Improving Click-Through Rate Prediction Accuracy in Online Advertising by Transfer Learning

Yuhan Su†, Zhongming Jin∧, Ying Chen∧, Xinghai Sun∧,
Yaming Yang∧, Fangzheng Qiao‡, Fen Xia∧, Wei Xu†
† Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China.
∧ Baidu, Inc., No.10 Shangdi Street, Haidian, Beijing 100084, China.
‡ School of Software, Nanjing University, Nanjing 210093, China.
syhmartin@yeah.net, jinzhongming888@gmail.com, {chenying09, sunxinghai, yangyaming}@baidu.com,
qiaofangzheng@gmail.com, xiafen@baidu.com, weixu@mail.tsinghua.edu.cn

ABSTRACT

As the main revenue source of Internet companies, online advertising is always a significant topic, where click-through rate (CTR) prediction plays a central role. In online advertising systems, there are often many advertisement products. Due to the competition in the bidding mechanism, some advertising products may get lots of data to train the CTR prediction model while some may lack high-quality data. However, to predict accurate CTR, a large amount of data is needed. Therefore, transfer knowledge from the large product (source) to the small product (target) is necessary. We propose a transfer learning method that iteratively updates the data weights to selectively combine source data with target data for training. To efficiently process huge advertisement data, we design a sampling strategy based on the gradient information, and implement the algorithm with a MapReduce-like machine learning framework. We do experiments on real advertisement datasets. The results show that our approach improves the accuracy of CTR prediction compared to the supervised learning method.

CCS CONCEPTS

•Information systems →Online advertising; •Computing methodologies →Transfer learning;

KEYWORDS

Online Advertising, Transfer Learning, CTR Prediction

WI '17, Leipzig, Germany

DOI: 10.1145/3106426.3109037

1 INTRODUCTION

Online advertising is a multi-billion dollar industry, and the major revenue source of many Internet companies. It is also a well-studied topic both in academia and industry. In a modern online advertising systems, online revenue is determined by three major factors: the *average click price (ACP)*, the *click-through rate (CTR)* and the number of *page views (PVs)*. Factors outside of the advertising system often affect the PV, and thus the key question for advertising is how to increase revenue given a certain PV.

Click through rate (CTR) is a key metric for advertising. It is the ratio between the number of clicks on an ad and the total number of impressions. Improving CTR directly improves the revenue. Also, a high CTR means more relevant ads on the page, indirectly improving user experience as well. A common way to optimize CTR is to *predict* the CTRs of a portfolio of ads based on a variety of features about the user, the page, the ad itself etc., before actually displaying one, and choose the ad with the highest CTR to display. Thus, it is important to accurately predict the CTR [5, 13, 18, 23], which is the goal of this paper.

In this paper, we focus on improving CTR prediction accuracy on Baidu's AllianceAds advertising service, a system similar to Google's AdSense. Baidu is the largest search engine provider in China, and one of the largest online advertising companies in the world. AllianceAds offers a variety of *advertisement products*. These products differ in many ways. For example, they target different websites or apps (e.g. games, shopping, news, social media), have different presentations (e.g. text, image, Flash animations), as well as have different cost structures (e.g. *cost-per-impression* or *cost-per-click*). AllianceAds delivers ads to hundreds of thousands of third-party websites and mobile apps every day.

In general, more data we have, the better a CTR prediction model can be. We are accumulating a huge amount of user browsing data every day. However, CTR prediction remains a challenge especially in two kinds of products: 1) Small,

niche-market products and 2) newly developed products. In either case, there is little click history to build an accurate CTR model. As a result, they do not perform as well as other well-tuned products in terms of CTR, or even revenue. Thus, the chances of them being chosen are even lower, further limiting their ability to accumulate more data, resulting in a vicious cycle.

It is a natural idea to use the vast amount of data from other large products to build the CTR model for these niche / new products. However, the problem is not trivial. Although the features are the same for different products, the *distribution* of the data can be quite different. In fact, as we will show in Section 5, naively using data from a different product can negatively affect the prediction accuracy.

