Predicting traffic of online advertising in real-time bidding systems from perspective of demand-side platforms

Conference Paper · December 2016

DOI: 10.1109/BigData.2016.7841012

Yi-Cheng Chen
Tamkang University
37 PUBLICATIONS **207** CITATIONS

Predicting Traffic of Online Advertising in Real-time Bidding Systems from Perspective of Demand-Side Platforms

Hsu-Chao Lai[1], Wen-Yueh Shih[1], Jiun-Long Huang[1], and Yi-Cheng Chen[2]
[1]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC
[2]Department of Computer Science and Information Engineering, Tamkang University, New Taipei City, Taiwan, ROC
Email: heyguys123.eecs99@g2.nctu.edu.tw, wen21318@hotmail.com, jlhuang@cs.nctu.edu.tw, ycchen@mail.tku.edu.tw

***Abstract*—Online advertising has been all the rage these years. Budget control and traffic prediction turn out to be important issues for the demand-side platforms (DSPs). However, DSPs cannot easily grab the information of audiences and media platforms. Although DSPs might have the information immediately, it is still hard to response the request of advertisements in real-time due to the high volume of features. Therefore, we propose a method predicting traffic of requests from perspective of DSPs. The features we used are simple to be extracted from historical data. The prediction model we chose is regression model with closed-form solution. Both the features and regression model make our prediction adaptive in real-time systems. Our method can detect traffic anomalies and prevent it from overwhelming prediction. Moreover, our method can also keep pace of the trend. Experiment results show that our method's error rate of prediction is about 0.9% in total, and 10% per time unit.**

***Index Terms*—Real Time Bidding, Online Advertisement, Request Traffic, Demand Side Platform**

I. Introduction

Since the internet, computers and smart devices are booming these years, people start to acquire information from browsing on the internet instead of watching news on televisions. This significant behaviour change leads to a new business opportunity in online advertising. Media platforms such as websites and mobile applications turn their blanks into advertising columns. Therefore those platforms can charge advertisers for their advertisements being seen or clicked by audiences. According to [1], the profit of online advertising has grown at least 17% every year since 2012, and peaked at 61.6 million U.S. Dollars in 2015 in Taiwan.

Online advertising works through real-time bidding systems (RTB). RTB will hold auctions between platforms and advertisers, and then the very one winner ad will be displayed to audiences. There are three main parts in RTB, which are demand-side platforms (DSPs), supply-side platforms (SSPs) and ad exchange (ADX). SSPs supply advertising opportunities, while DSPs handle advertisers' advertisements and decide which ad should be nominated to attend the auction held by ADX. RTB's workflow is shown in Fig. 1 and described as follows. (1) An audience opens a website and generates ad requests. The SSP sends the requests and information of both audience

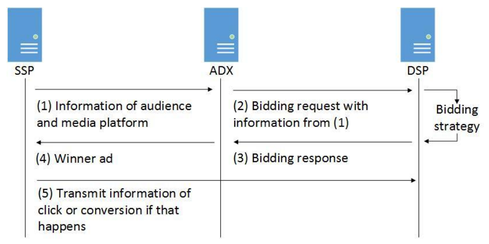
SSP
ADX
DSP
(1) Information of audience and media platform
(2) Bidding request with information from (1)
Bidding strategy
(4) Winner ad
(3) Bidding response
(5) Transmit information of click or conversion if that happens

Fig. 1. Workflow of real-time bidding systems

and website to the ADX. (2) ADX forwards the request from (1) to DSPs. (3) Every DSP decides and responses to the ADX that which ad can join the auction by bidding strategies. (4) After receiving DSPs' bidding responses, the ADX will return the ad which wins the auction to the SSP. The SSP will show the winner ad to the audience. (5) If the audience clicks or converses, such as installing an APP or buying stuffs from the ad, the SSP will transmit this information to the DSP. The DSP will record this information as the outcome of displaying this advertisement.

Every DSP has to choose one ad from those it manages to attend the auction. The bidding strategies of DSPs turn out to be significant. There are three main topics in DSPs' bidding strategies, including value evaluation [2]–[4], price prediction [5], [6] and budget control [7], [8]. Click-through rate (CTR) and conversion rate (CVR) are common value evaluations of the relationship between an ad and a bidding request. The higher the CTR or CVR, advertisers are more willing to pay more for winning the bidding. McMahan et al. proposed FTRL-Proximal in [4] and Kaiyanakrishnan et al. proposed to use decision trees in [3] to predict CTR. Lee et al. proposed to predict CVR by different combinations between data hierarchies in [2]. DSPs need to know how much the winning price will be after value evaluation. Wu et al. proposed to use both winning and losing logs to predict the winning price in [5]. Pacing models make strategies of how advertisers should spend their budget [7], [8]. Agarwal et al. proposed the bidding rate to be proportional to number of requests in [7].

If pacing models over-estimate the traffic of requests, the budget will be consumed too slow and cause residual. If pacing models under-estimate the traffic, the budget will be consumed early and miss high-value opportunities (e.g., potentially high-CTR requests) in the remaining time. Therefore, predicting traffic of requests turns out to be important. Traffic is highly correlated to humans' behaviour. Agarwal et al. and Cetintas et al. predict the number of user visits of websites in [9], [10]. However, it is not practical to predict every website's traffic whenever a time slot starts so that DSPs can finally take the sum of those traffic to be their predicting results. Time series analysis is also a method to predict traffic. Nevertheless, C. Chen and L.-M. Lin [11] have mentioned that "temporary change" is hard to model in time series model. If we do so, the temporary change will occur occasionally without reasons, which will harm the prediction badly. They then choose to ignore temporary changes to keep the level of prediction in general cases. This strategy is not suitable for predicting traffic because when traffic anomalies happen, it usually influences for a period of time. If we keep ignoring the anomalies, we would misestimate a lot of traffic.

To the best of our knowledge, this paper is the first research which predicts traffic of online advertising from perspective of DSPs. DSPs are asked to response to bidding request fast, so heavy feature preprocessing in related works are not suitable. We choose the features that are easy for DSPs to extract, and select the predicting models which can train and predict fast. Besides, our method can detect and handle outliers. The dataset used in experiments is real log of a DSP in Taiwan. There are 1,692,287,865 requests in testing data, and our prediction is 1,677,132,075 requests in total. The total error rate is about 0.9% and 10% per time unit (here we use one hour as unit).

In this paper, we are going to discuss related work in Section II. We will discuss our method in Section III and experiments in Section IV. Finally, the conclusion is in Section V.

II. Related works

Traffic prediction of online advertising plays an importatnt role in RTB. If we can predict traffic precisely, pacing models can allocate budget more efficiently. We will discuss related works about DSPs' bidding strategies, traffic prediction and time series analysis in this section.

A. DSPs' bidding strategies

Here we discuss what strategies DSPs will need after receiving bidding requests from ADXs. There are three main topics: value evaluation, price prediction and budget control. DSPs will decide how fast the budget should be spent. Then they will evaluate the value of the coming requests and predict the prices to win the auctions. In the end, DSPs will decide if their ads attend the auction according to the result of previous strategies.

1) Value evaluation: CTR and CVR are important evaluation metrics to online advertising. They can directly indicate that how the ads are appealed to audiences. In order to make clicks and conversions happen, finding conditional probability of clicks or conversions given feature combinations of audiences, platforms and ads is the general method.

Lee et al. proposed to predict CVR by combinations of feature hierarchies of audiences, platforms and ads in [2]. For example, audiences can be separated into males and females. Males can be further separated by different hobbies such as travelling, sports, etc. Then this method computes weak estimators of different combinations of every data point. Combinations start from the leaves of three hierarchies and up to the next level, which represent the aggregations CVR of similar features. All the weak estimators are used to be the features of predicting CVR. This method aggregates conversions which are very rare in real datasets. However, it cannot deal with new features.

Kaiyanakrishnan et al. proposed to use a decision tree to cluster features and predict CTR by similar clusters in [3]. Every node of the tree splits dissimilar feature combinations into two nodes. This method reduces high-dimensional features to a very low-dimensional space, which is only the height of the decision tree. The prediction time therefore reduces a lot, but the training time is long. Every split needs to compute every possible split to get the best result, which leads to long training time.

McMahan et al. proposed to use FTRL-Proximal to predict CTR by online learning in [4]. For the features they used, the $ProbabilisticFeatureInclusion$ is proposed to have a probability to add new features into their models , so the rare the features can be eliminated from feature space. They also modify online gradient descent's (OGD) objective function to make it more robust to sparse data, such as online advertising. However, FTRL-Proximal shares the same disadvantage with OGD which complicates convergence to the global minimum.

2) Price prediction: DSPs need to know if they can afford the price after value evaluation. RTB follows the rule of second price auction [12]. In other words, the highest bidding price wins the auction, but the winner just needs to pay the second highest bidding price. Therefore, prediction objectives, bidding price and winning price, represent the highest and the second highest price respectively.

Wu et al. proposed to predict winning price by both winning and losing logs in [5]. DSPs wouldn't have known the winning price if they lose the bids, so people usually predict winning price without using losing logs before this study. The authors use these losing logs as censored data, and use a censored regression to fit it. The combination of censored regression and a linear regression fitting winning logs is the final model predicting winning price.

Zhang et al. predicted bidding price by historical winning rate in [6]. They found that there is a high dependency between bidding price and winning rate. This method is also highly flexible, which can be adjusted by different optimization goals.

