Predicting the Location of Mobile Users: A Machine Learning Approach

Theodoros Anagnostopoulos
Pervasive Computing Research Group, Communication Networks Laboratory, Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilissia, Athens 15784, Greece, tel: +302107275409
thanag@di.uoa.gr

Christos Anagnostopoulos
Pervasive Computing Research Group, Communication Networks Laboratory, Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilissia, Athens 15784, Greece, tel: +302107275409
bleu@di.uoa.gr

Stathes Hadjiefthymiades
Pervasive Computing Research Group, Communication Networks Laboratory, Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilissia, Athens 15784, Greece, tel: +302107275409
shadj@di.uoa.gr

Miltos Kyriakakos
Pervasive Computing Research Group, Communication Networks Laboratory, Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilissia, Athens 15784, Greece, tel: +302107275409
miltos@di.uoa.gr

Alexandros Kalousis
Artificial Intelligence Laboratory, Department of Computer Science, University of Geneva, Uni-Dufour, Geneva 1211, Switzerland, tel: +41223797630
Alexandros.Kalousis@cui.unige.ch

ABSTRACT
Mobile context-aware applications experience a constantly changing environment with increased dynamicity. In order to work efficiently, the location of mobile users needs to be predicted and properly exploited by mobile applications. We propose a spatial context model, which deals with the location prediction of mobile users. Such model is used for the classification of the users' trajectories through Machine Learning (ML) algorithms. Predicting spatial context is treated through supervised learning. We evaluate our model in terms of prediction accuracy w.r.t. specific prediction parameters. The proposed model is also compared with other ML algorithms for location prediction. Our findings are very promising for the efficient operation of mobile context-aware applications.

Keywords
context-awareness, machine learning, location prediction, spatial context representation.

1. INTRODUCTION

During the recent past, we have witnessed an impressive growth in the domains of wireless-mobile telecommunications and context-aware services. The general framework that evolved from distributed systems, and is currently described under the term *mobile computing* has attracted the extensive interest of academia and the industry. New protocols, schemes, applications and services are developed and applied in real-life situations. This escalating development has also triggered discussions on other, more enhanced paradigms like *pervasive computing*.

ICPS'09, July 13–17, 2009, London, United Kingdom.

In order to render mobile applications intelligent enough to support modern users everywhere / anytime and materialize the so-called ambient intelligence, information on the present *context* of the user has to be captured and processed accordingly. Contextual information may refer to the user's position, time, physical properties like temperature or other general parameters (e.g., the specific devices that the user carries). The efficient management of context requires detailed and thorough data modeling along with specific processing, classification, inference and prediction capabilities.

A well-known definition of context is the following: "*context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the integration between a user and an application, including the user and the application themselves*" [26]. Context refers to the current values of specific parameters that represent the activity of an entity.

Context-awareness allows an entity to adapt to its environment, thus, offering a number of advantages and possibilities for new applications. One of the more intuitive capabilities of mobile context-aware applications is their *pro-activity*. Predicting user actions and contextual parameters enables the development of new, advanced applications.

Moreover, predicting the location of a mobile user is an inherently interesting and challenging problem. Location prediction has received increased attention driven by applications in location management, call admission control, smooth handoffs, and resource reservation for improved quality of service. Spatial information prediction can be used to facilitate the provision of advanced location-based services by preparing and

feeding them with the appropriate contextual information (in advance). Pro-active context-aware applications address context pre-evaluation / pre-determination introducing innovative proactive services, e.g., alerts related to traffic conditions, content pre-fetching and triggering actuation rules in advance (download company documents while entering downtown).

Context pre-evaluation may match with information classification and prediction, in the sense that, the values of certain contextual parameters are estimated /evaluated, clustered and classified in advance. The concept of predicting context through ML algorithms and techniques is quite novel. *ML refers to the study of algorithms that improve automatically through experience*. A wide spectrum of applications based on ML algorithms relates to search engines, medical diagnosis, object recognition in computer vision and natural language processing. ML tasks can be addressed through: *supervised learning*, (e.g., classification and regression), and *unsupervised learning*, (e.g., clustering and rules extraction). In this paper, we adopt classification in order to predict spatial context - location. The (training) *examples* used for the supervised learning task are vectors of attribute-value pairs. Each example is assigned to a specific *class* from a fixed set of classes (e.g., symbolic locations). The goal of the classification task is to develop a model from a set of examples capable of predicting the class of a given *unseen* example.

