TaxiRec: Recommending Road Clusters to Taxi Drivers Using Ranking-based Extreme Learning Machines

Ran Wang, Member, IEEE, Chi-Yin Chow, Senior Member, IEEE, Yan Lyu, Victor C. S. Lee, Member, IEEE, Sam Kwong, Fellow, IEEE, Yanhua Li, Senior Member, IEEE, and Jia Zeng, Senior Member, IEEE

Abstract—Utilizing large-scale GPS data to improve taxi services has become a popular research problem in the areas of data mining, intelligent transportation, geographical information systems, and the Internet of Things. In this paper, we utilize a large-scale GPS data set generated by over 7,000 taxis in a period of one month in Nanjing, China, and propose TaxiRec: a framework for evaluating and discovering the passenger-finding potentials of road clusters, which is incorporated into a recommender system for taxi drivers to seek passengers. In TaxiRec, the underlying road network is first segmented into a number of road clusters, a set of features for each road cluster is extracted from real-life data sets, and then a ranking-based extreme learning machine (ELM) model is proposed to evaluate the passenger-finding potential of each road cluster. In addition, TaxiRec can use this model with a training cluster selection algorithm to provide road cluster recommendations when taxi trajectory data is incomplete or unavailable. Experimental results demonstrate the feasibility and effectiveness of TaxiRec.

Index Terms—Extreme learning machine, passenger-finding potential, recommender system, taxi trajectory data analytics.

1 INTRODUCTION

Taxis have been considered as a major means of transportation in modern cities because of their dynamic nature as a transportation tool, which offer great convenience to our daily life. However, compared with other transportation services such as buses and subways, taxi services are more difficult to manage for both passengers and taxi drivers, due to the high flexibility of routes and operating time [1].

In many cities, taxis are equipped with GPS sensors, which enable them to report their locations periodically. A GPS record takes down the current information of a taxi, including the ID, time, longitude, latitude, speed, direction, and occupation, etc. Given the large number of taxis, and high data collection frequency, the GPS records are collected in a large scale of tens of Gigabytes on a daily basis. At the same time, they provide reliable information on the taxi trajectories, which reflect the behavioral patterns of both passengers and taxi drivers. Recent years have witnessed an increasing interest in the applications based on taxi trajectory data, such as urban planning [2], [3], visualization [4], route prediction [5], [6], and recommender systems [7], [8].

Recommender systems have shown high practical values for taxi services, which could help passengers or taxi drivers find their targets in a more effective manner. However, existing works on recommender systems only focus on using the knowledge learned from the historical trajectories, such as the passengers’ mobility patterns [9], the taxi drivers’ pick-up behaviors [7], [8], and the spatiotemporal distributions of taxi-passenger demands [10], [11], [12]. They neglect many objective influential factors, e.g.,

- Points-of-interest (POIs): Taxis are busy in the downtown area that covers a large number of POIs, but are inactive in suburbs.
- Road length: It is easier for passengers to find a vacant taxi on a longer road.
- Road types: Different road types have different regulations for taxis. For instance, a freeway does not allow vehicles to stop, thus, taxis cannot pick up passengers; a main road only allows taxis to pick up passengers in some specific segments but a minor road has no such a restriction.

Besides, most related works built recommender systems by utilizing landmark graphs, clustering techniques, and probability methods. However, only a few existing recommender systems adopted supervised learning, which can make accurate predictions on unknown samples by training a model based on a set of labeled samples. Following the direction of supervised learning, a strategy evaluation model is presented by using L1-norm support vector machine (SVM) [10]. This model can suggest taxi drivers with good passenger-hunting strategies (e.g., cruising or waiting) in a
specific region, but it fails to rank different regions based on the possibility of finding passengers.

In this paper, we propose a recommender system, TaxiRec, which employs a supervised learning model to recommend road clusters to taxi drivers for finding passengers. Artificial neural network (ANN) is adopted to build a regression model for TaxiRec to predict the pick-up frequency in a road cluster. ANN is a non-parametric model that has high capability in approximating nonlinear functions. It provides a better data representation ability with high tolerance to noise. Moreover, different activation functions may have different effects on the performance of ANN. Recent studies [13] have shown that if rectified linear units (ReLU), including parametric ReLU (PReLU) [14], leaky ReLU (LReLU), and noisy ReLU (NReLU) [15], are utilized as the activation functions of hidden nodes, ANN with deep architectures [16] can be realized in a supervised mode by preserving information about relative intensities as it travels through multiple layers. Thus, compared with other regression models, such as logistic regression (LR) [17], regression tree [18], and support vector regression (SVR) [19], ANN is more suitable for handling taxi data. However, traditional Back-Propagation (BP) based training methods for ANNs, such as Gradient-Descent (GD), Levenberg-Marquardt (LM), and Plwell-Beale (PB), usually suffer from high computational complexity, especially for deep architectures. In order to apply them to online real-time systems with general equipments, fast solutions with high reliability are required. In recent years, extreme learning machine (ELM) has been proposed for training single hidden-layer feed-forward neural networks (SLFNs) [20]. Different from BP-like algorithms, ELM gets the network parameters by solving a normal system of linear matrix equations, which exhibits an extremely fast learning speed without sacrificing the predictive capability. Thus, it has a great potential for recommender systems requiring high effectiveness and efficiency.

On the other hand, existing recommender systems for taxi drivers focus on suggesting a passenger-finding strategy in a certain region or the shortest path to a destination. TaxiRec can distinguish itself from existing recommender systems that it treats each road cluster as an evaluation unit and recommends the top-$K$ road clusters with the highest passenger-finding potentials to taxi drivers. The main contributions of this paper are listed as follows.

- We adopt a graph clustering algorithm to divide the road network of a city into smaller road clusters for evaluation.
- A set of features for each road cluster is extracted based on real-world data sets. The feature set of a road cluster reflects its properties with respect to different factors, such as its function, length, and number of POIs of each POI type.
- A ranking-based ELM model is proposed by taking the extracted feature set as an input, and the pick-up frequency as an output. This model can evaluate the passenger-finding potentials of all the road clusters based on their pick-up frequencies.
- We demonstrate how the proposed model can be used for a city with incomplete or no trajectory data. More specifically, some representative road clusters are selected in a city. Then, the pick-up frequencies in these clusters are recorded manually, which constitutes the training samples of the ELM model. The model is used to predict the pick-up frequencies in unlabeled clusters. The top-$K$ road clusters with the highest passenger-finding potentials are recommended to the taxi drivers.
- Extensive performance evaluation is conducted on a real trajectory data set of 7,000 taxis in a period of one month in Nanjing, China, a real road network of Nanjing obtained from OpenStreetMap, and POI data crawled from Weibo. Experimental results exhibit the effectiveness and efficiency of TaxiRec.

