Learning over Categorical Data using Counting Features

With an Application on Click-through Rate Estimation

Xuyang Wu
University College London
London, United Kingdom
xuyang.wu@outlook.com

Xinyang Gao
JPMorgan
London, United Kingdom
xinyang.gao@jpmorgan.com

Weinan Zhang
Shanghai Jiao Tong University
Shanghai, China
wnzhang@apex.sjtu.edu.cn

Rui Luo
University College London
London, United Kingdom
r.luo@cs.ucl.ac.uk

Jun Wang
University College London
London, United Kingdom
jun.wang@cs.ucl.ac.uk

ABSTRACT

Input data for many machine learning applications are often categorical and contain multiple fields. A common feature representation for such categorical data is *one-hot encoding*, which expresses data instances as high-dimensional sparse binary vectors. Given this encoding machine learning models, such as logistic regression or boosted trees, are trained. However, the following problems occur when dealing with large-scale data sets: (i) The binary feature space is sparse yet extremely large, which can require a large amount of computational resources. (ii) Models based on such a feature representation will typically need to be re-trained in order to keep up-to-date with any changes in the data distribution. (iii) The one-hot feature representation provides little generalisation ability. In this paper, we propose *counting features*, a novel statistics-based feature engineering paradigm, to address the above problems. Mathematically, we show a deterministic relationship between the optimal regression parameters of counting features and one-hot binary features. Then, in the context of click-through rate estimation in online advertising, we demonstrate that counting features indeed bring better generalisation ability. Our experiments on real-world large-scale datasets demonstrate that, despite their compressed nature, the proposed counting feature engineering outperforms the one-hot binary encoded features in various cases such as cold start training and cross-campaigns training.

1 INTRODUCTION

In many learning applications, e.g., web search, recommender systems and online advertising, the features of input data are categorical [13]. That is, multiple fields form the feature set and for each field there is a set of possible categories. For example, for the field `City`, there could be more than 10^4 different categories, e.g., `London` or `Paris`, in a dataset. An example of such a categorical data set is provided in Table 1.

DLP-KDD'19, August 5, 2019, Anchorage, AK, USA

A standard feature engineering paradigm for representing such categorical data in a mathematical model is *one-hot encoding*, in which each category in each field is treated as a distinct dimension of the feature space. Each field is represented as a sparse binary vector whose dimension is equal to the number of unique categories in the field. Each element of this vector represents a specific category of the field, and is set to 1 only for those data instances of which this field is specified as the corresponding category. For the `City` field example, if the category `London` is assigned with index of 3, then for any data instance with field `City:London`, the one-hot encoded binary feature for its `City` field is $(0, 0, 1, 0, \ldots, 0)_{(1\times10^4)}$. Given such a sparse binary vector for each field, the final feature vector is obtained through the concatenation of each such "field-wise" binary feature vector.

However, such one-hot feature encoding method causes the following inevitable problems when encountering the "**3Vs challenges**" of the big data [6]. **Volume**: The dimensionality of the binary feature vector is extremely large, which imports the curse of dimensionality and requires much computational resource. **Velocity**: Most machine learning models based on such a feature representation need to be re-trained with the latest data in order to keep up-to-date with any changes in the data distribution. **Variety**: The one-hot feature representation provides little generalisation ability.

In this paper we address these problems through the construction of a simple non-sparse feature transformation of categorical data. We refer to the corresponding feature representation as *counting feature* representation. Instead of indexing each unique category of a field as a new binary feature, counting features are continuous statistical values on the field level. Specifically, we use some statistics, such as frequency or average of the target value, to represent the category of the field. These numerical values, while preserving critical statistics of the data, make the categories meaningful and allow for comparison between them. It also dramatically reduces the dimensionality of the feature representation.

We establish in theory the deterministic relationship between coefficients of the proposed counting features and that of the sparse binary one-hot features for (generalised) linear regression problems. This relationship shows that counting features can be calculated by a linear/non-linear transformation on the binary features, which compresses information and reduces dimensionality. Furthermore, we prove that the optimal coefficients of counting features are also associated to that of binary features through a non-linear deterministic mapping.

We investigate the performances of linear regressions trained on binary and counting features in a clean environment based on synthetic data, where counting features demonstrate significant effectiveness in handling cold start and data drift problems. Later in the context of practical online advertising for click-through rate (CTR) estimation [22], the logistic regression and the boosted trees models [9] based on counting features are investigated. Our experiments on two real-world large-scale online advertising datasets confirm that despite its compressed representation, the performance of counting features is highly comparable to that of binary features. In the context of cross-campaign cold-start CTR prediction, we demonstrate that the counting features indeed bring better generalisation ability to the learning models due to the lower dimension and higher stability of the representation, which partly solves the cold-start training task with data drift problems across campaign and over time in online advertising.

2 RELATED WORK

CTR Estimation is a crucial machine learning task in online performance-driven advertising. With the binary click response as target and the context information as features, CTR estimation is formalised as a standard regression problem. The authors in [20] make use of logistic regression for CTR estimation. In [22] a boosted tree method is considered, and superior performance was obtained in comparison to the linear models. The practical engineering aspects of CTR estimation on complex huge real-world datasets, as well as the performances of traditional machine learning models, are discussed in [12]. Recently, the authors of [9] combine decision trees and logistic regression together, using decision trees to construct a non-linear transformation of the original feature space, and then applying logistic regression to this transformed feature space. The empirical results in [9] indicate that such an approach can provide a significant improvement in model performance. In our study, we will refer the GBRT model in [9], but rather than binary features they use, we will apply GBRT model on counting features.

Feature Engineering is common in machine learning [21]. A good feature engineering solution would produce a more flexible and meaningful feature space. In [9], the high-dimensional binary feature space is transformed into a low-dimensional space using decision trees. In our paper, different from the majority research work in CTR prediction model design and optimisation, we focus on a more fundamental problem, i.e., how to prepare a better feature space from the categorical data for effective model training. There are some implementations of building historical features mentioned CTR estimation tasks [9, 11]. Compare to them, our work is a comprehensive investigation on counting features from both theoretic and practical aspects with various scenarios.

Transfer Learning is different from a classic machine learning setting, where the difference between training and testing datasets is ignored. In real world, the training and prediction stages normally suffer from the problem of data drift [18]. In [16], the authors make a detailed discussion on transfer learning focusing on categorising transfer learning for classification problems. When the source and target tasks are the same and the source and target domains are different, *domain adaptation* is considered [5, 15]. In [2] the domain adaptation problem in sentiment classification is discussed. No labelled data for the new domain is needed, cross-domain prediction is solved based on the assumption that the source and target domains only share a part of the features, so the difference between the two domains can be solved by minimising the distances between the two domains. The work in [7] gives an example of the method *parameter transfer*, which aims to discover the shared parameters or priors between the source and target domains. This approach is based on the assumption that the marginal distribution of feature spaces for target-domain and source-domain are similar, and also conditional probabilities ought to be similar.

