On Predicting the Taxi-Passenger Demand: A Real-Time Approach

Luis Moreira-Matias[1,2,3], João Gama[2,4], Michel Ferreira[1,5], João Mendes-Moreira[2,3], and Luis Damas[6]

[1] Instituto de Telecomunicações, 4200-465 Porto, Portugal
[2] LIAAD-INESC TEC, 4200-465 Porto, Portugal
[3] Dep. de Engenharia Informática, Faculdade de Engenharia, U. Porto, 4200-465 Porto, Portugal
[4] Faculdade de Economia, U. Porto 4200-465 Porto, Portugal
[5] Dep. de Ciência de Computadores, Faculdade de Ciências, U. Porto, 4169-007 Porto, Portugal
[6] Geolink, Lda., Avenida de França, 20, Sala 605, 4050-275 Porto, Portugal
{luis.m.matias,jgama,joao.mendes.moreira}@inescporto.pt, michel@dcc.fc.up.pt, luis@geolink.pt

Abstract. *Informed driving* is becoming a key feature to increase the sustainability of taxi companies. Some recent works are exploring the data broadcasted by each vehicle to provide live information for decision making. In this paper, we propose a method to employ a learning model based on historical GPS data in a real-time environment. Our goal is to predict the spatiotemporal distribution of the Taxi-Passenger demand in a short time horizon. We did so by using learning concepts originally proposed to a well-known online algorithm: the *perceptron* [1]. The results were promising: we accomplished a satisfactory performance to output the next prediction using a short amount of resources.

Keywords: taxi-passenger demand, online learning, data streams, GPS data, auto-regressive integrated moving average (ARIMA), perceptron.

1 Introduction

The rising cost of fuel has been decreasing the profit of both taxi companies and drivers. It causes an unbalanced relationship between passenger demand and the number of running taxis, thus decreasing the profits made by companies and also the passenger satisfaction levels [2]. S. Wong presented a relevant mathematical model to express this need for equilibrium in distinct contexts [3]. An equilibrium fault may lead to one of two scenarios: (Scenario 1) excess of vacant vehicles and excessive competition; (Scenario 2) larger waiting times for passengers and lower taxi reliability [3,4]. However, a question remains open: Can we guarantee that the taxi spatial distribution over time will always meet the demand? Even when the number of running taxis already does?

The taxi driver mobility intelligence is an important factor to maximize both profit and reliability within every possible scenario. Knowledge about where the services (i.e. the transport of a passenger from a pick-up to a drop-off location) will actually emerge can be an advantage for the driver - especially when there is no economic viability of adopting random cruising strategies to find their next passenger. The GPS historical data is one of the main variables of this topic because it can reveal underlying running mobility patterns. Today, the majority of the public transportation networks have vehicles equipped with this technology. This kind of data represents a new opportunity to learn/predict relevant patterns while the network is operating (i.e. in real-time).

Multiple works in the literature have already explored this kind of data successfully with distinct applications like smart driving [5], modeling the spatiotemporal structure of taxi services [6–8] or building intelligent passenger-finding strategies [9, 10]. Despite their useful insights, the majority of techniques reported are tested using offline test-beds, discarding some of the main advantages of this kind of signal. In other words, they do not provide any live information about the passenger location or the best route to pick-up one in this specific date/time while the GPS data is mainly a data stream. In this work, we are focused into predicting the short-term spatiotemporal distribution of the taxi-passenger demand by using machine learning algorithms capable of learning and predicting in a real-time environment.

One of the most recent advances on this topic was presented by Moreira-Matias *et. al* in [4,11]. They propose a discrete time series framework to predict the event count (i.e. number of services) for the next P-minutes with a periodicity τ of 30 minutes. This framework handles three distinct types of memory: 1) short term (ARIMA - AutoRegressive Integrated Moving Average [12]), 2) mid term and 3) long term one (both based in time-varying poisson models [13]). This model presented three main contributions facing the existing literature [4]:

1. It builds accurate predictions on a stream environment(i.e. using a real-time test bed);
2. Part of the model is able to *forget* some past data by summarizing it into *sufficient* statistics;
3. It is able to update itself on a short amount of time[1] reusing the last real event count to learn about the novelty thereby introduced;

However, such approach presents two relevant limitations: 1) it just produces predictions each 30 minutes while the decision process is made in real-time (i.e. can we guarantee that a prediction made at 8:00am is still *informative* at 8:20am?); 2) the ARIMA weights are fitted (i.e. re-calculated using an offline learning process) and not updated before each prediction by reusing the entire time series of recent event counts plus the most recent one. A research question arises from this analysis: **Can this model handle with a real-time granularity** (i.e. to build predictions per each period of 5, 2, 1 minute or even *on demand*)**?**

[1] An averaged processing time of 99.77 seconds is reported in [4] to update the predictive model and to build the next prediction.

