COMPGW02 Web Economics Individual Report (Group 05)

Min Ying Ong
University College London
School of Computer Science
min.ong.16@ucl.ac.uk

1. INTRODUCTION

Real-time bidding (RTB) for online display advertising has revolutionised the advertising landscape. The process of a user loading a webpage, to the advertiser bidding to display their ads on the page now takes place within a fraction of a second. Much focus is now placed on computational approaches to maximise utility of the advertiser with an automated bidding strategy that is able the optimise the strategy for the advertisers goals.

This report covers exploratory data analysis of the dataset provided, followed by a proposed a bidding strategy consisting of two components, a click prediction model based on convolutional neural network (CNN) followed by a variation on linear bid price estimation strategy to allocate a bid for each impression. The strategy was implemented and evaluated against other bidding strategies.

The CNN click prediction model significantly outperforms the baseline logistic regression model, however the variation on linear bid price estimation did not outperform its baseline linear bid model. The source code is available at [5].

2. LITERATURE REVIEW

The following works [9] [11] provides an overview of the RTB landscape and iPinYou dataset. In particular, literature review of various deep neural network implementations for click prediction such as [2] [8] and [10] highlights the advantage and disadvantages of various neural network architectures to predict users' ad click response.

3. APPROACH AND RESULTS

3.1 Data Exploration

3.1.1 Data Format

The format of the data provided closely follows the format of the iPinYou dataset from a three-season global competition of RTB algorithms [11]. Each row represents a record of a bid for an advertisement by the DSP/advertiser. Each record contains information about the ad slot features, auction features, user features and bid prices (for validation and training sets).

Some of the differences in this dataset compared to the iPinYou dataset are highlighted here:

- ***click***: *logtype*=2 (click) information was moved to this separate *click* column.
- ***logtype*:** this column only contained entries of *logtype*=1 which corresponds to impressions. Conversion information (*logtype*=3) is not available and typical evaluation metrics for conversion cannot be used in this case.
- ***slotvisibility*:** contained a mix of text (e.g 'FirstView', 'SecondView') and numeric data (e.g 0, 1, 2, 255). The values are treated distinctly as no mapping between the text and numeric data was available.
- ***slotformat*:** was represented in numeric format (e.g 0, 1, 5) rather than text (e.g 'fixed','pop').

All monetary related numbers (e.g. columns *bidprice* and *payprice*) are in the currency of RMB and the unit of Chinese fen x 1000 i.e the cost-per-mille (CPM) pricing model.

Many columns also contained null values represented as 'Na' or 'null' when information is not available to the DSP.

3.1.2 Basic Statistical Analysis of Datasets

For the analysis, the following definitions are used.
Cost is in unit and currency of RMB.
Click-Through-Rate (CTR) is calculated as:

$$CTR = \frac{Total\ number\ of\ clicks}{Total\ number\ of\ impressions}$$

Average cost-per-mille (CPM) is calculated as:

$$AvgCPM = \frac{Cost \times 1000}{Total\ number\ of\ impressions}$$

Effective cost-per-click (eCPC) is calculated as:

$$eCPC = \frac{Cost}{Total\ number\ of\ clicks}$$

3.1.2.1 Overall analysis of datasets.

The datasets train, validation (val) and test provided only contain records of logtype=1 (impressions), i.e only contains records of ad auctions that were won. As such we can calculate some basic statistics shown in table 1 for the overall datasets provided.

Table 1: Overall statistics for the datasets provided

Data set	Num Imps	Num Clicks	Cost	CTR	Avg CPM	eCPC
Train	2,697,738	2,034	216,496	7.5e-04	80.3	106.4
Val	299,749	226	24,045	7.5e-04	80.2	106.4
Test	299,749	n/a	n/a	n/a	n/a	n/a

Table 1 shows that the training and validation sets provided are similarly distributed as the CTR, average CPM and eCPC are all similar despite the training set having 9 times more data than the validation set.

In terms of the number of clicks binary class, the ratio of click=0 to click=1 is 1326:1. This is a severely imbalanced dataset from that perspective and can impact the ability of machine learning algorithms to perform well. As such, basic resampling of the data set was carried out to change the ratio of click=0 to click=1 to be 4:1 by randomly undersampling the click=0 class, but still maintaining a large enough training set with approximately 10,000 impressions. Further methods used to tackle the class imbalance in the machine learning algorithm is detailed in section 3.2.1.2.