The remainder of this paper is organized as follows. In Section 2, we highlight related works. Section 3 describes road segment clustering and passenger-finding potential modelling. In Section 4, we propose the ranking-based ELM model for evaluating the passenger-finding potentials of road clusters. Experimental results are presented in Section 5. Finally, Section 6 concludes this paper.

## 2 Related Work

Many works have been proposed to solve challenges in urban computing by understanding and analyzing taxi GPS trajectory data.

**Pattern discovering.** This topic aims to discover frequent or unusual trajectory patterns. For instance, the frequent trajectory patterns can be discovered to help analyze and predict the regular behaviours of objects [21]. The time-dependent popular areas and movement patterns of taxis are mined to serve transport management, urban planning, and location-based services [22]. Besides, an anomalous route detection method is proposed to identify fraudulent behaviors [23].

**Route planning.** Given a destination, it is practical and profitable for a taxi driver to find the most efficient path. For instance, the fastest path is determined by summarizing the hierarchical travel experience information of road network [24], or by discovering the taxi drivers’ intelligence [25]. Furthermore, a routing algorithm is proposed by using multiple order Markov chains [26], a driving direction enhancement scheme is proposed by using a time-dependent landmark graph [5], [6]. In addition, some practical methods are presented for reducing cruising miles by suggesting profitable locations to taxi drivers [27], [28].

**Urban planning.** How to mine useful information from taxi trajectory data for city planning also has attracted a lot of research efforts. An interactive system is presented for traffic jam analysis by computing the traffic speed on each road segment [4]. The intra-urban human mobility is modeled to help us understand urban dynamics [2]. A taxi status (occupied, non-occupied or parked) inference method is proposed by using a local probabilistic classifier and a Hidden Semi-Markov Model (HSMM), which enables urban computing for improving a city’s transportation system and land usage [29]. Moreover, an empirical method is used for detecting region pairs with salient traffic problems [3], and a Service Disequilibrium Detection (SDD) framework is proposed for identifying regions of service disequilibrium [30]. By analyzing large scale electric taxi GPS trajectory data, an...
optimal charging station deployment framework for electric vehicles is developed [31].

**Recommender system.** Another popular topic is to provide recommendations for passengers or taxi drivers. The related works can be summarized into three categories. (1) **Services for passengers.** A waiting time prediction method is proposed for finding taxis [9], some scheduling methods are proposed for taxi ride-sharing [11], [32] or carpooling [33], [34], and a customized online model is presented for inferring the passenger demands with both historical and real-time data [35]. (2) **Services for taxi drivers.** A passenger-finding strategy (hunting or waiting) evaluation model is presented in [10], which adopts L1-norm SVM to determine the taxi performance. Moreover, a smart strategy is proposed in [36] for taxi drivers to maximize the profit for finding lucrative passengers. (3) **Services for both passengers and taxi drivers.** An incremental framework is used to predict the spatiotemporal distribution of the taxi-passerger demand [12], and a recommender system is proposed for finding passengers and vacant taxis by using passengers' mobility patterns and taxi drivers' pick-up/drop-off behaviors [7], [8].

In this paper, TaxiRec evaluates the passenger-finding potentials of road clusters based on supervised learning and recommends the top-K road clusters to taxi drivers. Although there are some existing works on recommender systems for taxi drivers, they cannot be applied to solve our problem. The main reason lies in the difference of the recommendation targets. For instance, existing systems such as T-Drive and T-Finder, aim to recommend some locations and road segments to taxi drivers. Another recommendation target is the strategy of hunting or waiting in a region, which is also different from our problem. Besides, in supervised learning, a training process is necessary to learn the model parameters, which can correlate between the input features and output target. This training process should be finished before the predicting step. However, existing works do not have training process for model parameters; they utilize a fixed model and make predictions from the historical data directly.

### 3 Road segment clustering and passenger-finding potential modelling

In this section, we will first introduce some preliminaries, and then present the road network clustering algorithm and passenger-finding potential model.

#### 3.1 Preliminaries

**Road network.** We use an undirected graph \( G(V, E) \) to model a road network, where \( V \) is the set of vertices (i.e., road intersections), and \( E \) is the set of edges (i.e., road segments connecting two vertices). The weight of an edge is determined as the reciprocal of its length, e.g., if the length of an edge is 0.5 km, the importance of the edge is 2. For long edges, each of them is divided into several shorter road segments with a maximum length of 500 meters\(^1\). In the graph of Nanjing, \( |V| = 120,008 \) and \( |E| = 120,021 \). After dividing long edges, there are 126,713 road segments.

**Taxi.** There are three possible statuses for a taxi: occupied (O), cruising (C), and parked (P). Intuitively, a taxi with status C and P are available for picking up passengers. An action of a taxi could be picking up passengers (\( P \rightarrow O \) and \( C \rightarrow O \)) or dropping off passengers (\( O \rightarrow P \) and \( O \rightarrow C \)).

**Problem definition.** Given a road network, POI data, and a set of taxi trajectories, we divide a road network into road clusters and estimate the passenger-finding potential for each road cluster based on a set of features extracted from the POI data and road properties. The top-K road clusters are recommended to taxi drivers. The key challenges are (1) how to design a set of representative features that can influence the passenger-finding potential of a road cluster, and (2) how to develop an effective and efficient supervised learning model to estimate the passenger-finding potential of a road cluster.

#### 3.2 Road Segment Clustering

In a road network, many road segments are short (e.g., the average length of road segments in our data set is about 120 meters). Such a short road segment is not a proper evaluation unit. Since the demand for a taxi is dynamic, if the evaluation unit is too small (e.g., only one location point or one short street), it is possible that the predicted demand has already changed when the taxi driver gets there. To this end, road segments should be grouped into road clusters.

