked by infrared beacons and a Bayesian network model that estimates the visitor's type as greedy, selective, or busy from the user's location and time spent at each location. Visitors are able to see different explanations about the same exhibits according to their visiting habits.

3 Learning and Predicting Future Movement

Figure 2 shows our movement prediction framework. First, we discover patterns of us er movement by clustering the location dataset (Step 1). This set comprises sequences of GPS records. A sequence represents a movement between places. Then, the model s are built (Step 2). User-preferred services are paired with related movement pattern models. These form user profiles. While the user travels, current movement is compar ed with the movement models (Step 3). Step 3 is repeated whenever the user travels s ome distance. If a movement model is significantly similar to the current movement, t he system will predict that the user will travel along the route of that model (Step 4). User-preferred services related to the selected movement pattern are offered to the use r immediately after the movement prediction.

To develop an easily adaptable system, it is necessary to automatically find what kinds of movements exist in a person's life with minimum pre-knowledge. The self-organized learning approach is suited to this purpose. We employed self-organizing maps to discover significant patterns of user movements from the location dataset.

The benefit of SOM is that it can provide a good approximation of the original input space. A SOM projects the continuous input space to the discrete output space. The output space of a SOM can be viewed as a smaller set of prototypes which store a large set of input vectors. This property helps simplify the problem. The sequences of raw GPS records can be transformed to the sequences of finite units by projecting them onto the SOM output space. We can state the transformed movement data as the state transition sequence of the user's movement. Thus, the user's movement patterns can be modeled more effectively by learning the transition sequences of finite states rather than learning the sequences of the vectors of two floating-point numbers.

A standard SOM is not able to discover significant movement paths because it cannot process temporal sequence data. In order to distinguish different movement patterns, temporal data processing is needed. To cope with this problem, the recurrent SOM (RSOM) is introduced. The RSOM processes the temporal sequence data by maintaining contextual information between the input samples. Even if the GPS data is captured at the same place, the RSOM projects the data into different output units

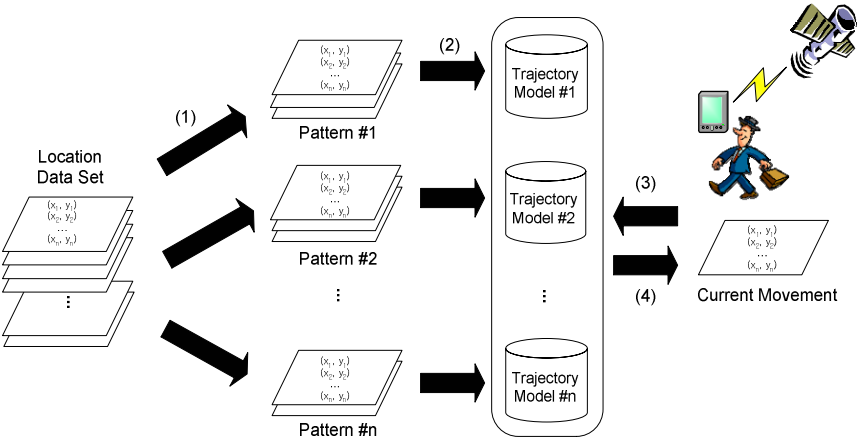

Fig. 2. Movement prediction framework

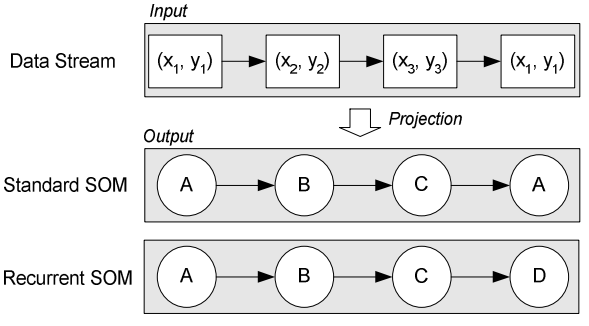

Fig. 3. Differences between the standard SOM and the recurrent SOM

with respect to past movement trajectories. Figure 3 illustrates the difference between the two kinds of SOM. When we project a two-dimensional vector sequence (of which the first and the last vectors are the same) into the two kinds of SOM, they yield different outputs. The standard SOM plots the first and last vectors into an identical output unit without regard to past input. However, in the RSOM, the last vector is mapped into another output unit. Hence, different movement paths and movement trajectories can be distinguished with the last output unit.

3.1 Discovering Patterns of User Movement

The SOM is a representative unsupervised neural network used to solve clustering an d vector quantization problems. The principal goal is to transform an incoming input pattern (of arbitrary dimensions) into a one or two-dimensional discrete map [9]. The output map L consists of a rectangular or hexagonal lattice of $n(i)$ units. The algorith m for training the SOM involves four essential processes: initialization, competition,

cooperation, and adaptation. These are summarized as follows. Here, $b(x)$ is the best matching unit to the input vector, *h* means the neighborhood function and η is the lea rning rate.