3) Budget control: DSPs usually use pacing models to control the speed of spending budget. Pacing models control budget by controlling the pacing rate (bidding rate) to make the budget able to be smoothly spent during the lifetime of campaign.

Agarwal et al. proposed to adjust the pacing rate according to traffic in different time slot in [7]. The pacing rate is set to be low whenever a time slot starts, and it will steadily grow if the goal haven't been achieve during the slot. The reason is that the model can learn the relationship between ads and requests. Experiments reveal this method results in 10% to 24% growth in buying impressions.

Xu et al. proposed to allocate budget by performance, such as CTR and CVR, in different time slot in [8]. This method computes historical performance of every requests and aggregates those having similar performance. When DSPs receiving high performance requests, DSPs will have a higher pacing rate to those requests. Therefore, this method can have a better performance under a limited budget.

B. *Traffic prediction*

Agarwal et al. predicted targeted user visits of guaranteed contracts in [9]. For example, advertisers would like to know how many 25 to 30-year-old users interested in travelling will visit a news website. However, there are billions of possible feature combinations making the training impractical without dimension reduction. They iteratively choose the most representative feature combinations as the basis set. When a query like the example above comes, the method finds the most similar feature combination in the basis set and computes the correlation between the query and the feature combination. Traffic prediction of the query will be the correlation times the feature combination's historical traffic. Although lowering the dimension of features, this method would take a lot of training time.

Cetintas et al. used probabilistic models to predict user visits of a website in [10]. They learned the latent classes that affect the number of user visits in different time. However, this model takes plenty of time. Besides, it is impractical for DSPs to predict every websites and aggregate them as its prediction.

C. *Time series analysis*

Box et al. proposed ARIMA model in [13]. ARIMA is widely used in time series analysis. It can analyse and predict stationary and seasonal time series data. However, [14] has mentioned that ARIMA cannot handle temporary change (TC) properly. If we model TCs into ARIMA's objective function, TCs will come out of nowhere in the prediction. It would harm the precision of the prediction.

Hillmer et al. separated ARIMA model into three parts: season, trend and white noise in [15]. This method solves the problems of TCs and boosts the training time of ARIMA. Although there are polynomial time solutions for models of season, trend and white noise respectively, there still exists the exponential possible combinations of parameters of their objective functions. The training time is still not feasible to online advertising.

III. The Proposed Method

A. *Problem definition*

In practices, the number of ad requests is highly dependent of humans' behaviour. As shown in Fig. 4, people usually sleep at night and work during day. Therefore, the traffic in day and night is obviously different. In weekday, traffic peaks at the time people eat and before sleep. In weekend, the peaks are not as obvious as those in weekday. The reason is that humans' behaviour is not as regular in weekend as it in weekday. Our method analyses humans' behaviour over time, and predicts traffic of online advertising accordingly.

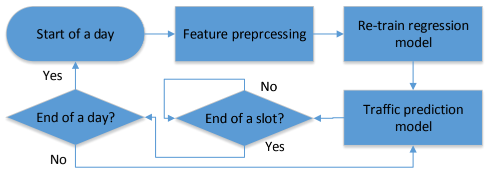

Fig. 2. System architecture

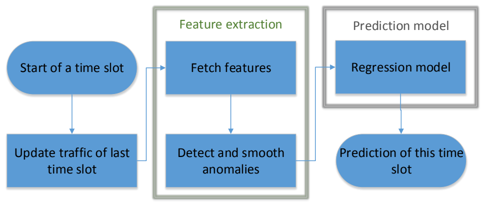
Feature extraction
Prediction model
Start of a time slot
Fetch features
Regression model
Update traffic of last time slot
Detect and smooth anomalies
Prediction of this time slot

Fig. 3. Module of traffic prediction

We slice every day into T equal-length time slots. For example, if $T = 24$, there will be 24 time slots in a day and every slot is 1 hour. We then formulate our problem as the following equation:

$$\arg_\theta \min \sum_{\forall d,t} loss(h(X_{d,t};\theta), Y_{d,t}),\ 1 \le d \le D,\ 1 \le t \le T \tag{1}$$

Suppose there are D days in our training data. Let $Y_{d,t}$ denote the number of requests during the t^{th} time slot in day d. The feature set which can describe $Y_{d,t}$ is denoted as $X_{d,t}$. Function $h(\cdot;\theta)$ predicts the traffic with feature set $X_{d,t}$ and parameters θ. Loss function $loss(\cdot)$ calculates the loss between the true value $Y_{d,t}$ and the predicted value $h(X_{d,t};\theta)$. The loss should be proportional to the absolute value of the difference between $Y_{d,t}$ and $h(X_{d,t};\theta)$. We train the prediction function h to minimize the sum of the losses.