We adopt graph clustering method to identify the road clusters. The Markov Cluster (MCL) algorithm [37], [38] is utilized, which simulates a stochastic flow in graphs. Intuitively, the MCL algorithm is designed based on the principle that “flow is easier within dense regions than across sparse boundaries”, then there will be many links within a cluster and fewer links between clusters. Given an undirected graph \( G(V, E) \), the MCL algorithm first creates an associated matrix based on the connectivity between vertices and the weights of the edges. Then, two processes, i.e., expansion and inflation, are alternated repeatedly on the associated matrix until a steady state is reached. Finally, the clusters are discovered from the resulting matrix. Since the MCL algorithm is not the contribution of this paper, we omit the details here and utilize the MCL toolbox\(^2\).

There are two control parameters in MCL, i.e., power parameter \( c \) and inflation parameter \( r \). Different parameter combinations will generate different numbers of clusters \( k \). In practice, given a certain driving speed (\( \text{speed} \)), a time period (\( \text{time} \)) and the total length of the road segments in a city (\( \text{length} \)), we can find a reference for \( k \) by first computing the driving distance \( \text{speed} \times \text{time} \) as the total length of road segments in a road cluster, and then determining \( k \) as \( \text{length}/(\text{speed} \times \text{time}) \). Consider our data set as an example, the average driving speed of taxis in Nanjing is 50 km/hour. Given a driving time of 20 minutes (which is defined as the longest driving time that a taxi driver is willing to spend for hunting passengers in a road cluster), the total driving distance is 16.67 km. Since the total length of the 126,713 road segments in Nanjing is 15,298 km, we have \( 15,298/16.67 \approx 918 \). Taking a reasonable approximation,

---

1. By statistics, most of the road segments (>85%) in our data set are shorter than 500 meters.
we set $k = 1,000$ and approximate this $k$ by tuning the parameter combination $(e, r)$.

It is noteworthy that the MCL algorithm only decides the cluster indices of the road intersections. We further determine the cluster indices of the road segments based on the following principles:

- If the two vertices of an edge belong to the same cluster $C_i$, all of the road segments in the edge also belong to $C_i$ (e.g., in Figure 1, $Seg_1 \sim Seg_4 \in C_1$ and $Seg_{10} \sim Seg_{13} \in C_2$).

- If the two vertices of an edge belong to two clusters $C_i$ and $C_j$, and the edge contains only one road segment, the cluster index of the road segment is randomly selected from $i$ and $j$ (e.g., in Figure 1, $Seg_9 \in C_1$ or $Seg_8 \in C_2$).

- If the two vertices of an edge belong to two clusters $C_i$ and $C_j$, and the edge contains multiple road segments, a separating point is randomly selected (e.g., in Figure 1, if the separating point is between $Seg_6$ and $Seg_{10}$, $Seg_9 \sim Seg_6 \in C_1$ and $Seg_{10} \sim Seg_8 \in C_2$).

As a result, the road network is divided into clusters, and each cluster is a connected sub-network. The clustering result in the urban area of Nanjing is shown in Figure 2.

### 3.3 Passenger-Finding Potential Modelling

In this section, we discuss the features for road clusters and the model for passenger-finding potentials.

#### 3.3.1 Features for Road Clusters

Our passenger-finding potential model considers three major types of features as described below.

**Road types.** The type of a road segment significantly influences the pick-up frequencies of taxis. For instance, a freeway does not allow vehicles to stop, thus, its pick-up frequency is zero; a primary road has a high flow rate of visitors, thus its pick-up frequency is high. In our work, we summarize all the road segments into 27 types as listed in Table 1, and the type of a road cluster is decided by the main type with the maximum number of road segments in it.

**Road length.** Although passengers are not uniformly distributed in reality, for two road segments of the same type, the longer one usually has a higher chance for a taxi to pick up passengers than the shorter one. In TaxiRec, the length of a road cluster is calculated as the sum of the lengths of all its road segments.

**Points-of-interest (POIs).** Another important factor that can influence the passenger-finding potential is the density of POIs near the road segment. As a consensus, a taxi will have a higher chance to pick up passengers on the road that is close to a larger number of POIs, and vice versa. In this paper, we extracted 66,160 POIs in Nanjing from Weibo\(^3\), a popular Chinese social network web site. All the POIs are categorized into 13 types, as depicted in Table 2.

Figure 3 visualizes the geographical distributions of four types of POIs in the urban area. Their overall distributions are roughly consistent, but each one exhibits some particular characteristics. For instance, the POIs of Restaurants are intensively located in the downtown area, where their densities in other areas are much lower; the POIs of Residence are concentrated in several smaller regions, rather than one big area; the POIs of Transportation are more uniformly distributed; and the POIs of Company are much sparser than others. In TaxiRec, we propose a distance-based method to calculate the numbers of the 13 types of POIs covered by each road cluster, which will be 13 POI features for the passenger-finding potential model.

First, a circular area is fixed for each POI; then, the distances of the POI to the road segments whose centers are located in the circular area are computed; finally, the POI is said to be covered by the road segments within a certain distance. Let $M_{1000 \times 13}$ be the POI feature matrix, where $M_{p,q}$ ($p = 1, \ldots, 1000$, $q = 1, \ldots, 13$) is the number of POIs of type $q$ covered by cluster $p$.

---

\(^3\) http://weibo.com
distance-based method is described in Algorithm 1. In this paper, we set $r = 1.0\text{km}$ and $d = 0.5\text{km}$.

It is noteworthy that multiple cases can happen for a POI as demonstrated in Figure 4.

- A POI can be covered by a single cluster or multiple clusters. For instance, the POI is covered by both the black and purple clusters in Figure 4(a), but is only covered by the black cluster in Figure 4(b).
- A POI can be covered by a cluster for one time or multiple times. For instance, the POI is covered twice by the black cluster in Figure 4(a), but is only covered once by the black cluster in Figure 4(b).
- If a POI is far away from all the road segments, it is not covered by any cluster.

3.3.2 Passenger-finding Potential Model

In the previous section, we have designed 15 features for a road cluster (i.e., type, length, and numbers of 13 types of POIs). It is necessary to make some investigations on the real taxi trajectory data, in order to validate their effectiveness in modeling the passenger-finding potentials. Since the road condition is largely influenced by the time, we divide a day into 48 time intervals, with the unit of 30 minutes (i.e., (1) 0:00 to 0:30, (2) 0:30 to 1:00, ..., (48) 23:30 to 0:00). The pick-up frequency ($P \rightarrow O$ and $C \rightarrow O$) is counted from the taxi trajectory data for each road cluster, and we observe ten randomly selected clusters as listed in Table 3.

The pick-up frequencies in the ten clusters in different time intervals are given in Figure 5. For different POI types, the percentages of POIs in the ten clusters to the total number of POIs are given in Figure 6. From these two figures, we can observe that the pick-up frequency of a cluster is influenced by the following factors:

- The type of the road cluster. For instance, Primary (Cluster 3) has higher chance than Tertiary (Cluster 8) and Tertiary link (Cluster 4) to pick up passengers.
- The length of the road cluster. For example, Clusters 1 & 6 or Clusters 9 & 10 are of the same type, the longer one (Cluster 6 or Cluster 10) has a higher chance than the shorter one (Cluster 1 or Cluster 9) to pick up passengers.
- The numbers of different types of POIs near the road cluster. For instance, Clusters 3, 6 & 10 have a larger number of POIs than other clusters for almost all the types, thus their pick-up frequencies are high.
contrary, Clusters 2 & 4 have a smaller number of POIs than other clusters for almost all the types, thus their pick-up frequencies are low. It is noteworthy that the numbers of different types of POIs for a cluster are not always consistent, e.g., in Cluster 7, only the number of Residence POIs for 
always consistent, e.g., in Cluster 7, only the number of Residence POIs is higher than others, and the pick-up frequency is in an intermediate level.

Having the above observations, it is possible to model the passenger-finding potentials of road clusters by the designed features. The basic idea is to treat each cluster as a sample \((x_i, y_i)\), where \(x_i\) is composed of \(n\) input features \(x_i = [x_{i1}, \ldots, x_{in}]^T\) (\(n = 15\) here), and \(y_i\) is an output decision value (pick-up frequency). Then, a regression model could be constructed on labeled samples, in order to find the relationship between \(x_i\) and \(y_i\). This model is used to predict the decision values of unlabeled samples.

4 RANKING-BASED ELM MODEL

In this section, we will propose the ranking-based ELM model for predicting the pick-up frequency in a road cluster.

4.1 Basic Framework of ELM

As mentioned in the introduction, in order to build up a supervised learning model for TaxiRec, ANN might be a suitable choice for handling the real-world data sets used in this paper. However, traditional training methods for ANN, i.e., BP-like algorithms [39], have high computational complexity. ELM [40], as an emerging technique for training SLFNs [20], can overcome this difficulty with a non-iterative training process. The structure of ELM is shown in Figure 7.

Given a training set \(X = \{(x_i, y_i)|x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^L, i = 1, \ldots, N\}\), where \(x_i = [x_{i1}, \ldots, x_{in}]^T\) is \(i\)-th training sample, \(y_i = [y_{i1}, \ldots, y_{iL}]^T\) is the output decision vector of \(x_i\), \(n\) is the number of features, \(L\) is the number of classes for a classification problem and \(L = 1\) for a regression problem. The standard SLFNs with \(\tilde{N}\) hidden nodes and activation function \(g(x)\) can be mathematically modeled as:

\[
t_i = \sum_{j=1}^{\tilde{N}} \beta_{ij} g(w_j \cdot x_i + b_j), \quad i = 1, \ldots, N, \tag{1}
\]

where \(t_i\) is the prediction result of \(x_i\), \(w_j = [w_{j1}, \ldots, w_{jn}]^T\) is the weight linking the input nodes and the \(j\)-th hidden node, \(b_j\) is the bias of the \(j\)-th hidden node, and \(\beta_{ij}\) is the weight linking the \(j\)-th hidden node and the output nodes. The standard SLFNs are proved to be capable of approximating the \(N\) training samples with zero error, i.e., \(t_i = y_i, i = 1, \ldots, N\). Thus, there exists \(\beta_{ij}, w_j, \) and \(b_j\) such that

\[
\sum_{j=1}^{\tilde{N}} \beta_{ij} g(w_j \cdot x_i + b_j) = y_i, \quad i = 1, \ldots, N. \tag{2}
\]

The \(N\) equations in Eq. (2) can be written compactly as the matrix form, i.e., \(H \cdot \beta = Y\), where

\[
H = \begin{bmatrix}
g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\
\vdots & \ddots & \vdots \\
g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}})
\end{bmatrix}_{N \times \tilde{N}},
\]
Algorithm 2: Training Process of ELM

**Input:** Training set \( \mathcal{X} = \{(x_i, y_i) | x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^L, i = 1, \ldots, N\} \); activation function \( g(x) \); the number of hidden nodes \( N \).

**Output:** Input weight \( w_j \); input bias \( b_j \); output weight \( \beta \).

1. Randomly assign input weight \( w_j \) and bias \( b_j \) from \( (0, 1) \), where \( j = 1, \ldots, N \);
2. Calculate the hidden layer output matrix \( H \);
3. Calculate the output weight \( \beta = H^\dagger Y \) where \( H^\dagger \) is the Moore-Penrose generalized inverse of matrix \( H \).

Algorithm 3: Testing Process of ELM

**Input:** Testing set \( \mathcal{T} = \{\hat{x}_i | \hat{x}_i \in \mathbb{R}^n, i = 1, \ldots, M\} \); output of Algorithm 2 (i.e., input weight \( w_j \); input bias \( b_j \); output weight \( \beta \)).

**Output:** Output value \( \hat{y}_i \) for each testing sample \( \hat{x}_i \).

1. Calculate the hidden layer output matrix \( \hat{H} = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_N \cdot x_1 + b_N) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_M + b_1) & \cdots & g(w_N \cdot x_M + b_N) \end{bmatrix}_{M \times N} \).
2. Calculate the testing output matrix \( Y_{M \times L} = \hat{H}_{M \times N} \beta_{N \times L} \).
3. if the problem is classification then
   4. \( \hat{y}_i = \text{argmax}_{j=1,\ldots,L} Y_{ij} \);
5. else if the problem is regression then
   6. \( \hat{y}_i = Y_{i1} \).
7. end

is the hidden layer output matrix, and
\[
\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_N^T \end{bmatrix}_{N \times L}, \quad Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}_{N \times L}.
\]

In general, the essence of ELM is composed of two parts, i.e., (1) randomly assign the input weights \( w_j \) and biases \( b_j \), and (2) analytically determine the output weights \( \beta_j \) by the Moore Penrose generalized inverse of the hidden layer output matrix. Having the above notations, the training and testing processes of ELM are presented in Algorithms 2 and 3, respectively. Note that (1) when the problem is classification, \( L \) is the number of classes, and \( y_{ij} \) is the class membership of \( x_i \) in the \( j \)-th class; (2) when the problem is regression, \( L = 1 \) and \( y_i \) is the regression value of \( x_i \).

It is stated in [41] that ELM has many advantages compared with other approaches. It not only exhibits an extremely fast learning speed, but also demonstrates a good generalization capability in many application domains [42], [43], [44], [45]. However, in order to apply ELM to the road cluster evaluation problem, some details need to be discussed further. We will discuss these details in Section 4.2 and Section 4.3.

4.2 Architecture Selection

Architecture selection is a fundamental problem in ANN, which aims to select the optimal network architecture (i.e., number of hidden nodes \( N \)) for a given data set. It is observed that the result of any type of ANN is largely influenced by the number of hidden nodes.

The most commonly used approach for architecture selection is \( k \)-fold cross validation (CV). This approach is experimentation-driven and is easy to implement, but it is very time consuming. In order to tune the network architectures for TaxiRec, we propose a localized generalization error model (LGEM) based method in this section.

Basically, LGEM [46] is a constrained error model by using stochastic sensitivity measure. It bounds the generalization error for unseen samples located within a predefined neighborhood of the training samples, while this neighborhood is called \( Q \)-neighborhood, and \( Q \) is a real value. In general, \( Q \)-value represents the size of the neighborhood in which the model is capable of predicting unseen samples with confidence. Obviously, when the error bound is fixed, the architecture with larger \( Q \)-value is preferred. Based on this statement, Wang et al. [47] proposed an extended LGEM for ELM with sigmoid activation function. In this paper, we further utilize LGEM to realize architecture selection for ELM. Given the number of hidden nodes \( N \) and an ELM model trained on \( \mathcal{X} \), it is easy to get the model parameters \( (w_j, b_j, \beta_j) \) and the empirical risk
\[
R_{emp} = \frac{1}{N} \sum_{i=1}^{N} (t_i - y_i)^2.
\]

Based on the theoretical analysis in [46], [47], the \( Q \)-value for an ELM architecture with \( N \) hidden nodes and sigmoid activation function can be computed as the following Eq. (6)

\[
Q(N) = \sqrt{3 \sqrt{\xi - \varepsilon} - \sqrt{R_{emp} - A}} / \sum_{j=1}^{N} \beta_j^2 \sum_{k=1}^{n} w_{jk}^2,
\]

where \( \xi \) is the localized generalization error bound, \( \varepsilon = B \sqrt{\ln \eta / 2N} \) is a very small constant when the confidence of the bound \((1 - \eta)\) is fixed, \( A \) is the difference between the maximum and minimum values of the target outputs, and \( B \) is the maximum possible value of mean squared error. In this paper, we set \( \xi = 0.25 \), which means that an architecture is worth investigating only if \( R_{emp} < 0.25 \). In this case, \( Q(N) = 0 \) if \( R_{emp} \geq \xi \) for an architecture.

On the other hand, a larger number of hidden nodes will improve the generalization capability, but it also increases the computational complexity and the risk of over-fitting. Thus, when the accumulation of empirical risk is lower than a certain threshold \( \alpha \), a smaller number of hidden nodes is preferred. Given a set of network architectures \( \mathcal{N} = \{N_1, \ldots, N_m\} \) where \( N_1 < \ldots < N_m \), our proposed architecture selection process is described as Algorithm 4. In this paper, we set \( \alpha = 0.002 \).

4.3 Ranking-based ELM for Road Cluster Evaluation

It is argued in [48], [49] that with the randomly assigned input weights, ELM can sometimes become unstable during
training, which shows a weak robustness to many applications. In other words, different input weights will generate different results. However, it is difficult to evaluate the quality of the randomly assigned input weights in the training process. One possible solution to overcome the unstable problem is to ensemble the results of a set of ELMs. By integrating multiple ELMs, the unstable factor caused by the random mechanism can be weakened.

Besides, taxi drivers are always interested in the ranking order of the road clusters, rather than the actual pick-up frequencies. Thus, it is important to treat the road cluster evaluation problem as a ranking problem, rather than a traditional regression problem. However, traditional loss functions for recommender systems, such as the ranking-based loss function proposed in [50], cannot be applied to TaxiRec, since they did not consider the input features for a learning system. Thus, we propose a ranking-based ELM model (Algorithm 5) for road cluster evaluation.

### 4.4 Training Cluster Selection Algorithm

After having the solution for ranking the road clusters, another important problem is how to select some representative training clusters for labeling. When taxi trajectory data is incomplete or unavailable, the labeling work has to be finished manually, which equals to recording the occurrence time and location of each pick-up action in each road cluster. Since the manual labeling work is expensive and time consuming, this selection process is significantly necessary. By selecting a set of representative clusters, a learning model with high generalization capability can be constructed. This model is used to predict the pick-up frequencies in the unlabeled clusters, which largely alleviates the manual labeling work.

Considering the features designed in Section 3.3.1, the selected clusters should cover all the possible road types with different lengths and numbers of POIs. With these considerations, we conduct the selection type by type. For each type, the length and number of POIs are integrated into one overall criterion, which ranks the clusters into a fixed order. Then, given a selection ratio, a sampling interval is calculated. Finally, clusters are sampled from the ranking order based on this interval. The detailed selection process is given in Algorithm 6, and the selection result in the urban area of Nanjing with the ratio of 0.1 is shown in Figure 8.