The approach of this paper is based on the assumption that tasks of old and new campaigns are the similar, which are binary regressors; however, the domains of the two will be distinct due to different data distribution and feature sets.

3 COUNTING FEATURES

3.1 Introduction to Counting Features

In order to handle the "3Vs challenges" of big data [6], a feature scheme should meet the requirements of *volume*, *velocity* and *variety*. **Volume**: The feature space should be of low dimension and the number of features should have no strong relationship with the amount of data instances. As such, there will be few new features introduced into the model when observing more and more data instances. **Velocity**: The feature representation should be robust to changes to the data distribution, and hence model performance should also be robust to any such changes. **Variety**: The feature space should provide generalisation capabilities, allowing models to make informed predictions in previously unseen parts of the feature space, or parts of the feature space that contain small amount of data. An important example is real-time systems, for which the model should generalise to new forms of data instance, such as previously unseen categories.

In order to meet the above three requirements, we propose the concept of *counting features*, which departs from the traditional one-hot sparsely encoded binary features. Counting features are essentially field-wise statistical features, i.e., certain statistics of the data for some or all of the fields. For example, one such feature may be the frequency of the data which has the same category for the `Gender` field, the average target value of the data instances which have the same category for the `City` field.

Given a data instance, the one-hot binary encoding of that instance can be written in the form,

$$\boldsymbol{x}_b^\top = [\boldsymbol{x}_{b,1}^\top \; \boldsymbol{x}_{b,2}^\top \; \cdots \; \boldsymbol{x}_{b,M}^\top] \,, \tag{1}$$

for which $\boldsymbol{x}_{b,m}^\top$ is the one-hot encoding of the m^{th} field, $m = 1, ..., M$. Given N data points, $\boldsymbol{x}_b^1, ..., \boldsymbol{x}_b^N$, we write the one-hot encoding of these instances in matrix notation, with

$$\mathrm{X}_b = \begin{bmatrix} \boldsymbol{x}_b^{1\top} \\ \boldsymbol{x}_b^{2\top} \\ \vdots \\ \boldsymbol{x}_b^{N\top} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_{b,1}^{1\top} & \boldsymbol{x}_{b,2}^{1\top} & \cdots & \boldsymbol{x}_{b,M}^{1\top} \\ \boldsymbol{x}_{b,1}^{2\top} & \boldsymbol{x}_{b,2}^{2\top} & \cdots & \boldsymbol{x}_{b,M}^{2\top} \\ \vdots & & & \vdots \\ \boldsymbol{x}_{b,1}^{N\top} & \boldsymbol{x}_{b,2}^{N\top} & \cdots & \boldsymbol{x}_{b,M}^{N\top} \end{bmatrix} . \tag{2}$$

Table 1: An example of categorical data.

Target	Gender	Weekday	City	Browser
1	Male	Tuesday	London	Chrome
1	Female	Tuesday	Paris	Chrome
0	Female	Wednesday	London	Firefox
*1	Male	Wednesday	Paris	Firefox
0	Male	Tuesday	Berlin	Safari

Similarly, we denote the target value of the N data points by, $y^1, \ldots, y^N$, and use Y to denote the N-dimensional vector of these target values. Given F counting functions, $f_1(\cdot), \ldots, f_F(\cdot)$, for each $f_k : \{0,1\}^p \to \mathbb{R}^q$, where p represent the dimensions of the one-hot binary encoding of a field and q the dimension of the corresponding counting encoding of that field, the counting feature encoding of a one-hot binary feature vector, $\boldsymbol{x}_b$, is given by,

$$\boldsymbol{x}_c^\top = [\boldsymbol{x}_{c,1}^\top \; \boldsymbol{x}_{c,2}^\top \; \cdots \; \boldsymbol{x}_{c,M}^\top], \tag{3}$$

with

$$\boldsymbol{x}_{c,m}^\top = [f_1(\boldsymbol{x}_{b,m}) \; f_2(\boldsymbol{x}_{b,m}) \; \cdots \; f_F(\boldsymbol{x}_{b,m})], \tag{4}$$

for $m = 1, \ldots, M$. Given M fields and F field-wise counting functions, there are only $M \times F$ counting features in total, which has no relationship with the amount of training data or the number of unique categories in each field. Such a feature scheme therefore naturally meets the **volume** requirement.

Several examples of counting function are the *frequency counting function*,

$$f_{\mathtt{freq}}(\boldsymbol{x}_{b,m}) = \frac{1}{N}\boldsymbol{x}_{b,m}^\top \sum_{n=1}^{N} \boldsymbol{x}_{b,m}^n, \tag{5}$$

which measures the frequency with which a category is observed in a field, the *average target value function*,

$$f_{\mathtt{avg}}(\boldsymbol{x}_{b,m}) = \frac{\boldsymbol{x}_{b,m}^\top \sum_{n=1}^{N} y^n \boldsymbol{x}_{b,m}^n}{\boldsymbol{x}_{b,m}^\top \sum_{n=1}^{N} \boldsymbol{x}_{b,m}^n}, \tag{6}$$

which measures the average value of the target function across data instances which contain the given category in the given field, and, similarly, the *averaged squared target value function*,

$$f_{\mathtt{avgsq}}(\boldsymbol{x}_{b,m}) = \frac{\boldsymbol{x}_{b,m}^\top \sum_{n=1}^{N} (y^n)^2 \boldsymbol{x}_{b,m}^n}{\boldsymbol{x}_{b,m}^\top \sum_{n=1}^{N} \boldsymbol{x}_{b,m}^n}. \tag{7}$$

Other counting functions includes quantiles and mode etc. Note that the summation over the data set in Eqs. (5, 6 & 7) can be calculated once and for all and stored in memory. The evaluation of these counting functions can then be done in $O(1)$ time. Furthermore, in situations where the data is arriving through a real-time data stream, such as online programmatic advertising, it is possible to update these quantities in $O(1)$ time. Counting features are therefore amenable to real-time situations.

An example of a counting feature vector associated with the data given in Table 1, which contains four fields of categorical data, `Gender`, `Weekday`, `City` and `Browser`.