1. Initialize the codebook vectors $w_i(0)$.
2. Compute difference and select the best matching unit

$$b(n) = \arg\min \left\| x(n) - w_j(n) \right\| \tag{1}$$

3. Update the codebook vectors

$$w_i(n+1) = w_i(n) + \eta(n) h_{b(n),i}(n)(x(n) - w_i(n)) \tag{2}$$

4. Repeat from 2 to 3 until the stop condition satisfies

Although the RSOM is specialized for temporal sequence processing, it inherits the original properties of the SOM [10]. The differences between the RSOM and the standard SOM are as follows. The RSOM allows the storing of temporal context from consecutive input vectors by putting the leaky integrator into the difference formula of the competition step.

$$y_i(n) = (1-\alpha) y_i(n-1) + \alpha(x(n) - w_i(n)) \tag{3}$$

where α is the leaking coefficient, $y_i(n)$ is the leaked difference vector at step *n* an d $x(n)$ is the input vector at step *n*. The best matching neuron criterion and codebook vector update rules are the same as the standard SOM. The best matching unit (BMU) at time step *n*, $b(n)$ is the unit with the minimum difference.

$$b(n) = \arg\min_i \left\| y_i(n) \right\| \tag{4}$$

The *i*th codebook vector at step *n* , $w_i(n)$ is updated as follows:

$$w_i(n+1) = w_i(n) + \eta(n) h_{b(n),i}(n) y_i(n) \tag{5}$$

The difference vectors are reset to zero after learning each input sequence and the alg orithm is repeated with the next input.

In this problem, the input vector $x(n)$ is a GPS record captured at time *n*, which is a two-dimensional vector composed of the user's specific longitude and latitude. A new GPS record is captured once or twice a second even if the user does not move. The meaningless data in the raw GPS records has to be filtered. Only after the user travels some distance, a new GPS record can be captured and used for training and predicting. Raw GPS data is never 100% accurate. There are many methods to allow for this margin of error. In this research, however, no error correction method was employed and the raw GPS records were directly used to focus on movement prediction.

The procedure of pattern discovery is as follows. First, we train the RSOM with the trajectory dataset obtained from the user. A trajectory dataset

$X_k = \{x(0), x(1), \ldots, x(N)\}$ is a sequence of GPS records captured during a movement *k*. Then, the trajectory dataset is transformed into the sequence of BMU $B_k = \{b(0), b(1), \ldots, b(N)\}$ by projecting it to the trained RSOM to group the similar trajectories. The transformed trajectory dataset is clustered according to the last BMU. A set of the transferred trajectory data that corresponds to the *i*th output unit of the RSOM, $C_i = \{B_1, B_2, \ldots, B_L\}$ represents a discovered pattern of user movement.

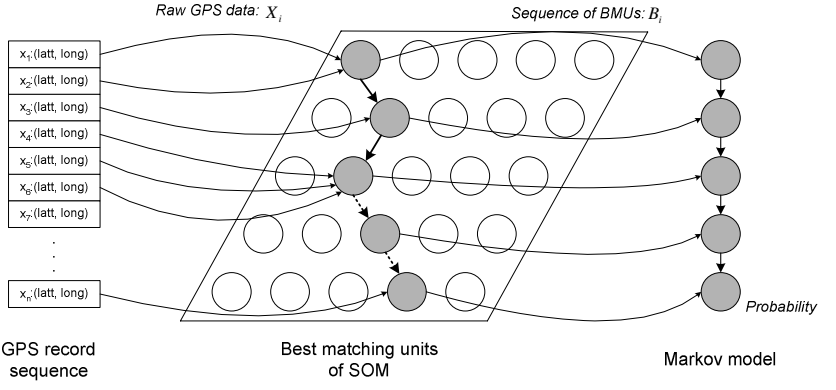

Fig. 4. Combination of RSOM and Markov model

3.2 Building Trajectory Models

A Markov model is a stochastic process based on the Markov assumption, under which the probability of a certain observation only depends on the observation that directly precedes it [11].The trajectory models are built using the first-order Markov models. A Markov model learns the sequences of the best matching units rather than those of the raw GPS data. Changes to the best matching units during the processing sequence can be considered as changes of state because the SOM approximates the input space. A trajectory model M_i is learned with a transformed trajectory dataset C_i. Figure 4 illustrates the combination of the RSOM and Markov model.