Transfer learning is a promising approach to solve the problem above. Specifically, we want to "transfer" some of the data from the large products (called the *source*) to the smaller products (called the *target*), in order to increase the training data size for the small products. This type of transfer learning is often called "instance transfer learning". To deal with the problem that the source and target have distinct distributions, we want to sample the source set to select the data points that are *similar to* those in the target set, and combine them into the target set for training.

We use an improved version of TrAdaBoost [8]. TrAdaBoost assigns a weight on each data point from both source and target set, and iteratively adjusts the weight. It adjusts the weight differently for source and target data: for a misclassified data point in the target set the algorithm increases its weight, allowing it to contribute more in the model in the next round; however, for a misclassified source data point, the weight is decreased as we can assume it is less similar to target data.

To make the approach scalable to the large advertising datasets, we made improvements both to the algorithm and to its implementation. To the algorithm, we add a sampling strategy based on the gradient information in each iteration. The strategy allows us to use large-scale data as input, and it provides an intuitive hyper-parameter to control the tradeoff on how much source dataset to use. We also use a model ensemble approach to combine the results from different iterations. On the implementation side, we provide a MapReduce-like implementation, allowing the algorithm to run in hundreds of servers in parallel.

We perform experiments on real data from two products with over 100 million data instances. The result shows that our approach significantly improves the CTR prediction performance over existing approaches, and trains 3x faster than the off-the-shelf TrAdaBoost algorithm.

In summary, we make the following three contributions in this paper.

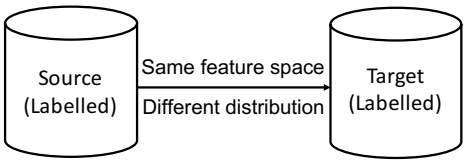

Figure 1: Transfer learning. Our source and target have the same feature space but different data distribution.

- We proposed a new transfer learning based approach to deal with the multi-product CTR prediction problem, greatly improving the prediction accuracy for niche / new products lacking training data.
- We design and implement an improved version of TrAdaBoost. Using sampling strategy and map reduce implementation, the algorithm achieves both computation efficiency and CTR prediction accuracy.
- We apply our method on real production data and demonstrate promising preliminary results.

The rest of the paper is organized as follows. In Section 2, we describe some related work in CTR prediction and transfer learning. Section 3 briefly introduces the advertising system of our Internet company and background knowledge of transfer learning. In the fourth section, we introduce our method. Section 5 shows the preliminary experiment results. Section 6 discusses limitations of our approach and concludes our work.

2 RELATED WORK

2.1 CTR Prediction

CTR prediction is of great significance for Internet companies, and thus people have applied all sorts of machine learning algorithms to improve it. Most of the work in the literature can be categorized broadly as either model development or feature development.

Regarding the model development, logistic regression [4] and decision trees [10] are the most popular models in the computational advertising literature. [5] factorizes CTR probability as two natural stochastic models to capture users' the different behaviors on different ads positions (a.k.a. positional bias).

In terms of feature development, [7] integrates multimedia features extracted from display ads into the click prediction models. [3] proposes click feedback features based on aggregated historical click data. [6] designs user specific features and demographic features.

Some work presents real industry experiences concerning CTR prediction. [18] summarizes valuable experiences, both positive and negative, at Google. [13] describes the Bayesian online learning algorithm in Microsoft's Bing search.

However, the above projects mainly focus on a single advertising product, while we are dealing with multiple products.

2.2 Transfer Learning

In transfer learning, the knowledge can be transferred among models in different ways. In addition to the instance transfer (i.e. transferring data instances) [8, 14] method we use in this paper, the other three popular types are: feature representation transfer [1, 26], parameter transfer [11, 15] and relational knowledge transfer [19, 20].

Transfer learning is also widely utilized in training deep learning models, such as the Deep Adaptation Network (DAN) [17] and the study on what kind of features in deep learning transfers well to other DNN models [24].

Given the computation cost involved in transfer learning, most of the applications only use small datasets like natural language articles or images [2, 8, 11, 26], rather than the large-scale click log data.

A few projects use transfer learning in online advertising. [22] proposes a two-stage transfer learning pipeline for targeted display advertising by transferring the prediction scores as a new feature to the target task. [16] proposes a method that constructs a joint feature space through data from multiple tasks. [9] trains a source model to get an informative prior and incorporate this Bayesian prior into the target loss function.