Applying the counting functions, $f_{\mathtt{freq}}(v_m)$ and $f_{\mathtt{avg}}(v_m)$, to each field v_m of the data, then the 4th data instance in Table 1 (denoted *) is expressed as a counting feature vector of the form,

$$[\underbrace{[\overbrace{0.6}^{f_{\mathtt{freq}}(\mathtt{Male})}, \overbrace{0.67}^{f_{\mathtt{avg}}(\mathtt{Male})}]}_{\mathtt{Gender}}, \underbrace{[0.4, 0.5]}_{\mathtt{Weekday}}, \underbrace{[0.4, 1.0]}_{\mathtt{City}}], \underbrace{[0.4, 0.5]}_{\mathtt{Browser}}].$$

Note that the dimensionality of one-hot sparse binary features will change when, say, there is a new city recorded in the dataset, but the dimensionality of the counting features will remain constant. As a special case, when the target value y is binary, i.e., $y \in \{0, 1\}$, the frequency counting function $f_{\mathtt{freq}}(v_m)$ and averaged target value function $f_{\mathtt{avg}}(v_m)$ will be sufficient statistics [19] of the model for target values, thus other statistics such as the standard deviation, quantiles are determined given the frequency and averaged target value in binary cases.

Moreover, with much lower dimension, the proposed counting features have the advantages of the unified and constant feature space compared to the one-hot encoded binary features, thus problem of feature space discrepancy between two data instances with one-hot features does not exist in counting features. In addition, according to [10], each pair of lower-dimensional random vectors tend to have a higher cosine/pearson correlation, hence reduce the feature **variety**. Another nice property of the resulting lower-dimensional feature spaces is the fact that the resulting models are much easier to handle when it comes to 'debugging' unexpected outputs, which matters in practice.

A severe problem with models based on a one-hot sparse feature encoding is that they frequently need to be retrained, i.e., updating the model coefficients, to keep up-to-date with changes in the data distribution. For instance, in the case of CTR estimation a unique identifier of an advertising campaign is often converted into a one-hot encoding and then used as a feature in the model. Campaigns usually run for only a relatively short period of time, and new campaigns typically start on a daily basis. Models built on a one-hot feature encoding will be unaware of the existence any new campaigns until data instances of these new campaigns are seen in the training data, and this requires models to be retrained multiple times a day. By contrast, given that the counting feature functions are incrementally updated in real-time, then the counting features will remain up-to-date with changes in the data distribution (without updating the model coefficients). Furthermore, as counting features generalise across the categories of a field, then a model based on counting features can make meaningful predicitions on data instances which contain categories that were not present in the training data. Given that the counting features of the new categories are not anomalous in comparison to the distribution of counting features over which the model was trained, then one would expect the model performance to be good on these new instances. In this manner counting features address the **velocity** and **variety** requirements of big data feature engineering.

In summary, counting features are a novel field-wise statistical feature engineering paradigm which satisfy the aforementioned "3Vs" of big data feature engineering. We will discuss on its mathematical properties and empirical performance in the rest of the paper.

3.2 Relation to Binary Features

In this section we present a mathematical relationship between the optimal linear regression coefficients (weights) of a sparse binary one-hot feature encoding and a counting feature encoding. We consider linear regression so that an analytic solution for the coefficients can be obtained and the relationship can be clearly observed. The assumption in this section is that there is a single counting feature for each field in the data. The extension of the result to multiple counting fields is trivial.

THEOREM 1. *Suppose we are given a set of one-hot binary features, $X_b \in \mathbb{R}^{N\times D}$. Additionally, suppose we are given a set of counting features, $X_c \in \mathbb{R}^{N\times M}$, constructed from the one-hot binary features, with M the number of distinct fields in the data set. Suppose that the one-hot binary features and the counting features are related through the mapping,*

$$X_c = X_b T(X_b), \tag{8}$$

with $T(X_b) \in \mathbb{R}^{D\times M}$ some matrix that depends on X_b. (For notational ease we shall use T to denote $T(X_b)$.)

Suppose that $\boldsymbol{w}_b \in \mathbb{R}^D$ and $\boldsymbol{w}_c \in \mathbb{R}^M$ are the optimal weights of the linear regressions problems, respectively,

$$y_1 = \boldsymbol{x}_b^\top \boldsymbol{w}_1 + \epsilon_1, \tag{9}$$

$$y_2 = \boldsymbol{x}_c^\top \boldsymbol{w}_2 + \epsilon_2. \tag{10}$$

Provided that $X_b^\top X_b$ and $T^\top X_b^\top X_b T$ are both invertible, then the optimal linear regression parameters of the one-hot binary features and the counting features satisfy the relation,

$$\boldsymbol{w}_c = \left(T^\top X_b^\top X_b T\right)^{-1} T^\top X_b^\top X_b \boldsymbol{w}_b. \tag{11}$$

PROOF. The optimal weights of the regression problem (9) satisfy the standard relation,

$$\boldsymbol{w}_b = \left(X_b^\top X_b\right)^{-1} X_b^\top Y. \tag{12}$$

Likewise, the optimal weights of the regression problem (10) are given by

$$\boldsymbol{w}_c = \left(X_c^\top X_c\right)^{-1} X_c^\top Y. \tag{13}$$

From the relation (8) this is equivalent to,

$$\boldsymbol{w}_c = \left(T^\top X_b^\top X_b T\right)^{-1} T^\top X_b^\top Y. \tag{14}$$

From (12), we have that $X_b^\top Y = \left(X_b^\top X_b\right)\boldsymbol{w}_b$, so that (14) can be written in the equivalent form,

$$\boldsymbol{w}_c = \left(T^\top X_b^\top X_b T\right)^{-1} T^\top X_b^\top X_b \boldsymbol{w}_b,$$

which completes the result. □

We note that the form of one-hot binary features infers that it is always possible to write the relationship between counting features and binary ones in the form (8). In particular, defining $T \in \mathbb{R}^{N\times M}$ such that $T_{n,m} = f(\boldsymbol{x}_{b,m}^n)$, for $m = 1, ..., M$ and $n = 1, ..., N$, it follows that (8) holds. Hence the result in Theorem 1 will always hold.

3.3 Learning Models with Counting Features

Consider the example of click-through rate prediction [8, 12], it is necessary to predict whether a user clicks an ad or not. In this case the target variable, $y \in \{0, 1\}$, denotes whether the user clicked on the ad: $y = 1$ if clicked, and the features, x, contain information such as the users' previous browsing history, the content of the website etc. Based on counting features representation of the data it is possible to learn a logistic regression model (LR) [12] that directly models the probability that the user will click on the ad,

$$P_{\text{LR}}(y = 1|\boldsymbol{x}_\text{c}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x}_\text{c})}, \tag{15}$$

and $P_{\text{LR}}(y = 0|x_\text{c}) = 1 - P_{\text{LR}}(y = 1|x_\text{c})$.