In fact, this framework is divided into four components that work according to distinct computational models: 1,2) while the two time varying Poisson models are incremental learning methods, 3) the ARIMA model employed is an offline one. Finally, the 4) ensemble method used is an online learning. To ease the interpretation of the system characteristics, some definitions about computational learning methods are presented below.

- **Offline Learning:** a method able to learn a predictive model from a ***finite*** set of instances where the post-training queries do not improve its previous training [14];

- **Incremental Learning:** a method able to learn and update its predictive model ***as long as*** the true labels of the input samples are known (i.e. a stepwise method where each step uses one or more samples) [15];

- **Online Learning:** an *incremental* learning method which is able to update the model ***every time*** a true label of a newly arrived sample is known (i.e. it learns from one instance at time) [14];

- **Real-Time Learning:** an online process able to operate in *real-time* (i.e. to use the last sample *true label* to update the predictive model ***before*** the next sample arrives) [16];

In this paper, we propose a way to minimize the limitations previously described as much as possible by 1) constantly updating the historical time series aggregation of events and by 2) proposing an *incremental* framework to update the ARIMA weights using just the most recent real event count. We tested our improved model by using two distinct case studies: A) a large-sized fleet of 441 vehicles running in the city of Porto, Portugal and B) a small-sized fleet of 89 vehicles running in the cities of Odivelas and Loures, Portugal. While case study A corresponds to a Scenario 1 city - where each vehicle waits on average 44 minutes to pick-up a passenger - case study B is a Scenario 2 one, where the expected waiting time to pick-up a passenger is just 21 minutes.

As input, we considered the data about the services dispatched in each stand over the time. As output, we produced predictions about the demand to arise in the next 30 minutes ($P = 30$) with a periodicity $\tau = 5$ minutes. The test-bed ran continuously for 9 and 6 months for case studies A and B, respectively. However, we just produced outputs for the last four and two months of data, respectively. The results demonstrated the usefulness of our contribution by reducing the typical processing time to each in more than 40% (37.92 seconds) maintaining a satisfying performance - a maximum aggregated error of 24% was accomplished on both case studies.

The remainder of the paper is structured as follows. Section 2 revises the predictive model, describing the extension hereby proposed. The third section describes how we acquired and preprocessed the dataset used as well as some statistics about it. The fourth section describes how we tested the methodology in two concrete scenarios: firstly, we introduce the experimental setup and metrics used to evaluate our model; then, the obtained results are detailed. Finally, conclusions are drawn as well as our future work.

2 Methodology

The model previously proposed in [4, 11] is mainly an incremental one - it just keeps sufficient statistics about the input data constantly updated using the majority of the recently arrived samples. Since the present time series of historical data is a discrete one, it is not easy to propose a true real-time predictive model. However, we can accomplish a good approximation by reducing the prediction periodicity τ from 30 to 5 minutes. The main challenge relies into doing this without increasing the needs on computational resources. In this section, we firstly revisit the model definition. Secondly, we detail how we can maintain an aggregation without recalculating the entire time series in each period. Finally, we propose an *incremental* ARIMA model.

2.1 A Review on the Predictive Model

The following model is deeply described in section II in [4]. Let $S = \{s_1, s_2, ..., s_N\}$ be the set of N taxi stands of interest and $D = \{d_1, d_2, ..., d_j\}$ be a set of j possible passenger destinations. Consider $X_k = \{X_{k,0}, X_{k,1}, ..., X_{k,t}\}$ to be a discrete time series (aggregation period of P-minutes) for the number of demanded services at a taxi stand k. Our goal is to build a model which determines the set of service counts $X_{k,t+1}$ for the instant $t+1$ per each taxi stand $k \in \{1, ..., N\}$. To do so, we propose three distinct short-term prediction models and a well-known data stream ensemble framework to use them all. We formally describe those models along this section.

