2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)
Singapore, 7-9 April 2015

Parking Availability Prediction for Sensor-Enabled Car Parks in Smart Cities

Yanxu Zheng, Sutharshan Rajasegarar, Christopher Leckie
National ICT Australia Victoria, Dept. of Computing and Information Systems,
The University of Melbourne, Australia
E-mails: {yanxuz@student., sraja@, caleckie@, }unimelb.edu.au

***Abstract*—The growth in low-cost, low-power sensing and communication technologies is creating a pervasive network infrastructure called the Internet of Things (IoT), which enables a wide range of physical objects and environments to be monitored in fine spatial and temporal detail. The detailed, dynamic data that can be collected from these devices provide the basis for new business and government applications in areas such as public safety, transport logistics and environmental management. There has been growing interest in the IoT for realising smart cities, in order to maximise the productivity and reliability of urban infrastructure, such as minimising road congestion and making better use of the limited car parking facilities. In this work, we consider two smart car parking scenarios based on real-time car parking information that has been collected and disseminated by the City of San Francisco, USA and the City of Melbourne, Australia. We present a prediction mechanism for the parking occupancy rate using three feature sets with selected parameters to illustrate the utility of these features. Furthermore, we analyse the relative strengths of different machine learning methods in using these features for prediction.**

I. Introduction

It is predicted that about 70% of the world's population will start living in cities and surrounding regions by 2050 [4]. In order to efficiently manage the critical infrastructure and services of a city, these need to evolve into a 'smart city'. A smart city [4] according to Forrester, is one that uses information and communications technologies (ICT) to make the critical infrastructure and services of a city, such as public safety, transportation and utilities, more aware, interactive and efficient [1], [4]. The realisation of the Smart City is now becoming possible with the emergence of the Internet of Things (IoT), which radically evolves the current Internet into a network of interconnected objects, such as sensors, parking meters, energy measuring devices and actuators [12]. These networked devices have the ability to compute, sense and interact with their surroundings in fine spatial and temporal detail, and generate a vast amount of data.

One of the key services that cities need to manage is car parking facilities and traffic. Finding an available parking spot in the city is always troublesome for drivers, and it tends to become harder with the increasing number of private car users. One study reveals that 30% of the congested traffic in the city is contributed by cars that are searching for parking spots [23]. If the drivers can be informed in advance about the availability of parking spaces at and around their intended destination, the traffic congestion can be efficiently controlled. This requires intelligent sensors to be deployed in the parking lots for monitoring the occupancy as well as intelligent data processing to gain practical insights from the vast amount of data collected.

With the development of sensor technology, many modern cities have been deploying various IoT devices in and around the cities for monitoring. Some examples of such deployments include the City of Melbourne IoT deployment in Australia [11], [22] and Smart Santander deployments [24]. Moreover, wireless in-ground sensors have been installed in parking lots, which record parking events or availability. Cities like San Francisco have made real-time parking information available to the public in order to help people make their decisions about parking. However, in order to efficiently utilise these parking facilities and the real-time data, an automated parking availability prediction mechanism is required. This will help people to plan their trips ahead of time, and therefore save time and traffic congestion in finding available parking spots.

In this paper, we use parking events and availability data collected from two major cities, namely (1) Melbourne, Australia, and (2) San Francisco, USA. We analyse the data and select some key parameters to form three feature sets to use as the input for the parking availability prediction models that we derive. We use three non-parametric algorithms for modelling the parking occupancy rate (OCCR), namely regression tree, neural network and support vector regression. The comparison of the performance of these feature set and model combinations for datasets from both Melbourne and San Francisco reveal that the regression tree with a feature set containing the history of observations, time of day and day of week provides the best performance. The parking availability prediction system that we present can be implemented in mobile app software in order to help people plan their trip and facilitate the way to find the available parking spaces. It can also be built into car navigation systems to help drivers choose parking slots based on their destination.

The rest of the paper is organised as follows: Section 2 presents the state-of-the-art in parking availability prediction. Section II formulates the problem statement and the algorithms used for modelling and prediction. Section III describes the real datasets in detail and evaluates the prediction results. Section V concludes with further research directions.

