Fast Click-Through Rate Estimation using Data Aggregates

Roman Wiatr[1], Renata G. Słota[1], and Jacek Kitowski[1,2]

[1] AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunication, Institute of Computer Science, Mickiewicza 30, 30-059 Krakow, Poland
[2] AGH University of Science and Technology, Academic Computer Centre CYFRONET AGH, Nawojki 11, 30-950 Krakow, Poland
rwiatr@gmail.com, {rena,kito}@agh.edu.pl

Abstract. Click-Through Rate estimation is a crucial prediction task in Real-Time Bidding environments prevalent in display advertising. The estimation provides information on how to trade user visits in various systems. Logistic Regression is a popular choice as the model for this task. Due to the amount, dimensionality and sparsity of data, it is challenging to train and evaluate the model. One of the techniques to reduce the training and evaluation cost is dimensionality reduction. In this work, we present Aggregate Encoding, a technique for dimensionality reduction using data aggregates. Our approach is to build aggregate-based estimators and use them as an ensemble of models weighted by logistic regression. The novelty of our work is the separation of feature values according to the value frequency, to better utilise regularization. For our experiments, we use the iPinYou data set, but this approach is universal and can be applied to other problems requiring dimensionality reduction of sparse categorical data.

Keywords: Real-Time Bidding, RTB · Click-Through Rate, CTR · dimensionality reduction · logistic regression

1 Introduction

Online advertising is a ubiquitous form of advertisement that uses the internet to display ads to the users. Y. Yuan et al. [9] define Real-Time Bidding (RTB) as a business model for automated online advertising with transaction time constraint between 10 and 100 milliseconds. There are three key players in the RTB setup: publishers - offering the internet traffic, generated by sites or applications, aggregated on Supply Side Platform (SSP); advertisers - running campaigns configured on a Demand Side Platform (DSP), offering ads to display on sites or applications provided by the SSPs; Ad Exchanges (AdEx) - platforms for facilitating the trade between multiple SSPs and DSPs. When a user visits a site with a display advertisement placement, the visit is being offered on AdEx,

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

where DSP can bid on behalf of the advertiser for that particular visit (Fig. 1). The winning advertiser displays the advertisement to the website visitor. On the DSP side, the advertiser configures one or more campaigns. Each campaign consists of preferred targets like age, country, operating system etc. and creatives i.e. images or videos.

Several event types can be tracked in the RTB environment. Each display of the advertisement generates an impression event. This event tells nothing about the user real interest in the advertisement. The next event is a click event where the user clicks on the displayed advertisement. The relation of clicks to impressions is called Click Through Rate (CTR). As reported in [1] CTR can be lower than 1% making the data very unbalanced. The final event is a conversion event. It is generated when the user takes further actions (like filling out a form or purchasing a product). The relation of conversions to clicks is called Conversion Rate (CVR). Typically conversions are very rare and can occur hours after the click.

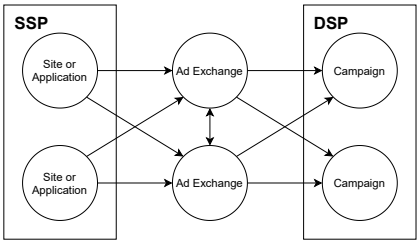

Fig. 1. RTB environment

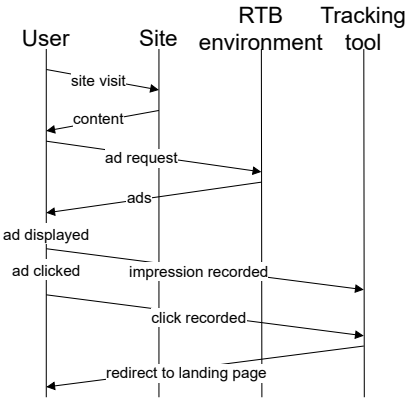

Fig. 2. Tracking events in RTB environment.

RTB environments are highly automated. Because of the number of transactions and time constraints, each transaction has to be made without human interaction. This forces every entity in the chain to monitor the state of the traded traffic. Fig. 2 shows a typical event chain for sites and how a tracking tool may be used by an advertiser to monitor the performance of the displayed ads. When a user visits a site that has a connection to an RTB environment (in this case SSP), a script is loaded that makes an ad request to the RTB environment. An offer is made on an AdEx and the winners are enabled to present their advertisements to the user. On advertisement load time an impression event is generated, indicating an advertisement was presented to the user. A subsequent click and conversion event can be generated by the user actions. All of these events are tracked using a tracking tool. A similar tool may be used on SSP, AdEx and DSP sides depending

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

on the business model. The traffic buyer is charged by click or by impression. Since the amount of ingested data is huge [1] it is crucial to store the data as an aggregated time series [8] for manual and automated decision making. Raw data are also stored for audit purposes and model building but tend to have a shorter lifespan due to space requirements.