A context model is proposed in order to support location prediction for mobile users. Such model predicts, with a certain accuracy level (portion of successful predictions), the future position (cell) of a mobile user in a cellular environment and can be used for the pro-active management of network resources (e.g., packets, proxy-cache content). The model can be trained using a variety of ML classification algorithms. In this paper we experiment with such algorithms, in order to perform location prediction. We can divide the assessed algorithms into two broad categories according to the way they construct their classification models: standard classification algorithms, (e.g., decision trees, *k*-nearest neighbors, support vector machines), and ensemble learning algorithms that learn to combine classification models constructed by base learning algorithms in order to improve predictive performance. Typical examples of this last category are the boosting and voting schemes. Finally, the performance of the ML models is compared with that of the most cited location prediction models namely the LeZi-Update [12] and Mobile Motion Prediction (MMP) [13] algorithms.

This paper is organized as follows: Section 2 reports certain ML schemes used for classification while in Section 3 the context representation model taking into account the user mobility pattern is proposed. In Section 4, we evaluate the discussed model and we compare it with the certain ML schemes. Section 5 reports prior work on that research area and, finally, Section 6 concludes the paper.

2. CLASSIFICATION IN MACHINE LEARNING

Classification is the task of learning to categorize (predict) an unseen example to a discrete class value. An *example* is a (m+1)-dimensional vector $\boldsymbol{e}$ of m attributes e_i, $i = 1, \ldots, m$ and a class attribute e_{m+1}, that is $\boldsymbol{e} = [e_1, \ldots, e_m, e_{m+1}]$. The e_i attributes determine the value of the class attribute e_{m+1} of $\boldsymbol{e}$. An e_i attribute assumes values from the corresponding domain $Dom(e_i)$. The input of a classification algorithm is a set E of s example vectors, $E = \{\boldsymbol{e}_i, i = 1, \ldots, s\}$, and the output is a classification model M(E). Such model is capable of predicting / estimating the value of the class attribute of an unseen and yet unlabeled (m+1)-dimensional example vector $\boldsymbol{q}$ based only on the values of its q_i attributes with $Dom(e_i) = Dom(q_i)$, $\forall i$.

The performance of a classification algorithm is estimated on the basis of the quality of predictions delivered by the trained models. In order to estimate the classification performance, a test phase is required. In the test phase we employ a test set V that was not used in the training phase. A classification model, M(E), is established through the training set, E, as a result of the learning phase. The produced model M(E) is then applied on the test set V and the correctness of its predictions is assessed and quantified. Prediction accuracy ε is a quantitative measure for the classification performance. It refers to the proportion of the correctly predicted examples $C \subset V$ out of V, i.e., the fraction $\varepsilon = |C| / |V|$, where $|C|$ denotes the cardinality of C. One method to estimate the prediction accuracy of a classification algorithm is a re-sampling method called *cross-validation* [1, 2]. In n-fold cross-validation, the training set E is divided into n subsets of equal size. A different model M is trained n times, each time setting aside one of the subsets which will be used as the test set on which the predictive accuracy will be computed. The final accuracy estimation is the averaged accuracy over the n

different repetitions of the training and testing phases.

A number of different learning paradigms have been developed over the years for classification. Since there is no single classification algorithm that is better than all the others irrespective of the application domain, each time face a new classification problem we have to assess anew the suitability of the algorithms. We have experimented with several classification algorithms trying to cover a range, as broad as possible, of different learning paradigms. We found a voting ensemble-learner to be the best for our application problem according to the estimated predictive performances. In the next paragraphs we give a brief description of the different types of learning paradigms that we are going to consider in this paper.

2.1 Bayesian learning

Bayesian classification algorithms are statistical learning algorithms based on the Bayes theorem. The Naïve Bayes algorithm (the simplest Bayesian classifier) [5] assumes that, the effect of the value of an attribute on the class attribute is independent of the values of the other attributes given the value of the class attribute (*conditional independence*).