### 5 Experimental Analysis

Existing recommender systems for taxi drivers focus on recommending the passenger-finding strategy in a certain region or the most efficient path to a given destination. To the best of our knowledge, TaxiRec is the first work treating each road cluster as an evaluation unit. Due to the different design objectives, we cannot compare the performance of TaxiRec with them; and thus we design three sets of experiments to validate the feasibility and effectiveness of TaxiRec. Our trajectory data are collected from over 7,000
taxis of one month in Nanjing, China, road network data are obtained from OpenStreetMap\(^4\), and POI data are crawled from Weibo.

5.1 Experiment Design

In each experiment, 48 data sets are tested, representing the 48 time intervals in a day (i.e., (1) 0:00 to 0:30, (2) 0:30 to 1:00, … (48) 23:30 to 0:00). All the experiments are conducted on different days separately and the average results are recorded. We design three experiments to conduct evaluation for TaxiRec. The first two experiments are designed for overall evaluations, which include all the road clusters, while the last one considers a practical scenario with a distance constraint.

- **Experiment 1** validates the feasibility of the ranking-based ELM model. More specifically, we realize Algorithm 5 with randomly selected training clusters. Since the random mechanism may cause some unstable problems, we repeat the experiment 20 times, and then observe the average value and standard deviation. In this experiment, architecture selection is realized according to the approach proposed in Algorithm 4.

- **Experiment 2** studies the effectiveness of the training cluster selection process through incorporating Algorithm 6 to select the training clusters. In this experiment, we compare the ranking-based ELM model with several state-of-the-art regression techniques, i.e., support vector regression (SVR), logistic regression (LR), classification and regression tree (CART), gradient boosting decision trees (GBDT), SLFN trained by BP method with PReLU (PReLU-S), and multiple-hidden layer feedforward neural network trained by BP method with PReLU (PReLU-M). Since the training and testing sets are fixed in this experiment, it is easy to make comparison among different methods. We utilize the same network architectures for ELM as determined in Experiment 1.

- **Experiment 3** considers a practical scenario, where a vacant taxi makes a request to TaxiRec at a certain location for top-\(K\) road clusters within a user-specified distance range. We randomly generate requests with various distance ranges \(\{2\text{ km}, 3\text{ km}, \ldots, 10\text{ km}\}\) from 200 randomly selected locations in the urban area. Figure 9 illustrates the 200 randomly sampled locations and the circular region with radius of 5 km for one location. In this experiment, Algorithm 6 is also used to select the training clusters, and the network architectures determined in Experiment 1 are utilized.

Each feature is normalized into \([0,1]\). The activation function in ELM is set as sigmoid function \(g(x) = \frac{1}{1+\exp(-x)}\), and the number of ELMs in Algorithm 5 is \(C = 10\). The network architecture is determined by Algorithm 4 from the candidates \(N = \{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}\). The experiments are performed under MATLAB R2011b, which are executed on a computer with an Intel Core i7-5500U CPU@2.40GHz, 8GB memory, and 64-bit Windows 8 system.

5.2 Performance Metric

Traditional recommender systems usually adopt precision@K to evaluate the performance, which defines the fraction of the recommended objects that are relevant. In TaxiRec, after all the road clusters are ranked into an order, the top-\(K\) clusters are recommended to the taxi driver. Thus, precision@K for this problem can be defined as

\[
\text{Precision@K} = \frac{|\text{Real top-}K\text{ clusters} \cap \text{Ranked top-}K\text{ clusters}|}{K}
\]

where \(K\) is a real positive integer that is smaller than the number of clusters.

5.3 Experimental Analysis

5.3.1 Results of the Ranking-based ELM Model

Before presenting the final result, we first make an investigation on the values of precision@K and execution time of different network architectures on an example data set, i.e., time interval 1, as shown in Figure 10. It is observed from Figure 10(a) and Figure 10(b) that with a larger number of hidden nodes, the precision becomes higher and the standard deviation becomes lower. That is to say, a larger value of $\tilde{N}$ can improve the prediction capability and the stability of the model. However, the improvement becomes very limited with the increasing of $\tilde{N}$. For instance, the result of $\tilde{N} = 5$ is much better than that of $\tilde{N} = 1$, but the result of $\tilde{N} = 30$ is just slightly better than that of $\tilde{N} = 20$, and the result of $\tilde{N} = 40$ is slightly worse than that of $\tilde{N} = 30$. In general, the performance converges when the value of $\tilde{N}$ reaches a certain level, afterwards, the result maintains a steady state with minor fluctuations.

Algorithm 5 is realized with randomly selected training clusters (i.e., Experiment 1). We observe the results with different combinations of $(K, r)$, where $K = \{2, 4, 6, \ldots, 50\}$ accounts for the recommendation percentage of $p = \{0.2\%, 0.4\%, 0.6\%, \ldots, 5\%\}$, and $r = \{0.1, 0.2, 0.3, 0.4, 0.5\}$ is the ratio of training clusters.
numbers of hidden nodes determined by Algorithm 4 for the 48 time intervals mostly concentrate on 20, 30, 40, and 50. Due to space limit, we omit the details.

Figure 11 depicts the values of precision@K and standard deviation of the ranking-based ELM. Due to space limit, we only list the results for time intervals 1, 13, 25, and 37, which correspond to 0:00 to 0:30, 6:00 to 6:30, 12:00 to 12:30, and 18:00 to 18:30, respectively. The average result of the 48 time intervals is given in Figure 12. Basically, we have two key observations. (1) A higher recommendation percentage $p$ results in a higher precision and a lower standard deviation. This is because when $p$ is higher, the number of evaluated clusters is larger, which gives a higher chance to recommend the potential ones. Obviously, when $p = 100\%$, the precision will be 1, and the standard deviation will be 0. (2) The ratio of training clusters $r$ can largely influence the result. When $p$ is fixed, a larger $r$ can help improve the precision and reduce the standard deviation. When taxi trajectory data are sufficient, it is possible to get a lot of training samples and set $r$ as a large ratio. However, when the training clusters are obtained manually, the labelling work will be very expensive and time consuming.