3.3 Predicting Future Movements

Figure 5 presents an algorithm which outlines the movement prediction phase. When a new GPS record is captured by the GPS receiver, the traveling distance after the last GPS record is calculated. If this is less than the minimum distance, the record is ignored. If not, it is inputted into the RSOM to get the BMU of the current GPS record. The new sequence of BMU is made by concatenating the newly obtained BMU $b(N)$ and the previously obtained BMUs $b(0), b(1), \ldots, b(N-1)$. Then, the BMU sequence is evaluated with the trajectory models. The state *i* in the Markov model corresponds to the *i*th output unit in the SOM because the sequences of the BMUs are used as

inputs. Hence, the probability that a sequence of the BMUs $B_k = \{b(0), b(1), \ldots, b(N)\}$ will occur from a given trajectory model can be computed using the following equation:

$$P(b(0), b(1), \ldots, b(N) \mid M_i) = q_{b(0)} \prod_{j=2}^{T} P_{b(t-1)b(t)} \tag{6}$$

The higher the probability of the trajectory model, the more likely the user moves similarly to the corresponding trajectory. The simplest way of selecting the most likely movement pattern is applying a threshold to the probability of the local model and selecting the local model whose probability exceeds the predefined threshold. However, this method suffers from a lack of flexibility because the level of probability varies according to the length of the movement. As the user moves, the overall level of probability decreases because the longer the user moves, the more the state transition probability of the Markov model is multiplied. The decision boundary has to vary according to the movement pattern.

```
input: the trained RSOM and the trajectory models
output: future movement pattern
begin
while (end-of-travel is true) do
  record = get-a-new-GPS-record();
  distance = get-traveling-distance(record);
  if (distance is less than minimum-distance)
   continue;
  end if
  bmu = get-the-best-matching-unit (RSOM, record);
  push-back (sequence, bmu);
  for (each trajectory model) do
    model.probability = evaluate-BMU-sequence(sequence, model);
  end for
  for (each trajectory model) do
    model.significance = compute-significance();
  end for
  max-significance = get-maximum-significance()
  if (max-significance exceeds  threshold)
    return model.pattern-number;
  end if
end while
```

Fig. 5. Outline of the movement prediction algorithm

Therefore, the method based on the relative significance of the trajectory model is employed instead of using the probability of the trajectory model directly. We select the outstandingly probable local model. The significance of the trajectory model is computed using the following equation:

$$significance(M_j) = p(B_k \mid M_j) - \sum_{i=1 st. k=I}^{I} \frac{P(B_k \mid M_j)}{I-1} \tag{7}$$

The significance of the trajectory model is defined as the difference between the probability that the current BMU sequence is generated from one model and the mean of the probabilities from the others. If there is the trajectory model with the significance exceeding the predefined threshold, we predict that the user will travel along the corresponding trajectory. In defining the threshold, the trade off between speed and accuracy has to be considered. The lower the threshold, the earlier we can predict the user's movement. If the threshold is set too high, the prediction will be made later with a relatively low risk of false prediction.

Table 1. User's movements in GPS data

Number	Starting Location	Ending Location	Count
1	Main Gate	Engineering Hall I	13
2	Engineering Hall I	College of Liberal Arts II	12
3	College of Liberal Arts II	Auditorium	13
4	Auditorium	College of Social Science	13
5	College of Social Science	Engineering Hall III	13
6	Engineering Hall III	Student Union	12
7	Student Union	Engineering Hall III	13
8	Engineering Hall III	Central Library	13
9	Central Library	College of Liberal Arts I	12
10	College of Liberal Arts I	Main Gate	12

4 Experiments

To test the proposed method, we collected a GPS dataset based on the actual campus life of Yonsei University students. The average student usually moves along 9 buildings for attending a lecture, having lunch, studying and participating in club activities along 10 kinds of paths. Four students walked along these predefined paths, each holding a GPS-enabled handheld computer. Each movement was discriminated by using the loss of the GPS signal and each trajectory was labeled according to its starting location and ending location. 130 trajectory datasets were collected in total, 13 sets for each class. Each trajectory dataset consisted of sequences of two dimensional vectors (longitude and latitude). However, four trajectory datasets were excluded from the experiments due to recording problems in the GPS receivers. Table 1 presents the description of each movement pattern. Due to GPS signal errors, the collected data could differ slightly from the real moving paths. In our experiments, an 8x8 map was used. The initial learning rate was 0.03 and the initial neighborhood radius was 4. The training algorithm was repeated 5000 times.

Prediction performance was evaluated using cross-validation because the size of the dataset was not large. First, the dataset was divided into 13 subsets. 9 subsets contained all kinds of classes and one class was omitted from the 4 subsets. We then chose one subset as the test dataset and the remaining 12 subsets were used for

training. Training and testing were repeated 13 times while changing the test subset. We repeated this cross-validation procedure ten times in order to evaluate the performance accurately. The results of the prediction experiments are given in Table 2 as a confusion matrix.