Our approach is different from the above work in that we directly transfer data samples from the source to the target, and also we handle a much larger data set.

3 BACKGROUND

3.1 AllianceAds System

Our AllianceAds system delivers advertisements to third-party partner websites. In a nutshell, the system keeps an inventory of ads from many advertisers and try to fill ads positions on third-party websites in real time. Separate teams have developed different *advertisement products* serving different advertiser categories, advertising positions, display formats and so on. AllianceAds system uses a real-time bidding (RTB) mechanism to decide which ad product gets a position; then it is up to the product to decide which ad to use.

Figure 2 shows the basic pipeline of AllianceAds. When a user opens a web page with an open advertising space, the web page sends a request to the Advertisement Exchange System (ADX) in AllianceAds with necessary information such as the page content, and the users' profile. ADX forwards the information to several products, who then offer bids at different prices. The product with highest bidding price wins the chance to display its ad. What ad to display is up to the product. ADX then routes the ad content from the winning product to display on the web page.

To build CTR models, we build the feature space including three feature sets: *user, advertiser* and *flow*. The feature set *user* includes the user profiles, such as the website they visit, the queries they issue and other historical behaviors. The feature set *advertiser* include the information of the advertisers, such as their brand category and so on. The feature set *flow* consists of the characteristics of the current web page, the advertisement content and other information related to the current web flow.

However, from the pipeline of AllianceAds we can find that only when the product wins in bidding, it can get the user and flow information. Due to the competition in the bidding, it is hard for the product that does not bid the highest price to get adequate data. Besides, if an advertising product is just released, it also lacks data compared to the large products that have accumulated many data. Therefore, to get accurate CTR prediction models, transferring knowledge from large products to small products is necessary.

In this paper, we present our experiment results on two products on AllianceAds. Product 1 and Product 2 are both display-ads on web pages. They bid on the same advertisement position, have similar advertising environment and have the same feature space format. However, they do have many differences that we summarize in Table 1. Note that for Product 2, we find that end-to-end prediction is not effective compared to multistage prediction. Therefore the first task of Product 2 is to predict CTR. Product 1 and Product 2 have the same feature space, but they have different distribution of the data. This scenario is what our method focuses on. In this paper, we use Product 1 as source, Product 2 as target to do transfer learning.

	Product 1 (source)	Product 2 (target)
Size	huge	10% of the former
Pricing type	cost-per-click	cost-per-action
Ad format	image, Flash, text	Flash, image

Table 1: Differences of the two products. We use Product 1 as source, Product 2 as target.

3.2 Transfer Learning

Based on [21], we here give the definition of transfer learning. A domain $\mathcal{D}$ consists of two components: a feature space $\mathcal{X}$ and a data distribution $P(X)$, where $X = \{x_i\}_{i=1}^n$, and $x_i \in \mathcal{X}$. Namely, $\mathcal{D} = \{\mathcal{X}, P(X)\}$. We use x_i to denote the data instance used in machine learning algorithms. A task $\mathcal{T}$ also consists of two components: a label space $\mathcal{Y}$ and a prediction function $h(\cdot)$. In other words, $\mathcal{T} = \{\mathcal{Y}, h(\cdot)\}$. $\mathcal{Y}$

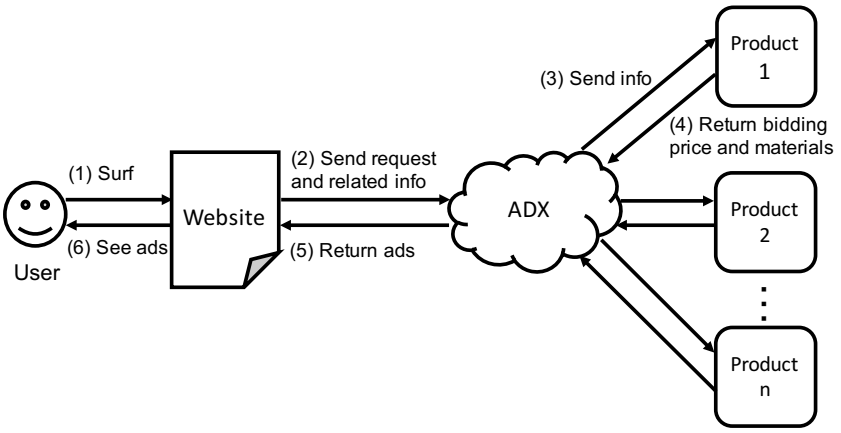

Figure 2: The workflow of AllianceAds. User opens a web page, and AllianceAds chooses ads to show to the user.

describes the label format in a machine learning problem, e.g., *true* or *false* in the binary classification problem. $h(\cdot)$ is the machine learning prediction function, the output of which is the predicted label, namely $h(x_i) = \hat{y}_i$.