Each time an ad request is done to the RTB environment, an AdEx makes an offer to multiple DSPs (or other AdEx), meaning a single site visit may trigger several DSP queries. Each time a DSP gets an offer, it has to evaluate the CTR/CVR model to decide on how to bid. The DSP does so by evaluating hand written rules as well as various models (fig. 3) in real time. The amount of bid requests combined with the volatility of traffic and campaigns [1] means the model has to be efficient in both training and evaluation.

In this work, we show how to exploit data aggregates to reduce resources and time required to train and execute the model. We evaluate the model performance on well-known iPinYou data set in a CTR prediction task. The bidding process is complicated and is beyond the scope of this paper. The process can be optimized by several components as stated in [2].

Section 2 presents the state of the art on which this work is based on. Section 3 describes our approach and proposed improvements. In Section 4 we compare three types of encodings: Dummy Encoding, Hashing Trick as the baseline and Aggregate Encoding as the proposed improvement. Section 5 describes the experimental setup using iPinYou data set and contains the results of experiments comparing Hashing Trick and Aggregate Encoding. In Section 6 we present the conclusions and proposals for future work.

2 State of the Art

A detail statistical analysis of the iPinYou data set as well as a benchmark for predicting CTR are provided by the authors in [11]. The iPinYou data set is available at [10].

In [4] authors present logistic regression as being widely referred to as the state-of-the-art and the most widely used algorithm for CTR prediction. They compare Area under the Receiver Operating Characteristic curve (AuROC) of several methods previously presented in the literature, including [11] on the iPinYou data set. Furthermore, the authors present the impact of certain features, feature generalization and training process parameters. The authors show the high importance of feature generalization and conclude that L2-regularized logistic regression with cross-entropy loss function is the state-of-the-art in RTB CTR estimation. The authors do not consider dimensionality reduction for sparse data that can be useful when feature conjunctions are introduced, leading to an exponential growth of feature vector size.

A scalable and easy to implement framework based on logistic regression is proposed in [1]. The model inputs are categorical variables and categorical variable conjunctions. The dimensionality problem is addressed by the use of hashing trick [6]. Authors present arguments for training one model for multiple

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

campaigns (multi-task learning) instead of separate models for each campaign. Due to class imbalance (positive class is lower than 1%) and huge amounts of data (9 billion events daily), negative class sampling is proposed to reduce the computational cost. Multiple dimensionality reduction techniques are considered in this work and Hashing Trick is considered to be the best one. The authors present practical implications of selecting such a model and propose a feature selection algorithm based on conditional mutual information. The algorithm allows assigning a score to features without retraining the model with each feature. The algorithm is used to select the best conjunction feature and only then the model is retrained. The authors show how fast the model gets outdated and they explain the degradation with the influx of new advertisements. Finally, the authors present an algorithm for large scale model training. This work is a comprehensive guide to build a CTR prediction system and as such it does not attempt to focus on novel techniques for feature encoding.

In [3] the authors focus on predicting Conversion Rate (CVR), a problem similar to CTR prediction with the difference that conversions are much rarer than clicks. The authors model three separate data hierarchies for user, advertiser and publisher. They build a conversion probability estimator for each of these hierarchies and combine the output of these estimators using logistic regression. They reduce the input of the logistic regression to three dimensions which are the success probability based on these estimators. This technique addresses data sparsity but does not address the problem of overfitting due to modelling rare and frequent features together.

An approach focusing on feature engineering is proposed in [5]. The authors present a novel Deep & Cross Network. The architecture consists of two modules: a deep neural network modelling feature interact and a novel cross-network that is better suited to learn cross product of features. This model achieves the state of the art performance and reduction in parameters compared to a deep neural network. Due to input sparsity, a learnable embedding layer is used and no other dimensionality reduction techniques are considered.

3 Research Goals and Approach

Our research focus on CTR prediction, which is a part of a DSP pipeline [2] described by Fig. 3. In our previous work [8] we point out that the marketer has access to aggregated impressions, clicks and conversions via a time series interface. In this work, we show how to reuse the aggregated data stored in a fast database to speed up the training process while preserving the quality of the produced models.