5.3.2 Effect of the Training Cluster Selection Algorithm

In this section, Algorithm 6 is further utilized to select the training clusters (i.e., Experiment 2). Furthermore, with the fixed training and testing sets, we consider the ranking-based ELM model with several state-of-the-art regression techniques, i.e., SVR, LR, CART, GBDT, PReLU-S, and PReLU-M. For SVR, we utilize the $c$-SVR in libsvm toolbox [51], where the radial basis function kernel with $\sigma = 1$ is adopted, the penalty term $C$ is set as 100 and the parameter $\epsilon$ in loss function is set as 0.01. For LR and CART, we utilize the regression toolboxes in MATLAB with default settings. For GBDT [52], we apply the toolbox provided in [53], the number of binary trees for ensemble is set as 10, the number of leaf nodes in each tree is set as 100, and the shrinkage parameter $\rho$ is set as 0.1. Finally, we realize PReLU-S and PReLU-M based on MATLAB $nnet$ toolbox.

The activation function $f_i$, i.e., PReLU, for a hidden node with input $x_i$ is defined as:

$$f(x_i) = \begin{cases} x_i, & \text{if } x_i > 0 \\ a_i x_i, & \text{if } x_i \leq 0 \end{cases}$$

We initialize all $a_i$ as 0.25, and adopt the momentum method [14] to update parameters, the learning rate is set as 0.01, the amount of momentum is set as 0.9, and the maximum number of epochs is set as 100. Furthermore, for PReLU-S and PReLU-M, we use the same manner as for ELM to tune the number of hidden nodes from $\mathbb{N} = \{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. As for PReLU-M, since it is time consuming to tune the number of hidden nodes for multiple layers, we fix a five-layer architecture with each layer consisting of ten nodes. These regression methods can predict the pick-up frequencies and provide the ranking order for the road clusters in a similar way as the ranking-based ELM model. Note that when the selection ratio $r$ in Algorithm 6 is fixed, the selected training clusters will also be fixed, thus there is no deviation.

Table 4: Average execution time in seconds

<table>
<thead>
<tr>
<th>Method</th>
<th>$r=0.1$</th>
<th>$r=0.2$</th>
<th>$r=0.3$</th>
<th>$r=0.4$</th>
<th>$r=0.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELM</td>
<td>0.0084</td>
<td>0.0127</td>
<td>0.0137</td>
<td>0.0143</td>
<td>0.0156</td>
</tr>
<tr>
<td>SVR</td>
<td>0.0011</td>
<td>0.0050</td>
<td>0.0081</td>
<td>0.0162</td>
<td>0.0276</td>
</tr>
<tr>
<td>LR</td>
<td>0.0097</td>
<td>0.0057</td>
<td>0.0046</td>
<td>0.0044</td>
<td>0.0119</td>
</tr>
<tr>
<td>CART</td>
<td>0.0055</td>
<td>0.0089</td>
<td>0.0900</td>
<td>0.0111</td>
<td>0.0181</td>
</tr>
<tr>
<td>GBDT</td>
<td>0.3099</td>
<td>0.3647</td>
<td>0.3972</td>
<td>0.4599</td>
<td>0.4633</td>
</tr>
<tr>
<td>PReLU-S</td>
<td>0.4165</td>
<td>0.3982</td>
<td>0.6697</td>
<td>0.4084</td>
<td>0.4154</td>
</tr>
<tr>
<td>PReLU-M</td>
<td>0.5199</td>
<td>0.9256</td>
<td>4.3533</td>
<td>0.4520</td>
<td>0.5733</td>
</tr>
</tbody>
</table>

48 time intervals with different settings of $r$. It is observed from Figure 13 that the performances of ELM, CART, GBDT, PReLU-S, and PReLU-M are much better than those of SVR and LR. This may due to the fact that both ANN and DT are non-parametric models that can learn the parameters automatically from the data. However, the performance of SVR significantly depends on the kernel parameter $\sigma$ and the penalty term $C$, which should be given in advance, and the performance of LR is sensitive to the linear dependency of the features, which is usually unsatisfactory when the number of features is large. Besides, the precisions of CART and GBDT are slightly higher than that of ELM when $p$ is small, while PReLU-S and PReLU-M can outperform ELM when $r = 0.2$ and $r = 0.4$. Overall speaking, ELM exhibits the most stable performance and outperforms the other methods with a relatively larger $p$. Finally, compared with the results in the previous section, the precision for ELM has been improved. This observation validates the effectiveness of the training cluster selection algorithm.

The average training and testing time for constructing the regression model and predicting the pick-up frequencies of the unlabeled clusters for the 48 time intervals are given in Table 4. It is observed that ELM, LR, SVR, and CART exhibit extremely fast learning speed in both training and testing, and this observation demonstrates a possibility to consider TaxiRec as an online recommender system. While the time complexities of GBDT, PReLU-S, and PReLU-M are relatively high.

Finally, we make some statistical tests on the results shown in Figure 13. Paired Wilcoxon’s signed rank tests and paired sign tests are performed, and the corresponding $p$-values are reported in Table 5. If the $p$-value is smaller than the significance level $\alpha$, the two referred methods are considered as statistically different. Table 5 depicts that the performances of PReLU-S and PReLU-M are very similar, and CART has no significant difference with ELM and GBDT regarding Wilcoxon’s signed rank test when $\alpha = 0.05$. However, almost all the methods are statistically different from ELM regarding Wilcoxon’s signed rank test when $\alpha = 0.1$, and sign test when $\alpha = 0.05$ and $\alpha = 0.1$.

5.3.3 Results for Practical Scenario

In this section, the evaluation is restricted to a user-specified range distance, which just includes a subset of the road clusters (i.e., Experiment 3). As listed in Table 6, with different
Fig. 13: **Experiment 2** (i.e., with training clusters selected by Algorithm 6). Average results of different regression models for the 48 time intervals.

![Graphs showing precision @ K for different models with varying radii](image)

Fig. 14: **Experiment 3** (i.e., practical scenario). Average results for the 200 locations in the 48 time intervals.