Given the notations above, we then define transfer learning. Transfer learning is the algorithm that aims to transfer knowledge from source to target. Given a source domain $\mathcal{D}^S$ and a source task $\mathcal{T}^S$, a target domain $\mathcal{D}^T$ and a target task $\mathcal{T}^T$, transfer learning is the algorithm that can improve $\mathcal{T}^T$ in $\mathcal{D}^T$ with the knowledge in $\mathcal{D}^S$ and $\mathcal{T}^S$, where $\mathcal{D}^S \neq \mathcal{D}^T$ or $\mathcal{T}^S \neq \mathcal{T}^T$. For example, some source and target may have different feature space and different data distributions. Some may have different label spaces.

4 OUR APPROACH

4.1 Problem Formulation

In our task, we want to use the data from the source product to improve the target product in CTR prediction. CTR is the ratio of the number of clicks on a certain advertisement link to the number of total views of this advertisement. Namely, $CTR = click/show$. The value of CTR ranges between 0 to 1. In our dataset, each data instance is labeled with the number of *show* and *click*. We have a set of online advertisement data $D = \{(x_i, y_i)\}_{i=1}^{n+m}$, where $x_i \in \{0,1\}^d$, $y_i \in [0,1]$ and d is a very large number over ten million. This d-dimensional feature space includes *user, advertiser* and *flow* information, which has been described in Section 3. The feature space is one-hot encoded and only hundreds of them are ones and most of them are zero. In a word, our data is large-scale, sparse and one-hot encoded. CTR prediction task is to predict the click-through rate y_i given an advertisement data x_i, namely to get $h(x_i) = \hat{y}_i$

For transfer learning, we assume there are n source data instances and m target data instances. We let $i = 1$ to n to represents the index of the source data while $i = n + 1$ to $n + m$ to represent the index of the target data. We add a S or a T in the upper right of the existing notations to denote whether it belongs to source or target. Our source data $D^S = \{(x_i, y_i)\}_{i=1}^{n}$ and target data $D^T = \{(x_i, y_i)\}_{i=n+1}^{n+m}$ are from Product 1 and Product 2 respectively. The two datasets have the same feature space but different data distribution. Therefore our transfer learning problem is formalized as follows.

$$\mathcal{X}^S = \mathcal{X}^T \subseteq \{0,1\}^d, \tag{1}$$

$$P(X^S) \neq P(X^T), \tag{2}$$

$$\mathcal{Y}^S = \mathcal{Y}^T \subseteq [0,1]. \tag{3}$$

Our goal is to get $h(\cdot)^T$ to predict accurate CTR.

4.2 Model

Our idea is similar to TrAdaBoost [8]. Compared to TrAdaBoost, we made the following changes to fit our problem setting. 1) We add a sampling strategy to sample the source data. 2) We use logistic regression as our learning model, with L1 regularization. 3) We use AdaGrad as our optimization algorithm. 4) We modify the model ensemble method. 5) We implement our approach in a MapReduce-like machine learning framework inside our company, which makes this algorithm run in more than one hundred computing nodes.

In each iteration, we first sample some source data instances. Since the source data is usually very large, it is unnecessary to use every source data in each iteration. So we design a sampling strategy to sample the source data. The sampling probability is proportional to the data's gradient in the trained model [25]. In the t-th iteration, the sampling probability is based on the $t-1$-th model's gradient information. The larger the gradient is, the larger parameter updates the model will get, which means the model needs

this source data instance more than others. Therefore, the sampling probability of this instance is larger. Before the first iteration, there is a pre-training target model to help on the first sampling. Let $L_i(\theta)^t$ denote the loss of logistic regression on data instance x_i in the t-th iteration, θ is the model parameter. The sampling probability of a source data instance x_i in the t-th iteration is

$$P_{samp}(x_i)^t = \min\{1, \alpha * |\nabla L_i(\theta)^{t-1}|\}, 1 \le i \le n, \quad (4)$$

where α is a hyper parameter that we can set, and $\nabla L_i(\theta)^{t-1}$ denotes the source data instance x_i's gradient in the trained model in iteration $t-1$. The larger $\alpha * |\nabla L_i(\theta)^{t-1}|$ is, the more the trained model needs this source data instance.