Our approach is to build aggregate-based estimators and use them as an ensemble of models weighted by logistic regression. The novelty of our approach is the separation of feature values of each estimator, to better utilise regularization.

We use the iPinYou data set [10] to conduct our experiments. All features are transformed into categorical variables. In our approach first, we create simple models based on aggregated features and then we use these aggregates to train

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

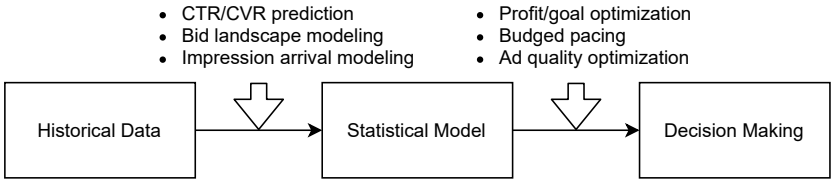

Fig. 3. DSP data pipeline [2]

a logistic regression model. We compare the baseline to our model in terms of AuROC and parameters used by the model. The code used in the experiments can be accessed via GitHub (see [7]).

4 Modelling Overview

One of the uses of raw data is to train models for CTR (or CVR) prediction. In CTR each impression ends in one of two events: an advertisement can be either clicked or not clicked making it a binary classification problem. The purpose of the model is to support the decision making process by providing it with the probability estimate that a particular user will click the advertisement (see Fig. 3).

The iPinYou data set is divided into test and train sets. The train set contains over 15 million impressions and over 11 thousand clicks. The test set contains over 4 million impressions and over 3 thousand clicks. The data is divided amongst nine advertisers from different industry sectors. For each advertiser, the test data contains events from a later period than the training data. In the iPinYou data set the majority of features are categorical. Meaning that each feature can take one of a predefined set of values. In our model we use the following features: 'weekday', 'hour', 'useragent', 'IP', 'region', 'city', 'adexchange', 'domain', 'url', 'urlid', 'slotid', 'slotwidth', 'slotheight', 'slotvisibility', 'slotformat', 'slotprice', 'creative', 'keypage', 'advertiser'. Feature 'slotprice' is first transformed into a categorical variable and then transformed accordingly.

A simple and efficient way to train a model is described in [1]. The raw data can be encoded using Hashing Trick [6]. Using the encoded feature vectors a logistic regression model is trained. While the model can be improved a new conjunction feature is selected based on conditional mutual information and the model is retrained. We argue that the process can be improved by exploiting existing data aggregate to boost the prediction accuracy of the model as well as reduce the size of feature vectors thus reducing overall training time.

4.1 Feature Encoding

The data set consists of N entries. Each entry has the form of $E = [F_1, F_2, ...]$ where F_k is a distinct categorical feature that can take value f_i where $i \in \{1, 2, ...\}$. Two features F_i and F_j might have different cardinality.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

Dummy Encoding in statistics is a standard technique used in regression tasks. The feature F_k is encoded as a $|F_k|$-dimensional vector $\mathbf{k}$. When feature $F_k = f_i$ then it is encoded as $k_i = 1$, e.g. if $|F_k| = 4$ and $F_k = f_2$ then $\mathbf{k} = [0, 1, 0, 0]$. Every categorical feature F_k has to be encoded into its corresponding vector and resulting vectors are concatenated into single vector $\mathbf{x}$. Continuous variables should be first divided into bins and then each bin should be treated as category value. Multi-value categorical variables may be treated as standard categorical variables, with one exception: they may produce vector $\mathbf{k}$ with more than one f_i for which $k_i = 1$. Dummy Encoding produces sparse vectors for each feature. If there are $|E|$ features and k-th feature has $|F_k|$ possible values, then eq. 1 is the dimensionality of $\mathbf{x}$. The dimensionality, d_{dummy}, can get very large if there is a lot of features with high cardinality.

$$d_{dummy} = \sum_{i=1}^{|E|} |F_i| \tag{1}$$

Hashing Trick addresses the problem of high dimensionality produced by Dummy Encoding. In [6] the authors outline significant compression of vectors while avoiding costly matrix-vector multiplication amongst the advantages of Hashing Trick. In [1] the authors further state that Hashing Trick is straightforward to implement and effective in terms of resource consumption when used for CTR prediction. They compare the Hashing Trick performance to other methods of parameter reduction stating that the Hashing Trick is slightly better in terms of model performance while being much more effective for real-time computation. Instead of encoding f_i value of F_k as k_i Hashing Trick calculates a hash $h(f_i)$ (eq. 2).