3.1.2.2 *Analysis of advertisers.*

There are a total of 9 unique advertisers in this dataset based on the advertiser ID in column *advertiser*. Carrying out similar statistical analysis on the training dataset yields the following numbers in table 2.

Table 2: Statistics per advertiser for the training set provided

ID	Num Imps	Num Clicks	Cost	CTR	Avg CPM	eCPC
1458	540,293	451	37,231	8.3e-04	68.9	82.6
2259	146,778	45	13,649	3.1e-04	93.0	303.3
2261	120,619	37	10,789	3.1e-04	89.4	291.6
2821	231,416	144	20,626	6.2e-04	89.1	143.2
2997	54,487	251	3,413	4.6e-03	62.6	13.6
3358	304,782	233	28,145	7.6e-04	92.3	120.8
3386	498,554	358	38,341	7.2e-04	76.9	107.1
3427	454,031	340	36,820	7.5e-04	81.1	108.3
3476	346,778	175	27,481	5.0e-04	79.2	157.0

From table 2, 8 of the 9 advertisers have CTR of less than 0.001 (0.10%). Advertiser 2997 stands out as having CTR of 0.0046 (0.46%). The features that influence the CTR for specific advertisers are explored further in section 3.1.3.

The average CPM for the 9 advertisers are fairly similar in the range of 60-95. Their eCPC however is as low as 13.6 for advertiser 2997 and as high as 303.3 for advertiser 2259. The features that influence the eCPC for specific advertisers are explored further in section 3.1.5.

3.1.3 *User Feedback*

In this section we examine the statistics of user feedback, by comparing the mean value of CTR (with 95% confidence interval) against some features

For clarity of presentation, we will look at just 3 advertisers in detail. Advertiser 2997 and 2259 because they stood out in section 3.1.2, and we also select an additional advertiser 3358.

From figure 1, we observe that in general the features impact CTR for each advertiser differently. For example:

- We observe that advertiser 2997 receives high CTR of 0.009 on Tuesdays, whereas advertiser 3358 has consistent CTR across the week.
- Advertiser 3358 has a noticable peak in CTR at 9PM. The large error bars for advertiser 2997 and 3358 on the peaks suggest that combining weekday and hour information might yield more specific day and hour combinations that exhibit high CTR.
- Windows as a user platform (OS) appears to have the poorest CTR for all advertisers.
- The slot height, slot width and user agent information suggests that advertiser 2997 only chooses to bid for specific slot size of 50x320 and predominantly obtains good CTR on Android platform.
- For advertiser 3358, slot visibility of 0 and 1 achieves noticeably better CTR, suggesting placement of ad is particularly important for advertiser 3358.

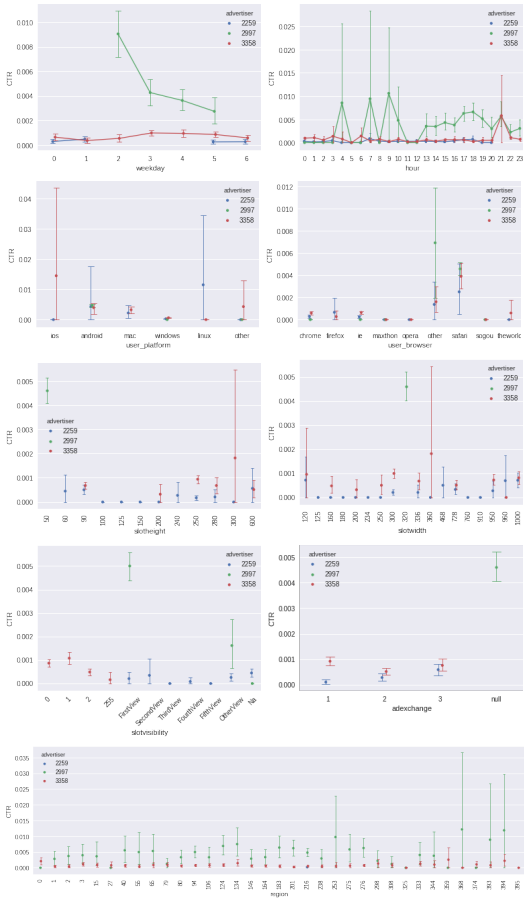

Figure 1: CTR distribution against features for advertisers 2259, 2997, 3358

From these observations, it suggests that advertiser ID should be included as a feature in modelling CTR prediction, as different advertisers exhibit different CTR distributions against the features explored.