After sampling, we combine these sampled source data and target data in training. For sampled source data, we use importance weighting to eliminate the bias [25] during the training process. For a target data instance, we will not do revision. But if there comes a source data, we use Eq.(5) to revise the gradient value.

$$\tilde{\nabla} L_i(\theta)^t = \frac{\nabla L_i(\theta)^t}{P_{samp}(x_i)^t}, 1 \le i \le n, \quad (5)$$

where $\tilde{\nabla} L_i(\theta)^t$ is the gradient after the revision. In this way, we modify the gradient value by multiplying $1/P_{samp}(x_i)$.

After the training process, we get the t-th model $h_t(\cdot)$. We calculate the training error on target data as

$$\epsilon_t = \frac{\sum_{i=n+1}^{n+m} w_i^t * |h_t(x_i) - y_i|}{\sum_{i=n+1}^{n+m} w_i^t}, \quad (6)$$

where w_i^t is the weight of the i-th data instance in the t-th iteration, and $h_t(x)$ is the prediction function in the t-th iteration. Initially, we set every data weight to 1.

Then, we use Eq.(7) to reweight the data [8]. If an instance is correctly classified by the model, its weight will not change. If a target data is misclassified, we increase its weight to make it contribute more in the next iteration. If a source data is misclassified, we decrease its data weight since we think it is less similar to target data.

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i)-y_i|}, 1 \le i \le n, \\ w_i^t \beta_t^{-|h_t(x_i)-y_i|}, n+1 \le i \le n+m, \end{cases} \quad (7)$$

where β and β_t is a parameter that ranges between 0 to 1. In each iteration, β and β_t [12] are set to

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t}, \quad (8)$$

$$\beta = \frac{1}{1+\sqrt{\frac{2\ln n}{N}}}. \quad (9)$$

Lastly, we output the ensemble model. As [8] proves, if the algorithm runs for N iterations, the average weighted training loss on source data from the $\lceil N/2 \rceil^{th}$ iteration to the N^{th} iteration converges to zero. Therefore, we do model ensemble from the $\lceil N/2 \rceil^{th}$ model to the N^{th} model. For example, if we run the algorithm for 10 iterations, we will do ensemble from the 5-th model to the 10-th model and output the ensemble model. The prediction function of ensemble model is as follows.

$$h(x) = \frac{-\sum_{t=\lceil N/2 \rceil}^{N} h_t(x) \ln \beta_t}{\sum_{t=\lceil N/2 \rceil}^{N} \ln \beta_t^{-1}}. \quad (10)$$

This prediction function is modified from [8] in order to fit our problem settings. The function value ranges from 0 to 1, indicating the probability that the user clicks the advertisement.

Algorithm 1 shows the pseudo code of our method.

Algorithm 1 Transfer Learning Algorithm

Input: Data: source data $D^S = \{(x_i, y_i)\}_{i=1}^{n}$, target data $D^T = \{(x_i, y_i)\}_{i=n+1}^{n+m}$, where n > m
Hyper-parameter: sample parameter α, learning rate γ, total iteration N, L1 regularization parameter λ, mini batch size s

```
Pre-train a model on target data.
for t = 1 to N do
    D̃^S = sample source data according to Eq.(4)
    h_t = train(D̃^S, D^T)
    Calculate the error on target data according to Eq.(6).
    if ε_t > 0.5 then
        break;
    end if
    Set β_t and β according to Eq.(8) (9)
    Update data weights according to Eq.(7)
end for
```

Output: Output the model according to Eq.(10)

5 EXPERIMENT

5.1 Datasets

We do experiments in the real online advertisement data from Baidu. Table 2 lists the sizes of our datasets. These data are just small samples of Product 1 and Product 2. The ratio of source size to target size is the same as the original ratio of Product 1 size to Product 2 size. We randomly choose 10% of the training set as the validation set. Every experiment result listed in this paper is the average number of three running results, varying the random data splits.