II. Related Work

Analysing parking data in terms of predicting parking lot availability has received attention in the literature. The main challenges of parking availability prediction are the accuracy of

978-1-4799-8055-0/15/$31.00 ©2015 IEEE

longterm prediction, the interaction between the parking lots in an area, and how user behaviors affect the parking availability.

In [6] and [13], a continuous-time Markov model is built to predict the parking availability in an ad hoc network. In [26], a recurrent neural network is used to forecast the time series parking occupancy up to 60 minutes ahead for the area of parking lots in Santander city. They also used a Weibull parametric model to predict the free parking duration, which consists of two functions, a survival function that computes the probability of available parking time in an area that is larger than a specific time, and a hazard function that computes the failure rate. In [7], Chen et al. presented Generalized Additive Models for availability prediction of shared bike and car parking lots in the centre of Dublin city. The models considerd the variables time of day, time of year, day type, weather, temperature, humidity and the past 2 steps of data, and chose these variables with corresponding functions based on the prediction type (short, medium and long term predictions). They predicted the waiting time utilising an exponential distribution of inter-arrival time. In [3], Beheshti et al. introduced a hybrid model consisting of a Markov chain Monte Carlo (MCMC) approach with an agent-based model, that generates the proposed distribution for MCMC, for parking and traffic prediction around the University of Central Florida. Moreover, they used Metropolis-Hastings to make the training of MCMC more practical. These previous works have focused on prediction models based only on one of the cites. Further, they have focused on a single model for prediction, ignoring any comparative analysis on finding the model that performs better with a given time horizon of prediction. In this paper, we focus on the analysis of three non-parametric models with three different time-series feature sets on datasets from Melbourne and San Francisco.

III. Methodology

Our aim is to forecast the occupancy rate $[0-1]$ of parking lots given a specific date and time, and analyse the effect of different feature sets on the model performance. We consider the following three kinds of feature sets:

- *Feature set 1*: we define the input as $X = \{t, dow\}$ and the output as $y = O(t)$, where $O(t)$ is the occupancy rate at time t, t is the time, which is converted from a 24 hour system $H : M$ to a floating number $H + M/60$, and dow is the day of week.
- *Feature set 2*: we define a feature set with the input as $X = \{O(t-n-k), \ldots, O(t-1-k), O(t-k)\}$ and the output as $y = O(t)$, where $O(t)$ is the same as above, $n+1$ is the number of previous observations and k is the number of steps ahead of occupancy to be predicted.
- *Feature set 3*: we combine feature sets 1 and 2 to form a feature set with the input defined as $X = \{O(t-n-k), \ldots, O(t-1-k), O(t-k), t, dow\}$ and the output is defined as $y = O(t)$.

We address two issues, namely (1) what features yield better predictions in the future on the two different parking data sets, and (2) the relative strengths of different machine learning methods in using these features for prediction.

Algorithm: Here we describe the algorithms that we use for modelling the occupancy rate and for prediction, namely regression tree, support vector regression and neural networks.

1) Regression tree (RT): RT [5] is similar to classification trees; the only difference is that the endpoint is a constant value rather than a class. The criterion for splitting a node is mean square error rather than entropy. RT is easy to build and fast to train. The model is easy to interpret after fitting the data, which can be seen from the layout of the tree whose example is shown in Figure 1(a). From the layout, it is easy to see which features are important based on the feature in the upper node, which is more important since it gives the smallest mean square error for regression or entropy for classification. It also handles parameters with nonlinear relationships. For data with missing values, instead of filling the missing values, we can average all the leaves in the subtree to do prediction although we are not able to reach a leaf at the bottom [9].

The steps involved in the algorithm learning process are as follows: For each node with feature j and split point s, the data can be split into two regions R_1 and R_2 [10]: $R_1(j,s) = \{X|S_j \leq s\}$ and $R_2(j,s) = \{X|S_j > s\}$. To find the best splitting point and feature, we need to solve

$$\min_{j,s}\left[\sum_{x_i \in R_1(j,s)}(y_i - c_1)^2 + \sum_{x_i \in R_2(j,s)}(y_i - c_2)^2\right]$$