Equally, with counting features it is possible to build other models, such as tree-based models. Compared with binary features, the continuous space of counting features naturally accommodates the space-splitting-based techniques of tree-based models.

We use gradient boosting regression trees (GBRT) [4] to obtain a non-linear CTR prediction model. Starting with an initial regression tree, the GBRT algorithm incrementally builds a collection of regression trees. After G iterations there will be G trees in the collection, and the GBRT regressor will take the form,

$$P_{\text{GBRT}}(y = 1|\boldsymbol{x}_c) = \sum_{k=1}^{G} g_k(\boldsymbol{x}_\text{c}), \tag{16}$$

where $g_k(\boldsymbol{x}_\text{c})$ denotes the prediction of the k^{th} tree on $\boldsymbol{x}_\text{c}$. At each stage of the algorithm a new tree is learnt by minimising the square of the residuals between the target variables and the predictions given by the current GBRT regressor. More details of the GBRT method can be found in [4, 9].

As will be discussed in Section 6.2, the performance of GBRT with counting features is highly comparable to that of LR and GBRT with binary features measured by area under the curve (AUC), which further verifies the effectiveness and rationality of counting features.

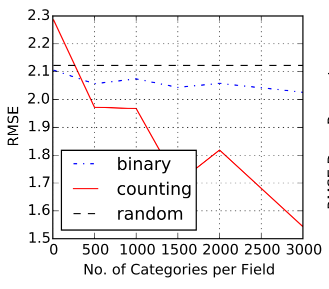
4 SYNTHETIC DATA EMPIRICAL STUDY

In this section, we perform an empirical study on comparing the machine learning models based on binary and counting features using synthetic data. With the synthetic data, we could create a clean test environment to get rid of various noise in different real-world scenarios to better study the properties of counting features. Specifically, we study the properties of the learning models on counting features in *cold start* and *data drift* tasks, respectively.

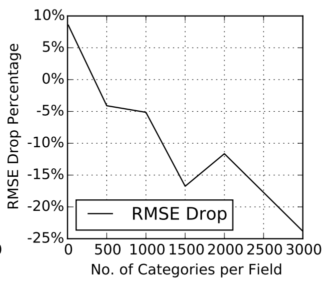
4.1 Cold Start Study

In this experiment, we study the cold start problem [14], where the data instance number is not sufficiently higher than the number of binary features (i.e. the total categories). The number of fields is set to be 3, as for the number of categories in each field, it is to be tuned and denoted as z. There is a fixed size of training and test dataset in this experiment, each contains 10,000 instances. For each data instance, the categories of the 3 fields are uniformly sampled from z categories[1]. These categories are then converted into one-hot binary features X_b^{train} and X_b^{test} and frequency, average target value counting features X_c^{train} and X_c^{test}. Then we sample the binary feature weight vectors on training and test datasets with the same Gaussian distribution, denoted as $\boldsymbol{w}_b^{\text{train}}$ and $\boldsymbol{w}_b^{\text{test}}$ respectively. With the binary features and sampled weights, the target value for each instance can be generated via $Y^{\text{train}} = X_b^{\text{train}}\boldsymbol{w}_b^{\text{train}} + \boldsymbol{\epsilon}$ and $Y^{\text{test}} = X_b^{\text{test}}\boldsymbol{w}_b^{\text{test}} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is the Gaussian white noise sampled from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Then with $(X_b^{\text{train}}, Y^{\text{train}})$ and $(X_c^{\text{train}}, Y^{\text{train}})$, we can learn the binary feature weight $\hat{\boldsymbol{w}}_b$ and counting feature weight $\hat{\boldsymbol{w}}_c$ using standard linear regression model, respectively. Then we test the root mean squared error (RMSE) performance of linear

[1]Other distributions, e.g., linear, exponential and power law, are also studied and provide the similar results.

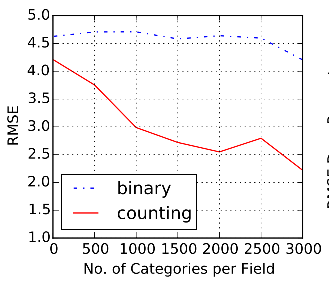


(a) Performance

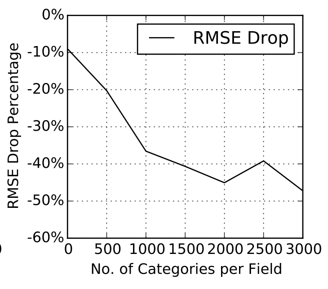


(b) RMSE Drop

Figure 1: Comparison with cold start setting.



(a) Performance



(b) RMSE Drop

Figure 2: Comparison with data drift setting.

regression models on test data with binary features $(X_b^{\text{test}}, Y^{\text{test}})$ and counting features $(X_c^{\text{test}}, Y^{\text{test}})$.

Figure 1 shows the models' prediction performance on test data against z. It can be observed that the estimation performance improvement of the counting feature trained model is much more significant than that of the binary feature trained one as the number of categories per field increases. The reason is that as the category number in each field increases, the binary feature number increases as well, which means more training instances are needed to learn a well-trained model. However, in the setting, the number of training instances is fixed to 10,000. Therefore the performance of binary feature trained model works unsatisfactory and just slightly better than the random guessing baseline as the number of category in each field increases. By contrast, the number of counting features does not increase; its values (i.e., frequency and averaged target value) are denser and more diverse when the categories number increases, which helps counting feature trained model to achieve better performance.

4.2 Data Drift Study

Different from the cold start setting, in this experiment the training instances are sufficient. The number of training instances is set to as 10 times as that of the binary features. In addition, the weight vectors of training and test data are sampled from two Gaussian distributions with different means (0.1 and 0.5) and variances (1 and 2), which means the average target value of each category is different. Moreover, the frequency distributions of the categories in each field are also different on two datasets. For the training data, we generate the categories by a uniform distribution, while for the test data we generate the categories by a linear distribution, i.e., the probability of generating each category is proportional to its index number. Again, we change the number of categories in each field and check the performances of two models.

Table 2: Feature property comparison.

Feature Types	Space	Sparsity	Dimension	Model Drift
Binary	Different	Sparse	High	Heavy
Counting	Same	Dense	Low	Light

As can be observed, in this data drift test, counting feature trained model consistently outperforms the binary feature trained one, which verifies that the low-dimensional counting features help build the model with higher generalisation ability. As the number of categories in each field increases, although both models perform better as there are richer feature representations and sufficient training instances, the improvement of counting feature trained model gets more significant than that of the other, which verifies the robust effectiveness of counting features on transferring knowledge from drifted data distribution.

From the discussions of Sections 3 and 4 the comparison between binary and counting features can be summarised in table 2.