2.2 Decision Tree learning

A decision tree consists of decision nodes and leaves. A leave is usually associated with a single class; the majority class of the training examples that arrive to that leave. Splits are introduced in the building of the tree according to the outcome of a function f (information gain ratio). When examples are classified a function is used in each split to determine the downstream path to be followed [6]. An indicative decision tree-based algorithm is the C4.5 classifier [7].

2.3 Rule-Induction learning

Rule-induction performs a depth-first search in a graph $G(V,E)$ generating one path. V is a set of attributes and E is the set of edges denoting dependencies among attributes. Such path represented as a *classification rule* [8] is a conjunction of conditions with discrete or numeric attributes. A rule is said to *cover* an example if the later fulfills all the conditions of that rule. A representative rule-induction algorithm is the RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [9].

2.4 Instance-based learning

An instance-based algorithm uses a distance function $||\boldsymbol{e} - \boldsymbol{q}||$ in order to determine which example vector $\boldsymbol{e}$ of the training set is closest to an un-*classified* example $\boldsymbol{q}$ (or *instance*). Once the nearest example vector has been determined, its class label is selected as the class label for $\boldsymbol{q}$. A representative instance-based algorithm is the k-nearest neighbor classifier.

2.5 Ensemble-Learning Algorithms

Ensemble-learning algorithms combine a number of base classification models, classifiers, produced by different learning algorithms or by different training sets, in order to achieve better classification performance than their constituents. Base models can be combined in different ways in order to generate ensemble-learning algorithms. Popular ensemble-learning algorithms are the following:

2.5.1 Voting

Each base classifier predicts, *votes*, a class. The final class is that, which assumes the greatest number of votes.

2.5.2 Bagging

Several training sub-sets E_i are formed from the initial training set E by random re-sampling with replacement. A base classifier is learned from each training sub-set. The final class is determined by voting of the base classifiers.

2.5.3 Boosting

In boosting also the diversity of the base classifiers is a result of different training sets. The method works in an iterative manner re-sampling the current training dataset by giving higher resample weights to instances that are hard to classify. The final class is determined by a weighted voting of the base classifiers where the weights are determined on the basis predictive performance of the base classifiers. A typical example of a boosting algorithm is AdaBoost M1 boosting [6].

3. CONTEXT REPRESENTATION

3.1 Spatial Context Model

The contextual information considered for classification refers to the history of user movements. Such history is represented by a $(m+1)$-dimensional vector $\boldsymbol{e}$ of m time-ordered visited locations e_i, that is, $e_j < e_i$ if the user visited location e_j before e_i, $i, j = 1, \ldots, m$. Such locations refer to the antecedent-part of a classification rule while the consequent-part is also the location e_{m+1}, which is the predicted location. In the considered context model, the user roams through a cellular network thus a network cell with a unique identifier represents a location. All attributes of the $\boldsymbol{e}$ vector assume values in the set $Dom(e)$ of

network cells identifiers. Assume that a user u is currently positioned at cell e, which becomes the point of reference. We want to predict which cell the user is going to move to in the next transition. Let $\boldsymbol{e}_{S(u)}$ denote the sequence of the last m cells from which user u went through together with the class attribute, e_{m+1}, i.e., the cell to which he/she is going to move in the next transition. Note that the $\boldsymbol{e}_{S(u)}$ vector does not model time. Instead, it only models cell transitions. Then, from the complete sequence of transitions of a user, we extract a number of transition sequences, $\boldsymbol{e}_{S(u)}$, applying a sliding window of length $m+1$, where :

$$\boldsymbol{e}_{S(u)} = [e_1, e_2, \ldots, e_m, e_{m+1}] \quad (1)$$

In the above vector e_m is the cell in which the user is currently positioned, e_{m+1} is the cell to which he/she is going to move in the next transition and which constitutes the prediction target and $[e_1, e_2, \ldots, e_{m-1}]$ is the sequence of cells from which the user passed before reaching e_m. We explain the above model through an example. Consider a set of cells $Dom(e) = \{e_1, e_2, e_3, e_4, e_5\}$. Assume a user that had the following sequence of transitions $[e_1, e_2, e_3, e_4, e_5]$. Applying a sliding window of length 3 derives the following three $\boldsymbol{e}_{S(u)}$ $[e_1, e_2, e_3]$, $[e_2, e_3, e_4]$, and $[e_3, e_4, e_5]$. Obviously the corresponding class labels are e_3, e_4, e_5. Figure 1 depicts an area divided into cells in which several sliding windows of a user movement are shown.