![Graphs showing precision @ K for different models with varying radii](image)

**TABLE 5:** Paired Wilcoxon’s signed rank tests and paired sign tests (p Values)

<table>
<thead>
<tr>
<th>Method</th>
<th>SVR</th>
<th>LR</th>
<th>CART</th>
<th>GBDT</th>
<th>PReLU−S</th>
<th>PReLU−M</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELM</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0599</td>
<td>0.0091</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0200</td>
<td>0.0001</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>SVR</td>
<td>0.4927</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>LR</td>
<td>0.0101</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>CART</td>
<td></td>
<td></td>
<td>0.0090</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>GBDT</td>
<td></td>
<td></td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>PReLU−S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0594</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2931</td>
</tr>
</tbody>
</table>

**Note:** In each comparison, the upper and lower results are respectively the p-values of the Wilcoxon’s signed rank test and sign test. For each test, † and ‡ represent that the two referred methods are significantly different with the significance levels 0.05 and 0.1, respectively.

The precision is influenced by all the three factors, i.e., r, p, and the range distance. The first two factors have obvious impacts on the result, where the precision is improved a lot with larger r and p. In comparison, the impact of the range distance is not obvious. Although a larger range distance can give a higher precision, the improvement is trivial, especially when the recommendation percentage is high.

**6 Conclusion**

In this paper, we have designed TaxiRec, a road cluster recommendation system, which aims to recommend top-K road clusters with the highest passenger-finding potentials to taxi drivers. In TaxiRec, we first conduct a clustering process to divide a road network into smaller road clusters. Then, a feature set is designed for each road cluster, which reflects its basic properties with regard to various factors...
(i.e., the main road type, total road length, and numbers of different types of POIs of a road cluster). Afterwards, a training cluster selection algorithm has been proposed to select some representative clusters for labeling (counting the pick-up frequencies during a certain time interval). These clusters are used to train an ELM regression model, which predicts the pick-up frequencies for the unlabeled clusters. Finally, we rank all the clusters in terms of their regression values and recommend top-K clusters. Finally, we rank all the clusters in terms of their regression values and recommend top-K clusters.

As pointed out in Section 4.3, ELM usually suffers from the unstable problem, which shows a weak robustness to many applications. In TaxiRec, the ensemble of multiple ELMs is used to overcome this drawback. How to further improve the stability of ELM to get a better recommendation result is also an issue worthy of study.

7 Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61772344, Grant 61402460, Grant 61732011, Grant 61472257, and Grant 61373092, in part by the Guangdong Provincial Science and Technology Plan Project under Grant 2013B040403005, in part by the HD Video R&D Platform for Intelligent Analysis and Processing in Guangdong Engineering Technology Research Centre of Colleges and Universities under Grant GCPX-A1409, in part by the Natural Science Foundation of SZU under Grant 2017060, and in part by CityU research grants (CityU Project No. 923113). Yanhua Li was supported in part by NSF CRII grant CNS-1657350 and a research grant from Pitney Bowes Inc.

References


**Ran Wang** (S’09–M’14) received her B.Eng. degree in computer science from the College of Information Science and Technology, Beijing Forestry University, Beijing, China, in 2009, and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2014. From 2014 to 2016, she was a Postdoctoral Researcher at the Department of Computer Science, City University of Hong Kong. She is currently an Assistant Professor at the College of Mathematics and Statistics, Shenzhen University, China. Her current research interests include pattern recognition, machine learning, fuzzy sets and fuzzy logic, and their related applications.

**Chi-Yin Chow** received the M.S. and Ph.D. degrees from the University of Minnesota-Twin Cities in 2008 and 2010, respectively. He is currently an assistant professor in the Department of Computer Science, City University of Hong Kong. His research interests include machine learning, spatio-temporal data management and analysis, GIS, mobile computing, and location-based services. He was the co-founder and co-organizer of ACM SIGSPATIAL. MobiGIS 2012–2016. He received the 10-Year Best Paper Award at the Best Paper Awards at ICA3PP 2015 and MDM 2009.

**Yan Lyu** received the M.S. degree in pattern recognition and intelligent systems from University of Science and Technology of China, China, in 2013, and the Ph.D. degree in computer science from City University of Hong Kong, Hong Kong, in 2016. She was a Postdoctoral Research Fellow with Hong Kong Baptist University, Hong Kong, in 2017. She is currently a Postdoctoral Research Fellow with National University of Singapore, Singapore. Her research interests include intelligent transportation systems, spatio-temporal data analytics, mobile computing, and location-based services.

**Victor C. S. Lee** (M’92) received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 1997. He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong. His research interests include intelligent transportation systems, data dissemination in vehicular networks, real-time databases, and performance evaluation. He is a member of the Association for Computing Machinery (ACM) and the IEEE Computer Society. He has been the Chairman of the IEEE Hong Kong Section Computer Chapter in 2006–2007.

**Sam Kwong** (M’93–SM’04–F’13) received the B.Sc. and M.S. degrees in electrical engineering from the State University of New York at Buffalo in 1983, the University of Waterloo, ON, Canada, in 1985, and the Ph.D. degree from the University of Hagen, Germany, in 1996. From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada. He joined Bell Northern Research Canada as a Member of Scientific Staff. In 1990, he became a Lecturer in the Department of Electronic Engineering, City University of Hong Kong, where he is currently a Professor and Head in the Department of Computer Science. His main research interests include evolutionary computation, video coding, pattern recognition, and machine learning.

Dr. Kwong is an Associate Editor of the IEEE Transactions on Industrial Electronics, the IEEE Transactions on Industrial Informatics, and the Information Sciences Journal.

**Yanhua Li** (S’09–M’13–SM’16) received two Ph.D. degrees in electrical engineering from Beijing University of Posts and Telecommunications, Beijing in China in 2009 and in computer science from University of Minnesota at Twin Cities in 2013, respectively. He has worked as a researcher in HUAWEI Noah’s Ark LAB at HUAWEI from Aug 2013 to Dec 2014, and has interned in Bell Labs in New Jersey, Microsoft Research Asia, and HUAWEI research labs of America from 2011 to 2013. He is currently an Assistant Professor in the Department of Computer Science at Worcester Polytechnic Institute (WPI) in Worcester, MA. His research interests are urban data analytics, data-driven cyber-physical systems (CPS), and smart cities.
Jia Zeng received the B.Eng. degree from the Wuhan University of Technology, Wuhan, China, in 2002, and the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2007. He is a principal researcher at Huawei Noah's Ark Lab. His research interests are machine learning and big data applications. He is a member of the CCF, the ACM and a senior member of the IEEE.