$$h : f_i \rightarrow \{1, 2, ..., z_{max}\} \tag{2}$$

$$d_{hash} = z_{max} \tag{3}$$

If each feature F_k has its own hash space, hash function maps feature value f_i element $k_{h(f_i)}$. If all features share the same space, each feature value is mapped to $x_{h(f_i)}$. Consider $h(F_a = f_a) = 2$ then the encoded vector of the i-th example is $\mathbf{x}^{(i)} = [0, 1, 0, ...]$. In case a collision occurs $h(F_b = f_b) = h(F_a = f_a) = 2$ for $\mathbf{x}^{(i)}$ the second feature may be ignored meaning $\mathbf{x}^{(i)} = [0, 1, 0, ...]$ or increased by 1 meaning $\mathbf{x}^{(i)} = [0, 2, 0, ...]$.

A variation of this method exists. Instead of adding 1, value of $sgn(h_2(f_i))$ is added, where h_2 is an independent hash function. In this case when $h(F_a = f_a) = 2$ the encoded vector is $\mathbf{x}^{(i)} = [0, sgn(h_2(F_a = f_a)), 0, ...]$, and in case of a collision, when $h(F_b = f_b) = h(F_a = f_a) = 2$, the resulting vector is $\mathbf{x}^{(i)} = [0, sgn(h_2(F_a = f_a)) + sgn(h_2(F_b = f_b)), 0, ...]$.

There is a second type of collisions when feature values occupy the same index but for different events. Consider $h(f_i) = h(f_j) = 2$, f_i is a feature value of $\mathbf{x}^{(i)}$ and f_j is a feature value of $\mathbf{x}^{(j)}$. In this case $\mathbf{x}^{(i)} = \mathbf{x}^{(j)} = [0, 1, 0, ...]$ effectively meaning that two features will share regression parameters. As we show later, this can diminish the effect of regularization, and thus may cause overfitting.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

It is easy to control the maximum dimensionality of the produced vector by changing the hashing function limit z_{max} (eq. 2). This causes the dimensionality to be equal to eq. 3 when the features share the hash space. One of the arguments for Hashing Trick is that it is straight forward to implement and the usage of a hashing function indicates $\mathcal{O}(|E|)$ complexity as it does not require any additional data structures. This is true for offline systems however for online environments such as DSP it may be probed and abused by exploiting hash collisions. In this case, an additional data structure with training set feature values has to be used to prevent unknown values from entering the system, meaning an additional $\mathcal{O}(d_{dummy})$ memory complexity.

4.2 Aggregate Encoding

We propose Aggregate Encoding for dimensionality reduction. It is designed as a set of probability estimators as first suggested in [3]. We propose the use of a single feature estimator but with the possibility of extending to feature conjunctions [1]. The conditional probability of success given that $F_k = f_i$ is given by eq. 4, where $success_i$ is the amount of successes and $attempts_i$ is the amount of attempts. We encode feature F_k from all events in the training set as vector $\mathbf{k}$. Each element of $\mathbf{k}$ corresponds to the conditional probability (eq. 4) of that particular event. In our case, the probabilities are very low so we normalize $\mathbf{k}$ to the range [0..1]. Each feature is encoded on a different position of $\mathbf{x}$ making it a $|F|$-dimensional vector.

$$P(success|f_i) = \frac{success_i}{attempts_i} \tag{4}$$

First Level Aggregate Encoding preserves the dimensionality of the original data and, as we show later, it behaves better than Hashing Trick with similar dimensionality. This approach, however, causes problems as it contradicts regularization by grouping features with different counts in a single representation. To address this issue we introduce quantile bins as the Second Level of Aggregate Encoding. Each value f_i, of F_k is assigned to a single bin_q where $q \in \{1, 2, ..., Q\}$ based on the quantile that $attempts_k$ belongs to. This causes features with a similar amount of attempts to be grouped in a single bin, reducing the negative effect that the original method has on regularization. This method produces vector $\mathbf{x}$ with the dimensionality given by eq.5.

$$d_{bin} = \sum_{i=1}^{|E|} \mathbf{min}(|F_i|, Q) \quad given \quad \forall_{i,j} attempts_i \neq attempts_j \tag{5}$$

As in First Level Aggregates, vector $\mathbf{x}$ is normalized per bin across all features. Aggregate Encoding requires storing a mapping from a feature value f_i to a probability of success $P(success|f_i)$ and a mapping from f_i to a bin_q. Since f_i is a feature from the training set the additional complexity is the same as for Hashing Trick when holding training set feature values in memory.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