5 PRACTICE ON AD CTR ESTIMATION

In this section, we discuss a practical application where the proposed counting feature representation would demonstrate its effectiveness of compact representation, efficient updating and high generalisation across different domains. Then detailed experiments are provided in Section 6.

5.1 Zero-Shot Cross-Campaign CTR Estimation

As discussed in [3], many traditional machine learning algorithms and models can be only used under the assumption that the training and test datasets are generated from the same distribution and with the same feature space, which is, however, rare in practice. When the distribution changes, the model needs to be re-trained with new data to keep up-to-date. In the context of online advertising with different ad campaigns, it is expensive and time-consuming to perform random ad display to collect the user click data and then train an initial ad click-through rate (CTR) model. In this case, transfer learning is needed which can borrow the knowledge learned from previous ad campaigns and apply to new campaigns to increase the efficiency for CTR prediction and decrease the cost of training new models. According to [17], however, the user behaviour across different campaigns are quite different. For example, a football sneaker campaign is probably preferred by young male users while lipstick campaign is only interesting to female users; a London tourism ticket campaign is only popular among UK users while such users will not get interested in a campaign of Shanghai hotel. It is non-trivial to perform knowledge transfer from a well-studied campaign to a new one.

At first, using the definitions in [16], for the source domain, we specify the data of a well-studied campaign as $X^s = \{\boldsymbol{x}_1^s, \boldsymbol{x}_2^s, ..., \boldsymbol{x}_n^s\}$, with each instance sampled from a p.d.f. $p^s(\boldsymbol{x})$, and the corresponding user click feedback as $Y^s = \{y_1^s, y_2^s, ..., y_n^s\}$. Assuming the data has already been processed by a certain feature engineering and X^s is encoded with feature representation. The learning model on the source data is denoted as $p_\theta(y^s|x^s)$, which is parameterised

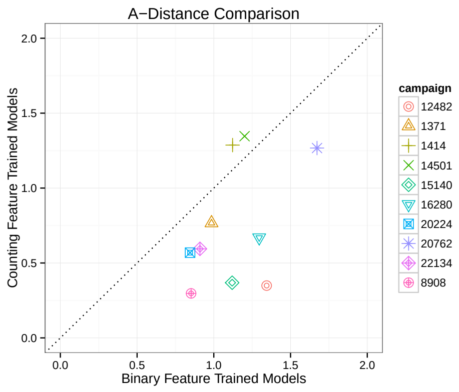


Figure 3: $\mathcal{A}$-Distances of the binary/counting feature trained models among campaigns.

by $\boldsymbol{\theta}$. Similarly, for the target domain, the data of a new campaign is denoted as $X^t = \{\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, ..., \boldsymbol{x}_n^t\}$, with each instance sampled from a p.d.f. $p^t(\boldsymbol{x})$, and its feedback as $Y^t = \{y_1^t, y_2^t, ..., y_n^t\}$. The prediction model is denoted as $p_{\boldsymbol{\vartheta}}(y^t|x^t)$, with another parameter $\boldsymbol{\vartheta}$.

From the above discussion, both the distributions of the data X^s and X^t and the true prediction models $p_{\boldsymbol{\theta}}(y^s|\boldsymbol{x}^s)$ and $p_{\boldsymbol{\vartheta}}(y^t|\boldsymbol{x}^t)$ are quite different from its counterparts. If we could find a good feature representation of the data $f(\boldsymbol{x})$ such that the underlying prediction model is close to each other $p_{\boldsymbol{\theta}}(y^s|f(\boldsymbol{x}^s)) \approx p_{\boldsymbol{\theta}}(y^t|f(\boldsymbol{x}^t))$ with the same parameter $\boldsymbol{\theta}$, the zero-shot CTR estimation transfer learning would be successful across campaigns.

5.2 Cross-Campaign Model Distance

In this section, we try to find a quantitative relationship between the models trained on any two different campaigns, which can be used to measure the difficulty of knowledge transfer.

Specifically, we borrow the concept of transfer distance, namely $\mathcal{A}$-distance proposed in [1], to measure the difference between the condition probability $p_{\boldsymbol{\theta}}(y^s|x^s)$ and $p_{\boldsymbol{\theta}}(y^t|x^t)$. With either binary features or counting features, we train a model on campaign s and perform CTR estimation on another campaign t. The $\mathcal{A}$-distance is calculated by

$$\mathcal{A}\text{-Distance} = 4 \times (1 - \text{AUC}), \tag{17}$$

which is in the range of $[0, 2]$. The lower the $\mathcal{A}$-distance is, the more similar the models on the two domain are. We performed 10 times of trials for each campaign pair and average the $\mathcal{A}$-distance for each source campaign to get the result as depicted in Figure 3.

It can be observed that in 8 out of 10 cases the $\mathcal{A}$-distance of counting feature trained models between the source domain to the rest of the target domains are lower than those trained by binary features. Thus we regard it is more feasible to accept $p_{\boldsymbol{\theta}}(y^s|\boldsymbol{x}^s) \approx p_{\boldsymbol{\theta}}(y^t|\boldsymbol{x}^t)$ as a reasonable assumption in our cross-campaign setting with counting features.

Therefore, we can directly apply the model learned from the old campaign to the new campaign and we still expect to get decent CTR prediction performance. Moreover, as discussed in Section 5.1, since the value of counting features can be updated in real-time with $O(1)$ time, by updating the counting values after observing each data instances, $p^t(\boldsymbol{x}^t)$ could efficiently approach its true distribution with increasing number of new incoming data instances, thus the CTR estimation performance could get improved.

6 EXPERIMENTAL RESULTS

In this section, the empirical results will be presented in two stages. First, in the standard single campaign CTR estimation task, we will demonstrate that the CTR performance of counting features is comparable to or even better than that of one-hot binary features. Second, we will show that the counting features perform significantly better than the binary features in cross-campaign CTR estimation task.

6.1 Experiment Setup

Datasets. Our experiments will be based on two large-scale real-world datasets. The first is a public dataset from iPinYou [23], which contains 19.5M ad impressions and 14.8K clicks from 9 campaigns. The second dataset is a proprietary dataset provided by Adform, a global digital media advertising technology company based in Copenhagen, Denmark. It consists of two weeks of online display ad logs collected during March 2015, including 10.2M ad impressions and 159.9K clicks from 486 advertisers' 2665 campaigns.

For iPinYou dataset, we follow the same processing method as in [23] to select 16 fields to train our model. For Adform dataset, we choose 14 fields data to train our model, which are categorised into user information, ad information and context information. The user information includes `user country ID`, which shows the region of the user, `log date` and `log time`, which show the time dynamics of user's online behaviour, and `visited domains` and `visited logpoints` which show the user's browsing history. The ad information includes `creative size`, `position ID`, showing the ad's basic information, `client ID`, showing the advertisers that the ad belongs to, representing the vertical of the ad, and `placement ID`, showing which specific campaign the ad is from, as well as `inventory source ID` which involves ad exchange information. The context information includes `OS ID` and `browser ID`, showing the device, operation system and browser information of the user.

Experiment Protocol. The whole process of feature generation is shown in Figure 4. For Adform dataset as example, the whole dataset is divided into training, test and counting datasets, where the test data is split from the last 7 days while the training and counting datasets are split randomly from the previous 7 days. For counting dataset, we use it for (i) building the feature index for binary features and (ii) calculating the counting feature values like frequency and average CTR values of each data instance. The reason we do this is to make sure that the counting feature values are independently generated from training or test datasets, so that overfitting can be to-some-extent avoided and the process for generating binary and counting features are the same to make the comparison fair. Another noteworthy point is that the model training has no dependency with the training data instances as the counting feature values are calculated based on the entire counting dataset.

For the experiment of cross domain learning, we divide the datasets into different campaigns using `client ID` and `placement`

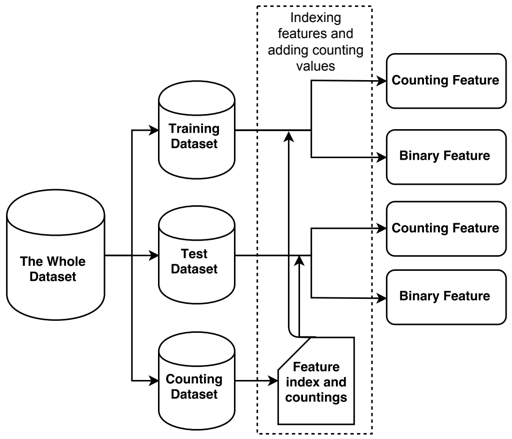


Figure 4: Process of label counting feature and binary feature.

ID, we use the other 12 data fields to perform CTR estimation in the experiment.

In the first stage of experiment, for each campaign, we will compare the performances of binary and counting features using logistic regression and GBRT, respectively, which is measured by AUC.

In the second stage of experiment, we will apply counting features in cold-start cross-campaign CTR estimation problems. Specifically, from the perspective of an online advertising agent serving for different clients' campaigns, based on the information from old campaigns, it hopes to get accurate CTR prediction results for new campaigns without training new models but directly making use the off-the-shelf models trained from the data of old campaigns. The counting features are leveraged to help the agent to achieve this goal and show they are superior to the traditional binary features.

To simulate the cross-campaign CTR estimation scenario, in the experiment, the whole 14-day Adform data will be split into sub-datasets based on the `client ID` so that in each subset all impressions are from a unique client. As shown in Figure 5, we will transform the categorical data in old campaign into binary features and counting features, and train logistic regression models respectively. When the data instances of the test new campaign arrive in time sequence, the feature values will be updated automatically in real time[2], and CTR prediction is performed based on the model trained from old campaigns (without any re-training) and feature values updated from the test new campaign data.

6.2 Single Campaign CTR Estimation

We use the two datasets to compare the CTR prediction performances of binary and counting features supporting logistic regression and GBRT model. The AUC and RMSE of all 9 campaigns in iPinYou dataset is shown in Table 3 and the largest 10 campaigns in Adform dataset is shown in Table 4, respectively.

From Tables 3 and 4, we have the following observations. When employing linear LR, counting features have no higher performance than binary features, which is reasonable because the counting features' dimension is much smaller than the binary features and its

[2]The feature values are updated for each real-time collected mini-batch with the size of 3000. Thus the ordering of the input data can be almost ignorable.

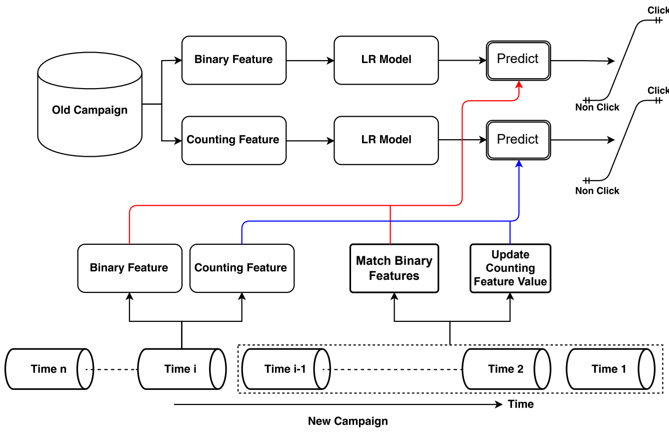


Figure 5: Estimate new campaign CTR using binary feature and counting feature comparison.

Table 3: Single-campaign performance (iPinYou).

Camp-	Binary Feature				Counting Feature			
	LR		GBRT		LR		GBRT	
aign	AUC	RMSE	AUC	RMSE	AUC	RMSE	AUC	RMSE
1458	**70.97%**	**2.85%**	68.87%	2.91%	60.47%	**2.85%**	70.28%	**2.85%**
2259	**71.24%**	1.73%	67.16%	1.73%	66.84%	1.73%	67.12%	1.73%
2261	60.96%	1.68%	59.94%	1.68%	52.07%	1.68%	**63.03%**	**1.52%**
2821	59.63%	2.38%	61.45%	2.38%	53.96%	2.39%	**61.84%**	**2.33%**
2997	**58.07%**	**5.68%**	55.81%	6.94%	53.99%	**5.68%**	56.24%	6.09%
3358	79.08%	2.98%	78.17%	2.98%	69.89%	2.98%	**79.20%**	2.98%
3386	78.44%	**2.77%**	79.04%	**2.77%**	63.77%	2.78%	**79.74%**	2.82%
3427	73.46%	**2.53%**	72.54%	2.54%	60.47%	**2.53%**	**75.25%**	**2.53%**
3476	65.97%	2.32%	64.57%	**2.31%**	61.01%	2.32%	**66.82%**	2.32%

Table 4: Single-campaign performance (Adform).