Table 2. Confusion matrix

		Predicted											
		1	2	3	4	5	6	7	8	9	10	Miss	Accuracy
Actual	1	**41**	0	0	0	0	0	0	0	0	0	89	0.32
	2	10	**110**	0	0	0	0	0	0	0	0	0	0.92
	3	0	0	**116**	0	0	0	0	0	0	0	14	0.89
	4	0	0	0	**130**	0	0	0	0	0	0	0	1.00
	5	0	0	0	0	**129**	0	1	0	0	0	0	0.99
	6	0	0	0	0	0	**88**	0	32	0	0	0	0.73
	7	0	9	0	0	0	0	**121**	0	0	0	0	0.93
	8	0	0	0	0	0	20	0	**110**	0	0	0	0.85
	9	0	0	0	0	4	0	0	0	**116**	0	0	0.97
	10	0	11	0	0	0	0	0	0	0	**99**	0	0.82

The 'Miss' column shows data which was not predicted because the significance did not exceed the threshold until the end of the movement. The prediction accuracy of movement path 1 is the lowest because it shows the most misses. One possible reason for this is because there was not enough time to exceed the threshold because the main gate and engineering hall I are so close to each other. However, besides the misses, no errors in prediction occurred. The lower accuracy of the path 1 reduces the average accuracy (0.84%). However, when the results from path 1 are removed, performance is acceptable. The average accuracy when excluding movement 1 is 0.9. In ambiguous situations (movements 6 and 8), the accuracies are 0.73 and 0.85, respectively. All errors in predicting paths 6 and 8 are due to the confounding of the two paths. The average travel time was 4 minutes 37 seconds and the average time elapsed until prediction was 1 minute 21 seconds. This result indicates that we can predict the user's future movement path before the user arrives at the destination.

5 Conclusion

In this paper, a novel method for learning user's movement patterns and predicting fut ure movements is presented. Our trajectory-based movement prediction method can le ad to more intelligent and proactive location-based services. In the future, we plan to i ncorporate more contexts into the trajectory models. If additional context information such as time of day, transportation mode and current activities are used, needs could b e estimated more accurately. Especially, information about the time of day may be a k ey factor when improving accuracy because many people travel along their regular ro utes during specific time periods.

Acknowledgement

This research was supported by Brain Science and Engineering Research Program sponsored by Korean Ministry of Commerce, Industry and Energy.

References

1. Ashbrook, D. and Starner, T., "Learning Significant Locations and Predicting User Movement with GPS," *Proceedings of IEEE Sixth International Symposium on Wearable Computing,* Seattle, WA. October 2002.
2. Stilp, L., "Carrier and End-User Applications for Wireless Location Systems," *Proceedings of SPIE*, vol. 2602, pp. 119-126, 1996.
3. Pousman, Z., Iachello, G., Fithian, R., Moghazy, J., and Stasko, J., "Design Iterations for a Location-Aware Event Planner," *Personal and Ubiquitous Computing*, vol. 8, no. 2, pp. 117-225, 2004.
4. Benford S., Anastasi R., Flintham M., Drozd A., Crabtree A., Greenhalgh C., Tandavanitj N., Adams, M., Row-Farr J., "Coping with uncertainty in a location-based game," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 34-41, 2003.
5. Cheok, A.D., Goh, K.H., Liu, W., Farbiz, F., Fong, S.W, Teo, S.L., Li, Y. and Yang, X., "Human Pacman: A Mobile, Wide-area Entertainment System based on Physical, Social, and Ubiquitous Computing," *Personal and Ubiquitous Computing*, vol. 8, no. 2, pp. 71-81, 2004.
6. Ashbrook, D. and Starner, T., "Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users," *Personal and Ubiquitous Computing*, vol. 7, no. 5, pp. 275–286, 2003.
7. Patterson, D., Liao, L., Fox, D., and Kautz, H., "Inferring High-Level Behavior from Low-Level Sensors," *Proceedings of the Fifth International Conference on Ubiquitous Computing*, pp. 73-89, Seattle, WA, October, 2003.
8. Sparacino, F., "Sto(ry)chastics: A Bayesian Network Architecture for User Modeling and Computational Storytelling for Interactive Spaces," *Proceedings of the Fifth International Conference on Ubiquitous Computing*, pp. 54-72, Seattle, WA, October 2003.
9. Haykin, S., *Neural Networks: A Comprehensive Foundation*, Second Ed, Prentice Hall, 1999.
10. Koskela, T., Varsta, M., Heikkonen, J., and Kaski, K., "Temporal Sequence Processing using Recurrent SOM," *Proceedings of Second International Conference on Knowledge-Based Intelligent Engineering Systems*, vol. 1, pp. 290-297, Adelaide, Australia, April 1998.
11. Winston, W.L., *Operations Research: Applications and Algorithms*, Belmont, CA: Duxbury, 1994.