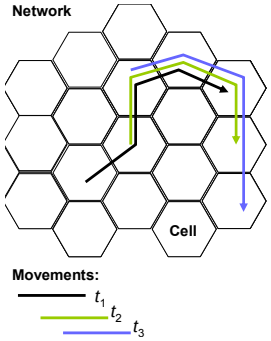

Figure 1. The three $e_{S(u)}$ vectors of a sliding window of length m = 4. Note that the sliding windows are overlapping.

3.2 User Mobility Profile

We introduce the parameter degree of movement randomness, $\delta \in [0, 1]$, in order to express the mobility behavior of a user, i.e., the way a user transits between cells and changes directions. Such degree is used for assessing the performance of the proposed models under various levels of uncertainty and unpredictability w.r.t. mobility behavior. The δ degree denotes the possible transition patterns of a user trajectory between locations. A certain trajectory can derive either from a deterministic movement (assuming a low value of δ) or a random movement (assuming a high value of δ). The adoption of δ provides an objective criterion for assessing movement prediction algorithms. It allows a correct interpretation of the performance evaluation results (e.g., a high accuracy may not necessarily indicate an efficient algorithm if the testing patterns were quite deterministic).

The deterministic trajectories represent regular movements (e.g., the route from home to work). On the other hand, random trajectories represent purely random movements between predefined locations (e.g., a quick detour for a coffee after leaving home and before getting to work). Therefore, a value of $\delta \sim 1.0$ does not mean an explicitly non-deterministic mobility behavior. Instead, such a movement (i.e., $\delta \sim 1.0$) is constrained by obstacles in the examined space.

In our experiments, we adopted the mobility pattern generator discussed in [10]. Through this generator we obtained trajectories with specific δ values in the set {0.0, 0.25, 0.5, 0.75, 1.0}. The five discrete values of δ range from the regular pattern (δ = 0.0 with 500 example trajectories) to completely random pattern (δ = 1.0 with 1000 example trajectories). It should be noted that, the value of δ influences the size of the training patterns (movement history) since the more random the movement is, the more transitions are generally required for a certain itinerary (i.e., moving from a given origin to a given destination).

Table I. The value of the majority class ρ w.r.t., degree of randomness δ.

Degree of Randomness (δ)	Majority Class ratio (ρ), (Default accuracy)
0%	9.06%
25%	6.53%
50%	5.48%
75%	3.81%
100%	3.69%

4. CONTEXT MODEL EVALUATION

We derive a number of context models that, in fact, correspond to the different classification models build from the different classification algorithms that we consider, and examine their relative predictive performance. All the considered classification algorithms are provided with the Weka machine-learning workbench [4]. We represent user trajectories through a series of waypoints. Each waypoint is defined by the location in terms of a cell and speed. The number of cells is 100, hence, $|Dom(e_{m+1})| = 100$. Traces

were obtained by the RMPG tool [10] which allows a controllable degree of randomness. We have used five discrete categories of randomness from the regular pattern (δ = 0.0 with 500 training instances) to completely disordered trajectories (δ = 1.0 with 1000 training instances).

We experimented with the following classifiers from Weka: (i) the Naïve Bayes classifier, (ii) the J48 Decision Tree-based classifier (an implementation of the C4.5 algorithm), (iii) the JRip Classification Rule classifier (an implementation of the RIPPER algorithm), and (iv) the IB*k* incremental algorithm (an implementation of *k*-nearest neighbor algorithm). From the category of the Ensemble learning algorithms we experimented with: (i) Voting with the base classifiers constructed by J48 and Ib*k*, (ii) Bagging were the base classifiers were learned by IB*k*, and finally (iii) the AdaBoost M1, where the base classifiers were also learned by IB*k*. The prediction accuracy ε for each classification algorithm is estimated by 10-fold cross-validation.

We assessed the level of statistical significance for the differences in the accuracies of the different classification algorithms using the *t*-test [11]. We extensively evaluated the performance of models produced by the best classification algorithm under different levels of movement randomness (δ) and different sizes *m* of the sliding window (parameter *m*).

4.1 Classifier Selection

At first we have to compare the predictive performance of each classification algorithm with the default accuracy, i.e., the prediction accuracy obtained from a naïve classifier that always predicts the majority class. A classification algorithm is considered appropriate for a certain classification problem if its classification accuracy is significantly better than the default accuracy. Table I depicts the values of default accuracy w.r.t., the different levels of randomness δ.

Figure 2 depicts the prediction accuracy ε of the IB*k*, J48, Naïve Bayes, JRip, Vote, Bagging and AdaBoost M1 classifiers for the $\boldsymbol{e}_{S(u)}$representation. The prediction accuracy of all classification algorithms is significantly better than the default accuracy for any given δ (see Table I). The classification algorithm that achieves the top performance (for all levels of randomness) is Vote.

Table II depicts in the *t*-test results of all the pairs of classifiers for δ = 0.25. The (*i*,*j*) element of the matrix denotes the statistical significance of the difference between the classification algorithm of the i^{th} row and that of the j^{th} column. Specifically, if the element at (*i*,*j*) is '*0*', then the difference $|\varepsilon_i - \varepsilon_j|$ between the i^{th} and the j^{th} classification algorithm is insignificant. If the element at (*i*,*j*) is '*' then the prediction accuracy of the i^{th} classification algorithm is better than that of the j^{th} algorithm and this is validated through statistical significance test; if the element at (*i*,*j*) is '*v*' then the j^{th} classifier is better. The threshold of the statistical significance test was set to a = 0.05.

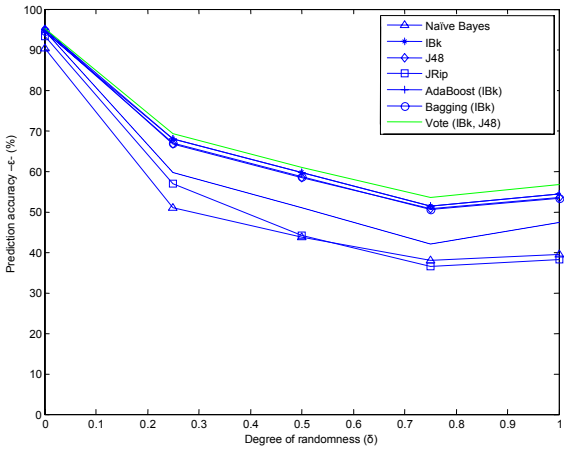

Figure 2. The behavior of the prediction accuracy ε of classifiers vs. degree of randomness δ.

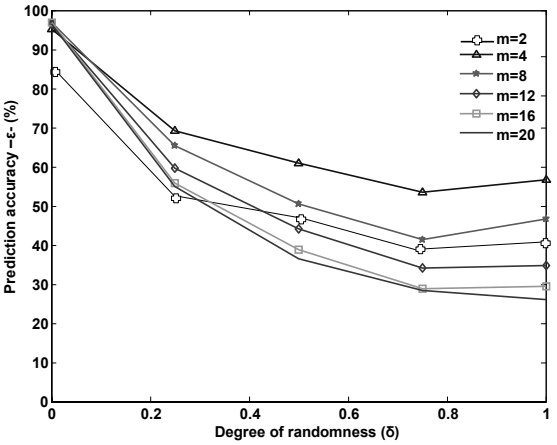

Figure 3. The behavior of the prediction accuracy ε of $e_{S(u)}$-Vote vs. the window length *m*.

4.2 Experimenting with the window length

An important factor of the proposed model is the selection of the sliding window size *m*. The appropriate value of *m* is estimated by experimentation. In Figure 3, we can observe the prediction accuracy of the $\boldsymbol{e}_{S(u)}$-Vote for $2 \le m \le 20$ with a training set size *s* = 40 days. The best results are obtained for *m* = 4. The case *m* = 4 assumes better prediction results even when the user exhibits a high degree of randomness. In case where the size *m* is small, the considered vectors refer to many different class attributes (almost equi-probable), thus, the prediction accuracy

decreases. As long as the size m raises then the considered vectors of cell identifiers is quite large to achieve a close match (between sample and test data), especially, when randomness increases. The size $m = 4$ reflects the regularity of a common user movement as reported in [27].