3.1.4 *Bidding Behaviour*

In this section we observe patterns in the payprice for advertisers 2259, 2997 and 3358. The payprice mean and confidence interval of 95% is shown in figure 2.

From figure 2, we observe that in general the bidding strategy for each advertiser also differs. For example:

- We observe that advertiser 2997 has a noticeable lower payprice than advertiser 2259 and 3358. The lower payprice combined with the high CTR led to the excellent eCPC observed in table 2.
- We also note that advertisers 2997 and 2259 do not appear to bid everyday, instead they appear to have a strategy of focusing on Tuesday to Friday for advertiser 2997 and Friday to Monday for advertiser 2259, which

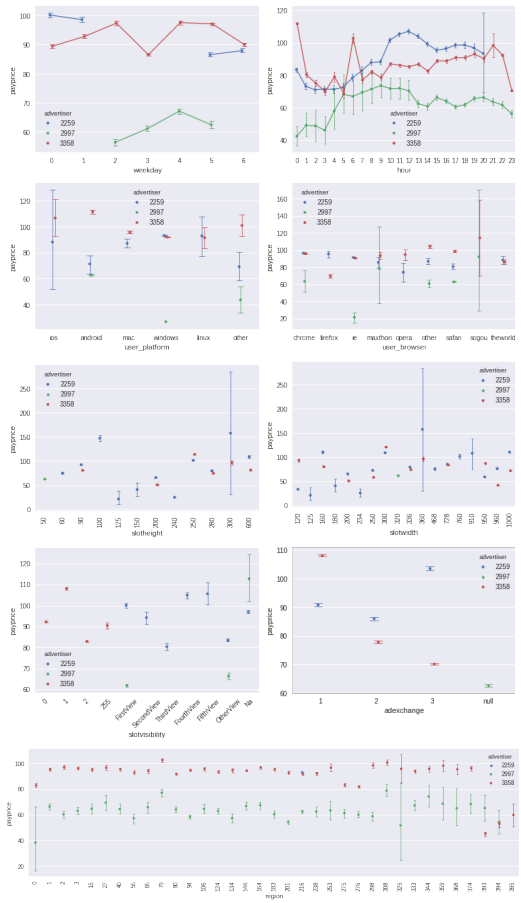

Figure 2: Payprice distribution against features for advertisers 2259, 2997, 3358

could be a strategy developed from previous analysis of RTB data.

- There are also noticeable variations in the competitiveness for the impressions relative to time of day and by advertiser. In general payprice is lower for all advertisers between 0-4 hours, which may reflect a lower expected CTR at those times.
- Payprice is observed to be heavily influenced by slot visibility as well, which is logical as that determines where on the page the ad is placed, and therefore more likely to receive focused attention from the user.
- Ad exchanges also influences the payprice heavily as different publishers connect to different exchanges, therefore some ad exchanges are noticably more competitive in price.
- We also observe that advertiser 2259 only appears to bid for impressions in region 238, which suggests a localised strategy.

In general the fluctuations in payprice relative to the features is less compared to clicks as the click observations as payprice is represented as integers whereas clicks are binary.

3.1.5 eCPC

Jointly observing the CTR from figure 1 and payprice from figure 2, allows us to estimate eCPC, which gives us an insight into return-on-investment (ROI) across different features. A lower eCPC means less money is spent to achieve one click, which implies a more cost effective bidding strategy. For example:

- Advertiser 2997 has two times higher CTR on Tuesday with the lowest payprice compared to Wednesday to Friday. Focusing on bidding only on Tuesdays for advertiser 2997 could be beneficial to maximising ROI.
- Advertiser 2259 have lowest payprices between 0-8 hours of the day, with no discernible differences in CTR. Bidding only in that window could be a way to maximise ROI for advertiser 2259.
- The ROI for impressions on windows platform is poor for advertisers 2259 and 3358, with similar payprice for significantly lower CTR.
- For advertiser 2259, high payprice for FourthView and FifthView and low CTR for those slot visibilities leads to poor ROI.