Time-Varying Poisson Model. Consider the probability to emerge n taxi assignments in a determined time period - $P(n)$ - following a **Poisson Distribution**. We can define it using the following equation

$$P(n; \lambda) = \frac{e^{-\lambda}\lambda^n}{n!} \tag{1}$$

where λ represents the rate (averaged number of the demand on taxi services) in a fixed time interval. However, in this specific problem, the rate λ is not constant but time-variant. So, we adapt it as a function of time, i.e. $\lambda(t)$, transforming the Poisson distribution into a non homogeneous one. Let λ_0 be the average (i.e. expected) rate of the Poisson process over a full week. Consider $\lambda(t)$ to be defined as follows

$$\lambda(t) = \lambda_0 \delta_{d(t)} \eta_{d(t),h(t)} \tag{2}$$

where $\delta_{d(t)}$ is the relative change for the weekday $d(t)$ (e.g.: Saturdays have lower day rates than Tuesdays); $\eta_{d(t),h(t)}$ is the relative change for the period

$h(t)$ on the day $d(t)$ (e.g. the peak hours); $d(t)$ represents the weekday 1=Sunday, 2=Monday, ...; and $h(t)$ the period in which time t falls (e.g. the time 00:31 is contained in the period 2 if we consider 30-minutes periods).

Weighted Time Varying Poisson Model. The model previously presented can be faced as a time-dependent average which produces predictions based on the long-term historical data. However, it is not guaranteed that every taxi stand will have a highly regular passenger demand: actually, the demand at many stands can often be **seasonal**. To face this specific issue, we propose a weighted average model based on the one presented before: our goal is to increase the relevance of the demand pattern observed in the previous week by comparing it with the patterns observed several weeks ago (e.g. what happened on the previous Tuesday is more relevant than what happened two or three Tuesdays ago). The weight set ω is calculated using a well-known time series approach to these kind of problems: the Exponential Smoothing [17].

AutoRegressive Integrated Moving Average Model. The AutoRegressive Integrated Moving Average Model (ARIMA) is a well-known methodology to both model and forecast univariate time series data. A brief presentation of one of the simplest ARIMA models (for non-seasonal stationary time series) is enunciated below. For a more detailed discussion, the reader should consult a comprehensive time series forecasting text such as Chapters 4 and 5 in [12].

In an autoregressive integrated moving average model, the future value of a variable is assumed to be a linear function of several past observations and random errors. We can formulate the underlying process that generates the time series (taxi service over time for a given stand k) as

$$\begin{aligned} R_{k,t} = \kappa_0 + \phi_1 X_{k,t-1} + \phi_2 X_{k,t-2} + ... + \phi_p X_{k,t-p} \\ +\varepsilon_{k,t} - \kappa_1 \varepsilon_{k,t-1} - \kappa_2 \varepsilon_{k,t-2} - ... - \kappa_q \varepsilon_{k,t-q} \end{aligned} \tag{3}$$

where $R_{k,t}$ and $\{\varepsilon_{k,t}, \varepsilon_{k,t-1}, \varepsilon_{k,t-2}, ...\}$ are the actual value at time period t and the *Gaussian white noise'* error terms observed in the past signal, respectively; $\phi_l(l = 1, 2, ..., p)$ and $\kappa_m(m = 0, 1, 2, ..., q)$ are the model parameters/weights while p and q are positive integers often referred to as the order of the model.

Sliding Window Ensemble Framework. How can we combine them all to improve our prediction? In the last decade, regression and classification tasks on streams attracted the community attention due to its drifting characteristics. The ensembles of such models were specifically addressed due to the challenge related with. One of the most popular models is the weighted ensemble [18]. The model we propose below is based on this one.

Consider $M = \{M_1, M_2, ..., M_z\}$ to be a set of z models of interest to model a given time series and $F = \{F_{1t}, F_{2t}, ..., F_{zt}\}$ to be the set of forecasted values to the next period on the interval t by those models. The ensemble forecast E_t is obtained as

$$E_t = \sum_{i=1}^{z} \frac{F_{it} * (1 - \rho_{iH})}{\Upsilon}, \Upsilon = \sum_{i=1}^{z} (1 - \rho_{iH}) \tag{4}$$

where ρ_{iH} is the error of the model M_i in the periods contained in the time window $[t - H, t]$ (H is a user-defined parameter to define the window size) while compared with the real service count time series. As the information is arriving in a continuous manner for the next periods $t, t+1, t+2, ...$ the window will also **slide** to determine how the models are performing in the **last *H* periods**.