It is known that in logistic regression the log loss, $LL(\theta)$, is given by eq. 6 and partial derivative for θ_j by eq. 7 where θ are the parameters of the model and σ is the sigmoid function. The cost function $J(\theta)$ with L_2 regularization is given by eq. 8 and partial derivative for θ_j by eq. 9. By minimizing $J(\theta)$ one can find θ that is optimal under the regularization constraints.

$$LL(\theta) = \sum_{i=1}^{N} y^{(i)} log[\sigma(\theta^T \mathbf{x}^{(i)})] + (1 - y^{(i)}) log[1 - \sigma(\theta^T \mathbf{x}^{(i)})] \tag{6}$$

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} [y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] x_j^{(i)} \tag{7}$$

$$J(\theta) = \frac{\lambda}{N} \theta^T \theta - \frac{1}{N} LL(\theta) \tag{8}$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{\lambda}{N} \frac{\partial}{\partial \theta_j} \theta^T \theta - \frac{1}{N} \frac{\partial LL(\theta)}{\partial \theta_j} = \frac{\lambda}{N} \theta_j - \frac{1}{N} \sum_{i=1}^{N} [y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] x_j^{(i)} \tag{9}$$

Let $\mathbf{x}_k$ and $\mathbf{x}_q$ be created by splitting feature F_j with $Q = 2$. Let $x_q^{(i)} = 0$ for $i \in \{1, 2, ..., n\}$ and $x_k^{(j)} = 0$ for $j \in \{n+1, n+2, ..., N\}$. In other words we create two quantile bins bin_k encoded on $\mathbf{x}_k$, where all rows with feature values from bin_q are encoded as $x_k^{(j)} = 0$ and bin_q encoded on $\mathbf{x}_q$, where all rows with feature values from bin_k are encoded as $x_k^{(j)} = 0$ as shown in Fig. 4. Knowing that we split eq. 9 in to eq. 10 and eq. 11.

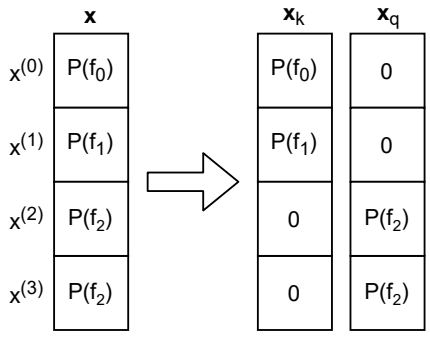

Fig. 4. Vector $\mathbf{x}$ encoding a single feature, as described by First Level Aggregate Encoding (subsection 4.2), is splitted into two vectors x_k and x_q, each containing features with similar occurrences. f_0 and f_1 occur once, f_2 occurs two times. $P(f_j)$ is $P(success|f_j)$ as defined in eq. 4. $x^{(j)}$ is the row index corresponding to the input data.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

$$\frac{\partial J(\theta)}{\partial \theta_k} = \frac{\lambda}{N}\theta_k - \frac{1}{N}\left[\sum_{i=1}^{n}[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})]x_k^{(i)} + \sum_{j=n+1}^{N}[y^{(j)} - \sigma(\theta^T \mathbf{x}^{(j)})] \times 0\right] \tag{10}$$

$$\frac{\partial J(\theta)}{\partial \theta_q} = \frac{\lambda}{N}\theta_q - \frac{1}{N}\left[\sum_{i=1}^{n}[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] \times 0 + \sum_{j=n+1}^{N}[y^{(j)} - \sigma(\theta^T \mathbf{x}^{(j)})]x_q^{(j)}\right] \tag{11}$$

We introduce $n_k = n$ and $n_q = N - n$ and we derive eq. 12 and eq. 13 by extracting the non-zero components as the gradient expectations when x is drawn from the train set where x_k or x_q is not zero, which are the mean gradient values at the current optimization step.

$$\frac{\partial J(\theta)}{\partial \theta_k} = \frac{\lambda}{N}\theta_k - \frac{n_k}{N}E_{X \sim train_k}[(y - \sigma(\theta^T \mathbf{x}))x_k] \tag{12}$$

$$\frac{\partial J(\theta)}{\partial \theta_q} = \frac{\lambda}{N}\theta_q - \frac{n_q}{N}E_{X \sim train_q}[(y - \sigma(\theta^T \mathbf{x}))x_q] \tag{13}$$

We can see that the regularization parameter θ is scaled by a constant that depends on the amount of data and the regularization parameter λ for both θ_k and θ_q. The gradients in both cases are scaled by different values that are corresponding to the element counts, respectively n_k for θ_k and n_q for θ_q. The amount of scaling depends on counts distribution of feature F_j. In our case $n_k < n_q$ meaning that the regularization for $\mathbf{x}_k$ has relatively higher impact than for $\mathbf{x}_q$. A similar effect will take place if Dummy Encoding is used and all features are separated.