In summary, both user feedback and bidding behaviour differs across advertisers. Independent models for CTR estimation and bid price strategy for each advertiser could prove be beneficial, however may not be scalable depending on the number of advertisers and the amount of training data available.

3.1.6 Data Consistency

The datasets were also verified to ensure data cleanliness and consistency. The following irregularities were found and addressed as follows:

- It was observed that approximately 1.2% of rows in the training and validation sets contained rows where payprice is higher than bidprice. As the dataset should only have contained impressions won i.e bidprice higher than or equal to payprice, these entries were pruned from the dataset used for training the bidding strategy.
- It was also observed that approximately 0.5% of rows in the training and validation sets contained rows where payprice is lower than slotprice. This should not be a valid situation as the payprice should be at minimum the reserve price (slotprice) for the impression.
- There was also approximately 0.02% of entries in the training and validation sets where payprice was 0. In a second price auction this would mean the ad would be placed for free. In the cases where we absolutely did not want to bid for the ad, we bid -1 value to avoid inadvertently winning these 0 payprice auctions.

3.2 Bidding Strategy

The bidding strategy model developed was split into two separate sub-models, one for utility estimation (click prediction model) and the other for cost estimation (bid price estimation model). The click prediction model developed is based on deep neural networks (DNN), and the bid price estimation model is a variant of linear bid price. A unified bidding strategy model based on DNN was also attempted but achieved very poor results on bid price estimation.

The evaluation of the models were carried out using metrics consistent across the group for comparing the models developed. This is discussed in further detail in the group report.

3.2.1 CNN Click Prediction Model

From dataset analysis in section 3.1, the complex interactions between the features and clicks would be suited to machine learning using neural networks for click prediction due to their ability to extract patterns from complex inputs.

Per section 2, Recurrent Neural Network (RNN) models would have the advantage if user historical click sequence is known, however this is not the case in our dataset. Convolutional Neural Network (CNN) models on the other hand has convolutional and pooling layers that can extract local-global key features from sequential ad impressions effectively.

This implementation is mostly based on the Convolution Click Prediction Model [8], however with a fixed sized one-hot encoded input and simpler convolutional model that is more akin to DNN with fully connected layers.

3.2.1.1 Input.

The majority of features in the dataset can be modelled as categorical data. Table 3 shows the unique categories for each of the potential input features based on the training set.

Table 3: Unique categories for each input feature

Feature	Num Uniq Categories	Included in Input
weekday	7	Yes
hour	24	Yes
bidid	2697738	No
logtype	1	No
userid	2591064	No
useragent	39	No*
userplatform	6	Yes*
userbrowser	9	Yes*
IP	515530	No†
IP block	124	Yes†
region	35	Yes
city	370	Yes
adexchange	5	Yes
domain	24087	Yes‡
url	833453	No
urlid	1	No
slotid	55983	No
slotwidth	21	Yes
slotheight	14	Yes
slotvisibility	11	Yes
slotformat	4	Yes
slotprice	284	Yes
creative	130	Yes
keypage	19	Yes
advertiser	9	Yes
usertag	69	Yes

The features labelled as included in input in table 3 were converted to one-hot encoded columns as input to the CNN using the rebalanced training dataset described in section 3.1.2.1. In general, columns that had more than 500 categories were discarded, and others with high number of categories were further processed as highlighted below.

* Useragent data was split into user platform (OS) and user browser columns and only the latter two included.

† IP data (in decimal IPv4 format) was split and only the first 8 bits were kept and included as IP block with 124 classes.

‡ The domain data was found to be heavily skewed, with 95% of domains having less than 15 occurrences. To speed up training and reduce memory requirements, the low frequency domains were grouped together into a new domain class. A total of 81 unique domains remained in the input data after this processing. This had negligible impact on the prediction performance while improving training speed.

A total of 907 one-hot encoded columns were used in the final training set. The validation and test sets were also converted to one-hot encoding using the 907 one-hot encoded columns from the training set. Any new classes found in the test and validation sets were discarded.

This approach to encoding the input data had benefits over using input embeddings in that the input size was known and fixed, and had a small memory footprint due to rebalancing of skewed classes like domain data. However this approach would be less flexible than using input embeddings described in [8] for adding new features or data with missing features.