Camp-	Binary Feature				Counting Feature			
	LR		GBRT		LR		GBRT	
aign	AUC	RMSE	AUC	RMSE	AUC	RMSE	AUC	RMSE
8908	80.26%	9.50%	**84.42%**	**9.16%**	72.99%	9.67%	83.65%	9.35%
14501	89.82%	7.90%	**91.56%**	**7.26%**	87.44%	8.21%	90.44%	7.36%
22134	67.12%	11.10%	**84.13%**	**10.71%**	62.60%	11.26%	83.96%	10.89%
1414	72.50%	17.72%	80.91%	16.95%	67.16%	18.04%	**81.43%**	**16.90%**
20224	90.02%	7.47%	**94.16%**	**7.23%**	79.59%	7.52%	94.02%	7.28%
16280	87.20%	4.30%	90.02%	**4.09%**	81.80%	4.51%	**90.32%**	4.20%
12482	80.13%	4.84%	**83.38%**	**4.67%**	76.62%	4.97%	82.05%	4.79%
1371	79.82%	9.28%	88.81%	8.55%	79.29%	10.99%	**89.34%**	**8.54%**
15140	85.50%	6.92%	90.19%	**6.76%**	77.39%	7.25%	**90.44%**	6.83%
20762	65.00%	2.02%	70.01%	2.02%	85.54%	1.71%	**94.17%**	**1.93%**

continuous value cannot take many advantages in a linear combination. However, with the non-linear tree model GBRT, counting features provide highly comparable performance or even better performance against binary features on both datasets. The reasons are probably two-fold. First, the counting feature value is continuous, which naturally fits the splitting process in tree model learning. On the contrary, each binary feature only provides one splitting point, i.e., 0 or 1. Second, the counting features are of low dimensions, i.e., 32 dimensions for iPinYou dataset and 24 dimensions for Adform dataset, while the binary features are of very high dimensions, i.e., 0.96M for iPinYou dataset and 4.84M for Adform dataset. As such, GBRT can easily make use of all counting features in the learning and prediction process, while it is impossible to build a GBRT with

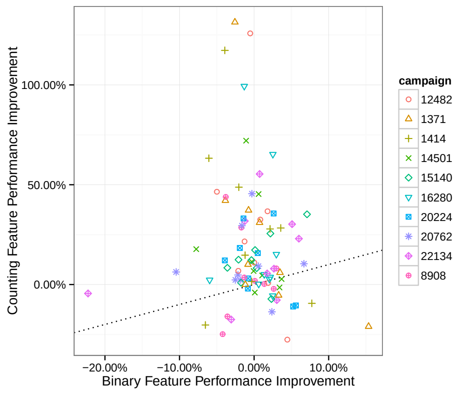


Figure 6: Performance comparison of binary and counting feature for cross-campaign CTR after counting convergence.

all of the binary features. Lots of binary features are given up when learning a GBRT.

In sum, the result demonstrates the effectiveness of counting features for discriminating the target values despite its heavy information compression from binary features. The statistical counting values make it feasible to compare different categories of each field via these statistic counting values, which is, however, infeasible on binary features.

Effective Counting Features Study. Below we show the counting feature coefficients with high absolute values.

$\texttt{ad size}_{\text{avg}}$	$+++$	$\texttt{position}_{\text{avg}}$	$++$	$\texttt{browser}_{\text{avg}}$	$+$
$\texttt{exchange}_{\text{avg}}$	$+++$	$\texttt{screen size}_{\text{avg}}$	$++$	$\texttt{hour}_{\text{freq}}$	$+$
$\texttt{os}_{\text{avg}}$	$+++$	$\texttt{user agent}_{\text{avg}}$	$+$	$\texttt{exchange}_{\text{freq}}$	$-$

Here "+" means the positive impact on CTR prediction and "−" means the negative one. It can be observed that 7 of 9 of the listed effective counting features are average CTR f_{avg} as in Eq. (6), while 2 of 9 are frequency f_{freq} as in Eq. (5), which is intuitive. For example, $\texttt{os}_{\text{avg}}$ is highly helpful for CTR estimation because the users' behaviour on different OSes are naturally discrepant; the positive impact of $\texttt{hour}_{\text{freq}}$ means the volume-peak hours also bring high CTR.

6.3 Cold-Start Cross-Campaign CTR Estimation

The second stage of experiment is concerned with the investigation of generalisation ability of counting/binary features in the scenario of cold-start cross-campaign CTR estimation. The experiment is conducted based on Adform dataset as it contains sufficient campaigns in the same period to perform the cross-campaign learning experiment. The dataset is split by `client ID` so that in each small dataset there are only the impressions from one unique client. Particularly, we focus on reporting the clients with more than 500,000 logged ad impressions in order to make the experimental results convincing. As such, 90 cross-campaign CTR estimation tasks are studied in this stage of experiment.

Figure 6 demonstrates the result that the counting features indeed improve the performance for cold-start cross-campaign CTR estimation. If we assume that the distribution of the new campaign is identical to the old ones, the performance of the CTR should be the same among all sub-dataset of the new campaign. We define $\texttt{AUC}_{\texttt{before}}$ as the CTR prediction performance on the test new campaign before any feature value update with the new incoming data. In such case the model can only predict the CTR totally based on the information (i.e., model parameters and feature values) from the old campaign. We also define $\texttt{AUC}_{\texttt{after}}$ as the AUC performance of binary or counting feature after sufficient information of the new campaign has been observed and its performance converges. In Figure 6 we compare the ratio $\texttt{AUC}_{\texttt{after}}/\texttt{AUC}_{\texttt{before}}$ for binary features and counting features, it shows that for binary feature there is no improvement for about half cases and the overall value is around 0 but for counting features the improvement is high: 80% cases have positive improvement and 33% cases have improvement higher than 25%. Note that in both binary and counting feature settings, the model performs no learning on the new campaign data. The value of each counting feature changes based on feeding each data instance and as shown in Section 3.1. Such changes are completed in $O(1)$ time, which is highly feasible.

Figure 7 provides the further details of the result, where each grid corresponds to the AUC performance improvement of a cross-campaign CTR estimation task. The model is trained on one campaign's (Y-axis) data and tested on another campaign (X-axis). The red grids indicate that the counting features perform better than binary features, and the value shows its improved performance in terms of AUC in percentage compared to binary feature. Conversely, blue grids indicate the counting feature performs worse than binary feature, and thus the improvement value is negative.