If the features are joined randomly, as in Hashing Trick, the effect diminishes, meaning that rare features will be regularized together with frequent features.

The maximum dimensionality of a data set is given by eq. 1. When d_{hash} approaches the dimensionality of Dummy Encoding (eq. 3) the amount of conflicts is reduced not entirely to zero due to the nature of the hash function. When $d_{hash} < d_{dummy}$ the conflicts occur, including situations where frequent and rare feature values occupy the same vector positions and might impact regularization. For Aggregate Encoding, if no two features have the same count, d_{bin} given by eq. 5 is equal at most d_{dummy}. In this case, the methods are identical. If some feature values have the same count, they are encoded in the same bin_q as the probability of success given by eq. 4 and will most likely be distinguishable by the model. If $d_{bin} < d_{dummy}$ our method groups features with similar counts together, thus preserving the properties of regularization.

5 Experiments and Results

In our experiments, we compare Hashing Trick with Aggregate Encoding. As we argue at the end of Section 4 both of the methods degenerate to Dummy Encoding. Aggregate Encoding exploits the fact that the model uses regularization and Hashing Encoding due to the hashing function fails to do so. In this section we

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

show the impact of this behaviour using the iPinYou [10] data set. We do not use conjunctions of the original variables. We train a separate model for each advertiser. The feature 'usertag' representing a tag given to a user, is a special feature that in sense that a single user can be tagged with none, one or several tags. In our experiments we treat each tag value as a separate feature but we omit the feature in our comparison as it requires special treatment.

5.1 Experiment Setup

Our Hashing Trick implementation encodes all features to single feature space and resolves conflicts by using the sign of an additional hashing function as described in subsection 4.1. Our Aggregate Encoding implementation does not resolve conflicts when feature values f_i and f_j have $attempts_i = attempts_j$.

To build training and test sets we sample the negative class without repetition to reduce the amount of time and memory required by the experiments. Negative class sampling was set to 20% for advertisers *1458 and *3386 and 50% for others due to memory constraints. We sample the sets before every iteration to measure how the method behaves depending on the input distributions.

For Hashing Trick we iterate over maximum dimensionality z_{max} from a predefined list. For each z_{max} we sample the test and train data set and set $d_{hash} = z_{max} + z_{delta}$ where z_{delta} is a random. We modify z_{max} by a random number from the range $[-5, 5]$, as we have noticed that the result may vary depending on the selected dimensionality. Next, we encode the data sets using the method described above, train the model and evaluate it using AuROC. For each (advertiser, z_{max}) pair we repeat the experiment five times.

For Aggregate Encoding, we iterate over bin size Q from a predefined list. For each Q we sample the test and train data set. d_{bin} varies between experiments as it depends on the sampled data. As with Hashing Trick, we encode the data sets, train the model and evaluate it using AuROC. For each (advertiser, Q) pair we repeat the experiment also five times. We do not introduce conjunction features but it is possible to do so in both methods.

5.2 Results

Comparison of Hashing Trick and Aggregate Encoding is shown in Tab. 1. To create the table first a quantile bin size bin_q value was selected amongst the values with the highest average AuROC. Then a hashing dimensionality z_{max} was selected with AuROC close to the corresponding bin_q results. The only exceptions from this rule are advertisers **2259 and **2997 where both bin_q and z_{max} had to be lowered due to a relatively small amount of unique features. The mean gain for AuROC is close to zero as intended. Gain in terms of feature shows almost a two times increase for Aggregate Encoding compared to Hashing Trick. This means that using Aggregate Encoding we were able to significantly reduce the amount of features preserving AuROC results.

Fig. 5 shows AuROC change depending on the number of features produced by Hashing Trick and Aggregate Encoding. We fit an exponential curve to illustrate

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

Table 1. AuROC and no. of features comparison. Average AuROC gain [A-H] is the difference of average AuROC for Aggregates and Hashing methods. Average number of features gain [H/A] is the quotient of Average number of features for Hashing and Aggregates methods. The average number of features is selected the to minimize the mean average AuROC gain.