3.2.1.2 Model.

The best performing architecture for the DNN model is shown in figure 3.

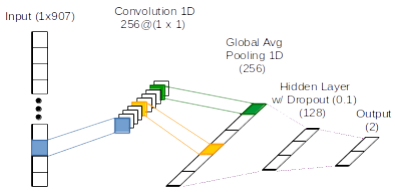

Figure 3: CNN model architecture

The 1x1 convolution kernel layer used to reduce the dimensionality of the one-hot encoded input from a 907 dimension vector, along with a non-linear ReLU activation layer to learn non-linear reductions of the features. The following global average pooling layer was used to squeeze the dimensions of the convolution layer to a 256 dimension vector, before being passed to a fully connected hidden layer with further reduced dimensions (128), again with non-linear ReLU activation, and also dropout of 0.1 to prevent overfitting to training data. This is finally passed into another fully connected layer with softmax activation to predict probablity of click=0 or click=1.

The model was trained with a class weight for scaling loss function as even after the resampling detailed in section 3.1.2.1, the ratio of training data for click=0 and click=1 was 4:1. Adam optimiser [6] was used with a learning rate of 0.0001. The model was implemented using Keras 1.22 [3] and Tensorflow 1.0 [1].

An alternate model with a fully connected layer replacing the 1x1 convolution kernel layer was also tested.

3.2.1.3 Tuning the model.

Initialisation of weights was found to be important in improving the performance of the model. The final model utilises lecun uniform initialisation in the convolutional layer [7], and glorot uniform initialisation in the other layers [4].

Dropout on the hidden layer was also tuned for regularisation to prevent overfitting. Early stopping was employed to

stop the training if validation accuracy no longer improved, ensuring optimum training and that overfitting did not occur. Input data was shuffled on every epoch as well. Three randomly resampled training datasets were also used to train three models and the output combined to reduce overfitting.

Different layer dimensions were also explored to arrive at the optimal layer dimensions used.

3.2.1.4 *Results.*

Table 4 shows the AUC score results for the CNN click prediction model and the alternate DNN click prediction model produced, compared to the baseline logistic regression model outlined in the group report.

Table 4: AUC score for Individual Click Prediction Models

CTR estimation model	AUC
Logistic Regression (baseline)	0.822
Convolutional Neural Network (main model)	0.868
Deep Neural Network (alternate model)	0.856

The results show that the CNN model comfortably outperforms the logistic regression baseline model by 4.6%. The DNN alternate model outperformed the baseline model by 3.4%. Both the CNN and DNN models exhibit similar patterns of prediction shown in figures 4 and 5 respectively.

From click=1 histogram (right green graph), it is observed that both models performed well in predicting click=1 as the majority of click=1 is predicted strongly at more than 0.8 probability, however the CNN model showed more consistent correct predictions than the DNN model. The click=0 (left red graph) decays as it approaches probability of 1, with the CNN model again predicting click=0 more strongly than the DNN model. This explains the AUC score difference between the two models.

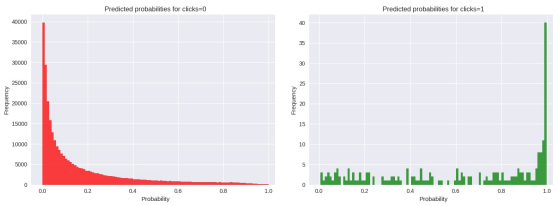

Figure 4: CNN histogram of click probabilities

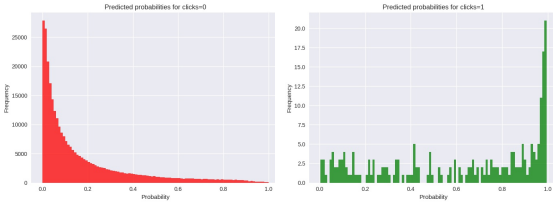

Figure 5: DNN histogram of click probabilities

3.2.2 *Linear Bid Price Variation Model*

The bidprice estimation model developed is based on the linear bid price model discussed in the group report. An alternate model using the CNN model to jointly predict the clicks and payprice was also attempted, but was unable to predict the payprice to a sufficient accuracy.

