# A Partition-based Partial Personalized Model for Points of Interest Recommendations

Elahe Naserian*, Xinheng Wang†, Keshav Dahal*, Jose M. Alcaraz-Calero*, Honghao Gao‡
* University of the West of Scotland, Paisley, Scotland, UK
Email: {elahe.naserian, keshav.dahal, jose.alcaraz-calero}@uws.ac.uk
†University of West London, London, UK
Email: {xinheng.wang}@uwl.ac.uk
‡ Shanghai University, Shanghai, China

**Abstract**—Providing location recommendations has become an essential feature for location-based social networks (LBSNs) because it helps users explore new places and makes LBSNs more prevalent to users. Existing studies mostly focus on introducing new features that affect users' check-in behaviours in LBSNs. Despite the difference in the type of the features exploited, they mostly follow the same principle - characterizing dependencies between the probability of a user visiting a points-of-interest (POI) and each feature separately. However, the decision of a user on where to go in an LBSN is driven by multiple features that act simultaneously. On the other hand, applying a full model which considers all the features jointly suffers from over fitting, as for each user there is limited available data. We propose an intermediate solution by fragmenting the model into multiple partial models which each takes the subset of the features as the input. The proposed approach focuses on building the personalized partial models which then combine them by applying an additive approach. We further introduce a partition-based approach to identify the hidden patterns from the geographically clustered check-in data. Experiments on two datasets from Foursquare show that our proposed method outperforms the state-of-the-art approaches on POI recommendation.

**Index Terms**—Location-based social network (LBSN), Recommendations, Point-of-Interest (POI)

## 1 Introduction

With the advancement of location acquisition devices and wireless communications [1], [2] location-based social networks (LBSNs), such as Foursquare, Gowalla, and Facebook places, have attracted millions of users. In an LBSN, users can share experiences of visiting locations, also known as points-of-interest (POIs), like restaurants, stores, and museums. The act of visiting a POI is known as a check-in activity and can be used to learn the preferences of the user of the LBSN and to utilize them for making POI recommendations [3]. This would help users to explore new places, and also makes LBSNs more attractive to them by increasing its effectiveness based on personalized recommendations.

A traditional approach to address the creation of recommendation systems is applying collaborative filtering (CF) techniques [4], [5], [6], [7], [8], [9], [10] over users' check-in data to generated models about their preferences. Although these techniques are a good starting point for simple recommendation systems, they lack to consider contextual information about the check-in actions performed by users, and instead the merely focus on defining similarity functions to allow the comparison of behavior between users taking in considering mainly the visiting places.

There are other more appealing approaches to address the creation of recommendation systems which make use of different features available in the check-in data such as geographical and temporal information [11], [12], [13], [14], [15].

Despite the difference in the type of features exploited, most of the existing approaches follow the same principle

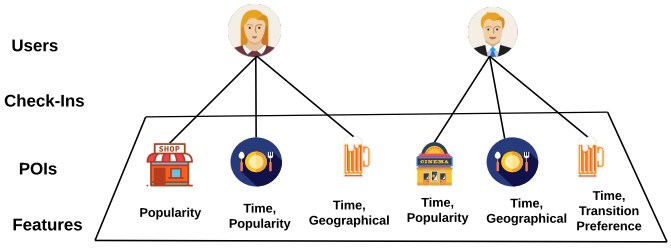



Fig. 1: Influential features for visiting various POIs for different users.

- a Naive Bayes model that characterizes dependencies between the outcome (probability of a user visiting a POI) and each feature (geographical, time, transition, etc.) separately. A clear example is proposed in [15] where authors model the user preference, social influence, and geographical influence independently, and then conclude the final result by simply fusing the individual models.

However, Naive Bayes models are based on the strong assumption that features are independent. This approach may be not accurace enough to model scenarios where a user who mostly visits POIs in nearby locations also travels long distances for the most popular POIs which is in fact something very common nowadays []. Therefore, to model the users' preferences, incorporating the interactions between features is essential.

Fig. 1 shows an example of the influential features pre-

viously described and how different users are influenced differently by such features when they carry out different activities. For example, someone can choose to go for the nearby bars during the evening time (geographical influence and time influence), and the other one prefers to go for the bar after having the dinner (time influence and transition influence).

In this paper, we argue that the decision of a user on where to go is driven by multiple features that act simultaneously. Also, different users may be affected differently by these features. These two assumptions are mostly ignored in state-of-the-art studies. This lead to a significant degradation of accuracy in the existing POI recommendation systems which has been our main motivation. Therefore, we aim to propose a personalized POI recommendation model which takes the features' interactions into account. For this purpose, we formulate the POI recommendation as a supervised learning problem given multiple influential features. Instead of defining a sophisticated model that exploits all the features such as the proposed by [16], [17], [18], we develop a principled way to combine multiple partial models. The proposed framework has been named as Additive Personalized Point-of-Interest Recommendator (APPR) focuses on building partial models about user preferences to learn the binary decision tree classifier from the subsets of the features. Then, the set of partial models are combined by applying an additive approach. We finally apply the proposed model to compute the probability of a user visiting a POI to recommends the $top(K)$ POIs with the highest probabilities.

Our APRM Recommendation Model has major differences with respect to existing methods. First, it exploits the features' interactions in POI recommendation and thus not assuming independence between them assuming that user mobility in LBSNs is driven by several features acting synchronously. Second, it train the model about the importance of the different features for differentiating the interesting POIs for each user from those which do not attract them. Third, it is a non-parametric approach. Most of the algorithms in the state-of-art are sensitive to the values of their parameters. As such, the performance of the recommendation may easily degrade if the values do not match the specific dataset. Finally, it does not require assumptions about the underlying statistical models. Other proposed methods mostly build the recommendation model based on the assumption about the form of distribution of the check-in data.

The main contributions of this paper can be summarized:

- It is proposed an Additive Personalized POI Recommendation model (APPR) which takes the features' interactions into account. For this purpose, it is formulated POI recommendation as a supervised learning problem given multiple influential features.
- It is further designed a partition-based approach to identify hidden patterns of partitions obtained from geographically clustered check-in data. This approach is embedded with a feature selection process which chooses the most appropriate features to represent an underlying PRM pattern.
- It has been conducted extensive experiments to evaluate the performance of APPR using two large-scale real data sets obtained from Foursquare. Experimental results show that APPR outperforms over other state-of- the-art recommendation techniques including the Naive Bayes [19], [20], [21], [22], [23], [24] approaches and Full-joint models [16], [17], [18] in terms of recommendation precision, recall, and accuracy.

The remainder of this paper is organized as follows. Section 2 highlights related work. Section 3 provides the full explanation of the proposed recommendation approach. In Section 4 we propose our partition-based approach. We introduce a set of influential features in Section 5. In Sections 6 and 7, we evaluate the POI recommendation model and analyze the experimental results. Finally, we conclude the paper in Section 8.

## 2 Related Work

With the rapid increase of LBSNs like Foursquare, Gowalla, Facebook places, etc., POI recommendation has become prevalent [3]. In a very recent work, Liu in [25] evaluates the state-of-the-art POI recommendation models. In general, existing POI recommendation approaches can be divided according to the features they exploit and the recommendation model they apply.