	Avg. AuROC (%)			Avg. no. of features		
Advertiser	Hashing	Aggregates	Gain [A-H]	Hashing	Aggregates	Gain [H/A]
*1458	63.75	64.35	0.60	499.00	242.60	2.06
3358	74.55	74.58	0.03	501.00	259.80	1.93
*3386	75.44	75.98	0.54	499.75	255.50	1.96
3427	71.10	70.84	-0.27	499.75	273.00	1.83
3476	62.64	63.08	0.43	498.25	261.00	1.91
**2259	66.75	67.17	0.42	198.40	209.33	0.95
2261	62.34	61.88	-0.46	500.40	275.80	1.81
2821	59.57	59.83	0.26	497.75	250.25	1.99
**2997	57.02	56.04	-0.98	400.60	207.33	1.93
Mean			0.06			1.82

the trend. AuROC for most advertisers behaves better when Aggregate Encoding is used. In most cases, the improvement is most evident for low feature count and diminishes with the feature increase. This behaviour is expected due to two facts. As shown earlier Aggregate Encoding is expected to exploit the benefits of regularization on low dimensional data compared to a Hashing Trick which is not optimized for this behaviour. With the increase of feature count, as shown in subsection 4.2, both of the methods are being reduced to Dummy Encoding with one difference. Error introduced by Aggregate Encoding may be easily reduced to zero opposed to the error introduced by Hashing Trick that is fully dependent on the hashing function used.

For each advertiser, we normalize the feature count and AuROC to the values of Aggregate Encoding as they tend to have less variance. Then we fit exponential curves to both normalized Hashing Trick and normalized Aggregate Encoding data. Fig. 6 shows the normalized dependency between AuROC and average no. of features count on combined data. As observed for most advertisers, the most significant gain of AuROC is for low dimensional data, and the difference slowly diminishes as we move towards the dimensionality used by Dummy Encoding.

6 Conclusions and Future Work

In this work, we present Aggregate Encoders - a method for dimensionality reduction with preservation of regularization properties of logistic regression. Using iPinYou data set, we empirically show that it behaves as good as the popular Hashing Trick, producing 42% smaller feature vectors (1.82 mean gain [H/A] in Tab. 1).

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

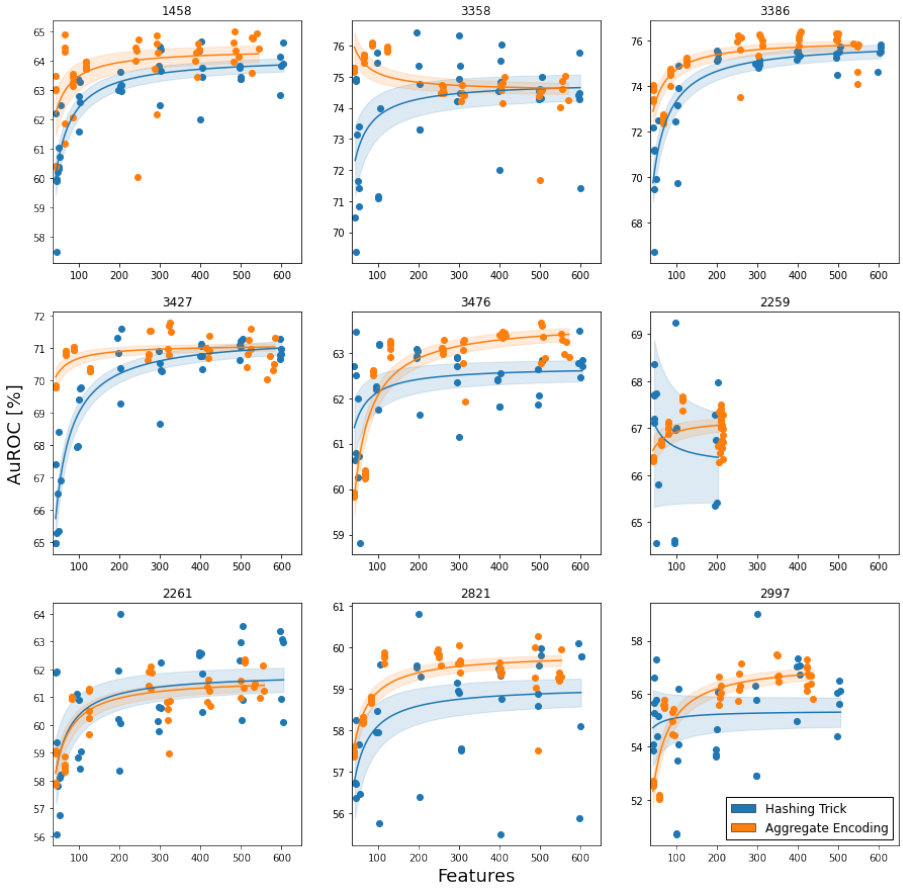

Fig. 5. Results for individual advertisers.