3.2.2.1 *Model.*

The Linear Bid Price Variation Model developed takes the following steps:

1. An initial bid price is obtained using the linear bid equation $bid = basebid * (pCTR/avgCTR)$ where $pCTR$ is the probability of a click as determined by the click prediction model, and the $avgCTR$ determined by the average CTR in the training set.
2. The initial bid is increased to meet the minimum slot price + 10%. This is to ensure minimum price is met.
3. A bid prune threshold determines which bids are removed and set to -1 if the predicted click probability from the click prediction model is below the threshold. This is to improve CTR by not bidding on any ad slots where we are confident will not get a click.

Base bid and bid prune threshold are the two parameters that are tuned for the model.

3.2.2.2 *Tuning the model.*

The model is tuned using gridsearch to determine the best values for the base bid and bid prune threshold parameters using the common blended metric score for the group as detailed in the group report.

Average CTR on the full training set was taken as 0.00075. The best values were found to be 0.25 CPM for the base bid, and 0.1 for the prediction threshold with the CNN model as the click prediction estimator.

3.2.2.3 *Results.*

The results of the CNN model and linear bid variation model developed was compared to the baseline linear bid model (i.e step 1 in section 3.2.2.1) on top of the CNN model with an optimum base bid of 0.325 CPM. Table 5 shows the evaluation metric scores for the two models.

Table 5: Bidding Strategies Performance Metrics

Model	Num Imps	Num Clicks	Cost	CTR	Avg CPM	eCPC	Blended Score
Baseline	111,304	177	6,245	1.6e-03	56	35	0.323
Variation	97,704	174	6,020	1.8e-03	62	35	0.317

The linear bid variation model was very similar in performance to the baseline linear bid model. Whilst it performed better on CTR, it did not receive any additional clicks.

3.3 Conclusion

The CNN click prediction model significantly outperforms the baseline logistic regression model, however the variation on linear bid price estimation did not outperform its baseline linear bid model.

Future potential direction for this work includes having independent models for CTR estimation and bid price strategy for each advertiser (or clusters of advertisers with similar attributes). The CNN model could also benefit from input embedding for features with variable class sizes rather than one-hot encoding for flexibility.

Ensemble methods such as combining gradient boosted trees with CNN as explored in the group report could also prove beneficial to improving the click predictions.

3.3.1 Group roles

The group for this project consisted of Kah Siong Tan (KS), Chun Siong Poh (CS) and myself (Min). Collaboration on the code and report was carried out using Github and Sharelatex which provided efficient platform for concurrent work.

The team agreed early on that CS would focus on gradient boosted tree models, KS on factorisation machine models and Min on neural network models. With that as the main split of work, we collaborated on the common items like data conversion and resampling (CS, KS), feature extraction (All), baseline models (All), bidding strategies (All), evaluation metrics (All). The group report was also written collaboratively by all members.

Overall I think the team workload was distributed well and everyone contributed ideas, code and suggestions to achieve the best model we could in the timeframe.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[2] Qiao-Hong Chen, Shi-Min Yu, Zi-Xuan Guo, and Yu-Bo Jia. 2016. Estimating AdsâĂŹ Click through Rate with Recurrent Neural Network. In *ITM Web of Conferences*, Vol. 17. 1–5.

[3] François Chollet. 2015. Keras. https://github.com/fchollet/keras. (2015).

[4] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. 249–256. http://www.jmlr.org/proceedings/papers/v9/glorot10a.html

[5] Min Ying Ong Kah Siong Tan, Chun Siong Poh. 2017. Web Economics assignment Group 5. (Apr 2017). https://github.com/jax79sg/webecon2017

[6] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

[7] Y. LeCun, L. Bottou, G. Orr, and K. Muller. 1998. Efficient BackProp. In *Neural Networks: Tricks of the trade*, G. Orr and Muller K. (Eds.). Springer.

[8] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A Convolutional Click Prediction Model. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. ACM, New York, NY, USA, 1743–1746. DOI: http://dx.doi.org/10.1145/2806416.2806603

[9] Jun Wang, Weinan Zhang, and Shuai Yuan. 2016. Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting. *CoRR* abs/1610.03013 (2016). http://arxiv.org/abs/1610.03013

[10] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *European Conference on Information Retrieval*. Springer, 45–57.

[11] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Real-Time Bidding Benchmarking with iPinYou Dataset. *CoRR* abs/1407.7073 (2014). http://arxiv.org/abs/1407.7073