B. *System architecture*

The architecture of our system is shown in Fig. 2. We insert traffic of yesterday into training dataset and pop the first day in the set to maintain that there are always D days in the training dataset. Then we re-train the prediction model h. Because our input and output data are all numerical data, regression would be a proper solution of prediction model. Whenever a time slot starts, our system will predict the number of requests by traffic prediction module shown in Fig. 3. Repeat the prediction until the end of a day, and start the procedure over.

Prediction module will be activated when a time slot starts. The module first updates the traffic of last time slot. Then it fetches features from historical data, and proceeds anomalies

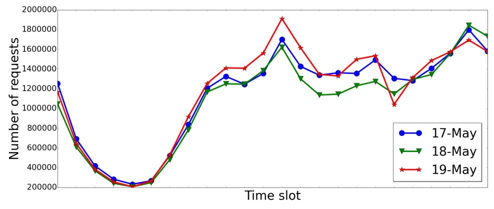

Fig. 4. Example of $LastNDayReqs$

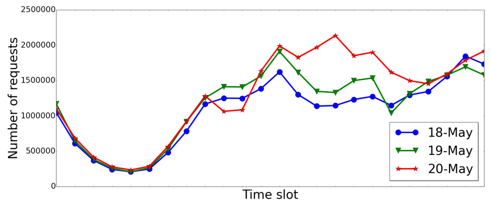

Fig. 5. Example of $LastSlotReqs$

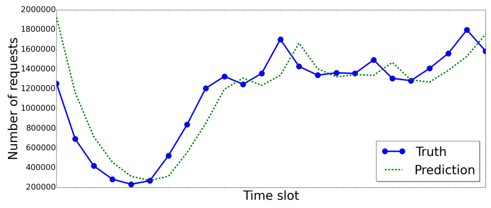

Fig. 6. Example of predicting without $SLotNumber$

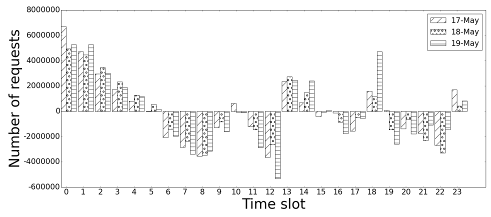

Fig. 7. Differences of real and prediction values

detection and smoothing on those features. Details of handling anomalies will be discussed in Section III-D. After the features are ready, the module will predict the traffic by those features and the underlying regression model.

C. Feature extraction

In order to predict and train in real time, we decide to extract the features from historical traffic without using the features of online advertising. The advantage is that we do not have to handle billions of sparse features, such as information of audiences and platfroms, which are hard for DSPs to fetch. We use three types of features which can represent humans' behaviour:

1) $LastNDayReqs$: traffic during the same slot in previous N days;
2) $LastSlotReqs$: traffic during the last slot;
3) $SlotNumber$: the numbering of the slot.

The first two features represent long-term and short-term effect respectively. By observing the real dataset[1] from a DSP in Taiwan, as shown in Fig. 4, the traffic of the same slot in consecutive three days is close to each other. The reason is that humans' behaviour is usually regular and highly correlated to time. For example, people usually wake up, commute between home and office, have meals and sleep at almost the same time every day, implicitly making the traffic correlated to time. Therefore, we take the traffic during the same slot in previous N days as a feature and denote it as $LastNDayReqs$.

As shown in Fig. 5, the third day in the figure shows continuous traffic anomalies from the 15^{th} to 19^{th} time slots. The reason causing the anomalies may be short-term activities, such as important news or limited free virtual goods of online games. However, it's hard for DSPs to know when the activities will start or end. As a result, we need a feature to keep pace with the short-term effect. We propose traffic of the last slot, denoted as $LastSlotReqs$, to be the solution. Directly using $LastSlotReqs$ will harm the prediction result because sometimes the bias between real $LastSlotReqs$ and expected traffic is too large. We will discuss our strategies to handle a severe anomaly in Section III-D.

Predicting by $LastNDayReqs$ and $LastSlotReqs$ is not enough. The result of predicting only with the two features is shown in Fig.6. When the traffic is descending, the effect of $LastSlotReqs$ will make the prediction descend slower than it should be and vice versa. To solve this problem, we observe the difference between the every prediction value and real value, which is shown in Fig.7. It's obvious that the differences in the same time slots are either over-estimated or under-estimated. Moreover, the values of the differences are very close in the same time slots. Based on this observation, we use the third feature $SlotNumber$ to represent such differences. $SlotNumber$ is a T-length binary array. Every value in $SlotNumber$ is 0 except the one whose index of the array is equal to the numbering of the time slot. For example, if T=24, $SlotNumber$ will be $[1, 0, ..., 0]$ for the data during 0 to 1 a.m. This design makes every single data affected by one coefficient, which is expected to be the difference observed above, at a time. $SlotNumber$ therefore can compensate the insufficiency of predicting only by $LastNDayReqs$ and $LastSlotReqs$.

[1]Details of the dataset are described in Section IV-A.

D. Detecting and smoothing anomalies

We mentioned that we fetch features from historical traffic data. Nevertheless, there are anomalies which bias a lot from historical data occasionally. If we predict with those anomalies, the prediction result would be degraded hardly. We now discuss how to handle these severe anomalies.

There are two possible reasons for anomalies. One is purely unusual traffic, and the other is a part of a trend. We should be aware that some of the anomalies are from trends while designing the detection of anomalies. When we are going to predict traffic of the t^{th} time slot, $LastSlotReqs$ will be examined by traffic of the $(t-1)^{th}$ time slots in the previous k days. Traffic of these k time slots is mapped to Gaussian distribution. If original $LastSlotReqs$ is tol standard deviations farther away from the mean, it will be treated as a severe outlier. After detecting a severe outlier, our strategy is to smooth $LastSlotReqs$ by the traffic of previous k days at the same time unit. The smoothing technique we use is geometric mean because it is more robust to outliers than arithmetic mean does. The pseudo code is shown in Algorithm 1. Lines 1 to 8 are the detecting part and Line 10 is the smoothing part.

Algorithm 1 SmoothedLastSlotReqs

```
Input: $Reqs$: one-dimensional array storing historical request traffic
    $len$: denotes the latest updated index
    $T$: total number of slots in one day
    $k$: number of days we used to draw a Gaussian distribution
    $tol$: tolerance of anomaly in z-score scale
Output: Smoothed $LastSlotReqs$ if anomaly is detected
    Original $LastSlotReqs$ if it's not an anomaly
 1: $OriginalLastReqs = Reqs[len]$
 2: $History = []$
 3: for $i \in [1, k]$ do
 4:     Append $Reqs[len - i \times T]$ to $History$
 5: end for
 6: $avg$ = arithmetic mean of $History$
 7: $std$ = standard deviation of $History$
 8: if $\frac{\|OriginalLastReqs - avg\|}{std} > tol$ then
 9:     Append $OriginalLastReqs$ to $History$
10:     $LastSlotReqs$ = geometric mean of $History$
11: else
12:     $LastSlotReqs = OriginalLastReqs$
13: end if
```

For example, suppose that $k = 3$, $tol = 1$. The original $LastSlotReqs$ is 40,000, while the traffic of previous 3 days are 12,000, 15,000 and 18,000. The mean and standard deviation of the previous three days are 15,000 and 5,000 respectively. It's obvious that the original $LastSlotReqs$ is more than one standard deviation farther away from the mean. As a result, it is an significant outlier in our model. The geometric mean $\sqrt[4]{12,000 \times 15,000 \times 18,000 \times 40,000} = 18,973$ will be the smoothed $LastSlotReqs$ and used in prediction. Arithmetic mean, which is 21,250, is harmed by outliers hardly, so geometric mean is better in this case.

We smooth $LastNDayReqs$ in a similar way. However, we don't detect anomalies here but calculate the geometric mean of traffic in the t^{th} time slots in previous N days. The reason is that this feature represents a long-term effect. If the trend of traffic keeps growing, this feature should also show the growth. The pseudo code is shown in Algorithm 2.

E. Regression model

From our feature extraction, both input and output of prediction function h in Eq. 1 are all numerical data. Regression models are proper solutions to prediction model due to numerical input and output. We next discuss how to combine the features and the regression model.

Algorithm 2 SmoothedNDayReqs

```
Input: $Reqs$: one-dimensional array storing historical request traffic
    $len$: denotes the latest updated index
    $T$: total number of slots in one day
    $N$: number of days we used to make $LastNDayReqs$
Output: Geometric mean of the traffic during the same slot in previous N days
 1: $History = []$
 2: for $i \in [1, N]$ do
 3:     Append $Reqs[len - i \times T + 1]$ to $History$
 4: end for
 5: return geometric mean of $History$
```

First of all, we re-write Eq. 1 into the following regression model's from:

$$\begin{aligned}&\arg_{w,b}\min\sum_{\forall d,t} loss((w^T X_{d,t} + b) - Y_{d,t}),\\ &1 \le d \le D,\ 1 \le t \le T\end{aligned} \tag{2}$$

$X_{d,t}$ is composed of $LastNDayReqs$, $LastSLotReqs$ and $SlotNumber$. w and b are the coefficients and intercept of a regression model. We optimize w and b to minimize the error. Prediction method is shown in Algorithm 3. Line 1 updates the recent receiving traffic to training data, Lines 2 to 7 are feature preprocessing, Lines 9 to 11 eliminate the oldest data and Line 12 will return the prediction value.

Algorithm 3 PredictTraffic

```
Input: $Reqs$: one-dimensional array storing historical request traffic
    $T$: total number of slots in one day
    $N$: number of days we used to make $LastNDayReqs$
    $h(\cdot; w, b)$: regression model with parameters $w$ and $b$
Output: Prediction of traffic in the coming time slot
 1: Append the recently obtained traffic to $Reqs$
 2: $len$ = the last index in $Reqs$ whose value isn't NULL // the newest updated index of $Reqs$
 3: $t = (len + 1) \mod T$ // $t$ denotes the index of the coming slot
 4: $LastNDayReqs_t = SmoothNDayReqs(Reqs, len, T, N)$
 5: $LastSlotReqs_t = SmoothedLastSlotReqs(Reqs, len, T, k, tol)$
 6: $SlotNumber =$ array in length $T$ filling with 0s
 7: $SlotNumber[t] = 1$
 8: $X_t = [LastNDayReqs_t, LastSlotReqs_t, SlotNumber]$
 9: if $t$ is 0 then
10:     Pop first $T$ elements of $Reqs$ and shift the rest of data
11: end if
12: return $h(X_t; w, b)$
```

IV. Experiments

A. Dataset and environments

The data we use are real logs from a DSP in Taiwan and the information is shown in Table I. It starts from March 1st, 2016 to May 24th, 2016. However, the data of five days are hardly damaged, so we discard the logs in those days. There are about 1.9 billions logs in 80 days. Machine we use has 8 Intel Cores i7-3770, 16GB memories and the operating system is Ubuntu 14.04.3. We use Python 2.7.6 to implement our system.

TABLE I
Dataset Information

Month	discarded dates	days	traffic	average traffic per day
March	13, 23, 26	28	686,702,891	24,525,103
April	7	29	615,301,006	21,217,276
May	1, 25-31	23	587,593,967	25,547,563
Total	-	80	1,889,597,864	23,619,973

In the following experiments, we start the prediction from March 8th. The first seven days are preserved to be initial training data. We will discuss how to select those parameters in our method in Section IV-C, and the experimental result is given in Section IV-D.

B. Evaluation metrics

We use root mean squared error (RMSE) and mean absolute percentage error (MAPE) as our evaluation metrics.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i^{predict} - Y_i)^2}{n}} \tag{3}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{Y_i^{predict} - Y_i}{Y_i}| \tag{4}$$

$Y_i^{predict}$ and Y_i are the prediction value and real value in the same time slot respectively. RMSE has larger penalties on larger error. Comparing to traditional mean absolute error, RMSE will increase a lot if there exists large errors. MAPE shows the percentage of error, which makes us understand the error in a regular scale.

C. Parameters selection

1) Number of training days: We want to know how many days training data should be for the re-training step when a day starts. Here we slice one day into 24 time slots, and every slot is 1 hour long. The prediction starts from March 8th and the result is shown in Fig. 8. No matter RMSE or MAPE, when the training days increased from 3 to 4, the performance improved the most. The improvement slows down after training days come to 4. The reason is that when the model is lack of training days, it's hard for $SlotNumber$ to learn good coefficients.

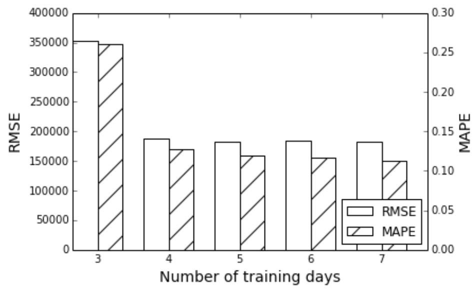

Fig. 8. Training days

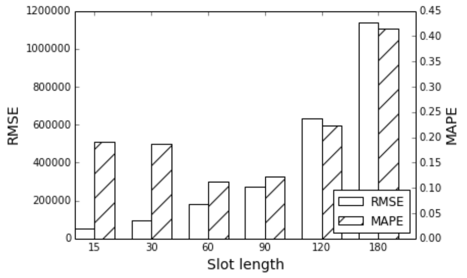

Fig. 9. Slot length

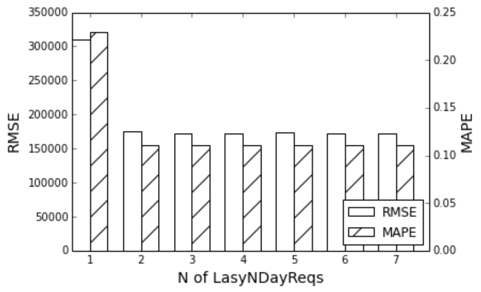

Fig. 10. N of $LastNDayReqs$

2) Time slot length and N of LastNDayReqs: Length of time slots should show the behaviour of humans' life. If the length is too short, the prediction would be too sensitive. Otherwise, every prediction error would be too large. The result is shown in Fig. 9. MAPE shows that the best length is 60 minutes for a time slot. However, RMSE is not adequate here. The reason is that the length of time is not equal, and the scale of traffic is not the same, neither.

N of $LastNDayReqs$ represents how many days we use to interpret long-term effect. The result is shown in Fig. 10. We found that $N = 2$ is enough. The reason is that geometric mean does not need too many days to converge.

3) Days and tolerance of detecting anomalies: We use Gaussian distribution of traffic of the same slot in previous k days to detect anomalies. If the traffic is tol standard deviations farther away from the mean, we use geometric mean to smooth the anomaly with previous traffic.

The result is shown in Fig. 11. Different bars in the histogram represent different values of k. Although the MAPE is lower as k goes larger, the effect is still not significant

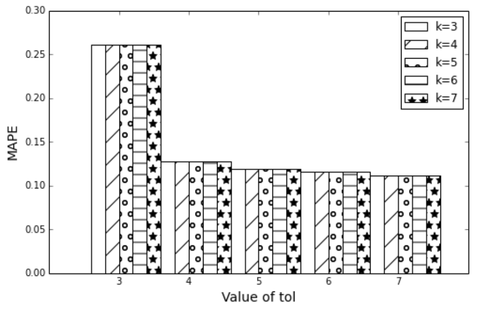

Fig. 11. Parameters of anomaly detection

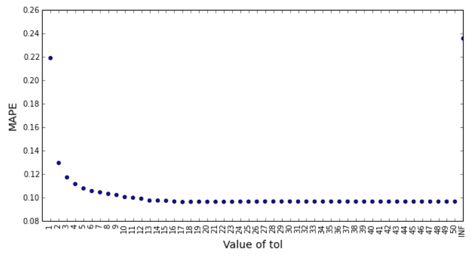

Fig. 12. Effect of tol

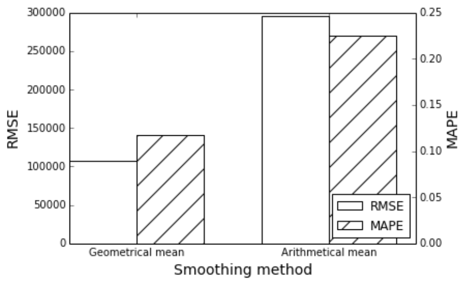

Fig. 13. Smoothing methods

enough. However, when tol increases from 1 to 1.5, there is an obvious improvement of MAPE. From Fig. 12, we can see that when tol comes to 17, there exists the minimum of MAPE. When the tol comes to infinite, which means we don't smooth anything, the MAPE is the worst.

4) Comparison of smoothing method: We compare two different smoothing methods here: arithmetic and geometric means. The result is shown in Fig. 13. Geometric mean outperforms arithmetic mean in both evaluation metrics. The reason is that arithmetic mean could be influenced a lot by anomalies.

D. Experiments result

1) Regression selection: Different regression models have their advantages on specific data. We discuss the selection of regressions in this section. There are three candidates: linear regression, Ridge regression [16] and Lasso regression [17]. Ridge and Lasso are linear regression's relatives, which add norm-2 and norm-1 regularization term after linear regression's objective function respectively. The regularization terms make them prevent from overfitting outliers better than linear regression. Lasso regression has the ability of feature selection due to the norm-1 penalty. Linear and Ridge regressions have closed-form solution so that they can be computed faster than Lasso does. The result is shown in Fig. 14. RMSE shows that Ridge is the worst and linear is the best. However, MAPE shows that Ridge is the best and linear is the worst. The reason is that the regularization term makes Ridge hard to change modification in very few outliers. Nevertheless, Ridge still outperforms linear when the traffic is stable.

2) Predict with or without SlotNumber: In this section, we demonstrate the importance of the feature $SlotNumber$. The result is shown in Fig. 15. RMSEs of predicting with and without $SlotNumber$ are 105,960.86 and 320,458 respectively. MAPEs of predicting with and without $SlotNumber$ are 0.112 and 0.266. Both of the evaluation metrics indicate that there is at least 2 times of improvement after adding $SlotNumber$ as a feature.

3) Other temporal features: We discuss other possible temporal features in this section. We preserve all the data in March to be training data and predict from April 1st to the end of the dataset. The result is shown in Fig. 16. "Original" is the model we discussed above. The second model, named "Feature", adds a feature indicating that every data point is weekday or weekend. The third model, named "MixModel", combines two original models together. One of them trains and predicts data from weekends, and the other trains and predicts data from weekdays. It can be represented by the following

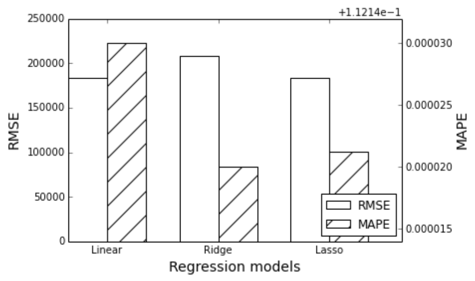

Fig. 14. Regression selection

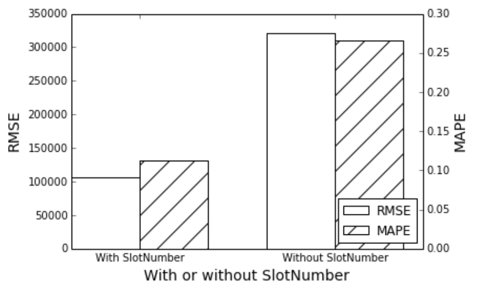

Fig. 15. Prediction with SlotNumber or not

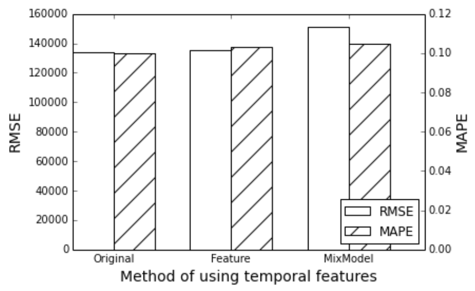

Fig. 16. Method of using temporal features

equation:

$$h_{mixed}(\cdot; w_1, w_2, b_1, b_2) = I_{weekend} \times h_{weekend}(\cdot; w_1, b_1) + (1 - I_{weekend}) \times h_{weekday}(\cdot; w_2, b_2) \quad (5)$$

$I_{weekend}$ is an indicator. If it's weekend data, $I_{weekend} = 1$. Otherwise, $I_{weekend} = 0$.

No matter model "Feature" or "MixModel", RMSE and MAPE both increase a little. The reason for not improving the prediction is that the training data for weekends is too few to converge.

V. Conclusion and Future Works

In this paper, we proposed a method to predict traffic from perspective of DSPs. The features we chose are more convenient for DSPs to fetch, and also boost the speed of our method, which made our method practical for real-time systems. In experiments, we found that one-hour time slots can interpret humans' behaviour. When the training days increase to 4, MAPE will come to about 11%. Using geometric mean to smooth anomalies makes the MAPE drop from 23% to 11%. When detecting anomalies, the most important parameter is how the anomalies are far away from the mean. The result showed that when the distance increases from 1 to 1.5 standard deviations, MAPE decreases from 20% to 13%. Every regression model had their advantages on predicting traffic. However, Ridge has the best MAPE and a closed-form solution. These advantages made Ridge a good choice.

In the future, we expect to have more temporal features if we could have sufficient data. Besides, we expect to model the anomalies to make the prediction more precise when anomalies occur.

References

[1] Digital Marketing Association, "Taiwan digital advertising statistics report," http://www.dma.org.tw/trend/, 2016.

[2] K.-c. Lee, B. Orten, A. Dasdan, and W. Li, "Estimating conversion rate in display advertising from past erformance data," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 768–776.

[3] S. Kalyanakrishnan, D. Singh, and R. Kant, "On building decision trees from large-scale data in applications of on-line advertising," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 669–678.

[4] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin *et al.*, "Ad click prediction: a view from the trenches," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1222–1230.

[5] W. C.-H. Wu, M.-Y. Yeh, and M.-S. Chen, "Predicting winning price in real time bidding with censored data," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1305–1314.

[6] W. Zhang, S. Yuan, and J. Wang, "Optimal real-time bidding for display advertising," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 1077–1086.

[7] D. Agarwal, S. Ghosh, K. Wei, and S. You, "Budget pacing for targeted online advertisements at linkedin," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1613–1619.

[8] J. Xu, K.-c. Lee, W. Li, H. Qi, and Q. Lu, "Smart pacing for effective online ad campaign optimization," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 2217–2226.

[9] D. Agarwal, D. Chen, L.-j. Lin, J. Shanmugasundaram, and E. Vee, "Forecasting high-dimensional data," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 1003–1012.

[10] S. Cetintas, D. Chen, and L. Si, "Forecasting user visits for online display advertising," *Information retrieval*, vol. 16, no. 3, pp. 369–390, 2013.

[11] C. Chen and L.-M. Liu, "Forecasting time series with outliers," *Journal of Forecasting*, vol. 12, no. 1, pp. 13–35, 1993.

[12] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[13] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[14] F. J. Trívez, "Level shifts, temporary changes and forecasting," *Journal of Forecasting*, vol. 14, no. 6, pp. 543–550, 1995.

[15] S. C. Hillmer and G. C. Tiao, "An arima-model-based approach to seasonal adjustment," *Journal of the American Statistical Association*, vol. 77, no. 377, pp. 63–70, 1982.

[16] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, 2000.

[17] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.