We implement our algorithm on an internal MapReduce-like machine learning framework in our company. This

framework provides user-friendly APIs and an efficient running environment. During the experiment, we use 100 computing nodes.

To measure the performance of the methods in CTR prediction, we use AUC, the area under the ROC curve. ROC curve is a plot that illustrates the true positive rate against the false positive rate of a binary classifier, as the discrimination threshold varies. AUC is the area under this curve, ranging from 0 to 1. The larger AUC indicates that the method has a low false positive rate and a high true positive rate at a certain threshold. When AUC=1, the method has a threshold that makes the false positive rate equal 0 and the true positive rate equal 1. Thus, the larger the AUC is, the more accurate the prediction is. An improvement in AUC means more accurate advertisement to users, thus helping the Internet company make more profit.

Dataset	# of data instances
Source	135,187,742
Target-training	11,703,741
Target-validation	1,300,315
Target-testing	1,705,122

Table 2: Data Size. Our approach uses sampled source and target-training.

5.2 Results

We now describe the results of our method. In our experiment, we use pure supervised learning method logistic regression (LR) as our comparison method. We run LR on different training data. We use LR_T to denote that we train logistic regression only on target data. LR_S means we train logistic regression only on source data. LR_{ST} means we directly combine source data and target data for LR training. Note that the test data in these different settings is the same set of target data *Target-testing*, as Table 2 shows.

We carefully tuned the hyperparameters to get the optimal combination. Our best hyperparameters are as follows: $N = 6, \gamma = 0.01, \lambda = 0.1, \alpha = 2, s = 100,000$. The experiment results are showed in Table 3. The first column of the table indicates different method names. The second column shows the training data we use. The third column shows testing AUC.

From LR_T and LR_S we can see that the model trained on source has very low AUC on the target testing data, and we believe it is due to the fact that source and target have very different distribution. LR_T and LR_{ST} together show that directly combining source data and target data is not useful to improve AUC. The last row shows our method outperforms the above traditional supervised learning method. Besides, our method has slightly better AUC of TrAdaboost, and greatly reduces training time due to the sampling strategy. Our sampling strategy automatically selects the data instances the model needs, while TrAdaboost only uses all the data it has, thus making our approach performs better. Besides, the total running time of TrAdaboost is about 220 minutes, while our approach runs only for about 70 minutes due to the sampling strategy. Our approach saves 68.2% time compared to TrAdaboost. This time saving is very important to scale the algorithm to very large datasets that is common in the online advertising scenario.

Method	Training data	AUC
LR_T	Target-training	0.7123
LR_S	Source	0.6513
LR_{ST}	Source + Target-training	0.7069
TrAdaboost [8]	Source + Target-training	0.7301
Our method	**Source + Target-training**	**0.7341**

Table 3: Experiment Results. Our method has better AUC and less training time.

5.3 Parameter Sensitivity

In this section, we discuss about the sensitivity of the hyperparameters of our algorithm. We mainly focus on the total iteration N and the sampling parameter α, since the two are special parameters in our algorithm, while the other hyperparameters work similar in our algorithm compared to other machine learning algorithms.

The Total Iterations N. The parameter N controls the total number of iterations in our algorithm. Since our algorithm outputs the ensemble model according to the iteration, so N also controls the number of ensemble models. We run our algorithm using different N and record the AUC as Figure 3 shows. The blue curve is our method, while the red dashed line means LR_T. Since the method LR_T is not affected by N, this line is horizontal. We can see from the figure that the AUC firstly increases and then decreases, and reaches the maximum at $N = 6$. When $N = 6$, the algorithm uses $\lceil 6/2 \rceil = 3$ to $N = 6$, altogether 4 models to do ensemble. The current experiment shows this is the most suitable choice in our problem settings. We think when N is small, the number of ensemble model is too small so the algorithm does not work well. When N is large, the number of ensemble model is too large so that the algorithm tends to overfit. So it is better to select an N that is neither too small nor too large.