2.2 How Can We Update the Time Series Aggregation without a High Computational Effort?

In this paper, we propose to use the model previously described to build predictions with a periodicity τ for the next period of P-minutes (where $P >> \tau \wedge P \bmod \tau = 0$). However, it requires a *newly* calculated discrete time series each τ-minutes. How can we do this calculation without a high computational effort? One of the main ways to do so is to perform an **incremental discretization** [19]. An event count X_t in an interval $[t, t + P]$ will be very *similar* to the count X_{t+1} in the interval $[t+\tau, t+P+\tau]$ (as much as $\tau \sim 0$). We can formulate it as

$$X_{t+1} = X_t + X'_{[t+P,t+P+\tau]} - X'_{[t,t+\tau]} \tag{5}$$

where X' represents both the continuous event count on the first τ-minutes of the interval $[t, t + P]$ and on the τ-minutes immediately after the same period. By generalization, we can define two discrete time series of services demand on a taxi stand k as $X_k = \{X_{k,0}, X_{k,1}, ..., X_{k,t}\}$ and $Y_k = \{Y_{k,0}, Y_{k,1}, ..., Y_{k,t'}\}$ (where $t' \geq t$) using granularities of P and τ minutes, respectively. Let X'_k be the discrete time series needed to predict the event count on the interval $[t', t' + \tau]$. We can define the event count at the time period $[t', t' + P]$ as following

$$X'_{k,t'} = \begin{cases} X'_{k,t'-1} + Y_{k,t'+\theta-1} - Y_{k,t'-1}, \theta = P/\tau \text{ if } t' > t \\ X_{k,t} \text{ if } t' = t \end{cases} \tag{6}$$

We take advantage of the *additive* characteristics of both time series to rapidly calculate a new series of interest maintaining two aggregation levels/layers: P and τ. An illustrative example about how this series can be calculated is presented in Fig. 1.

2.3 An *Incremental* ARIMA Model

The ARIMA model relies on calculating the present event count using a linear combination of previous samples. In eq. 3 the $\phi_l (l = 1, 2, ..., p)$ and $\kappa_m (m = 0, 1, 2, ..., q)$ are the model weights. Such weights usually need to be fitted using the entire historical time series every time we build a new prediction (i.e. an offline learning process). This operation can represent a high computational cost if we employ it at such a large scale as we do here.

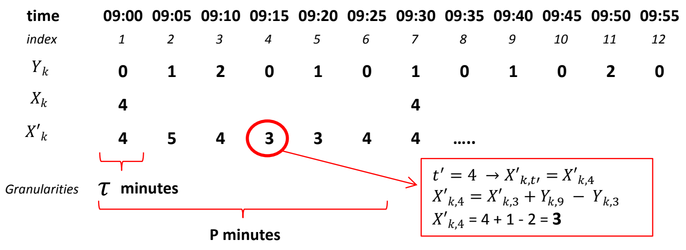

Fig. 1. An example about how can we *additively* calculate one term of the series $X'_{k,t}$

To overcome this issue, we propose to use the *delta rule* to update these weights recursively instead of re-calculating them iteratively as we did so far. The *delta rule* is a gradient descent learning rule commonly used for updating weights of online learning algorithms. It was firstly introduced in [1] to update the weights of the artificial neurons in a single-layer *perceptron*. This rule consists of updating the weights by increasing/decreasing them using a *direct proportion* of the difference between the predicted and the real output. Consider $R = \{R_{k,1}, R_{k,2}, ..., R_{k,t}\}$ to be a time series with the number of services predicted for a taxi stand of interest k in the period $[1, t]$ and $X = \{X_{k,1}, X_{k,2}, ..., X_{k,t}\}$ the number of services actually emerged in the same conditions. Let $w_{k,t} = \{w_{k,t,1}, w_{k,t,2}, ..., w_{k,t,z}\}$ be a set of z weights of a predictive model of interest (like ϕ and κ in the ARIMA one) used to calculate $R_{k,t}$. The update set $\Delta w_{k,t} = \{\Delta w_{k,t,1}, ..., \Delta w_{k,t,j}\}$ can be calculated as follows

$$\Delta w_{k,t,j} = \beta(R_{k,t} - X_{k,t})w_{k,t,j} \tag{7}$$

where β is an user-defined proportionally constant which sets how reactive the model should be and $j \in \{1, ..., z\}$. This way, the ARIMA weights can be *incrementally* updated.

3 Data Acquisition and Preprocessing

In case study A, we focused on the event data stream of a taxi company (which contains 441 running vehicles) operating in Porto, Portugal between Aug. 2011 and April 2012. This Scenario 1 city is the center of a medium size area (consisting of 1.3 million habitants) which contains 63 taxi stands. In case study B, we considered the data broadcasted by the 89 vehicles of a company running in the cities of Odivelas and Loures, Portugal from June to Dec. of 2012. These two cities are part of outer Lisbon - the urban area surrounding the capital of Portugal which has more than 3 million inhabitants. Two maps containing the spatial distribution of the taxi stands of each case study are presented in Fig. 2.

In this section, we describe the data acquisition process and the preprocessing applied to it.

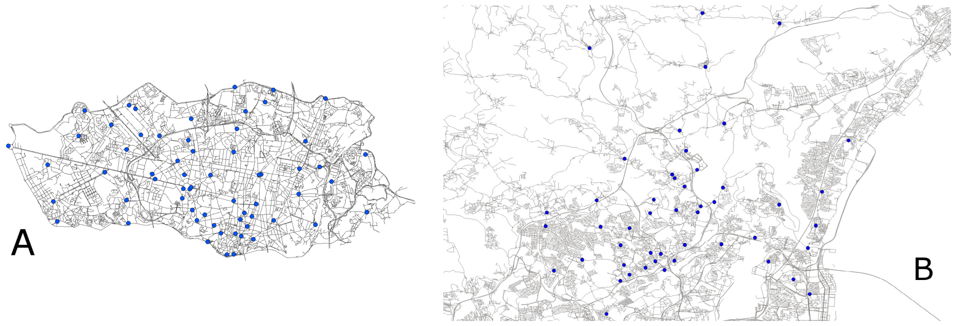

Fig. 2. Taxi Stand spatial distribution over the case studies A and B, respectively

3.1 Data Acquisition

The data was continuously acquired using the telematics installed in the vehicles. These two taxi centrals usually run in one out of three 8h shifts: midnight to 8am, 8am-4pm and 4pm to midnight. Each data chunk arrives with the following six attributes: (1) TYPE – relative to the type of event reported and has four possible values: *busy* - the driver picked-up a passenger; *assign* – the dispatch central assigned a service previously demanded; *free* – the driver dropped-off a passenger and *park* - the driver parked at a taxi stand. The (2) STOP attribute is an integer with the ID of the related taxi stand. The (3) TIMESTAMP attribute is the date/time in seconds of the event and the (4) TAXI attribute is the driver code; attributes (5) and (6) refer to the LATITUDE and LONGITUDE corresponding to the acquired GPS position.

Our study only uses as input/output the services obtained directly at the stands or those automatically dispatched to the parked vehicles (more details in the section below). We did so because the passenger demand at each taxi stand is the main feature to aid the taxi drivers' decision.

3.2 Preprocessing and Data Analysis

As preprocessing, two time series of taxi demand services aggregated were built: one with a periodicity of P-minutes and other with τ minutes. There are three types of accounted events: (1) *busy* set directly at a taxi stand; (2) *assign* set directly to a taxi parked at a taxi stand and (3) *busy* set while a vacant taxi is cruising. We consider both a type 1 and type 2 event as service demanded. However, for each type 2 event, the system receives a *busy* event a few minutes later – as soon as the driver effectively picked-up the passenger – this is ignored by our system.

Table 1 details the number of taxi services demanded per daily shift and day type in the two case studies. Additionally, we could state that, in both cases, the central service assignment is 24% of the total service (*versus* the 76% of

Table 1. Taxi Services Volume (Per Daytype/Shift)

Case Study	**Daytype Group**	**Total Services Emerged**	**Averaged Service Demand per Shift** 0am to 8am	8am to 4pm	4pm to 0am
A	Workdays	957265	935	2055	1422
	Weekends	226504	947	2411	1909
	All Daytypes	1380153	1029	2023	1503
B	Workdays	247694	267	831	531
	Weekends	57958	275	920	674
	All Daytypes	354304	270	826	559

the one demanded directly in the street) while 77% of the service demanded directly in the street is demanded in the stand (and 23% is assigned while they are cruising). In case study A, the average waiting time (to pick-up passengers) of a taxi parked at a taxi stand is 42 minutes while the average time for a service is only 11 minutes and 12 seconds. Such low ratio of busy/vacant time reflects the current economic crisis in Portugal and the inability of the regulators to reduce the number of taxis in Porto. It also highlights the importance of our recommendation system, where the shortness of services could be mitigated by getting services from the competitors. Conversely, the average waiting time in case study B is just 21 minutes.

4 Results

4.1 Experimental Setup

We used a H-sized sliding window to measure the error of our model before each new prediction about the service count of the next period (the metrics used to do so are defined in section 4.2). Each data chunk was transmitted and received through a socket. The model was programmed using the R language [20].

The aggregation period of 30-minutes was maintained ($P = 30$) and a new time series with an aggregation period of 5-minutes ($\tau = 5$) was created according the definition presented in 2.2. Both the ARIMA model (p, d, q values and seasonality) and the weight set ϕ and κ were firstly set (and updated each 24h) by learning/detecting the underlying model (i.e. autocorrelation and partial autocorrelation analysis) running on the historical time series curve of each stand during the last two weeks. To do so, we used an automatic time series function in the `[forecast]` R package - *auto-arima* – and the *arima* function from the built-in R package `[stats]`. The weight set is then *incrementally* updated for each 24h period according with the eq. 7.

The time-varying Poisson averaged models (both weighted and non-weighted) were also updated every 24 hours. A sensibility analysis carried out with data previous to the one used on these experiments determined the optimal values for the parameters α, β and H as 0.4, 0.01 and 4 (i.e. it represents a sliding window of 20 minutes), respectively. The hardware configuration is equivalent to the one used in [4, 11].

4.2 Evaluation Metrics

We used the data obtained from the last four and two months to evaluate our framework of case studies A and B, respectively. A well-known error measurement was employed to evaluate our output: the Symmetric Mean Percentage Error (*sMAPE*) [21]. However, this metric can be too intolerant with small magnitude errors (e.g. if two services are predicted at a given period for a taxi stand of interest but no one actually emerges, the error measured during that period would be 1). To produce more accurate statistics about series containing very small numbers, we can add a Laplace estimator [22] to the previous definition of (*sMAPE*) in [21]. In this case, we will do it by adding a constant c to the denominator (i.e.: originally, it was added to the numerator to estimate a success rate [22]). Therefore, we can re-define $sMAPE_k$ as follows

$$sMAPE_k = \frac{1}{t}\sum_{i=1}^{t} \frac{|R_{k,i} - X_{k,i}|}{R_{k,i} + X_{k,i} + c} \tag{8}$$

where c is a user-defined constant. To simplify the estimator application, we will consider its most used value: $c = 1$ [22].

This metric is focused just on one time series for a given taxi stand k. However, the results presented below use an weighted average of the error as evaluation measure. The weight of each stand error is the number of services emerged in the stand during the test period.

4.3 Results

The error measured for each model in the two case studies considered is presented in Table 2. The results are firstly presented per shift and then globally. The overall performance is good: the maximum value of the error using the ensemble was **25.90%** during the evening shift. The sliding window ensemble is always the best model in every shift and case study considered. The models just present

Table 2. Error Measured on the Models using *sMAPE*

Case Study	**Model**	**Periods**			
		00h–08h	**08h–16h**	**16h–00h**	**24h**
A	Poisson Mean	27.67%	24.29%	25.27%	25.32%
	W. Poisson Mean	27.27%	24.62%	25.66%	25.28%
	ARIMA	28.47%	24.80%	25.60%	26.21%
	Ensemble	**24.86%**	**23.14%**	**24.07%**	**23.77%**
B	Poisson Mean	28.66%	21.10%	23.34%	23.99%
	W. Poisson Mean	26.27%	22.01%	24.32%	23.83%
	ARIMA	31.88%	21.10%	23.63%	25.53%
	Ensemble	**25.90%**	**19.97%**	**22.09%**	**21.80%**

slight discrepancies within the defined shifts. Our model took - in average - 37.92 seconds to build the next prediction about the spatiotemporal distribution of the demand by all stands.

5 Final Remarks

In this paper, we proposed a method to apply a complex learning model [4,11] to build predictions about the taxi passenger demand in a real-time environment. We did so by extending the typical definition of an ARIMA model [12] to an *incremental* one using **the delta rule** - a rule firstly introduced in the perceptron algorithm [1] which is able to update its weights step by step.

We tested this approach using two case studies with distinct scenarios in Portugal: A) in the city of Porto, where the number of vehicles is larger than the demand and B) in Odivelas and Loures, which have the opposite situation. Our model was able to produce predictions about the spatiotemporal distribution of the demand during the next 30 minutes $P = 30$ with a periodicity τ of 5 minutes. The results demonstrated the relevance of our contribution: we maintained the aggregated error ratio lower than 24% and 22% in the case studies A and B, respectively. On the other hand, we were able **to reduce the typical computational time** used to build each prediction **by 40%** (from the 99.77 seconds firstly proposed in [4] to 37.92 seconds accomplished by the present framework).

Acknowledgements. The authors would like to thank Geolink, Lda. and its team for the data supplied. This work was supported by the VTL: "Virtual Traffic Lights" and KDUS: "Knowledge Discovery from Ubiquitous Data Streams" projects under the Grants PTDC/EIA-CCO/118114/2010 and PTDC/EIA-EIA/098355/2008 grants.

References

1. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386 (1958)
2. Schaller, B.: Entry controls in taxi regulation: Implications of us and canadian experience for taxi regulation and deregulation. Transport Policy 14(6), 490–506 (2007)
3. Wong, K., Wong, S., Bell, M., Yang, H.: Modeling the bilateral micro-searching behavior for urban taxi services using the absorbing markov chain approach. Journal of Advanced Transportation 39(1), 81–104 (2005)
4. Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., Damas, L.: Online predictive model for taxi services. In: Hollmén, J., Klawonn, F., Tucker, A. (eds.) IDA 2012. LNCS, vol. 7619, pp. 230–240. Springer, Heidelberg (2012)
5. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: driving directions based on taxi trajectories. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 99–108. ACM (2010)

6. Deng, Z., Ji, M.: Spatiotemporal structure of taxi services in Shanghai: Using exploratory spatial data analysis. In: 2011 19th International Conference on Geoinformatics, pp. 1–5. IEEE (2011)
7. Liu, L., Andris, C., Biderman, A., Ratti, C.: Uncovering taxi drivers mobility intelligence through his trace. IEEE Pervasive Computing 160, 1–17 (2009)
8. Yue, Y., Zhuang, Y., Li, Q., Mao, Q.: Mining time-dependent attractive areas and movement patterns from taxi trajectory data. In: 2009 17th International Conference on Geoinformatics, pp. 1–6. IEEE (2009)
9. Li, B., Zhang, D., Sun, L., Chen, C., Li, S., Qi, G., Yang, Q.: Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In: 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 63–68 (March 2011)
10. Lee, J., Shin, I., Park, G.: Analysis of the passenger pick-up pattern for taxi location recommendation. In: Fourth International Conference on Networked Computing and Advanced Information Management (NCM 2008), pp. 199–204. IEEE (2008)
11. Moreira-Matias, L., Gama, J., Ferreira, M., Damas, L.: A predictive model for the passenger demand on a taxi network. In: 15th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1014–1019 (2012)
12. Cryer, J., Chan, K.: Time Series Analysis with Applications in R. Springer, USA (2008)
13. Ihler, A., Hutchins, J., Smyth, P.: Adaptive event detection with time-varying poisson processes. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 207–216. ACM (2006)
14. Burke, E., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.: A classification of hyper-heuristic approaches. In: Handbook of Metaheuristics, vol. 146, pp. 449–468. Springer, US (2010)
15. Chalup, S.: Incremental learning in biological and machine learning systems. International Journal of Neural Systems 12(06), 447–465 (2002)
16. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Real-time learning capability of neural networks. IEEE Transactions on Neural Networks 17(4), 863–878 (2006)
17. Holt, C.: Forecasting seasonals and trends by exponentially weighted moving averages. International Journal of Forecasting 20(1), 5–10 (2004)
18. Wang, H., Fan, W., Yu, P., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235. ACM (2003)
19. Gama, J., Pinto, C.: Discretization from data streams: applications to histograms and data mining. In: Proceedings of the 2006 ACM Symposium on Applied Computing, pp. 662–667. ACM (2006)
20. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2012)
21. Makridakis, S., Hibon, M.: The m3-competition: results, conclusions and implications. International Journal of Forecasting 16(4), 451–476 (2000)
22. Jaynes, E.: Probability theory: The logic of science. Cambridge University Press (2003)