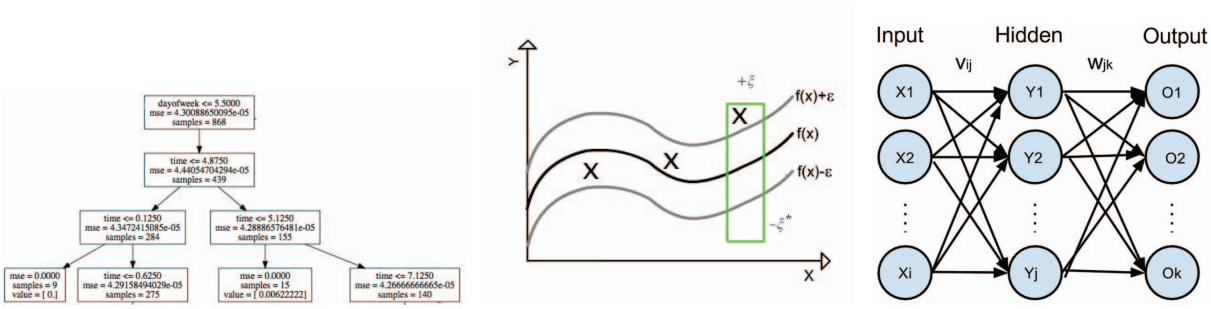
where the best c_1 and c_2 are the average values of the corresponding region [10]. This step also implicitly performs variable screening.

Node splitting is repeated until meeting a stopping criterion, which is an important parameter to set. If the tree is too large, it may cause over-fitting; if it's too small, it may not be able to reveal the characteristics of the data. So a stopping criterion needs to be set, which could be the maximum depth, maximum number of leaf nodes, minimum number of samples at the leaf node and so on [5].

2) Support vector regression (SVR): SVR is an implementation of a Support Vector Machine (SVM) [25] with a slight difference in that it is used to predict real values rather than classes. For the non-linear problem, we can map the input data into a higher dimensional space using a kernel function, which makes the problem linearizable [2]. One of the popular kernel functions is the radial basis function (RBF), $K(x, x') = \exp\left(-\frac{\|x-x'\|_2^2}{2\gamma^2}\right)$ where x and x' are the input vectors, and γ determines the influence area over the data space by a support vector [25]. If γ is set to a large value, the decision surface will be smoother and the decision boundary will be more regular [25]. Similar to SVM, the output can be computed using $f(X) = \sum_i w\varphi(X) + w_0$, where w is the margin, $\psi(X)$ is the kernel function of X, $X = \{x_i : i = 1 \ldots n\}$ and w_0 is the offset value. In order to fit the model, we need to optimize

$$\min_w \tfrac{1}{2}w^T w + C\sum_i(\xi_i + \xi_i^*)$$

subject to: $y_i - \hat{y_i} < \varepsilon + \xi_i$, $\hat{y_i} - y_i < \varepsilon + \xi_i^*$ and $\xi_i, \xi_i^* \geq 0$, where positive constant C is the trade-off between the flatness of $f(X)$ and the number of larger deviations that can be tolerated; ξ and ξ^* are the slack variables, which are used to allow the machine to have a larger error tolerance [25]. ε determines the tolerance threshold of the margin; y_i is the real value of the output and $\hat{y_i}$ is the estimated value. As the example in Fig 1(b) shows, only the points outside the margin ε will be counted in the loss function so that only part of the datasets are contributed to the training process of the model.



(a) (b) (c)

Fig. 1. (a) An example of a regression tree (b) Support vector regression (c) Neural networks

3) Neural network (NN): NN [8] is a machine learning algorithm that imitates the behavior of biological neural networks. There are three layers in the network system: input layer, hidden layer and output layer, as shown in Figure 1(c). The number of input and output layers are always one, while the number of hidden layers can be more than one. Normally the number of hidden layers is less than two, which is enough to model and solve most types of problems. Usually there is only one hidden layer used for most NNs [8].

In order to decide the number of nodes in each layer, we can follow these guidelines: For the input layer, in most cases, it is equal to the number of features in the data. While for time series data, it needs to be optimized by seeing how many steps ahead the prediction data can show the trend. For the hidden layer, there is no theory about the number to be chosen. Based on a rule-of-thumb, the following can be used: (1) the number of nodes in the hidden layer equals the number of input features, (2) the number of nodes in the hidden layer equals the mean number of input features and the output dimension. For the output layer, it is determined by the data to be predicted. Normally there is only one output node. A NN with one hidden layer is shown in Figure 1(c), where X is the vector of input nodes, Y is the vector of the hidden nodes and O is the vector of the output nodes. The mathematical expression is as follows. For the hidden nodes $Y_j = h(V_j X)$ where, $V_j =< V_{0j}, V_{1j}, V_{2j}, \cdots >$ and $X =< X_1, X_2, \cdots >$. For the output node $O_k = f(W_k Y)$ where the weights $W_k =< W_{0k}, W_{1k}, W_{2k}, \cdots >$ and $Y =< Y_1, Y_2, \cdots >$. To fit the regression model, we need to minimise the sum-of-squared errors $R(\theta) = \sum_{k=1}^{K}(O_k - f_k(X_i))^2$. To achieve the minimum error, a common optimisation approach is to use the backpropagation algorithm (BPA). Taking the derivatives of R by v and w respectively, we get

$$\begin{aligned} \frac{\partial R_i}{\partial W_{jk}} &= -2(O_k - f_k(X_i))f_k'(W_k^T Y_j)Y_j \\ \frac{\partial R_i}{\partial V_{ij}} &= -2(O_k - f_k(X_i))f_k'(W_k^T Y_j)W_{jk}h'(V_i^T X_i)X_i \end{aligned}$$

Then the parameters v and w can be updated during the training of each data point as: $W_{jk}^{(r+1)} = W_{jk}^{(r)} - \gamma^{(r)} \sum_i \frac{\partial R_i}{\partial W_{jk}^{(r)}}$, $V_{ki}^{(r+1)} = V_{ki}^{(r)} - \gamma^{(r)} \sum_i \frac{\partial R_i}{\partial V_{ki}^{(r)}}$ where, $\gamma^{(r)}$ is the learning rate [10]. A larger learning rate will give a faster learning speed, but the trained model may perform poorly if the data has a large variability. A smaller learning rate will make the training process slower and may make the optimization converge to a local minimum. So setting an appropriate value of the learning rate is important.

IV. Implementation and Results

In this section, we implement these three algorithms based on three kinds of feature sets on two different datasets and compare the performance of these models.

Datasets: In this section, we describe the datasets used in more detail. The datasets we considered are the parking sensor information collected from two cities, namely San Francisco and Melbourne.

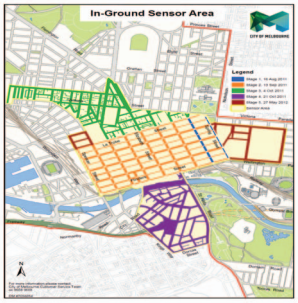
San Francisco data: The city of San Francisco has deployed 8200 parking sensors in city parking lots [21]. Sensors are installed in on-street metered parking spaces and gate controlled off-street facilities [21]. The distribution of sensors in San Francisco is shown in Figure 2(a). The blue paths are the street blocks with sensors that provide real-time parking availability information to the public, and the yellow paths are the street blocks with sensors that do not provide the real-time parking availability information to the public, which are used as the control group. The real time data are collected and made available online for public use and research activities. These sensor data make it more convenient to find vacant parking spots by drivers, bicyclists, pedestrians, visitors, residents and merchants. In addition to the parking availability map available on the SFpark.org website, information on parking availability is distributed via a mobile apps and the regional phone system. By checking parking availability before leaving home, drivers will know where they can expect to find parking and how much it will cost.

We used the public API data feed to collect parking data over a period between 12-08-2013 and 22-09-2013 at a sampling rate of 15 minute time intervals. In total, we have collected 3948 records for each of the 572 street blocks from all the parking sensors deployed in the city. The features collected include (among other features) the number of spaces currently occupied and the number of spaces currently operational for a location. We use these to calculate the *occupancy rate* (OCCR) as follows:

$$OCCR = \frac{\textit{Number of slots currently occupied}}{\textit{Number of slots currently operational}}$$

Melbourne data: The City of Melbourne has installed around 7114 in-ground sensors distributed around 270 street blocks (75 streets) in 23 areas across the Melbourne CBD (central business district) [16], [17]. The installation process is sectioned into five stages. The areas covered for each stage are Spring and Exhibition Streets, the remainder of the CBD

(a) San Francisco, USA [20]



(b) Melbourne, Australia [17]

Fig. 2. Parking sensor distribution

grid, CBD north of La Trobe St and West Melbourne, Southbank and West Melbourne and east of the CBD respectively. Figure 2(b) shows the distribution of sensors in the Melbourne CBD. The path with different colors means different stages of installation.

We use the historic data collected over a period from 01-10-2011 to 30-09-2012, and is available from [15]. There are 12,208,178 parking event records in total. Data features include the area name, street name, between street 1, between street 2, side of street, street marker, arrival time, departure time, duration of parking events (in seconds), sign, in violation, street ID and device ID. Since the occupancy rate is not directly available from the original dataset, we used the following procedure, utilising the parking event recorded by each sensor. Each data vector is formed in the following format; $data = \{street\ block\ name : \{time : occupancy\ delta\}\}$. If the time is the arrival time, we add 1 to the occupancy delta; if it is the departure time we subtract 1 from the occupancy delta. For each street block name, we sorted the list of time and also obtained the occupancy delta list in the same order of the time list. Assuming that the occupancy number is 0 before the sensor turns on, we accumulated the occupancy delta to get the number of slots currently occupied. We sampled the timeline with 15 minute intervals and attached the occupancy number in the nearest occupancy change time point before the sampling point to it as the occupancy number. Therefore, the number of operational parking lots is the maximum occupancy number that has occurred. We then use equation (1) to obtain the occupancy rate.

Data Pre-processing: After retrieving the data, we have performed some pre-processing on the data obtained from both San Francisco and Melbourne. Below, we describe the practical issues that we encountered, and how we solved them in order to use the data in our analysis.

For the San Francisco dataset, the following issues were observed:

- collection failure due to events, such as network issues and power failures, caused missing data
- malfunction of sensors affected both the number of occupancy and operational parking slot values to become zero.

These missing and abnormal data are filled using the average value of the data values at the nearby 15 minutes or 30 minutes time steps. If the 15 minutes data is also missing, then the average value of data at the same time of the day over all dates is used.

For the Melbourne dataset, the following issues were observed:

- For most of the parking lots, the City of Melbourne (council) turns on the sensors at the start of the operational time (e.g., early morning at 7.30 AM) and turns off the sensors at the end of the operational time (e.g., night at 6.30 PM). This caused the occupancy number to increase and decrease drastically. These are abnormal and may interfere with the analysis and prediction of the data.
- The operational time of sensors may be different over a day even for the same parking space. This might cause bias when calculating the mean value. For most streets and times, the parking sensors are turned on at 7.30 AM and turned off at 6.30 PM. For some other streets, the sensors may operate the whole day or only for a specific period during a day.
- There is no data or the occupancy value is zero during holidays or weekends for some of the streets. This makes the analysis and prediction of occupancy at holidays and weekends for some of the streets impossible.
- There may be no records for a period of time. This can be because no parking events happened during that time or the sensor has been turned off.

In order to address these issues, in the first case, we truncated the first and the last data of the day to ignore the abnormal behaviours. For the second issue, we only calculate the mean value over the days for the common times all these days have. For the third case, we have to ignore these days. For the last case, we assumed there are no records in the period between the sensor turning on and off.

We use three metrics to measure the performance of our prediction models; (1) *Mean square error (MSE)* $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ (2) *Mean absolute error (MAE)* $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$ and (3) *Coefficient of determination* $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$ where n is the number of data points and y_i, $\hat{y}_i$ and $\bar{y}_i$ are the true, predicted and mean values respectively.

Results and Discussion: In the experiment, we select the street blocks with more than 50 parking lots for training and testing. In order to prevent overfitting, we used 5-fold cross validation for training. For the experiment involving one step (15 minutes) ahead prediction, we used all the three algorithms on each dataset. For the $k(> 1)$ steps ahead prediction, we only used RT and NN, and omitted the SVR method due to the long computation time. The algorithms are implemented using Python. Particularly, RT and SVR are implemented using scikit-learn [18], and NN is implemented using PyBrain [19].

Feature Set 1 with 1 Step Ahead Prediction: Feature set 1 consists of $X = \{t, dow\}$, $y = O(t)$. For the regression tree, we set the minimum number of samples in the leaf to be 5 to prevent over fitting. For neural networks, we set the model structure to 2-2-1 (2 nodes in the input layer, 2 nodes in the hidden layer and 1 node in the output layer) to fit the data, where the number of nodes in the hidden layer is optimised by using model evaluation with 5-fold cross validation. The optimization method used is the back propagation algorithm (BPA) with a learning rate of 0.01. For SVR, we set the parameters as $C = 1$, $\epsilon = 0.1$, $kernel = RBF$ and $\gamma = 0.1$.

TABLE I. PERFORMANCE OF MODELS USING FEATURE SET 1

	Regression Tree		Neural Network		SVR	
	MELB	SF	MELB	SF	MELB	SF
Mean MSE	0.013	0.010	0.045	0.054	0.038	0.011
Mean MAE	0.079	0.057	0.172	0.194	0.091	0.070
Mean R^2	0.645	0.825	0.004	0.059	0.359	0.804

The average values obtained for the performance metrics using both the datasets with the three algorithms for one step ahead prediction are shown in Table I. From mean MAE, we can see that the RT is 13% better than SVR and 53% better than NN. From mean MSE, we can see that the RT is 65% better than SVR and 70% better than NN. This reveals that the RT performs much better even at higher occupancy levels. Moreover, we can observe from the R^2 valus that the RT fits the data better than the other algorithms, with the R^2 values of 65% and 83% for the Melbourne (MELB) and San Fransisco (SF) data respectively. NN achieves only 0.5% (MELB) and 6% (SF), while SVR achieves 36% (MELB) and 80% (SF) respectively.

The reason for NN's poor performance is due to the fact that it considers the inner correlation of its input features in the modelling. Note that the time of day and the day of week do not have strong correlation among them. Overall, the performance of the RT is better than that of the other two algorithms in all the three performance criteria using the feature set 1.

Feature Set 2 with 1 Step Ahead Prediction: Feature set 2 consists of $X = \{O(t-n-k), \ldots, O(t-1-k), O(t-k)\}$, and $y = O(t)$. In this section we perform one step ahead prediction, i.e., $k = 1$.

In order to find the most appropriate parameter k (number of previous observations) for the feature set, we used 5-fold cross validation to estimate the performance of the models with different n values for the RT. To prevent overfitting, we also set the minimum number of samples in the leaf to 5 and the previous setting, the number of history observations in the input, also have the same effect. For the neural networks, we use grid search with 5-fold cross validation to find the best combination of n and the number of nodes in the hidden layer. We observed that the best n is 1 for the RT and 5 for the NN, which means the prediction is based on the previous 1 and 5 observations, respectively. The number of hidden layer nodes is set to be 5. So the structure of the NN is 5-5-1. For SVR, using 5-fold cross validation, $C = 1$, $\epsilon = 0.1$, $kernel = RBF$ and $\gamma = 0.1$.

Table II shows the average performance of one step ahead prediction for the three algorithms. The performance of RT is still better than NN and SVR, but the difference becomes smaller. From R^2 values, we can see that NN fits 81% (MELB) and 74% (SF) of the data. Compared with feature set 1, we can see that all the models with feature set 2 are remarkably better for one step ahead prediction with an increasing rate of at least 72% for MSE. The RT fits 28% more data, NN fits 80% more data, while the SVR fits 35% more data. This reveals that incorporating the previous observations in the modelling results in more correlation with the prediction occupancy than considering the time of day and the day of week features alone for the one step ahead prediction. Further, it shows that a few past observations can reveal the trend in occupancy more precisely.

TABLE II. PERFORMANCE OF MODELS ON ONE STEP AHEAD PREDICTION USING FEATURE SET 2

	Regression Tree		Neural Network		SVR	
	MELB	SF	MELB	SF	MELB	SF
Mean MSE	0.003	0.001	0.007	0.013	0.004	0.002
Mean MAE	0.035	0.019	0.059	0.089	0.045	0.038
Mean R^2	0.922	0.979	0.811	0.744	0.909	0.957

TABLE III. PERFORMANCE OF MODELS ON ONE STEP AHEAD PREDICTION USING FEATURE SET 3

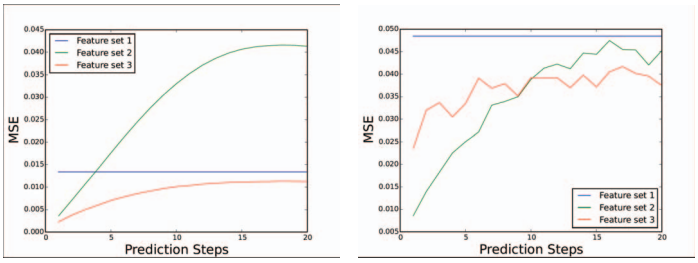
	Regression Tree		Neural Network		SVR	
	MELB	SF	MELB	SF	MELB	SF
Mean MSE	0.002	0.000	0.022	0.039	0.005	0.005
Mean MAE	0.032	0.013	0.109	0.154	0.043	0.053
Mean R^2	0.942	0.986	0.462	0.257	0.903	0.899

Feature set 3 with 1 step ahead prediction: Feature set 3 consists of $X = \{O(t-n-k), \ldots, O(t-1-k), O(t-k), t, dow\}$ and $y = O(t)$. The evaluation results shown in Table III reveal that the RT with the parameter, minimum number of samples in a leaf, set to 5 is better than the NN with model structure 7-7-1 and the SVR with RBF kernel. Comparing with the feature set 2, we can observe a significant performance increase for the RT for the MELB data, with a 37% increase for the mean MSE. There is a 30% improvement observed for the RT using the SF data.

k steps ahead prediction: Now we evaluate the performance of the RT and the NN for increasing numbers of steps ahead prediction. Due to the long computation time, the SVR results are not produced. Figures 3(a) and 3(b) show the performance of up to 20 steps (300 minutes; note that one step corresponds to 15 minutes) ahead prediction with RT and NN algorithms using the three feature sets for the MELB datasets.

It can be seen from the MSE vs the number of prediction steps ahead graph that the RT with feature set 3 is the best for all the steps. Its performance decreases with the increasing number of steps, and gradually approaches the performance of RT using feature set 1. The RT with feature set 2 is only better than RT with feature set 1 until 4 steps (60 minutes). We can see the performance improvement from RT using feature set 2 to RT using feature set 3, is more significant with increasing number of steps, until it reaches a maximum value. It indicates that the time of day and the day of week feature have higher weights towards the prediction with increasing number of steps. However, for NN, the model with feature set 2 is the best until 8 steps and then it is outperformed by that with a feature set 2. Therefore, for short-term prediction (up to 2 hours), the previous observations included in the feature set dominate the performance of the NN, and the time of day and day of week features have a negative effect on the prediction. While for prediction after 2 hours, the latter two features start to dominate the performance. Next we analyse the computational complexity of these prediction methods in detail.

Computational complexity: The computational complexities incurred by each of the prediction algorithms for learning are shown in the Table IV, where p is the number of instances, q is the number of features and r is the total number of



(a) (b)

Fig. 3. k steps ahead prediction for Melbourne data. (a) Regression tree (b) Neural networks

parameters in the neural networks, $r = \sum_{i=1}^{N} n_i$, n_i is number of parameters in node i and N is the total number neurons in the network [14]. It can be seen that the complexity of SVR is the largest and that of RT is the smallest. The recorded time consumption of these models with 5-fold cross validation used in the above sections are also shown in the Table. The experiments were run on a MacOS featuring a 1.8 GHz Intel Core i5 with a 4GB, 1600 MHz, DDR3 RAM. It shows that the RT has the lowest time cost for all the three feature sets used, and the SVR has the largest time cost.

TABLE IV. COMPLEXITY AND TIME COST OF EACH MODEL (SECONDS)

	Regression Tree	Neural Networks	SVR
Complexity	$O(pq \log p)$	$O(pr)$	$O(p^3 q)$
Feature Set 1	0.831	1104.108	21291.527
Feature Set 2	0.524	999.532	342.149
Feature Set 3	1.676	1045.655	5052.197

V. CONCLUSION

Analysing large data collected by smart city deployments is an important task to enable intelligent management of the infrastructure that has been monitored using IoT devices. In this work, we analyse the parking occupancy data collected from two cities, namely San Francisco, USA and Melbourne, Australia. We employed various prediction mechanisms for the parking occupancy rate using three different feature combinations. Further, we compared the relative strengths of different machine learning methods in using these features for prediction. The evaluation reveals that the regression tree, which is the least computational intensive algorithm compared to neural networks and support vector regression, using a feature set that includes the history of the occupancy rates along with the time and the day of the week performs best for parking availability prediction on both the data sets. In the future, we would like to incorporate additional factors into the model that may affect the parking availability predictions, such as events (e.g., social) and the effect of nearby parking lots.

ACKNOWLEDGEMENT

We thank the National ICT Australia; the ARC grants LP120100529 and LE120100129; and the EU FP7 SocIoTal.

REFERENCES

[1] "IoT," http://issnip.unimelb.edu.au/research_program/Internet_of_Things, 2014.

[2] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.

[3] R. Beheshti and G. Sukthankar, "A hybrid modeling approach for parking and traffic prediction in urban simulations," *AI & SOCIETY*, pp. 1–12, 2014.

[4] J. Belissent, "Getting clever about smart cities: New opportunities require new business models," in *http://www.forrester.com/rb/Research/getting_clever_about_smart_cities_new_opportunities/q/id/56701/t/2*, 2013.

[5] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[6] M. Caliskan, A. Barthels, B. Scheuermann, and M. Mauve, "Predicting parking lot occupancy in vehicular ad hoc networks," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*. IEEE, 2007, pp. 277–281.

[7] B. Chen, F. Pinelli, M. Sinn, A. Botea, and F. Calabrese, "Uncertainty in urban mobility: Predicting waiting times for shared bicycles and parking lots," in *Proc. of the 16th Int. IEEE Annual Conf. on Intelligent Transportation Systems, The Hague, Netherlands*, 2013.

[8] M. T. Hagan, H. B. Demuth, M. H. Beale *et al.*, *Neural network design*. Pws Boston, 1996, vol. 1.

[9] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. MIT press, 2001.

[10] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*. Springer, 2009, vol. 2, no. 1.

[11] IoT Deployment, "IoT deployment in the City of Melbourne," http://issnip.unimelb.edu.au/research_program/Internet_of_Things/iot_deployment, 2015.

[12] J. Jin, J. Gubbi, T. Luo, and M. Palaniswami, "Network architecture and QoS issues in the Internet of Things for a Smart City," in *Proc. of the ISCIT*, 2012, pp. 974–979.

[13] A. Klappenecker, H. Lee, and J. L. Welch, "Finding available parking spaces made easy," in *Proceedings of the 6th International Workshop on Foundations of Mobile Computing*. ACM, 2010, pp. 49–52.

[14] C. Leondes, *Neural Network Systems Techniques and Applications: Advances in Theory and Applications*, ser. Control and Dynamic Systems, Neural Network Systems Techniques and Applications, Seven-Volume Set. Elsevier Science, 1998. [Online]. Available: http://books.google.com.au/books?id=BoHXLluvRlMC

[15] MELBDATA, "City of melboune car parking data," https://data.melbourne.vic.gov.au/Transport/Parking-Events-From-Parking-Bays-With-Sensors/8nfq-mtcn?, 2014.

[16] C. of Melbourne, "Melbourne cbd in-ground sensor implementation map," http://www.melbourne.vic.gov.au/ParkingTransportandRoads/Parking/Pages/InGroundSensors.aspx, 5 2012.

[17] ——, "City of melbourne," http://www.melbourne.vic.gov.au/, 2014.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber, "PyBrain," *Journal of Machine Learning Research*, vol. 11, pp. 743–746, 2010.

[20] SFPark, "San francisco parking sensor locations," http://sfpark.org/how-it-works/the-sensors/, 7 2012.

[21] SFpark, "Sfpark," http://sfpark.org, 2013.

[22] A. Shilton, S. Rajasegarar, C. Leckie, and M. Palaniswami, "DP1SVM: A dynamic planar one-class support vector machine for internet of things environment," in *Proc. of the IEEE 10th Intl. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (IEEE ISSNIP))*, April 2015.

[23] D. Shoup, "Free parking or free markets," *ACCESS Magazine*, vol. 1, no. 38, 2011.

[24] SmartSantander, "Smart Santander," http://www.smartsantander.eu/, 2014.

[25] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[26] E. I. Vlahogianni, K. Kepaptsoglou, V. Tsetsos, and M. G. Karlaftis, "Exploiting new sensor technologies for real-time parking prediction in urban areas," in *Transportation Research Board 93rd Annual Meeting*, no. 14-1673, 2014.