**Influential features.** Existing approaches for location recommendation in LBSNs differ mainly in terms of the different aspects of the check-in data that they exploit. The geographical proximity between POIs affects the check-in behaviours of users on the POIs. In this regard, several methods have been proposed to exploit geographical influence for improving the quality of location recommendation. Some studies define a distance range for recommended locations [11], [26], some gives more priority to the closer locations [27], and some researchers model the distance between two locations visited by the user as a distribution, like a multi-center Gaussian model [12], or personalized distribution [15], [19].

Time is another feature which influences the visiting behaviour of users in LBSN, i.e. visiting a restaurant at noon or visiting a bar at night. The time-dependent recommendation techniques split a day into time slots to infer users' preferences on locations at each time slot [20], [21], [28], [29]. It has been shown that the user's preference transitions over POIs are not random. In this regard, Some studies investigate the patterns of users' preference transitions over POIs as the influential feature in users' visiting behaviour [23], [30], [31].

Different approaches have been developed to exploit different types of the check-in information, however, these approaches are usually developed for a specific type of feature and it is difficult to generalize them to handle another type of feature.

**Recommendation model.** Despite the difference in the type of features exploited, the POI recommendation models follow one of two strategies: Naive Bayes model, or Full-joint model.

The first strategy is based on this assumption that features are independent given the outcome, such that the probability of visiting a POI and each influential feature is

modelled separately. The final result is then calculated by fusion of the individual models. Most of the state-of-the-art studies apply this model [23], [19], [24]. In [22], authors define a conditional probability model incorporating three influential features, geographical, social and temporal features, which each is modelled independently of the others. The authors in [31] apply the product fusion rule to combine the models obtained from geographical, social and sequential features. In [21] the authors separately consider spatial and temporal features of user activities and then propose a context-aware fusion framework to infer user activity preference. However, the independence assumption does not match with the characteristics of the check-in data; the mobility of users in an LBSN is driven by several features that act simultaneously.

Another direction is to consider the Full-joint model, in which the probability of visiting a POI depends on all influential features jointly [16], [17], [18]. In [16] the authors first define a set of features that may drive users' behaviour, then they apply a supervised learning model to predict the next top-k POIs that user may visit. The same strategy has been applied in [17] and [18] with incorporating different influential features. This model could be practical if a user has sufficient historical check-in data. However, in a real-world LBSN, users' check-in data is often sparse, and then applying the full model might lead to the overfitting and low accuracy of the POI recommendation.

We can distinguish our work from previous studies as follows: (1) In contrast with the Naive Bayes models, we take the features' interactions into account. (2) Contrary to the Full-joint models, we fragment the model into multiple partial models which each takes the subset of the features as the input.

## 3 Modelling Additive Personalized POI Recommendation (APPR)

In this section, we investigate the POI recommendation problem through applying a supervised learning framework. First, we build the partial models that capture the subsets of the influential features. The final model, then, is obtained by combining the partial models additively. The recommendation model then is applied to determine the probability of a user visiting a location.

### 3.1 Problem Definition

The problem of POI recommendation can be formulated as finding a function $f : X \rightarrow Y$ where $X$ is the input feature space and $Y$ is the output space. In our setting, the input space is the set of influential features, and the output space is the set of POIs. Let $D = \{\{x(u,t), y(u,t)\}, u = 1, .., N\}$ be a dataset of $N$ users, which $y(u,t)$ is a visited POI by the user $u$ at time $t$ and $x(u,t)$ encodes the values of the features of the visited POI. To calculate the output $y(u,t)$ of the feature space $x(u,t)$, we can only use data $\{x(u,t'), y(u,t')\}t' = 1, .., t-1$ to learn the function $f$.

We consider a probabilistic approach where we learn the distribution, $P(Y|X)$, from the training set where $X = \{X_1, X2, ..., X_F\}$ denotes the set of $F$ features and $Y$ refers to the POI. To obtain the above distribution, we learn a supervised model over the feature space to calculate the probability of a user visiting a POI. To train the model, we consider all the POIs visited before the prediction time $t$ as the positive instances. Positive instances are those locations (POIs) that have been visited by the users and negative instances are those locations not visited by her. Then, we retrieve the negative labeled instances by sampling at random across all other places in the city up to the point where there is the same number of positives and negatives. This approach has been already used by XXXX []. This method of training a model by providing feedback in the form of user preference corresponds to an effective reduction of the ranking problem to a binary classification task which has been established in the past [32].

Applying the supervised learning for recommendation problem has been previously used in the literature [16], [17], [18], and it brings the following advantages; This approach allows us to train the model what the essential characteristics are for differentiating the places that attract a user from those which would not. Contrary to the state-of-the-art approaches which consider the same set of features with equal importance for all users, we build a personalized model for each user which determines the significance of features according to their influence on the user's movement. Furthermore, this approach let us take the interactions between the features into account which is ignored by the previous works and results in the more accurate recommendation.

Accordingly, we formulate the problem of POI recommendation as follows:

- Given: a dataset containing users' historical check-in data; the training and test instances drawn from it.
- Find: a recommendation model based on the training instances.
- Objective: improving the performance of the recommendation calculated by evaluating the model on the test instances.

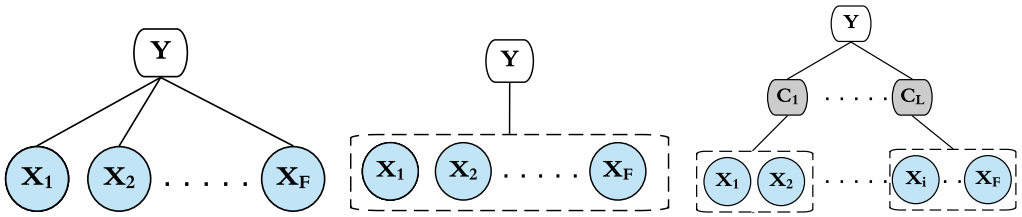
Our goal is to built a recommendation model, denoted as $H$, which takes a POI's feature vector $x$ as its input and then outputs the $H(x)$ as the probability of the user visiting the POI. Then the $top(K)$ POIs with the highest probabilities are returned as the recommendations. When the recommended POI is visited by the user, we consider the model to correctly recommend the POI. The notations used in this paper are summarized in Table I.

Our approach is focused on an off-line approach where every given time there is an update the model using the new historical data in order to allow the model to evolve its accuracy along the time. It has been decided an off-line approach to optimize response times into the recommendation framework.

### 3.2 Different Approaches

### 3.3 Theoretical Background

This is obvious that the more accurate the feature space is, the more precise the recommendation might be. In other words, incorporating more relevant features, result in more precise model. For example, considering both popularity and distance could be more effective than using each one




(a) Naive Bayes model (b) Full-joint model (c) Fragmented model

Fig. 2: Different Recommendation Models.

alone. However, as the feature space $X$ increases, more data is needed for accurately estimating the distribution $P(Y|X)$. Therefore, the primary challenge is to build a model that can efficiently utilize the available features from the limited training instances.

A promising way is to consider the full model (Fig. 2b), such that the output $Y$, the probability of visiting a POI, depends on all features $X_i$ jointly, similar to the proposed model in [16]. Theoretically, if a user has sufficient historical check-in data, this could be the ideal model. However, in a real-world LBSN, users' check-in data is often sparse, and then applying the full model might lead to the overfitting and low accuracy of the probability estimation. The opposite case is considering the Naive Bayes approach in which the dependencies between the output $Y$ and each feature $X_i$ is modelled separately, Fig. 2a. Most of the state-of-the-art studies in the POI recommendation follow this approach [19], [22], [23], [24]. However, the Naive Bayes model is based on the strong assumption that features are independent given the outcome, which does not match the real characteristics of the check-in data in LBSN - the mobility of users in an LBSN is driven by several features that act simultaneously [16].

Unlike these two extreme approaches, we propose an intermediate solution by fragmenting the model into multiple partial models which each model take the subset of $X$ as the input features, Fig. 2c. We learn decision tree binary classifier $R_l$ from the input feature $C_l$. We call it Partial Personalized Model (PRM) because it builds the model by considering not all the features, but the subset of them. For each user, the model learns the importance of different features for differentiating the interesting POIs from those which don't attract the user. We then formally define a PRM as follows.

*Definition 1 (Partial Personalized Model (PRM) )* Let $C_l \subset X$ and the training set containing the values of features in $C_l$ be denoted as $D_l$. PRM $R_l$ is a decision tree binary classifier learned from $D_l$, such that $R_l^+$ is used to calculate the probability of visiting the POI considering the feature subset $C_l$, denoted as $Prob(1|C_l)$, and $R_l^-$ as the probability of not visiting the POI, denoted as $Prob(-1|C_l)$.

In this section we learn the PRMs from all the possible subsets of the feature space then combine them to build the recommendation model. In Section 4, we introduce a partition-based approach for learning the PRMs in an efficient manner.

To build PRMs and then collectively utilize them, we incorporate an additive learning process based on the boosting algorithm [33]. This iteratively process chooses a PRM at each layer $m$ and builds a predictive model with $M$ layers of PRMs at the end of $M$ iterations. From each $R_m$, we learn hypothesis $h_m$ where $h_m(x) \in \mathbb{R}$, $x$ is a feature vector of an instance POI and $\mathbb{R}$ is the domain of real numbers. Consequently, $h_m^+$ has been learned from $R_m^+$, and $h_m^-$ has been learned from $R_m^-$. At the end of $M$ iterations, the final hypothesis is modelled as:

$$H(x) = \sum_{m=1}^{M} h_m(x) \tag{1}$$

where itself can be divided to $H^+(x)$ and $H^-(x)$, which are obtained from $h^+(x)$ and $h^-(x)$, accordingly. As we are looking for determining the probability of visiting a POI, unless otherwise specified, $H(x)$ represents the $H^+(x)$. This additive learning approach mainly applies the same principle of real-valued confidence-rated boosting approach [34], which allows using the probability estimates from decision trees , $R_m$, to update the additive model. The objective is to find the best $h_m$ at each iteration which yields the least prediction error on training instances. To this end, we first recode the output of the training instances with a 2-dimensional vector $V = (v_1, v_2)$, such that $V = (1, -1)$ if $x$ is the negative instance, and $V = (-1, 1)$ if $x$ is the positive instance. Then the generalization of the exponential loss function L at iteration $m$ follows:

TABLE 1: KEY NOTATIONS

| Symbol | Description |
|---|---|
| $X$ | set of features |
| $Y$ | set of POIs |
| $X_i$ | $ith$ feature |
| $D$ | dataset of check-ins |
| $C_l$ | a subset of features |
| $D_l$ | dataset containing the values of features in $C_l$ |
| $M$ | number of iterations |
| $R_l^+$ | probability of visiting a POI considering the feature subset $C_l$ |
| $R_l^-$ | probability of not visiting a POI considering the feature subset $C_l$ |
| $h_m^+$ | hypothesis learned from $R_l^+$ |
| $h_m^-$ | hypothesis learned from $R_l^-$ |
| $w(i)$ | weight of instance i |
| $H(x)$ | final hypothesis |
| $k$ | number of recommended POIs |
| $g$ | Number of clusters |

$$L(V, h_m) = e^{\Big(\frac{-1}{2} V^T h_m\Big)} \tag{2}$$

Where $V^T$ is defined as the transpose of V. To simplify the presentation, we use $h(x)$ to represent the $h_m(x)$ hereafter. Accordingly, the minimization problem is formulated as:

$$\begin{aligned} \underset{h(x)}{\arg\min} \quad & E\Big(exp\Big(\frac{-1}{2}(v_1 h^-(x) + v_2 h^+(x))\Big)|x\Big) \\ \text{subject to} \quad & h^-(x) + h^+(x) = 0 \end{aligned} \tag{3}$$

The Lagrange of this constrained optimization problem can be written as:

$$\begin{aligned} exp\Big(-h^-(x)\Big) Prob(-1|x) + exp\Big(-h^+(x)\Big) Prob(1|x) \\ -\lambda\Big(h^-(x) + h^+(x)\Big) \end{aligned} \tag{4}$$

where $\lambda$ is the Lagrange multiplier. This model of the error rate as minimization problem has been already addressed in [34]. Thus, taking derivatives with respect to $h$ and $\lambda$, we reach

$$\begin{aligned} -exp\Big(-h^-(x)\Big) Prob(-1|x) - \lambda = 0, \\ -exp\Big(-h^+(x)\Big) Prob(1|x) - \lambda = 0, \\ h^-(x) + h^+(x) = 0 \end{aligned} \tag{5}$$

Solving this set of equations, we obtain the $h(x)$

$$h(x) = \frac{1}{2}\Big(\frac{logProb(1|x)}{logProb(-1|x)}\Big) \tag{6}$$

unless otherwise specified, $h(x)$ represents the $h^+(x)$ and $h^-(x)$ is calculated in opposite way of $h^+(x)$. We can use probability estimates from decision trees, $R$, as the approximations to the conditional expectation.

$$h(x) = \frac{1}{2}\Big(\frac{logR^+(x)}{logR^-(x)}\Big) \tag{7}$$

### 3.4 Additive Recommendation Model

Based on the theoretical analysis given in Subsection 3-2, we first propose our POI recommendation model $BuildAPPR$ which mainly follows the SUMME.R boosting algorithm [34]. This function will be executed by the dataset of every users individually. Then, the function will iteration a number of times in order to allow the model of user's preference to converge. Furthermore, we propose $PruneAPPR$ function to reevaluate the resulted model, to prevent the final hypothesis $H$ from overfitting and also reduce the size of the PRM set. This function will be executed against all the user's preference model individually following a similar iterative approach in order to trade-off time and accuracy. Accordingly, we divide the dataset of check-in records of user $u$, $D_u$, into two subsets, growing dataset (GrowSet) and validation dataset (PruneSet). The former dataset is used for the $BuildAPPR$ function and the latter one is used for the $PruneAPPR$ function.

**Algorithm 1** : BuildAPPR

**Input**:
**GrowSet**: Growing dataset.
**Output**:
**H**: The hypothesis of the final model.
**RList**: Set of PRMs.

1: **buldAPPR(user u, iterations M)**
2: **for** $m = 1..M$ **do**
3: **Normalize** the weights of the training instances.
4: **Learn** the Partial Personalized Models (PRMs).
5: **Select** a PRM as $R_m$.
6: **Obtain** the probability estimates from $R_m$.
7: **Set**:
$h_m(x) = \frac{1}{2}\Big(\frac{logR_m^+(x)}{logR_m^-(x)}\Big)$
8: **Update** weights:
$w_{m+1}(i) = w_m(i).exp\Big(-\frac{1}{2} V_i^T logR_m(x_i)\Big)$
9: **end for**
10: **Final hypothesis** is defined as:
11: $H(x) = \sum_{m=1}^{M} h_m(x)$
12: **Return: RList**

#### 3.4.1 BuildAPPR Function

According to the Algorithm1, we first normalize the weight of the training instances at each iteration $m$ to make it a probability distribution such that $\sum_i w_m(i) = 1$, where $w_m(i)$ is the weight of instance $i$ at iteration $m$. The set of PRMs, then, is learned from the training data. Afterwards, we choose a PRM $R$ randomly and add that to the $RList$. According to the probability estimates calculated from the chosen PRM, $R^+$ and $R^-$, $h_m(x)$ is obtained, equation 7. To give the data instances which are not recognized by $h_m$ correctly, more attention in the next iteration, we exponentially lower the weights on those instances which are correctly recognized by $h_m$ and increase the weight of those which are incorrectly recognized by $h_m$. The weights of the instances are updated as follows:

$$w_{m+1}(i) = w_m(i).exp\Big(-\frac{1}{2} V_i^T logR_m(x_i)\Big) \tag{8}$$

At the end, the result is a list of $PRMs$, $RList = \{R_1, R_2, ..., R_M\}$ , which their corresponding hypothesis are $\{h_1, h_2, ..., h_M\}$. The final hypothesis can be obtained according to the equation 1. To convert the $H(x)$ to a probability distribution, we set $H(x) = exp\Big(H(x)\Big)$, and then we normalize it.

#### 3.4.2 PruneAPPR Function

This part addresses the question of whether all of the resulted PRMs are improving the performance of our model. PruneAPPR function, shown in Algorithm 2, is used to reevaluate the PRM set returned by $BuildAPPR$ through calculating the error rate $error$ on the PruneSet to solve a possible overfitting problem. The goal is obtaining the subset of PRMs which gives the best performance on PruneSet. We take $h^+(x) > 0$ as the positive prediction, and

**Algorithm 2** : PruneAPPR

**Input**:
**PruneSet**: Validation dataset.
**RList**: The output of the BuildAPPR.
**Output**:
**RList**: Pruned PRM set.

**pruneAPPR(user u, iterations M)**
$Stop = False$.
**Calculate** $error$ using PruneSet.
**Set** $error_{min} = error$.
**repeat**
**Pop** the top $R$ from the queue of $RList$.
**Calculate** $error$ using PruneSet.
**if** $error < error_{min}$ **then**
**set** $error_{min} = error$
**else**:
**Push** $R$ back to $RList$
**Set** $Stop = True$
**end if**:
**until** $(Stop = True)$ *or* $(only\ one\ PRM\ left\ in\ RList)$
**Return: RList**

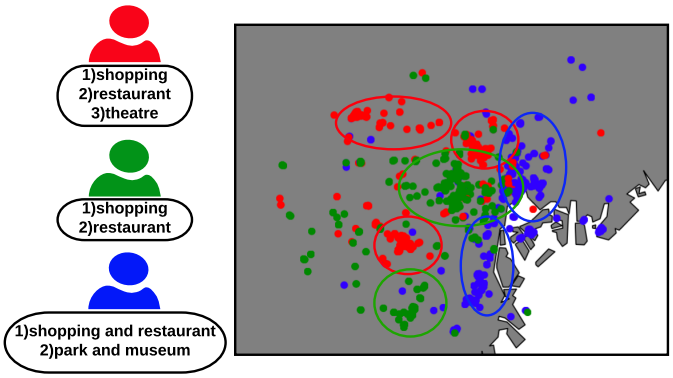
$h^-(x) > 0$ as the negative result. The error then is calculated as the number of instances which are not recognized correctly by the model. This function repeatedly removes an R from RList until the minimum value of $error$ is reached or there is only one PRM left in RList. In the end, the RList with the minimum error is returned. The error is defined as the number of incorrect recommendations given to the user, i.e. recommendations that has not been converted to real check-in activities.

## 4 Partition-based Additive Personalized POI Recommendation (P-APPR)

The process of building the PRMs is very efficient in terms of accuracy but it is inefficient in terms of scalability as we identify PRMs from all the possible subsets of the feature space. In this section, we propose an partition-based approach to learn the partial models. This approach improves the performance of the recommendation, and also reduces the training time while keeping and even improve accuracy.

For the partition-based approach, instead of identifying the PRMs from all the combinations of features, we identify one PRM for each partition. The important features automatically be learned from the data, instead of randomly choosing them.

This approach is possible due to the fact that Users' activities in LBSNs often present strong preference bias in the areas that they frequently visit [21], [10]. In other word, similar activities are driven by the similar influential features. Users only perform a few types of activities (i.e., visit POIs of a few categories) in each of their frequented areas. For example, the area(s) a person goes for the shopping is different from the the area(s) he/she goes for visiting the museums or art galleries. Fig. 3 shows check-ins of three users (represented by red, green, blue colors) in Tokyo in our dataset. First, we realize that most of the user's check-ins are occurring in certain geographic areas, as plotted in




(a) Tokyo

Fig. 3: Spatial Distribution of checking of three users in Tokyo which are plotted in red, green, and blue colors, respectively

the Figure. This observation shows the strong geographic preference of the check-in behaviors. It also indicates that different users usually have their own frequented areas. Second, by examining users' activities in their frequented areas, we find out that their activities are often limited to a few kinds of activities for the majority of their frequented areas. The dominant activities for the frequented areas for each user is shown in Fig. 3. Third, the check-ins from the same frequented area show strong similarities in their influential features.

To this end, we design a partition-based approach to identify the hidden patterns of partitions obtained from geographically clustered check-in data. Assuming that there exists a pattern within each partition, we learn a binary decision tree classifier from each partition to represent the underlying pattern, that we call it Partition-based Partial Personalized Model (PPRM). Thus, a feature selection process is embedded in our Partition-based PRM Discovery which chooses the most appropriate features to represent the underlying PPRM. We then formally define a PPRM as follows.

*Definition 2 (Partition-based Partial Personalized Model (PPRM) )* Let $P_l$ be a partition of the check-in data. PPRM $R_l$ is a decision tree binary classifier learned from $P_l$, which involves the feature subset $C_l$. Accordingly, $R_l^+$ is used to calculate the probability of visiting the POI considering the feature subset $C_l$, $Prob(1|C_l)$, and $R_l^-$ as the probability of not visiting the POI, $Prob(-1|C_l)$.

To capture the PPRMs from the partitions at different granularity levels, we cluster the check-ins with different similarity degrees in terms of geographical location by varying the number of clusters. It makes natural grouping of data by different locations in order to better understand the user's preferences. Notice that every of the dataset analyzed contains the check-in preferences for a given city. It means that such city will be cluster in different geographically per each user. It should be noted that the extracted partitions are not-mutually-exclusive from which the features can be learned more efficiently. For example, we cluster the check-ins of a user in Fig. 4 at different levels which different cluster numbers, g =1,2,3 and 4. The patterns then are identified from the extracted clusters. In order to combine the

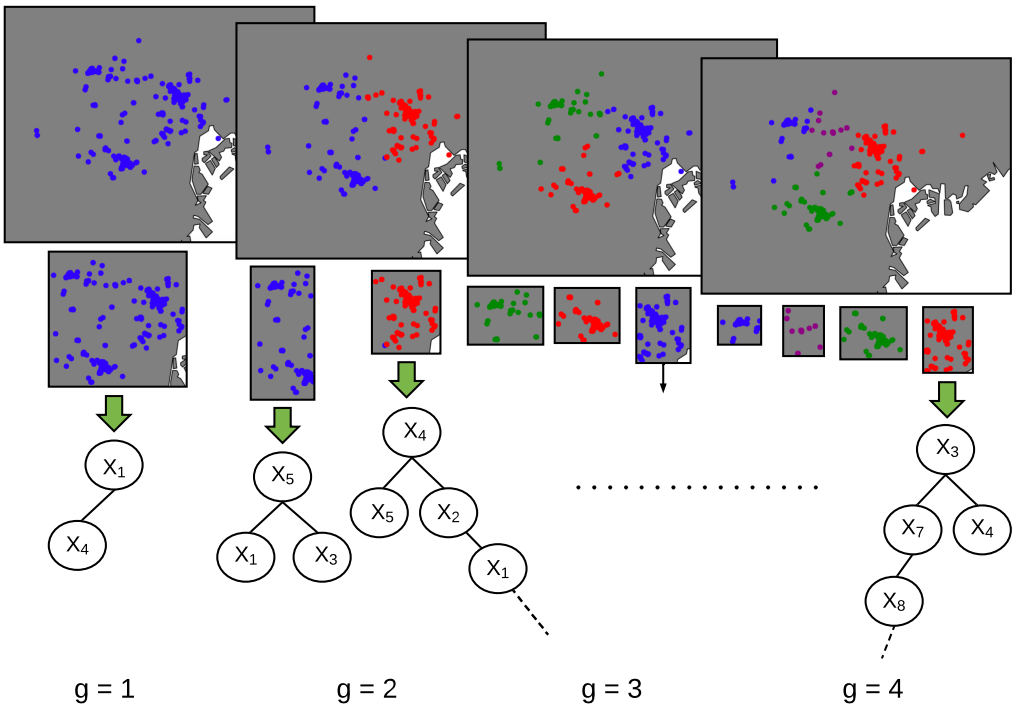



Fig. 4: An illustration of the PPRM construction. Through the partitioning process, the check-in instances are segmented into non-mutually-exclusive clusters (the points filled with the same color). We learn a decision tree binary classifier from this type of spatial distribution to represent a Partition-based Partial Personalized Model (PPRM).

TABLE 2: DATASET STATISTIC

| Dataset | New York (Foursquare) | Tokyo (Foursquare) |
|---|---|---|
| $Users$ | 824 | 1,939 |
| $POIs$ | 38,336 | 61,858 |
| $Check-ins$ | 227,428 | 573,703 |

obtained PPRMs, we apply the same strategy as Subsection 3.3. Applying a partition-based approach for identifying the PPRMs take the spatial characteristics of the check-in data into account, which leads to the more accurate recommendation. Besides, it reduces the training time in comparison with the previous approach which considers all the possible subsets of the feature space.

## 5 Influential Features

We define a set of features $X = \{X_1, X_2, ..., X_F\}$ such that cover different aspects of users' movements in LBSNs; the features covering an individual user's preferences, such as historical visits, and those derived taking the knowledge about the whole system into account such as the popularity of places, their geographic distance and user transitions between the places. We also define a set of features that exploit the temporal information of users' movements. We consider $t^{'}$ and $y^{'}$ as the time and location of the current check-in, where $C(y^{'})$ determines the category of POI $y^{'}$, and $tod(t^{'})$ returns the hour of a day, and $dow(t^{'})$ returns the day of the week of time $t^{'}$. The value of all the features is calculated regarding data up to the current time, $t^{'}$. Note that employed features depend on the task, and different type of features can be defined according to the application.

*POI Preferences:* The number of visits of user $u$ at POI $p$. This feature measures to what extent the next check-in of a user is likely to appear at a place that has been visited by the user in the past. Formally we have:

$$X_1(p) = |\{(y,t) \in D_u : t < t^{'} \wedge y = p\}|$$

*Categorical Preferences:* To identify the importance of different categories of POIs (cinema, coffee shop, restaurant etc.) for a given user $u$, we consider the number of check-ins user $u$ has performed at a specific category:

$$X_2(p) = |\{(y,t) \in D_u : t < t^{'} \wedge C(y) = C(p)\}|$$

where $C(p)$ is the category of POI $p$.

*POI Popularity:* The total number of check-ins that have been performed by all the users $U$ in the dataset in a POI $p$:

$$X_3(p) = |\{(y,t) \in D : t < t^{'} \wedge y = p\}|$$

*Geographic Distance:* Considering $y^{'}$ as the current location of user $u$, we measure the distance between POI $p$ and the current location as an influential feature:

$$X_4(p) = dist(y^{'}, p)$$

*POI Transition Preference:* The user's transition between the POIs is not random [23]. Considering $T_u$ as the set of tuples for the POIs involved in consecutive transitions before the current time $t^{'}$ for user $u$, we define the following feature:

$$X_5(p) = |\{(v_1, v_2) \in T_u : v_1 = y^{'} \wedge v_2 = p\}|$$

*Category Transition Preference:* Identifies the preference of a given user $u$ in transition between the category of the current location and the target POI's category:

$$X_6(p) = |\{(v_1, v_2) \in T_u : C(v_1) = C(y^{'}) \wedge C(v_2) = C(p)\}|$$

*POI Transition Popularity:* The total number of transitions that has been done between the the current location ant the target POI $p$, by all the users:

$$X_7(p) = |\{(v_1, v_2) \in T : v_1 = y^{'} \wedge v_2 = k\}|$$

*Category Transition Popularity:* The total number of transitions that has been done between the POIs with the same category of the current location, and the POIs with the same category as the target location's category, by all the users:

$$X_8(p) = |\{(v_1, v_2) \in T : C(v_1) = C(y^{'}) \wedge C(v_2) = C(p)\}|$$

*POI Time-aware Popularity:* We also take the temporal pattern of visiting a POI into our consideration as the influential feature. We define the POI Hour Popularity and POI Day Popularity, as the sum of past check-ins at a POI in a given hour $h$ of the day and a given day $d$ of the week.

$$X_9(p) = |\{(y, t) \in D : t < t^{'} \wedge y = p \wedge tod(t) = tod(t^{'})\}|$$

$$X_{10}(p) = |\{(y, t) \in D : t < t^{'} \wedge y = p \wedge tow(t) = tow(t^{'})\}|$$

where $tod(t)$ returns the hour of a day, and $dow(t)$ returns the day of the week of time $t$.

*Category Time-aware Popularity:* Determining the temporal pattern of visiting a specific category during different hours and different days of the week:

$$X_{11}(p) = |\{(y, t) \in D : t < t^{'} \wedge C(y) = C(p) \wedge tod(t) = tod(t^{'})\}|$$

$$X_{12}(p) = |\{(y, t) \in D : t < t^{'} \wedge C(y) = C(p) \wedge tow(t) = tow(t^{'})\}|$$

For the specific user $u$, we extract the above feature set $X$ for each of the visited POIs. Following the steps in Subsection 3-3, the recommendation model for user $u$ is obtained.

For each POI, then, we extract the feature set $X$, and calculate the probability of visiting the POI by the user. Finally, the top-k POIs with the highest probability is returned as the recommendation list, $PList$.

## 6 EXPERIMENTAL SETTINGS

In this section, we describe our experimental settings for evaluating the performance of proposed POI recommendation models against the state-of-the-art POI recommendation techniques.

### 6.1 Datasets and Influential Features

We use two publicly available real check-in datasets [1] that were crawled from Foursquare between April 2012 to February 2013 [21]. It contains the check-ins of active users (defined as users who have visited at least three POIs per week) in two big cities, New York and Tokyo. The statistics of the datasets are shown in Table II.

### 6.2 Baselines

The baseline recommendation techniques implemented in our experiments are divided into two categories based on the employed strategy for the POI recommendation.

*Naive Bayes approach (base1)*: This approach characterizes dependencies between the probability of visiting the POI and each influential feature, $X_i$, separately. This technique is widely used in the literature [19], [20], [21], [22], [23], [24]. To implement this approach, we rank the POIs based on different features, and the final rank is the multiplication of the individual rankings.

*Full-joint model (base2)*: This approach characterizes dependencies between the probability of visiting the POI and all the influential features, $X$, jointly, i.e. [16], [17], [18]. We implemented this model according to [16], which applies the M5 decision tree to predict the $next - k$ POIs of a user. It should be noted that this approach is the most similar model to our proposed method.

### 6.3 Evaluation Metrics

To evaluate the quality of POI recommendations, we employ three standard metrics: accuracy, precision and recall. First, we assess the quality of recommending exactly the next location. To this end, we define accuracy:

- $Accuracy$ is 1, if the next POI is discovered among the top-k POIs, $PList$. Average accuracy is then calculated as the fraction of successful instances over the total number of recommendation tasks.

To evaluate the quality of location recommendations, it is also essential to find out how many locations visited by the target user in the testing dataset are identified by the recommendation method. For this purpose, we define precision and recall:

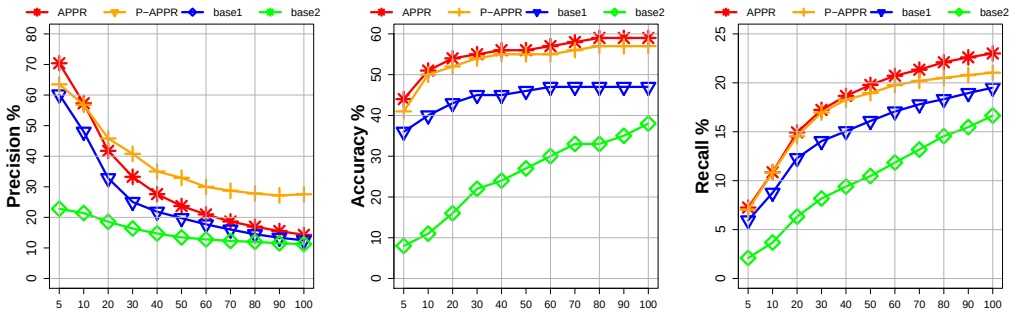
- $Precision$ defines the ratio of the number of discovered POIs to the $k$ recommended POIs, such that:

$$precision = \frac{numberofdiscoveredPOIs}{k}$$

- $Recall$ defines the ratio of the number of discovered POIs to the number of positive POIs, which have been visited by the target user in the testing set, such that:

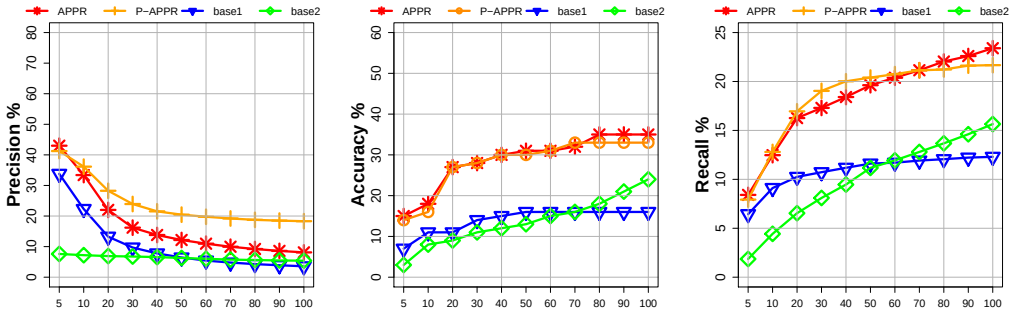
$$recall = \frac{numberofdiscoveredPOIs}{numberofpositivePOIs}$$

1. The check-in datasets used for our experiments can be downloaded from https://sites.google.com/site/yangdingqi/home/foursquare-dataset




(a) Tokyo (b) Tokyo (c) Tokyo

Fig. 5: Recommendation performance with respect to top-k recommended values for Tokyo Dataset




(a) New York (b) New York (c) New York

Fig. 6: Recommendation performance with respect to top-k recommended values for New York Dataset

### 6.4 Evaluation Plan

As we can only use the past check-in data to predict the future check-ins, each dataset is divided into the training set and the testing set regarding the check-in time. We use the first eight-month check-ins as training dataset and the last two months check-ins as the test dataset. The training set is used to learn the recommendation model as it is described in Section 3 to predict the testing data. In our experiments, we examine the accuracy, precision and recall of evaluated recommendation techniques concerning a range of top-k from 5 to 100. The number of iteration, $M$, is set to 50, and number of clusters, $g$, for evaluating the P-APPR is set to 4.

## 7 EXPERIMENTAL RESULTS

This section analyzes the experimental results. First, we compare our APPR against two POI recommendation baselines regarding the recommendation accuracy, precision and recall. We then address some important findings. Finally, we study the impact of $M$, number of iterations, on the quality of the recommendation of APPR.

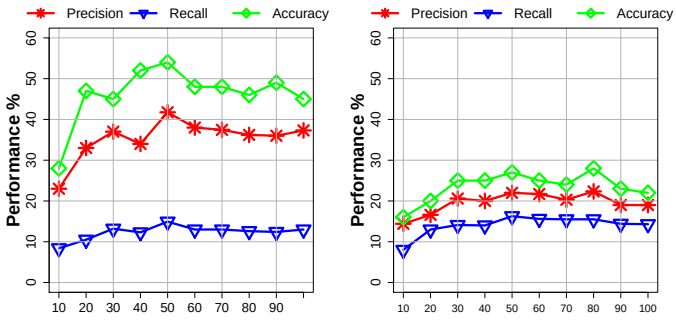
### 7.1 Comparison of Performance

In this section, we compare the performance of evaluated POI recommendation techniques, considering a range of top-k values as it is shown in Fig.5 and Fig.6 for the Tokyo and New York datasets respectively.

*Naive Bayes Approach:* This approach simply assumes that features are independent given the outcome. It models the probability of visiting a POI and each feature $X_i$ separately, and then combine them by multiplying. As a result, it cannot take advantage of the interaction between the features in POI recommendations. Considering that user behaviour in an LBSN is influenced by multiple features acting synchronously, $base1$ returns the less accurate POIs, regarding precision (Fig. 5a and Fig. 6a) and accuracy (Fig. 5b and Fig. 6b), than APPR. This model also misses more POIs visited by the target users, regarding the recall, than APPR. This has been represented in Fig. 5c and Fig. 6c. This approach also neglects the fact that different users may be affected differently by the features and considers the same importance for all the features.

*Full-joint Model:* To overcome the limitation of Naive Bayes approach, this model characterizes dependencies between the probability of visiting a POI and all the influential features $X_i$ jointly. Similar to our work, it applies a supervised learning strategy to model the POI recommendation. Therefore, it considers the features' interaction and also distinguishes the importance of different features on users' behaviour. However, it returns the most inaccurate POIs in terms of precision and accuracy and misses most POIs actually visited by target users regarding the recall. The reason is that users' check-in data is often sparse in an LBSN and Full-joint Model leads to the overfitting and low performance of the probability estimation.

*Additive Personalized POI Recommendation (APPR):* In con-




(a) Tokyo (b) New York

Fig. 7: Effect of M (number of iterations) on quality of APPR

trast, APPR inherits the superiorities of both Naive Bayes approach and Full-joint model including the combining the individual models, taking the features interactions into account, and distinguishing the features' importance for different users. Thus, it substantially exhibits the better performance regarding various top-k values on both datasets.

*Partition-based Additive Personalized POI Recommendation (P-APPR):* In comparison with APPR, partition-based approach substantially improves the precision of the recommendation in both datasets, Fig. 5a and Fig. 6a. However, considering the recall and the accuracy of the recommendation, P-APPR is a bit less than the APPR approach. As only the most appropriate features are involved in the partition-based APPR, not all of them, it results in the more precise recommendation than APPR. On the other hand, all the possible subsets of the feature space are involved in building the APPR. This results in a model that can identify more POIs that the user would like to visit which leads to the higher recall. The same reason goes for the accuracy. Nevertheless, APPR-P closely follows the APPR regarding the recall and accuracy.

### 7.2 Discussion on Quality of the Proposed Framework

**Is Naive Bayes or Full-joint more effective?** According to Figures 5 and 6, $base1$ applying the Naive Bayes strategy is inferior to $base2$ applying the Full-joint approach on both datasets. The main reason is, as $base1$ models each feature separately, it deals better with the data sparsity problem. On the other hand, because of the limited available training set, $base2$ faces the overfitting which results in the poor performance. Fortunately, the superiorities of both strategies can be integrated into a unified recommendation framework through APPR.

**Recommendation performance on various top-k values.** From Figures 5 and 6, we can see that with rising $k$, the recall is gradually increasing, but the precision decreases steadily on two datasets. Our explanation is that, when more POIs are returning to users, it can identify more locations that users would like to visit which results in the higher recall. On the other side, since the recommendation techniques return the POIs with the top-k highest probabilities (scores), the additional recommended POIs are less likely to be visited by the users because of the lower visiting probabilities of these POIs which results in lower precision. The same explanation as the recall goes for the accuracy, by returning more locations, the chance of discovering the next location that user visits increases.

Another interesting observation is that with rising the number of recommended POIs, $k$, the superiority of APPR-P regarding the precision becomes more evident. On the other side, recall of the APPR-P goes below the APPR. It shows that, with increasing the $k$, more true-positive POIs are discovered by APPR; however, more false-positive results also are returned by it. This leads to the higher recall and lower precision than the APPR-P.

### 7.3 Effect of Number of Number of Iterations (M)

In this part, the convergence of our approach is indicated by the number of iterations needed to reach certain recommendation performance. For this experiment we set $k = 20$ (The number of recommended POIs). As represented in Fig. 7 and Fig. 8, for both datasets, the performance obtained from the APPR and P-APPR reaches its ceiling around $M = 50$, and setting the $M > 50$ doesn't necessarily improve the performance of the recommendation models.

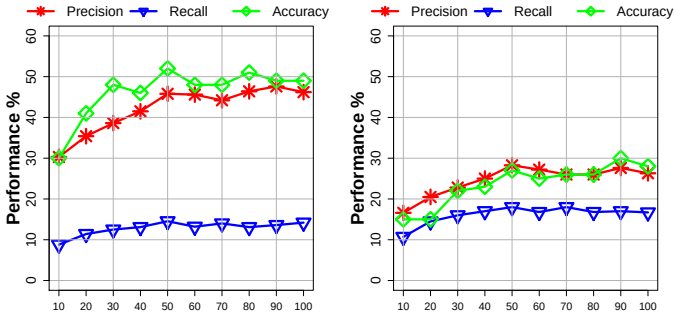
### 7.4 Effect of Number of Clusters (g)

In this section, we evaluate the impact of the number of clusters on the performance of APPR. The number of clusters, $g$, is usually decided based on the amount of available training data, which is equal to the number of check-ins that have been performed by the user. During this experiment, we set the other two parameters $M = 50$ (The number of iterations) and $K = 20$ (The number of recommended POIs). $g$ is evaluated using the two obtained datasets- New York and Tokyo. The results of these experiments are shown in Fig. 9. When $g = 1$, there is essentially no clustering and the entire data set is the only cluster. This is equal to the Full-joint Model. Therefore, the results obtained from the settings of $g = 1$ are then used as the baseline to compare with other setting of $g$. We observe that using clustering yields better overall precision, recall and accuracy. This is because using clustering to find PRMs successfully captures the hidden patterns from frequented areas visited by the user.

We also find that the performance converges at a certain level around $g = 4$ for both datasets. On the other side, setting with $g > 4$ doesn't necessarily improve the performance. The reason is when the size of the cluster is too small, the PRM fits the training data too precisely which may cause overfitting. However, as it's mentioned before, the convergence level depends on the amount of available training data, the more check-in data is available, the more clusters can be extracted.

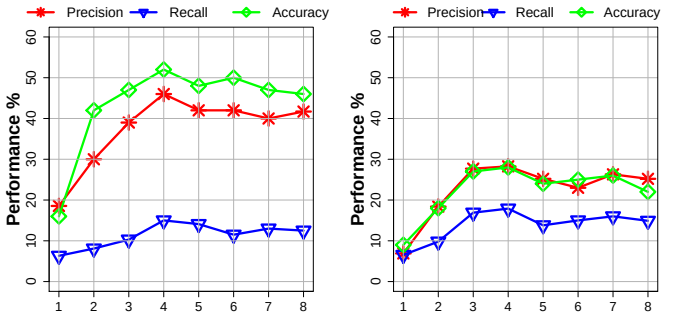
### 7.5 Complexity Analysis

To construct the APPR, the BuildAPPR function has to identify PRMs from all the possible subsets of the feature space. Let's assume that there are $F$ features, then in every iteration, $2^F$ PRMs should be identified. Therefore, this process is computationally expensive. According to the P-APPR, the BuildAPPR function only needs to identify the PRMs from the extracted clusters. Considering there are $g$ clusters, only $g$ PRMS then should be identified. It should be




(a) Tokyo (b) New York

Fig. 8: Effect of M (number of iterations) on performance of P-APPR




(a) Tokyo (b) New York

Fig. 9: Effect of g (number of clusters) on recommendation performance of P-APPR

noted, $g < 2^F$, specially when many features are involved. Fig. 10 shows the training time of APPR and P-APPR on Tokyo dataset. As it is clear, the training time of P-APPR is significantly lower than the APPR, while it also improved the precision of the recommendation, and showed the comparable recall and accuracy in comparison with APPR. It should be noted, with increasing the number of features, the training time difference will increase further.

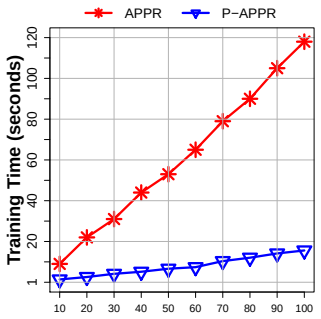



Fig. 10: Average training time of APPR and P-APPR for each user with respect to M (number of iterations) on Tokyo dataset

## 8 CONCLUSION AND FUTURE WORK

In this paper, we have explored the problem of POI recommendation in LBSNs. Unlike the Naive Bayes based approaches, APPR does not put this assumption that features are independent given the output. On the other hand, considering all the features jointly might lead to the overfitting, as each user visited the limited number of POIs. We have proposed APPR, an intermediate solution, which learns the partial personalized models (PRMs) from different subsets of the feature space, then combines them to build the recommendation model. To construct PRMs and then collectively utilize them, our approach iteratively chooses a PRM, at each layer $m$ and builds a recommendation model with $M$ layers of PRMs at the end of $M$ iterations. Furthermore, to solve a possible overfitting problem, we reevaluate the PRM set to prune the unnecessary PRMs. The final hypothesis is the sum of the single hypothesis obtained from the final PRM set. We further propose a partition-based approach to identify the hidden patterns of partitions obtained from geographically clustered check-in data, Partition-based Partial Personalized Model (PPRM). Unlike considering all the possible subsets of the feature space, only the most appropriate features are involved in a PPRM, not all of them. Finally, we have conducted experiments to evaluate the performance of APPR using two real datasets from Foursquare. Experimental results show that APPR provides much better POI recommendations than other recommendation techniques evaluated in our experiments. We have two directions for future study: (1) Although choosing the PRMs randomly at each iteration represents the high performance; we would like to propose more advances selection strategy. (2) In this work, we applied some general features directly extracted from the data. In our future work, we will obtain more sophisticated features.

## REFERENCES

[1] X. Wang, C. Zhang, F. Liu, Y. Dong, and X. Xu, "Exponentially weighted particle filter for simultaneous localization and mapping based on magnetic field measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 7, pp. 1658–1667, 2017.

[2] S. M. Ghoreyshi, A. Shahrabi, and T. Boutaleb, "Void-handling techniques for routing protocols in underwater sensor networks: Survey and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 800–827, 2017.

[3] J. Bao, Y. Zheng, D. Wilkie, and M. Mokbel, "Recommendations in location-based social networks: a survey," *GeoInformatica*, vol. 19, no. 3, pp. 525–565, 2015.

[4] M. Ye, P. Yin, and W.-C. Lee, "Location recommendation for location-based social networks," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2010, pp. 458–461.

[5] E. H.-C. Lu, C.-Y. Chen, and V. S. Tseng, "Personalized trip recommendation with multiple constraints by mining user check-in behaviors," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012, pp. 209–218.

[6] D. Zhou, B. Wang, S. M. Rahimi, and X. Wang, "A study of recommending locations on location-based social network by collaborative filtering," in *Canadian Conference on Artificial Intelligence*. Springer, 2012, pp. 255–266.

[7] L. Yao, Q. Z. Sheng, X. Wang, W. E. Zhang, and Y. Qin, "Collaborative location recommendation by integrating multi-dimensional contextual information," *ACM Transactions on Internet Technology (TOIT)*, vol. 18, no. 3, p. 32, 2018.

[8] E. Naserian, X. Wang, K. Dahal, Z. Wang, and Z. Wang, "Personalized location prediction for group travellers from spatial–temporal trajectories," *Future Generation Computer Systems*, vol. 83, pp. 278–292, 2018.

[9] E. Naserian, X. Wang, X. Xu *et al.*, "A framework of loose travelling companion discovery from human trajectories," *IEEE Transactions on Mobile Computing*, 2018.

[10] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 739–748.

[11] J. Bao, Y. Zheng, and M. F. Mokbel, "Location-based and preference-aware recommendation using sparse geo-social networking data," in *Proceedings of the 20th international conference on advances in geographic information systems*. ACM, 2012, pp. 199–208.

[12] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks." in *Aaai*, vol. 12, 2012, pp. 17–23.

[13] X. Li, G. Cong, X.-L. Li, T.-A. N. Pham, and S. Krishnaswamy, "Rank-geofm: A ranking based geographical factorization method for point of interest recommendation," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 433–442.

[14] Z. Lu, H. Wang, N. Mamoulis, W. Tu, and D. W. Cheung, "Personalized location recommendation by aggregating multiple recommenders in diversity," *GeoInformatica*, vol. 21, no. 3, pp. 459–484, 2017.

[15] J.-D. Zhang and C.-Y. Chow, "igslr: personalized geo-social location recommendation: a kernel density estimation approach," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 334–343.

[16] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "Mining user mobility features for next place prediction in location-based services," in *Data mining (ICDM), 2012 IEEE 12th international conference on*. IEEE, 2012, pp. 1038–1043.

[17] J. J.-C. Ying, E. H.-C. Lu, W.-N. Kuo, and V. S. Tseng, "Urban point-of-interest recommendation by mining user check-in behaviors," in *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*. ACM, 2012, pp. 63–70.

[18] J. J.-C. Ying, W.-N. Kuo, V. S. Tseng, and E. H.-C. Lu, "Mining user check-in behavior with a random walk for urban point-of-interest recommendations," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, p. 40, 2014.

[19] J.-D. Zhang and C.-Y. Chow, "Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 443–452.

[20] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, "Time-aware point-of-interest recommendation," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 363–372.

[21] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 129–142, 2015.

[22] J.-D. Zhang and C.-Y. Chow, "Ticrec: A probabilistic framework to utilize temporal influence correlations for time-aware location recommendations," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 633–646, 2016.

[23] X. Liu, Y. Liu, K. Aberer, and C. Miao, "Personalized point-of-interest recommendation by mining users' preference transition," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, pp. 733–738.

[24] J.-D. Zhang, C.-Y. Chow, and Y. Li, "igeorec: A personalized and efficient geographical location recommendation framework," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 701–714, 2015.

[25] Y. Liu, T.-A. N. Pham, G. Cong, and Q. Yuan, "An experimental evaluation of point-of-interest recommendation in location-based social networks," *Proceedings of the VLDB Endowment*, vol. 10, no. 10, pp. 1010–1021, 2017.

[26] Q. Fang, C. Xu, M. S. Hossain, and G. Muhammad, "Stcaplrs: a spatial-temporal context-aware personalized location recommendation system," *ACM Transactions on Intelligent systems and technology (TIST)*, vol. 7, no. 4, p. 59, 2016.

[27] H. Wang, M. Terrovitis, and N. Mamoulis, "Location recommendation in location-based social networks using user check-in data," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 374–383.

[28] H. Gao, J. Tang, X. Hu, and H. Liu, "Modeling temporal effects of human mobile behavior on location-based social networks," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013, pp. 1673–1678.

[29] ——, "Exploring temporal effects for location recommendation on location-based social networks," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 93–100.

[30] J.-D. Zhang and C.-Y. Chow, "Spatiotemporal sequential influence modeling for location recommendations: A gravity-based approach," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 7, no. 1, p. 11, 2015.

[31] J.-D. Zhang, C.-Y. Chow, and Y. Li, "Lore: Exploiting sequential influence for location recommendations," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2014, pp. 103–112.

[32] W. W. Cohen, R. E. Schapire, and Y. Singer, "Learning to order things," in *Advances in Neural Information Processing Systems*, 1998, pp. 451–457.

[33] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[34] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.