The Sampling Parameter α. The parameter α controls the sampling probability of the source data. We run our algorithm using different α and Figure 4 shows the results.

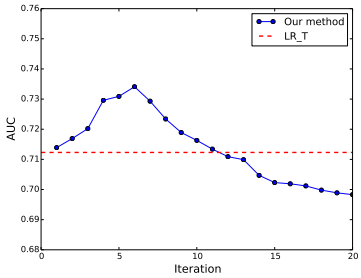

Figure 3: AUC-iteration curve. As iteration increases, AUC firstly increases and then decreases.

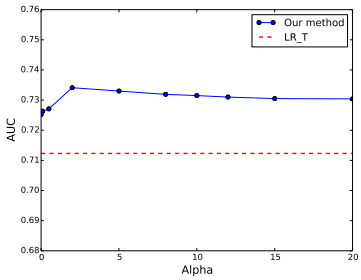

Figure 4: AUC-alpha curve. Source is helpful to improve AUC, but too many source data will not help.

LR_T is a horizontal line since the method LR_T only uses target data, not affected by α. The figure indicates that source data help to improve AUC compared to only using target data ($\alpha = 0$). However, sampling every source data instance ($\alpha = 20$) may not be very useful since target and source have very different data distribution. In our experiment the best choice is $\alpha = 2$. In this setting the sampling ratio, namely $\#\ sampled\ source/\#\ total\ source$ is about 0.113. The ratio of $\#\ target/\#\ sampled\ source$ is about 0.85. In our algorithm, each iteration has different sizes of sampling source data, so we only count the number in the first iteration to calculate the sampling ratio. If α is zero, no source data will be used. The algorithm only uses target data and do model ensemble so this algorithm will become somewhat similar to Adaboost [12]. If α is too large, the probability value will be larger then 1. Thus, every source data instance will be sampled. In this way, the algorithm becomes TrAdaboost. We choose an α that is between the two extremes as a good practical tradeoff.

5.4 Data Size Ratio

We will talk about the ratio of source data size to target data size in this subsection. We fix the target data size according to $Target - training$ in Table 2 and vary the source data size. Since we have discussed the sampling ratio in the previous

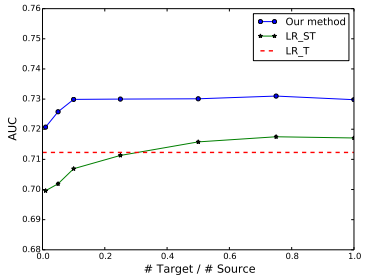

Figure 5: Data size ratio. It is necessary to carefully adjust the data size ratio instead of over utilizing the source data.

subsection, we run this experiment with *all* the source data to focus on the data size ratio. We compare our method with LR_{ST} and LR_T. Figure 5 shows the results. The blue curve is our method, while the green curve means the method LR_{ST}. LR_T is horizontal since we fix the target data size. In the figure, the starting point of is $\#target/\#source = 0.01$, which means source data is 100 times the size of target data. The ending point is 1 since in our settings Product 1 is much larger than Product 2, so the ratio large than 1 is meaningless. It is also the case in most transfer learning problems.

When the ratio is 0.01, we can see that AUC is very low even compared to LR_T. It is because in our datasets, source and target have very different distributions. Using too much source data will hurt the training data quality, thus making the AUC drop greatly. As the ratio increases, namely the source data decreases, AUC increases gradually. When the ratio reaches around 0.8, our method reaches the max AUC. Meanwhile, the LR_{ST}'s AUC keeps increasing gradually, but with a decreasing speed. If we use too few source data, then AUC will start to drop. If the source size is zero, we can imagine the AUC of the two curves will get close to the line LR_T.

In real cases, the data size is often given. One can adjust the data size ratio though our sampling parameter α. Manually selecting data amount is possible but not convenient. People often do not know which part of the data they should use. Our sampling strategy makes this choice fully automatic. Based on a given α, the sampling strategy can automatically select the data that the model needs. Different problems and different datasets may have different optimal points, but we believe that it is necessary to carefully adjust the data size ratio instead of over utilizing the source data.