4.3 Comparison with other Machine Learning algorithms

We compare $\boldsymbol{e}_{S(u)}$-Vote with the LeZi-Update [12] and MMP [13] schemes by means of prediction accuracy. Specifically, in [14], the mobility tracking problem in a cellular network has been considered in the information theoretic framework. Comparison of user mobility models has been based upon the concept of entropy. A *dictionary* of user's path updates is built and maintained by the proposed scheme. Such dictionary supports an adaptive online algorithm that learns the profiles of users. This technique is based on ideas and concepts coming from the area of *lossless compression* and, specifically, the Lempel-Ziv algorithm. This algorithm is also called "LeZi-update" and is exploited to reduce the location update related costs while its predictive power is used to reduce paging cost. In [12], the LeZi-Update scheme is applied in an intelligent home-environment in order to track down an inhabitant, both inside and within surroundings in order to satisfy connectivity requirements. After processing the input trace of the user through the LeZi-update algorithm, a *blending* strategy called *exclusion* is used to predict the next location of the user (see Figure 4).

Table II. Significance difference between the classifiers.

classifier	Naive Bayes	IB*k*	J48	JRip	AdaBoost M1	Bagging	Vote
Naïve Bayes	-	*v*	*v*	*v*	*v*	*v*	*v*
IB*k*	-	-	*	*	*0*	*	*v*
J48	-	-	-	*	*v*	*0*	*v*
JRip	-	-	-	-	*v*	*v*	*v*
AdaBoost M1	-	-	-	-	-	*	*v*
Bagging	-	-	-	-	-	-	*v*
Vote	-	-	-	-	-	-	-

Moreover, the algorithm discussed in [13] is based on Mobile Motion Prediction (MMP) scheme for the prediction of the future location of a roaming user according to his / her movement history pattern. The scheme consists of Regularity-Pattern Detection (RPD) algorithms and Motion Prediction Algorithm (MPA). Regularity Detection is used to detect specific patterns of user movement from a properly structured database (IPB: Itinerary Pattern Base). Two RPD algorithms are proposed: (i) Movement Circle (MC) for the detection of long way routes and (ii) Movement Track (MT) for the detection of regular routes. Three classes of matching schemes are used for correlation analysis of the MCs or MTs namely the *state matching*, the *velocity* or *time matching* and the *frequency matching*. The Motion Prediction Algorithm (MPA) is invoked for combining regularity information with stochastic information (and constitutional constraints) and thus reach a decision - prediction for the future location (or locations) of the mobile user. Figure 5 shows an overview of the suggested scheme.

Figure 6 depicts the prediction accuracy of the $\boldsymbol{e}_{S(u)}$-Vote with that of the LeZi-update and MMP schemes with $m = 4$ and $s = 40$ days for four same training sets. Specifically, $\boldsymbol{e}_{S(u)}$-Vote achieves better performance than the other two algorithms for each degree of user randomness.

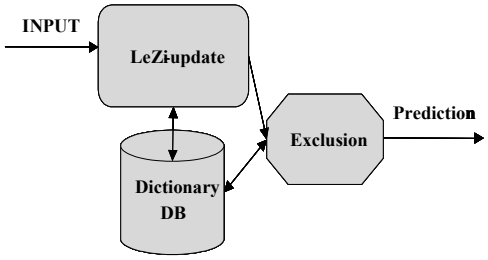

Figure 4. The LeZi-update scheme.

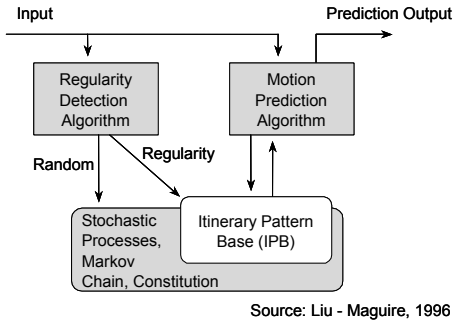

Figure 5. The Predictive Mobility Management Algorithms.

5. PRIOR WORK

There are a lot of prediction models based on Machine Learning techniques. Specifically, the probabilistic model in [15] is based on the user movement history of handover behavior. This

model considers the history of all handovers that occurred in a given cell using the Naïve Bayes classification. The authors in [16] report a probability - based and a learning - based model for trajectory prediction. The algorithm in [17] predicts the next inter - cell movement of a mobile user. The user mobility patterns are mined from the history of the trajectories resulting in mobility rules extraction. The location prediction is based on such rules.

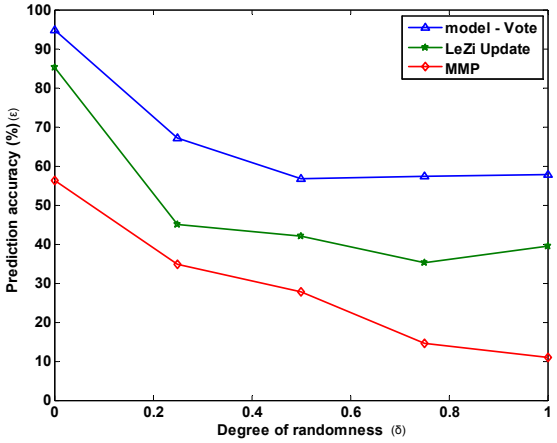

Figure 6. The behavior of the prediction accuracy ε of eS(u) -Vote vs. the LeZi-update and MMP algorithms.

The authors in [18] implement an algorithm based on user mobility patterns discovery. Such patterns, which derived from trajectory clustering, are used for location prediction and dynamic resource allocation. Moreover, an efficient online (incremental) algorithm that classifies *routes* between *important* locations and predicts the future location is reported in [19]. Specifically, clusters of cell - sequences are built to represent *physical routes*. The prediction is based on destination probabilities and temporal reasoning. A data - mining algorithm is proposed in [20], which efficiently discovers sequential mobile patterns. These patterns exploit both spatial and application context (e.g., service requests). Moreover, the mobility tracking in a cellular network is based on information theory by using the compression Lempel-Ziv algorithm [21]. The algorithm [13] consists of the *regularity-pattern detection* and the *model prediction* processes used for predicting the user movements. The work presented in [22] discusses pattern matching techniques and extended self-learning Kalman filters in order to estimate the future location. In addition, a learning automaton that follows a linear *reward - penalty* scheme is used in [23] in order to facilitate location prediction. The authors in [24] apply evidential reasoning based on the Dempster-Shafer theory in mobility prediction when adequate knowledge about the history of user's travelling patterns is not available. Finally, the authors in [25] refer to a conceptual context prediction model based on geometrical location prediction.

6. CONCLUSIONS

We propose a context model for location prediction based on the user mobility behaviour. We present how Machine Learning is applied to context-awareness for predicting future locations in pervasive computing environments. The proposed context model exploits the user history and degree of movement randomness in order to classify and predict future movements. The model is evaluated with the 10-fold cross validation method with sets of simulated user movements with varying degrees of randomness. We experiment with several ML classifiers and evaluate the model through certain parameters derived from the ML field in order to choose the appropriate classifier for location prediction. Our findings show that, the Voting classification scheme is appropriate for location prediction since it exhibits satisfactory prediction results for diverse user mobility behaviour. Moreover, we compare the performance of the proposed $\boldsymbol{e}_{S(u)}$-Vote model with that of the LeZi-Update and MMP algorithms justifying the importance of the ML classification in predicting spatial context.

Our model can be enhanced with more semantics and contextual information, like temporal context (e.g., time period within a day), application context (e.g., service requests), proximity of people (e.g., social context) and destination / velocity of the user movement. Finally, the combination of several local classification models has to be considered in order to exploit local spatial and temporal contextual information.

7. REFERENCES

[1] M. Weiss, and C.A. Kulikowski, *Computer Systems That Learn: Classification and Prediction Methods from Statistics*, Neural Networks, Machine Learning and Expert Systems, Morgan Kaufmann, 1991.

[2] M. Plutowski, S. Sakata and H. White, *Cross-validation estimates integrated mean squared error*, Advances in Neural Information Processing Systems, vol. 6, 1994.

[3] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, Wiley-Interscience, 2001.

[4] I. H. Witten and E.Frank, *Data Mining: Practical Machine Learning Tool sand Techniques*, Morgan Kaufmann Series in Data Management Systems, 2005.

[5] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Series in Data Management Systems, 2001.

[6] E. Alpaydin, *Introduction to Machine Learning*, The MIT Press, 2004.

[7] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Series in Machine Learning, 1993.

[8] M. Mahmood, M. Zonoozi, and P. Dassanayake, *User Mobility Modeling and Characterization of Mobility Patterns*, IEEE Communications, vol. 15, no. 7, 1997.

[9] W. Cohen, A. Prieditis and S.J. Russel, *Fast Effective Rule Induction*, Proc .Int. Conf. on Machine Learning, 1995.

[10] M.Kyriakakos, N.Frangiadakis, S.Hadjiefthymiades and L.Merakos, *RMPG: A Realistic Mobility Pattern Generator for the Performance Assessment of Mobility Functions*, Simulation Modeling Practice And Theory, vol. 12, no. 1, Elsevier, 2004.

[11] F. Pernkopf, J. Bilmes, *Discriminative versus Generative Parameter and Structure Learning of Bayesian Network Classifiers*, ICML, 2005.

[12] S.K.Das, D.J.Cook, A.Bhattacharya, E.O.Helerman and T-Y.Lin, *The Role of Prediction Algorithms in the MavHome Smart Home Architecture*, IEEE Wireless Communications, vol. 9, no. 6, pp. 77-84, December 2002.

[13] L. George and G. Maguire, A *class of mobile motion prediction algorithms for wireless mobile computing and communications*, Mobile Networks and Applications 1, J.C. Baltzer AG, Science Publishers, 1996.

[14] A.Bhattacharya, and S.K.Das, LeZi *Update: An Information Theoretic Approach to Track Mobile Users In PCS Networks*, ACM/IEEE Mobicom '99, Seattle, USA, August 1999.

[15] S. Choi and K. G. Shin, *Predictive and adaptive bandwidth reservation for hand-offs in QoS-sensitive cellular networks*, Proc. ACM SIGCOMM '98, 1998.

[16] L. Xiong and H. A. Karimi, *Location Awareness through Trajectory Prediction*, Computers, Environment and Urban Systems, Elsevier, 2006.

[17] G. Yavas, D. Katsaros, O. Ulusoy, and Y.Manolopoulos, *A data mining approach for Location Prediction in Mobile Environments*, Data & Knowledge Engineering vol. 54, 2005.

[18] D. Katsaros, A. Nanopoulos, *Clustering Mobile Trajectories for Resource Allocation in Mobile Environments*, Intelligent Data analysis, 2003.

[19] K. Laasonen, *Clustering and Prediction of Mobile User Routes from Cellular Data*, Proc. PKDD, 2005.

[20] V. S. Tseng, and K. W. Lin, *Efficient mining and prediction of user behavior patterns in mobile web systems*, Information and Software Technology vol. 48, Elsevier, 2006.

[21] A. Bhattacharya and S.K. Das, *LeZi update: An Information Theoretic Approach to Track Mobile Users in PCS Networks*, Proc. ACM/IEEE Mobicom '99, 1999.

[22] T. Liu, P. Bahl and I. Chlamtac, *Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks*, IEEE JSAC vol. 16, no. 6, 1998.

[23] S.Hadjiefthymiades, L.Merakos, *Proxies + Path Prediction: Improving Web Service Provision in Wireless-Mobile Communications*, ACM/Kluwer Mobile Networks and Applications, Special Issue on Mobile and Wireless Data Management, vol.8, no. 4, 2003.

[24] A. Karmouch, N. Samaan, *A Mobility Prediction Architecture Based on Contextual Knowledge and Spatial Conceptual Maps*, IEEE Transactions on Mobile Computing, vol. 4 no. 6, 2005.

[25] C. Anagnostopoulos, P. Mpougiouris, S. Hadjiefthymiades, *Prediction Intelligence in Context-aware Applications*, Proc. ACM Mobile Data Management (MDM05), pp. 137-141, 2005.

[26] A. Dey, *Understanding and using context*, Personal and Ubiquitous Computing, vol. 5, no. 1, pp 4-7, 2001.

[27] M.Kyriakakos, Mobility Simulation and Path Prediction in Wireless Mobile Communications, PhD thesis, University of Athens, Dept. of Informatics and Telecommunications, 2005.