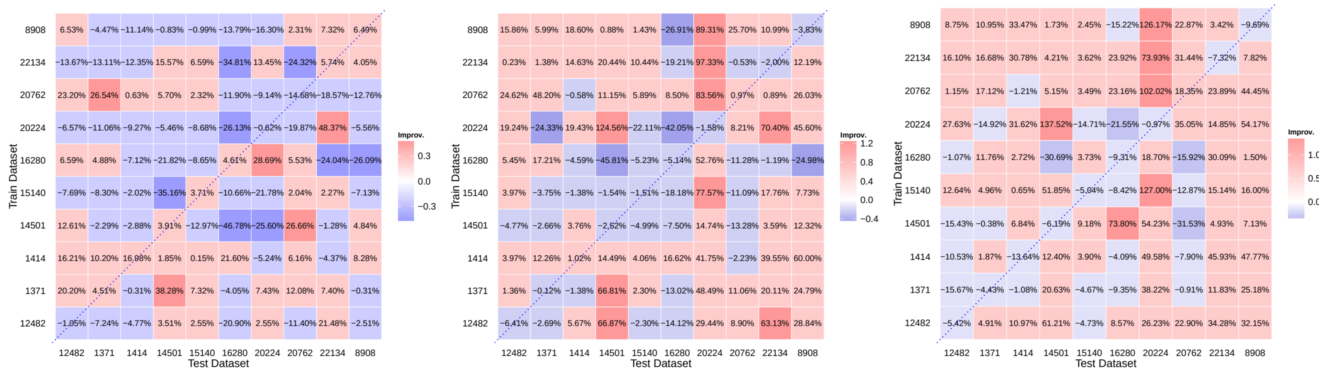
The ratio $(\texttt{AUC}_{\texttt{before(count)}} - \texttt{AUC}_{\texttt{before(bi)}})/\texttt{AUC}_{\texttt{before(bi)}}$ is calculated in Figure 7(a) to show how counting feature improves the performance compared to binary feature before updating the feature values. It can be obversed that with no observation of any data of the test campaign, the counting features and binary features are comparable.

In Figure 7(b) we show the same calculation for that but after updating the feature values until convergence when observing sufficient incoming data instance, without updating the model. The ratio $(\texttt{AUC}_{\texttt{after(count)}} - \texttt{AUC}_{\texttt{after(bi)}})/\texttt{AUC}_{\texttt{after(bi)}}$ is shown in the matrix showing that excluding the diagonal in which the train and test datasets are the same, in 61 out of 90 experiments counting features perform better than binary features, much better than that in Figure 7(a). This result demonstrates that the real-time updating of counting feature value without the re-training the model indeed brings an efficient way of cold-start cross-campaign CTR estimation.

Finally, Figure 7(c) calculates the relative improvement of AUC based on counting features over that based on binary features:

$$\left(\frac{\texttt{AUC}_{\texttt{after(count)}}}{\texttt{AUC}_{\texttt{before(count)}}} - \frac{\texttt{AUC}_{\texttt{after(bi)}}}{\texttt{AUC}_{\texttt{before(bi)}}}\right) \Big/ \left(\frac{\texttt{AUC}_{\texttt{after(bi)}}}{\texttt{AUC}_{\texttt{before(bi)}}}\right). \quad (18)$$

As a result, in 68 out of 90 cross-campaign CTR estimation tasks, counting features provide better AUC improvement ratio than binary features. This is benefited from counting features' advantages of identical low-dimension feature space across different campaigns, which effectively avoids feature discrepancy and provides good generalisation.



(a) Before counting value updating (b) After convergence (c) Relative improvement

Figure 7: AUC improvement of counting features against binary features in cross-campaign CTR estimation.

In sum, based on the experiment results we claim that counting features actually bring a better generalisation ability to the learning models and lead to higher AUC performance in the cold-start cross-campaign CTR estimation, which is a problem of practical importance for online advertising.

7 CONCLUSIONS

In this paper, we introduced the concept of counting features, which is a novel statistics-based and highly scalable feature engineering paradigm for big data. We performed both theoretic and empirical analysis about counting features. Theoretically, we derived mathematical relationship between the learned model weights based on counting features and binary features, and then showed that such highly compressed feature representation could lead to high generalisation ability of the learning models. In the empirical study, we conducted comprehensive experiments of ad CTR estimation tasks based on two real-world large-scale online advertising datasets, both with single-campaign and cross-campaign settings. We observed that in the single-campaign setting, counting features provided highly comparable or even better prediction than binary features. In the cold-start setting, we found the models trained with counting features provided much higher generalisation ability in cold-start campaign CTR estimation tasks with data, which is a strong evidence of the effectiveness of counting features. To the best of our knowledge, our work is the first systematical research on counting features, both theoretically and empirically.

In the future work, we will perform deeper investigations on more general learning scenarios with different counting features, such as the non-binary regression and the multi-label classification problems. Also, it will be interesting to test the model performance with the ensemble of counting features and binary features. In addition, as it transfers the high dimensional binary features into low-dimensional continuous features, it is interesting to explore the potential usage of counting features on deep learning models.

REFERENCES

[1] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. *NIPS* (2007).
[2] John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.
[3] Leon Bottou. 2015. Two big challenges in machine learning. http://goo.gl/Ip0Kju. ICML Invited Talk.
[4] Tianqi Chen. 2014. Introduction to Boosted Trees. https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf
[5] Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *JAIR* (2006).
[6] Pinal Dave. 2013. Big Data - What is Big Data - 3 Vs of Big Data - Volume, Velocity and Variety. http://goo.gl/J7oLLd.
[7] Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. 2008. Knowledge transfer via multiple model local structure mapping. In *KDD*.
[8] Thore Graepel, Joaquin Q Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML*.
[9] Xinran He et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *ADKDD*.
[10] John Hopcroft and Ravindran Kannan. 2014. Foundations of Data Science. (2014).
[11] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past erformance data. In *ADKDD*. ACM.
[12] H Brendan McMahan et al. 2013. Ad click prediction: a view from the trenches. In *KDD*.
[13] B Mirkin. 1976. Analysis of categorical features. *Finansy i Statistika Publishers, Moscow* (1976), 166.
[14] Richard J Oentaryo, Ee-Peng Lim, Jia-Wei Low, David Lo, and Michael Finegold. 2014. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *WSDM*. ACM.
[15] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22, 2 (2011), 199–210.
[16] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *TKDE* 22, 10 (2010), 1345–1359.
[17] Sinno Jialin Pan, Vincent Wenchen Zheng, Qiang Yang, and Derek Hao Hu. 2008. Transfer learning for wifi-based indoor localization. In *Proc. Workshop Transfer Learning for Complex Task of AAAI*.
[18] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2009. *Dataset shift in machine learning*. The MIT Press.
[19] Edward C Real. 1996. Feature extraction and sufficient statistics in detection and classification. In *ICASSP*. IEEE.
[20] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *WWW*. ACM, 521–530.
[21] Sam Scott and Stan Matwin. 1999. Feature engineering for text classification. In *ICML*.
[22] Ilya Trofimov, Anna Kornetova, and Valery Topinskiy. 2012. Using boosted trees for click-through rate prediction for sponsored search. In *ADKDD*.
[23] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Real-Time Bidding Benchmarking with iPinYou Dataset. *arXiv preprint arXiv:1407.7073* (2014).