6 DISCUSSIONS AND CONCLUSIONS

Online advertising is always a hot topic in Internet companies. Due to the bidding mechanism, the data size of different products varies much. To improve the CTR prediction of

the small product (target) with the help of the large product (source), we propose an iterative transfer learning method. Our sampling strategy and MapReduce-like implementation make the algorithm more scalable. We do experiment on real advertisement data and the results show that our approach improves AUC of CTR prediction compared to the supervised learning method.

While our experiment shows good preliminary results, there are some limitations in our approach. Firstly, the current version of sampling strategy only uses the information of gradient. This strategy selects the source data that are most needed by the trained model. We still need to figure out what sampling strategy is the best for improving AUC and reducing training time. Also, we do not take the sparsity of the advertisement data into consideration, which we believe can further improve the algorithm performance. Besides, our approach only uses one source and one target. Actually there are multiple products in the company. How to efficiently do *multiple-source* transfer is challenging. In addition, we use L1 regularization in our current approach to avoid overfitting, and we will explore other techniques to reduce overfitting.

Our approach shows a promising direction. The current algorithm is used only in CTR prediction. Actually, it can also be used in ACP (average click price) prediction, which is another important topic in online advertising. In our company, the ACP dataset has a similar format of CTR dataset. Thus, we are also going to adapt to ACP prediction.

ACKNOWLEDGMENTS

This research is supported in part by the National Natural Science Foundation of China (NSFC) Grant 61532001, Tsinghua Initiative Research Program Grant 20151080475, MOE Online Education Research Center (Quantong Fund) Grant 2017ZD203, and gift funds from Huawei and Ant Financial. Thanks to Baidu Inc. for providing the datasets.

REFERENCES

[1] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. *Advances in Neural Information Processing Systems*, 19:41, 2007.

[2] J. Blitzer, M. Dredze, F. Pereira, et al. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Annual Meeting of the Association of Computational Linguistics*, volume 7, pages 440–447, 2007.

[3] D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. In *International Conference on World Wide Web*, pages 417–426. ACM, 2008.

[4] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology*, 5(4):61, 2015.

[5] Y. Chen and T. W. Yan. Position-normalized click prediction in search advertising. In *International Conference on Knowledge Discovery and Data Mining*, pages 795–803. ACM, 2012.

[6] H. Cheng and E. Cantú-Paz. Personalized click prediction in sponsored search. In *International Conference on Web Search and Data Mining*, pages 351–360. ACM, 2010.

[7] H. Cheng, R. v. Zwol, J. Azimi, E. Manavoglu, R. Zhang, Y. Zhou, and V. Navalpakkam. Multimedia features for click prediction of new ads in display advertising. In *Iinternational Conference on Knowledge Discovery and Data Mining*, pages 777–785. ACM, 2012.

[8] W. Dai, Q. Yang, G. R. Xue, and Y. Yu. Boosting for transfer learning. In *International Conference on Machine Learning*, 2007.

[9] B. Dalessandro, D. Chen, T. Raeder, C. Perlich, M. Han Williams, and F. Provost. Scalable hands-free transfer learning for online advertising. In *International Conference on Knowledge Discovery and Data Mining*, 2014.

[10] K. S. Dave and V. Varma. Learning the click-through rate for rare/new ads from similar ads. In *International Conference on Research and Development in Information Retrieval*, pages 897–898. ACM, 2010.

[11] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *International Conference on Knowledge Discovery and Data Mining*, 2004.

[12] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of online learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37. Springer, 1995.

[13] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *International Conference on Machine Learning*, pages 13–20, 2010.

[14] J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *Annual Meeting of the Association of Computational Linguistics*, volume 7, pages 264–271, 2007.

[15] N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In *International Conference on Machine Learning*, page 65. ACM, 2004.

[16] Y. Liu, S. Pandey, D. Agarwal, and V. Josifovski. Finding the right consumer: optimizing for conversion in display advertising campaigns. In *International Conference on Web Search and Data Mining*, 2012.

[17] M. Long and J. Wang. Learning transferable features with deep adaptation networks. *CoRR, abs/1502.02791*, 1:2, 2015.

[18] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *International Conference on Knowledge Discovery and Data Mining*, pages 1222–1230. ACM, 2013.

[19] L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *AAAI Conference on Artificial Intelligence*, pages 608–614, 2010.

[20] L. Mihalkova and R. J. