An Online Adaptive Model for Location Prediction

Theodoros Anagnostopoulos, Christos Anagnostopoulos, and Stathes Hadjiefthymiades

Pervasive Computing Research Group, Communication Networks Laboratory, Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilissia, Athens 15784, Greece
Tel.: +302107275127
`{thanag,bleu,shadj}@di.uoa.gr`

Abstract. Context-awareness is viewed as one of the most important aspects in the emerging pervasive computing paradigm. Mobile context-aware applications are required to sense and react to changing environment conditions. Such applications, usually, need to recognize, classify and predict context in order to act efficiently, beforehand, for the benefit of the user. In this paper, we propose a mobility prediction model, which deals with context representation and location prediction of moving users. Machine Learning (ML) techniques are used for trajectory classification. Spatial and temporal on-line clustering is adopted. We rely on Adaptive Resonance Theory (ART) for location prediction. Location prediction is treated as a context classification problem. We introduce a novel classifier that applies a Hausdorff-like distance over the extracted trajectories handling location prediction. Since our approach is time-sensitive, the Hausdorff distance is considered more advantageous than a simple Euclidean norm. A learning method is presented and evaluated. We compare ART with Offline *k*Means and Online *k*Means algorithms. Our findings are very promising for the use of the proposed model in mobile context aware applications.

Keywords: Context-awareness, location prediction, Machine Learning, online clustering, classification, Adaptive Resonance Theory.

1 Introduction

In order to render mobile context-aware applications intelligent enough to support 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. 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*" [1]. Context refers to the current values of specific ingredients that represent the activity of an entity / situation and environmental state (e.g., attendance of a meeting, location, temperature).

One of the more intuitive capabilities of the mobile context-aware applications is their *proactivity*. Predicting user actions and contextual ingredients enables a new class of applications to be developed along with the improvement of existing ones.

One very important ingredient is location. Estimating and predicting the future location of a mobile user enables the development of innovative, location-based services/applications [2], [12]. For instance, location prediction can be used to improve resource reservation in wireless networks and facilitate the provision of location-based services by preparing and feeding them with the appropriate information well in advance. The accurate determination of the context of users and devices is the basis for context-aware applications. In order to adapt to changing demands, such applications need to reason based on basic context ingredients (e.g., time, location) to determine knowledge of higher-level situation.

Prediction of context is quite similar to information classification / prediction (*offline* and *online*). In this paper, we adopt ML techniques for predicting location through an adaptive model. ML is *the study of algorithms that improve automatically through experience*. ML provides algorithms for learning a system to cluster pre-existing knowledge, classify observations, predict unknown situations based on a history of patterns and adapt to situation changes. Therefore, ML can provide solutions that are suitable for the location prediction problem. Context-aware applications have a set of pivotal requirements (e.g., flexibility and adaptation), which would strongly benefit if the learning and prediction process could be performed in real time. We argue that the most appropriate solutions for location prediction are offline and online clustering and classification. Offline clustering is performed through the Offline *k*Means algorithm while online clustering is accomplished through the Online *k*Means and Adaptive Resonance Theory (ART). Offline learners typically perform complete model building, which can be very costly, if the amount of samples rises. Online learning algorithms are able to detect changes and adapt / update only parts of the model thus providing for fast adaptation of the model. Both forms of algorithms extract a subset of patterns / clusters (i.e., a knowledge base) from an initial dataset (i.e., a database of user itineraries). Moreover, online learning is more suited for the task of classification / prediction of the user mobility behavior as in the real life user movement data often needs to be processed in an online manner, each time after a new portion of the data arrives. This is caused by the fact that such data is organized in the form of a data stream (e.g., a sequence of time-stamped visited locations) rather than a static data repository, reflecting the natural flow of data. Classification involves the matching of an unseen pattern with existing clusters in the knowledge base. We rely on a Hausdorff-like distance [5] for matching unseen itineraries to clusters (such metric applies to convex patterns and is considered ideal for user itineraries). Finally, location prediction boils down to location classification w.r.t. Hausdorff-like distance.

We assess two training methods for training an algorithm: (i) the “nearly” *zero-knowledge* method in which an algorithm is incrementally trained starting with a little knowledge on the user mobility behavior and the (ii) *supervised* method in which sets of known itineraries are fed to the classifier. Moreover, we assess a learning method for the online algorithms regarding the success of location prediction, in which a misclassified instance is introduced into the knowledge base updating appropriately the model.

We evaluate the performance of our models against the movement of mobile users. Our objective is to predict the users’ future location (their next move) through an on-line adaptive classifier. We establish some important metrics for the performance assessment process taking into account low system-requirements (storage capacity) and effort for model building (processing power). Specifically, besides the prediction accuracy, i.e., the precision of location predictions, we are also interested in the size of the derived knowledge base; that is the produced clusters out of the volume of the training patterns, and the capability of the classifier to adapt the derived model to unseen patterns. Surely, we need to keep storage capacity as low as possible while maintaining good prediction accuracy. Lastly, our objective is to assess the *adaptivity* of the proposed schemes, i.e., the capability of the predictor to detect and update appropriately the specific part of the trained model. The classifier (through the location prediction process) should rapidly detect changes in the behavior of the mobile user and adapt accordingly through model updates, however, often at the expense of classification accuracy (note that an ambient environment implies high dynamicity). We show that increased adaptivity leads to high accuracy and dependability.

The rest of the paper is structured as follows. In Section 2 we present the considered ML models by introducing the Offline *k*Means, Online *k*Means and ART algorithms. In Section 3 we elaborate on the proposed model with context representation. Section 4 presents the proposed mobility prediction model based on the ART algorithm. The performance assessment of the considered model is presented in Section 5, where different versions of that model are evaluated. Moreover, in Section 6, we compare the ART models with the Offline / Online *k*Means algorithms. Prior work is discussed in Section 7 and we conclude the paper in Section 8.

2 Machine Learning Models

In this section we briefly discuss the clustering algorithms used throughout the paper. Specifically, we distinguish between offline and online clustering and elaborate on the Offline/Online *k*Means and ART.

2.1 Offline kMeans

In Offline *k*Means [3] we assume that there are $k > 1$ initial clusters (groups) of data. The objective of this algorithm is to minimize the reconstruction error, which is the total Euclidean distance between the instances (patterns), $\mathbf{u}_i$, and their representation, i.e., the cluster centers (clusters), $\mathbf{c}_i$. We define the reconstruction error as follows:

$$E\left(\{\mathbf{c}_i\}_{i=1}^{k} \mid U\right) = \frac{1}{2}\sum_t \sum_i b_{i,t} \| \mathbf{u}_t - \mathbf{c}_i \|^2 \tag{1}$$

where

$$b_{i,t} = \begin{cases} 1, & \text{if } \| \mathbf{u}_t - \mathbf{c}_i \| = \min_l \| \mathbf{u}_t - \mathbf{c}_l \| \\ 0, & \text{otherwise} \end{cases},$$

$U = \{\mathbf{u}_t\}$ is the total set of patterns and $C = \{\mathbf{c}_i\}$, $i = 1,\ldots, k$ is the set of clusters. $b_{i,t}$ is 1 if $\mathbf{c}_i$ is the closest center to $\mathbf{u}_t$ in Euclidean distance. For each incoming $\mathbf{u}_t$ each $\mathbf{c}_i$ is updated as follows:

$$\mathbf{c}_i = \frac{\sum_t b_{i,t}\mathbf{u}_t}{\sum_t b_{i,t}} \tag{2}$$

Since the algorithm operates in offline mode, the initial clusters can be set during the training phase and cannot be changed (increased or relocated) during the testing phase.

2.2 Online kMeans

In Online *k*Means [3] we assume that there are $k > 1$ initial clusters that split the data. Such algorithm processes unseen patterns one by one and performs *small* updates in the position of the appropriate cluster ($\mathbf{c}_i$) at each step. The algorithm does not require a training phase. The update for each new (unseen) pattern $\mathbf{u}_t$ is the following:

$$\mathbf{c}_i = \mathbf{c}_i + \eta \cdot b_{i,t} \cdot (\mathbf{u}_t - \mathbf{c}_i)$$

This update moves the *closest* cluster (for which $b_{i,t} = 1$) toward the input pattern $\mathbf{u}_t$ by a factor of η. The other clusters (found at bigger distances from the considered pattern) are not updated. The semantics of $b_{i,t}$, η and $(\mathbf{u}_t - \mathbf{c}_i)$ are:

- $b_{i,t} \in \{0, 1\}$ denotes which cluster is being modified,
- $\eta \in [0, 1]$ denotes how much is the cluster shifted toward the new pattern, and,
- $(\mathbf{u}_t - \mathbf{c}_i)$ denotes the distance to be learned.

Since the algorithm is online, the initial clusters should be known beforehand[1] and can only be relocated during the testing phase. The number of clusters remains constant. Therefore, the algorithm exhibits limited flexibility.

2.3 Adaptive Resonance Theory

The ART approach [4] is an online learning scheme in which the set of patterns U is not available during training. Instead patterns are received one by one and the model is updated progressively. The term *competitive learning* is used for ART denoting that the (local) clusters *compete* among themselves to assume the "responsibility" for representing an unseen pattern. The model is also called *winner-takes-all* because one cluster "wins the competition" and gets updated, and the others are not updated at all.

The ART approach is *incremental*, meaning that one starts with one cluster and adds a new one, if needed. Given an input $\mathbf{u}_t$, the distance b_t is calculated for all clusters $\mathbf{c}_i$, $i = 1, .., k$, and the closest (e.g., minimum Euclidean distance) to $\mathbf{u}_t$ is updated. Specifically, if the minimum distance b_t is smaller than a certain threshold value, named the *vigilance*, ρ, the update is performed as in Online *k*Means (see Eq.(3)). Otherwise, a new center $\mathbf{c}_{k+1}$ representing the corresponding input $\mathbf{u}_t$ is added in the model (see Eq.(3)). It is worth noting that the vigilance threshold refers to the criterion of considering two patterns equivalent or not during the learning phase of a classifier. As it will be shown,

[1] One possible approach to determine the initial k clusters is to select the first k distinct instances of the input sample U.

the value of vigilance is considered essential in obtaining high values of corrected classified patterns. The following equations are adopted in each update step of ART:

$$b_t = \| \mathbf{c}_i - \mathbf{u}_t \| = \min_{l=1}^{k} \| \mathbf{c}_l - \mathbf{u}_t \|$$
$$\begin{cases} \mathbf{c}_{k+1} \leftarrow \mathbf{u}_t & \textit{if } b_t > \rho \\ \mathbf{c}_i = \mathbf{c}_i + \eta(\mathbf{u}_t - \mathbf{c}_i) & \textit{otherwise} \end{cases} \tag{3}$$

3 Context Representation

Several approaches have been proposed in order to represent the movement history (or history) of a mobile user [15]. We adopt a spatiotemporal history model in which the movement history is represented as the sequence of 3-D points (3DPs) visited by the moving user, i.e., time-stamped trajectory points in a 2D surface. The spatial attributes in that model denote latitude and longitude.

Let $\mathbf{e} = (x, y, t)$ be a 3DP. The *user trajectory* $\mathbf{u}$ consists of several time-ordered 3DPs, $\mathbf{u} = [\mathbf{e}_i] = [\mathbf{e}_1, \ldots, \mathbf{e}_N]$, $i = 1, \ldots, N$ and is stored in the system's database. It holds that $t(\mathbf{e}_1) < t(\mathbf{e}_2) < \ldots < t(\mathbf{e}_N)$, i.e., time-stamped coordinates. The x and y dimensions denote the latitude and the longitude while t denotes the time dimension (and $t(\cdot)$ returns the time coordinate of $\mathbf{e}$). Time assumes values between 00:00 and 23:59. To avoid state information explosion, trajectories contain time-stamped points sampled at specific time instances. Specifically, we sample the movement of each user at $1.66 \cdot 10^{-3}$ Hertz (i.e., once every 10 minutes). Sampling at very high rates (e.g., in the order of a Hertz) is meaningless, as the derived points will be highly correlated. In our model, $\mathbf{u}$ is a finite sequence of N 3DPs, i.e., $\mathbf{u}$ is a $3 \cdot N$ dimension vector. We have adopted a value of $N = 6$ for our experiments meaning that we estimate the future position of a mobile terminal from a movement history of 50 minutes (i.e., 5 samples). Specifically, we aim to query the system with a N-1 3DP sequence so that our classifier / predictor returns a 3DP, which is the predicted location of the mobile terminal.

A *cluster trajectory* $\mathbf{c}$ consists of a finite number of 3DPs, $\mathbf{c} = [\mathbf{e}_i]$, $i = 1, \ldots, N$ stored in the knowledge base. Note that a cluster trajectory $\mathbf{c}$ and a user trajectory $\mathbf{u}$ are vectors of the same length N. This is because $\mathbf{c}$, which is created from ART based on unseen user trajectories, is a representative itinerary of the user movements. In addition, the *query trajectory* $\mathbf{q}$ consists of a number of 3DPs, $\mathbf{q} = [\mathbf{e}_j]$, $j = 1, \ldots, N$-1. It is worth noting that $\mathbf{q}$ is a sequence of N-1 3DPs. Given a $\mathbf{q}$ with a N-1 history of 3DPs we predict the $\mathbf{e}_N$ of the closest $\mathbf{c}$ as the next user movement.

4 Mobility Prediction Model

From the ML perspective the discussed location prediction problem refers to an m+l model [13]. In m+l models we have m steps of user movement history and we want to predict the future user movement after l steps (the steps have time-stamped coordinates). In our case, $m = N$-1, i.e., the query trajectory $\mathbf{q}$, while $l = 1$, i.e., the predicted $\mathbf{e}_N$. We develop a new spatiotemporal classifier (C) which given $\mathbf{q}$ can predict $\mathbf{e}_N$. Specifically, $\mathbf{q}$ and $\mathbf{c}$ are trajectories of different length thus we use a Hausdorff-like measure for calculating the ||**q** - **c**|| distance. Given query **q**, the proposed classifier C attempts to find the nearest cluster **c** in the knowledge base and, then, take $\mathbf{e}_N$ as the *predicted* 3DP. For evaluating C, we compute the Euclidean distance between the predicted 3DP and the *actual* 3DP (i.e., the real user movement). If such distance is greater than a preset error threshold θ then prediction is not successful. After predicting the future location of a mobile terminal, the C classifier receives feedback from the environment considering whether the prediction was successful or not, and reorganize the knowledge base accordingly [14]. In our case, the feedback is the actual 3DP observed in the terminal's movement. Thus the C classifier reacts with the environment and learns new patterns once an unsuccessful prediction takes place.

Specifically,

- in case of an unsuccessful prediction, the C appends the actual 3DP to **q** and updates (i.e., learns) such extended sequence in the model considering as new knowledge, i.e., an unseen user movement behavior.
- in the case of a successful prediction, C dos not need to learn. A successful prediction refers to a well-established prediction model for handling unseen user trajectories.

The heart of the proposed C classifier is the ART algorithm. ART clusters unseen user trajectories to existing cluster trajectories or creating new cluster trajectories depending on the vigilance value. ART is taking the $\mathbf{u}_1$ pattern from the incoming set U of patterns and stores it as the $\mathbf{c}_1$ cluster in the knowledge base. For the t-th unseen user trajectory the following procedure is followed (see Table 1): The algorithm computes the Euclidean distance b_t between $\mathbf{u}_t$ and the closest $\mathbf{c}_i$. If b_t is smaller than the vigilance ρ then $\mathbf{c}_i$ is updated from $\mathbf{u}_t$ by the η factor. Otherwise, a new cluster $\mathbf{c}_j \equiv \mathbf{u}_t$ is inserted into the knowledge base. The ART algorithm is presented in Table 1.

Table 1. The ART Algorithm for the C classifier

```
1.    $j \leftarrow 1$
2.    $\mathbf{c}_j \leftarrow \mathbf{u}_j$
3.    For ($\mathbf{u}_t \in U$) Do
4.    $b_t = ||\mathbf{c}_j - \mathbf{u}_t|| = \min_{l=1,\dots,j}||\mathbf{c}_l - \mathbf{u}_t||$
5.       If ($b_t > \rho$) Then  /*expand knowledge*/
            $j \leftarrow j + 1$
6.          $\mathbf{c}_j \leftarrow \mathbf{u}_t$
7.       Else
8.          $\mathbf{c}_j \leftarrow \mathbf{c}_j + \eta(\mathbf{u}_t - \mathbf{c}_j)$  /*update model locally*/
9.       End If
10.   End for
```

Let T, P be subsets of U for which it holds that $T \subseteq P \subseteq U$. The T set of patterns is used for training the C classifier, that is, C develops a knowledge base corresponding to the supervised training method. The P set is used for performing on-line predictions. We introduce the C-T classifier version, which is the C classifier trained with the T set. In addition, once the T set is null then the C classifier is not trained beforehand corresponding to the zero-knowledge training method and performs on-line prediction with the set *P*. In this case, we get the C-nT classifier corresponding to the C classifier, when the training phase is foreseen.

Moreover, in order for the C classifier to achieve prediction, an approximate Hausdorff-like metric [5] is adopted to estimate the distance between **q** and **c**. Specifically, the adopted formula calculates the point-to-vector distance between $\mathbf{e}_j \in \mathbf{q}$ and **c**, $\delta'(\mathbf{e}_j, \mathbf{c})$, as follows:

$$\delta'(\mathbf{e}_j, \mathbf{c}) = \| \mathbf{f}_i - \mathbf{e}_j \|_{\min_{\mathbf{f}_i} |t(\mathbf{f}_i) - t(\mathbf{e}_j)|}$$

where $\|\cdot\|$ is the Euclidean norm for $\mathbf{f}_i \in \mathbf{c}$ and $\mathbf{e}_j$. The $\delta'(\mathbf{e}_j, \mathbf{c})$ value indicates the minimum distance between $\mathbf{e}_j$ and $\mathbf{f}_i$ w.r.t. the time stamped information of the user itinerary, that is the Euclidean distance of the closest 3DPs in time. Hence, the overall distance between the *N*-1 in length **q** and the *N* in length **c** is calculated as

$$\delta_{N-1}(\mathbf{q}, \mathbf{c}) = \frac{1}{N-1} \sum_{\mathbf{e}_j \in \mathbf{q}} \delta'(\mathbf{e}_j, \mathbf{c}) \tag{4}$$

Figure 1 depicts the process of predicting the next user movement considering the proposed C classifier. Specifically, once a query trajectory **q** arrives, then C attempts to classify **q** into a known $\mathbf{c}_i$ in the knowledge base w.r.t. Hausdorff metric. The C classifier returns the predicted $\mathbf{e}_N \in \mathbf{c}_i$ of the closest $\mathbf{c}_i$ to **q**. Once such result refers to an unsuccessful prediction w.r.t. a preset error threshold θ then the C-T (or the C-nT) extend the **q** vector with the actual 3DP and insert **q** into the knowledge base for further learning according to the algorithm in Table 1 (feedback).

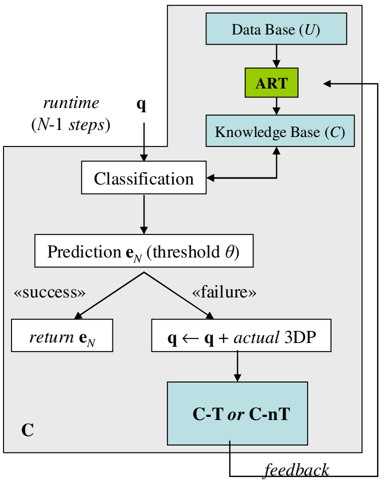

Fig. 1. The proposed adaptive classifier for location prediction

5 Prediction Evaluation

We evaluated our adaptive model in order to assess its performance. In our experiments, the overall user movement space has a surface of 540 km^2. Such space derives from real GPS trace captured in Denmark [6]. The GPS trace was fed into our model and the performance of the C system w.r.t. predefined metrics was monitored. Table 2 indicates the parameters used in our experiments.

Table 2. Experimental Parameters

Parameter	**Value**	**Comment**
Learning rate (n)	0.5	In case of a new pattern $\mathbf{u}_t$, the closest cluster $\mathbf{c}_i$ is moved toward $\mathbf{u}_t$ by half the spatial and temporal distance.
Spatial coefficient of vigilance (ρ_s)	100 *m*	Two 2D points are considered different if their spatial distance exceeds 100 meters.
Temporal coefficient of vigilance (ρ_t)	10 *min*	Two time-stamps are considered different if their temporal distance exceeds 10 minutes.
Precision threshold / location accuracy (θ)	10 *m*	The predicted location falls within a circle of radius 10 meters from the actual location.[2]

The GPS traces including 1200 patterns were preprocessed and we produced two training files and two test files as depicted in Figure 2. The first training file, *TrainA*, is produced from the first half of the GPS trace records. The second training file, *TrainB*, consists of a single trace record. The first test file, *TestA*, is produced from the entire set of trace records, including -in ascending order- the first half of the GPS traces and the other half of unseen traces. Finally the second test file, *TestB*, is produced from the entire set of the GPS trace records, including -in ascending order- the second half of unseen traces and the first half of the GPS traces. During the generation of the training/test files, white noise was artificially induced into the trace records.

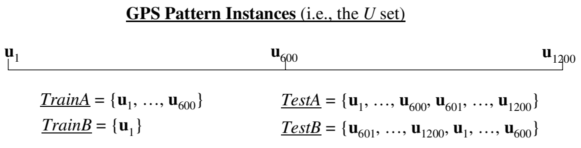
GPS Pattern Instances (i.e., the U set)
u1
u600
u1200
TrainA = {u1, …, u600}
TrainB = {u1}
TestA = {u1, …, u600, u601, …, u1200}
TestB = {u601, …, u1200, u1, …, u600}

Fig. 2. The generated GPS trace files for experimentation

[2] Such accuracy level is considered appropriate for the kind of applications where location prediction can be applied (see the Introduction section or [12]).

We have to quantitatively and qualitatively evaluate the proposed model. For that reason, we introduce the following quantitative and qualitative parameters: (a) the precision achieved by the prediction scheme –the higher the precision the more accurate the decisions on the future user location- (b) the size of the underlying knowledge base –we should adopt solutions with the lowest possible knowledge base size (such solutions are far more efficient and feasible in terms of implementation) and (c) the capability of the model to rapidly react to changes in the movement pattern of the user/mobile terminal and re-adapt. We define *precision*, p, as the fraction of the correctly predicted locations, p_+, against the total number of predictions made by the C system, p_{total}, that is,

$$p = \frac{p_+}{p_{total}}$$

In the following sub-sections, we evaluate the diverse versions of the C classifier w.r.t. training methods by examining the classifier convergence (speed of learning and adaptation) and the derived precision on prediction future locations.

5.1 Convergence of *C-T* and *C-nT*

The C classifier converges once the knowledge base does not expand with unseen patterns, i.e., the set U does not evolve. In Figure 3, we plot the number of the clusters, $|U|$, that are generated from the C-T/-nT models during the testing phase. The horizontal axis denotes the incoming (time-ordered) GPS patterns. The point (.) marked line depicts the behavior of the C-T-1 model trained with *TrainA* and tested with *TestA*. In the training phase, the first 600 patterns of *TrainA* have gradually generated 70 clusters in U. In the testing phase, the first 600 patterns are known to the classifier so there is no new cluster creation. On the other hand, in the rest 600 unseen patterns, the number of clusters scales up to 110 indicating that the ART algorithm learns such new patterns.

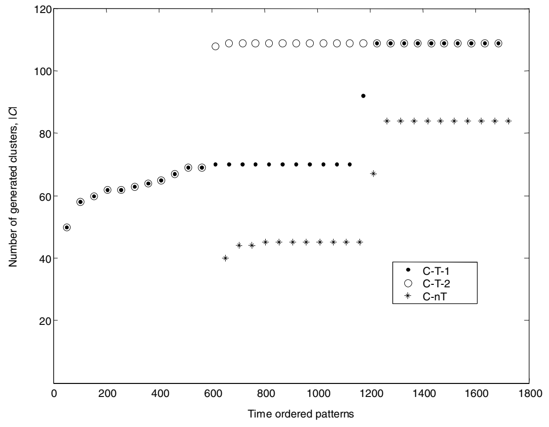

Fig. 3. Convergence of C-T/-nT

The circle (o) marked line depicts the C-T-2 model, which is trained with *TrainA* and tested with *TestB*. Since the train file is the same as in the C-T-1 model, the first generated clusters are the same in number ($|U| = 70$). In the testing phase, we observe a significant difference. ART does not know the second 600 unseen patterns, thus, it learns new patterns up to 110 clusters. In the next 600 known patters, C-T-2 does not need to learn additional clusters thus it settles at 110 clusters.

We now examine the behavior of the C-nT model corresponding to the zero-knowledge training method. The asterisk (*) marked line depicts the training phase (with *TrainB*) followed by the testing phase (with *TestA*) of C-nT. In this case, we have an incremental ART that does not need to be trained. For technical consistency reasons, it only requires a single pattern, which is the unique cluster in the knowledge base at the beginning. In the testing phase, for the first 600 unseen patterns of *TestA* we observe a progressive cluster creation (up to 45 clusters). For the next 600 unseen patterns, we also observe a gradual cluster creation (up to 85 clusters) followed by convergence. Comparing the C-T-1/-2 and C-nT models, the latter one achieves the minimum number of clusters (22.72% less storage cost). This is due to the fact that C-nT starts learning only from unsuccessful predictions in an incremental way by adapting pre-existing knowledge base to new instances. Nevertheless, we also have to take into account the prediction accuracy in order to reach safe conclusions about the efficiency and effectiveness of the proposed models.

5.2 Precision of *C-T* and *C-nT*

In Figure 4 we examine the precision achieved by the algorithms. The vertical axis depicts the precision value *p* achieved during the testing phase. The point (.) marked line depicts the precision of the C-T-1 model trained with *TrainA* and tested with *TestA*. During the test phase, for the first 600 known patterns C-T-1 achieves precision value ranging from 97% to 100%. In the next 600 unseen patterns, we observe that for the first instances the precision drops smoothly to 95% and as C-T-1 learns, i.e., learn new clusters and optimize the old ones, the precision converges to 96%.

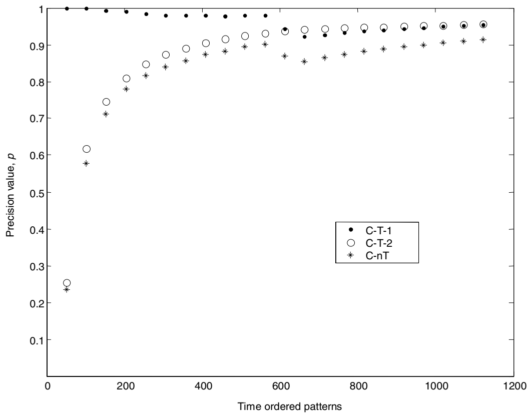

Fig. 4. Precision of C-T/-nT

The circle (o) marked line depicts the precision behavior for the C-T-2 model tested with *TestB* and trained with *TrainA*. With the first 600 totally unseen patterns during the test phase, C-T-2 achieves precision from 26% to 96%. This indicates that the model is still learning during the test phase increasing the precision value. In the next 600 known patterns, the model has nothing to learn and the precision value converges to 96%.

The asterisk (*) marked line depicts the precision behavior of the C-nT model tested with *TestA* and trained with *TrainB*. In this case, C-nT is trained with only one pattern instance, i.e., the algorithm is fully incremental, thus, all the instances are treated as unseen. In the test phase, for the first 600 patterns, the model achieves precision, which ranges from 25% to 91% In the next 600 patterns, we can notice that for the first instances the precision drops smoothly to 88% and as the model learns, precision gradually converges to 93%.

Evidently, the adoption of the training method, i.e., the C-T-1/-2 models, yields better precision. However, if we correlate our findings with the results shown in Figure 3, we infer that a small improvement in precision has an obvious storage cost. Specifically, we need to store 110 clusters, in the case of C-T, compared to 85 clusters in the case of C-nT (22.72% less storage cost). Furthermore, the user movement patterns can be changed repeatedly over time. Hence, by adopting the training method, one has to regularly train and rebuild the model. If the mobile context-aware application aims at maximizing the supported quality of service w.r.t. precision, while keeping the storage cost stable, the C-nT model should be adopted.

6 Comparison with Other Models

We compare the C-nT model with other known models that can be used for location prediction. Such models implement the Offline *k*Means and Online *k*Means algorithms. Such models require a predefined number of $k > 1$ initial clusters for constructing the corresponding knowledge base. We should stress here that, the greater the k the greater the precision value achieved by Offline/Online *k*Means. In our case, we could set $k = 110$, which is the convergence cluster-count for the C models (Section 5). For C-nT, we use *TrainB* for the training and *TestA* for the testing phase (such model adopts the zero-knowledge training method). Moreover, for the Offline/Online *k*Means models we use *TrainA* for the training and *TestA* for the testing phase because both models require $k > 1$ initial clusters.

Figure 5 depicts the precision achieved by the C-nT (the point (.) marked line), Offline *k*Means (the asterisk (*) marked line) and Online *k*Means (the circle (o) marked line) models. The horizontal axis represents the ordered instances and the vertical axis represents the achieved precision. We can observe in the first 600 patterns C-nT achieves precision levels ranging from 25% to 91% indicating adaptation to new knowledge. This is attributed to the learning mechanism (C-nT recognizes and learns new user movements). In the next 600 patterns we notice that for the first instances, the precision drops smoothly to 88% and as the knowledge base adapts to new movements and optimizes the existing ones, precision converges to 93%.

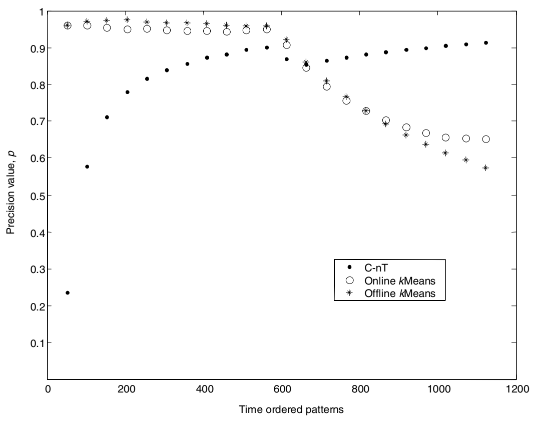

Fig. 5. Comparison of C-nT with the Offline/Online *k*Means models

In the case of Offline *k*Means, we observe that for the first 600 patterns, it achieves precision levels ranging from 96% to 98% once the initial clusters are set to $k = 110$. In the next 600 patterns we notice that the precision drops sharply and converges to 57% as the knowledge base is not updated by unseen user movements. By adopting Online *k*Means we observe that for the testing phase (the first 600 patterns) it achieves precision levels ranging from 94% to 97% given the train file *TrainA*. In the next 600 patterns we notice that for the first instances the precision drops rather smoothly to 86% and, as the knowledge base is incrementally adapting to new patterns, the precision value converges to 65%. Evidently, by comparing such three models, the most suitable model for location prediction is the C-nT since (i) it achieves greater precision through model adaptation and (ii) requires a smaller size of the underlying knowledge base (i.e., less clusters) than the Offline/Online *k*Means models.

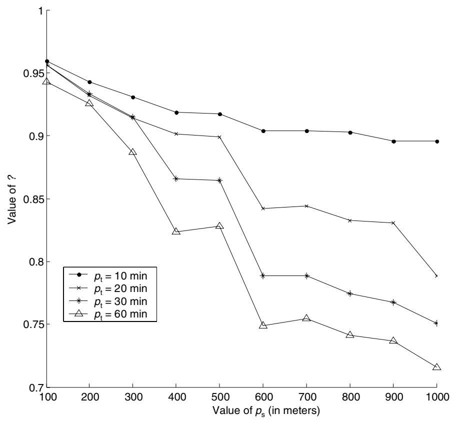

Fig. 6. The behavior of the γ parameter vs. temporal and spatial coefficients of the vigilance threshold

Up to this point we have concluded that the C-nT model achieves good precision with limited memory requirements, which are very important parameters for mobile context-aware systems. However, we need to perform some tests with C-nT in order to determine the best value for the spatiotemporal parameter vigilance ρ. In other words, we aim to determine the best values for both spatial ρ_s and temporal ρ_t vigilance coefficients in order to obtain the highest precision with low memory requirements. We introduce the weighted sum γ as follows:

$$\gamma = w \cdot p + (1 - w) \cdot (1 - a)$$

where a is the proportion of the generated clusters by the classifier (i.e., the size of the knowledge base in clusters) out of the total movement patterns (i.e., the size of the database in patterns), that is: $a = |C|/|U|$; $|C|$ is the cardinality of the set C. The weight value $w \in [0, 1]$ indicates the importance of precision and memory requirements; a value of $w = 0.5$ assigns equal importance to a and p. In our assessment, we set $w = 0.7$. We require that a assumes low values minimizing the storage cost of the classifier. A low value of a indicates that the applied classifier appropriately adopts and learns the user movements without retaining redundant information. The value of γ indicates which values of ρ_s and ρ_t maximize the precision while, on the same time, minimize the memory requirements. Hence, our aim is to achieve a high value of γ indicating an adaptive classifier with high value of precision along with low storage cost. As illustrated in Figure 6, we obtain a global maximum value for γ once $\rho_s = 100m$ and $\rho_t = 10min$ (which are the setting values during the experiments – see Table 2).

7 Prior Work

Previous work in the area of mobility prediction includes the model in [7], which uses Naïve Bayes classification over the user movement history. Such model does not deal with fully / semi- random mobility patterns and assumes a normal density distribution for the underlying data. However, such assumptions are not adopted in our model as long as mobility patterns refer to real human traces with unknown distribution. Moreover, the learning automaton in [8] follows a linear reward-penalty reinforcement learning method for location prediction. However, such model does not provide satisfactory prediction accuracy, as reported in [8]. The authors in [9] apply evidential reasoning in mobility prediction when knowledge on the mobility patterns is not available (i.e., similarly to this paper). However, such model assumes large computational complexity (due to the adopted Dempster-Schafer algorithm) once the count of possible user locations increases and requires detailed user information (e.g., daily profile, preferences, favorite meeting places). Other methods for predicting trajectory have also been proposed in the literature [10] but these have generally been limited in scope since they consider rectilinear movement patterns only (e.g., highways) and not unknown patterns. A closely related work to ours has been reported in [11], where a GPS system is used to collect location information. The proposed system then automatically clusters GPS data taken into meaningful locations at multiple scales. These locations are then incorporated into a similar Markov model to predict the user's future location. The authors in [16] adopt a data mining approach (i.e., rule extraction) for predicting user locations in mobile environments. This approach achieves prediction accuracy lower than ours (i.e., in the order of 80% for deterministic movement). In [17], the authors adopt a clustering method for the location prediction problem. Prediction accuracy is still low (in the order of 66% for deterministic movement). The authors in [18] introduce a framework where for each user an individual function is computed in order to capture its movement. This approach achieves prediction accuracy lower than ours (i.e., in the order of 70% for deterministic movement). In [19], the authors apply movement rules in mobility prediction given the user's past movement patterns. Prediction accuracy is still low (i.e., in the order of 65% for deterministic movement). The authors in [20] introduce a prediction model that uses grey theory (i.e., a theory used to study uncertainty). This approach achieves prediction accuracy lower than ours (i.e., in the order of 82% for deterministic movement).

8 Conclusions

We presented how ML techniques can be applied to the engineering of mobile context-aware applications for location prediction. Specifically, we use ART (a special Neural Network Local Model) and introduce a learning method. Furthermore, we deal with two training methods for each learning method: in the supervised method the model uses training data in order to make classification and in the zero-knowledge method the model incrementally learns from unsuccessful predictions. We evaluated our models with different spatial and temporal parameters. We examine the knowledge bases storage cost (i.e., emerged clusters) and the precision measures (prediction accuracy). Our findings indicate that the C-nT model suits better to context-aware systems. The advantage of C-nT model is that (1) it does not require pre-existing knowledge in the user movement behavior in order to predict future movements, (2) it adapts its on-line knowledge base to unseen patterns and (3) it does not consumes much memory to store the emerged clusters. For this reason, C-nT is quite useful in context-aware applications where no prior knowledge about the user context is available. Furthermore, through experiments, we decide on which vigilance value achieves the appropriate precision w.r.t. memory limitations and prediction error. Finally, in the Neural Networks Local Models literature there are other models (e.g., Self-Organizing Maps) that we have not examined in this paper. We intent to implement and evaluate them with C-nT by means of knowledge base requirements, precision of the location prediction and adaptation.

References

1. Dey, A.: Understanding and using context. Personal and Ubiquitous Computing 5(1), 4–7 (2001)
2. Hightower, J., Borriello, G.: Location Systems for Ubiquitous Computing. IEEE Computer 34(8) (August 2001)
3. Alpaydin, E.: Introduction to Machine Learning. MIT Press, Cambridge (2004)
4. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley-Interscience, Hoboken (2001)

5. Belogay, E., Cabrelli, C., Molter, U., Shonkwiler, R.: Calculating the Hausdorff Distance between Curves. Information Processing Letters 64(1), 17–22 (1997)
6. Site, `http://www.openstreetmap.org/traces/tag/Denmark`
7. Choi, S., Shin, K.G.: Predictive and adaptive bandwidth reservation for hand-offs in QoS-sensitive cellular networks. In: ACM SIGCOMM (1998)
8. Hadjiefthymiades, S., Merakos, L.: 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 8(4) (2003)
9. Karmouch, A., Samaan, N.: A Mobility Prediction Architecture Based on Contextual Knowledge and Spatial Conceptual Maps. IEEE Trans. on Mobile Computing 4(6) (2005)
10. Viayan, R., Holtman, J.: A model for analyzing handoff algorithms. IEEE Trans. on Veh. Technol. 42(3) (August 1993)
11. Ashbrook, D., Starner, T.: Learning Significant Locations and Predicting User Movement with GPS. In: Proc. Sixth Int'l Symp. Wearable Computes (ISWC 2002), October 2002, pp. 101–108 (2002)
12. Priggouris, I., Zervas, E., Hadjiefthymiades, S.: Location Based Network Resource Management. In: Ibrahim, I.K. (ed.) Handbook of Research on Mobile Multimedia. Idea Group Inc. (May 2006)
13. Curewitz, K.M., Krishnan, P., Vitter, J.S.: Practical Prefetching via Data Compression. In: Proceedings of ACM SIGMOD, pp. 257–266 (1993)
14. Narendra, K., Thathachar, M.A.L.: Learning Automata – An Introduction. Prentice Hall, Englewood Cliffs (1989)
15. Cheng, Jain, R., van den Berg, E.: Location prediction algorithms for mobile wireless systems. In: Wireless Internet handbook: technologies, standards, and application, pp. 245–263. CRC Press, Boca Raton (2003)
16. Yavas, G., Katsaros, D., Ulusoy, O., Manolopoulos, Y.: A data mining approach for location prediction in mobile environments. Data and Knowledge Engineering 54(2) (2005)
17. Katsaros, D., Nanopoulos, A., Karakaya, M., Yavas, G., Ulusoy, O., Manolopoulos, Y.: Clustering Mobile Trajectories for Resource Allocation in Mobile Environments. In: Berthold, M.R., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 319–329. Springer, Heidelberg (2003)
18. Tao, Y., Faloutsos, C., Papadias, D., Liu, B.: Prediction and Indexing of Moving Objects with Unknown Motion Patterns. In: ACM SIGMOD (2004)
19. Nhan, V.T.H., Ryu, K.H.: Future Location Prediction of Moving Objects Based on Movement Rules. In: ICIC 2006. LNCIS, vol. 344, pp. 875–881. Springer, Heidelberg (2006)
20. Xiao, Y., Zhang, H., Wang, H.: Location Prediction for Tracking Moving Objects Based on Grey Theory. In: IEEE FSKD 2007 (2007)