We use this method in a RTB setup but it is universal and can be applied to other problems. Small feature space is crucial for model training and model evaluation. During model training lower feature space reduces memory and time requirements, and during the evaluation, it reduces CPU consumption. Both of these properties are crucial in environments that are processing tens of billions of impressions [1] and potentially hundreds of times more offers daily. In this case, our method can reduce the overall training time and evaluation costs without sacrificing model performance. In [5] the authors present a deep neural network model with better log loss than logistic regression but with more parameters. Given the number of events, logistic regression can be used as the first stage of CTR assessment before a more expensive deep model is evaluated.

We leave three closely related problems for future work. All of them consider the challenge of exploiting cross-feature dependencies. The first one is to measure the effect of conjunction feature aggregates on the AuROC metric. We expect

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

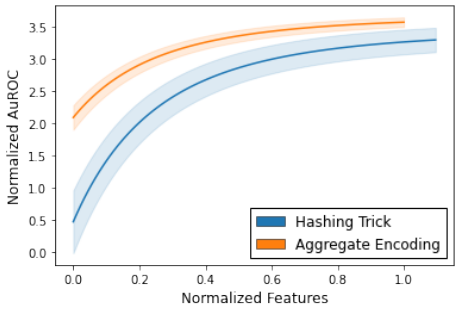

Fig. 6. Combined normalized results for all advertisers.

the improvement will be similar than as for Hashing Trick with the exception that Hashing Trick uses a preset vector size, and Aggregate Encoding increases the dimensionality of the vector with each added feature. The second one is using Aggregate Encoding in ensemble methods where each model instead of one value, might use our technique and return a vector divided by frequencies as this could be beneficial for regularization in the meta-classifier. The last proposal for future work is to investigate Aggregate Encoders in conjunction with deep neural networks.

Acknowledgements

We are grateful for support from the subvention of the Polish Ministry of Education and Science assigned to AGH University.

References

1. Chapelle, O., Manavoglu, E., Rosales, R.: Simple and scalable response prediction for display advertising. ACM Transactions on Intelligent Systems and Technology (TIST) **5**(4), 1–34 (2014)
2. Grigas, P., Lobos, A., Wen, Z., Lee, K.c.: Profit maximization for online advertising demand-side platforms. In: Proceedings of the 2017 AdKDD and TargetAd Workshop held in conjunction with the ACM SIGKDD'17 Conference on Knowledge Discovery and Data Mining. ACM (2017)
3. Lee, K.c., Orten, B., Dasdan, A., Li, W.: 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. pp. 768–776 (2012)
4. Szwabe, A., Misiorek, P., Ciesielczyk, M.: Logistic regression setup for RTB CTR estimation. In: Proceedings of the 9th ICMLC 2017 International Conference on Machine Learning and Computing. pp. 61–70 (2017)
5. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. In: Proceedings of the 2017 AdKDD and TargetAd Workshop held in conjunction with the ACM SIGKDD'17 Conference on Knowledge Discovery and Data Mining. ACM (2017)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54

6. Weinberger, K., Dasgupta, A., Langford, J., Smola, A., Attenberg, J.: Feature hashing for large scale multitask learning. In: Proceedings of the ICML'09 Annual International Conference on Machine Learning. pp. 1113–1120. ACM (2009)
7. Wiatr, R.: Code repository for this work (updated 12012021), `https://github.com/rwiatr/agge`
8. Wiatr, R., Lyutenko, V., Demczuk, M., Słota, R., Kitowski, J.: Click-fraud detection for online advertising. In: Proceedings of the 13th PPAM'19 International Conference on Parallel Processing and Applied Mathematics. pp. 261–271. Springer (2019)
9. Yuan, Y., Wang, F., Li, J., Qin, R.: A survey on real time bidding advertising. In: Proceedings of 2014 IEEE IntThe 8th International Workshop on Data Mining for Online Advertising in conjunction with ACM SIGKDD'14 international Conference on Service Operations and Logistics, and Informatics. pp. 418–423. IEEE (2014)
10. Zhang, W.: iPinYou Data Set (accessed 25082020), `https://github.com/wnzhang/make-ipinyou-data`
11. Zhang, W., Yuan, S., Wang, J., Shen, X.: Real-time bidding benchmarking with iPinYou dataset. Tech. rep., UCL (2014), arXiv preprint arXiv:1407.7073

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_54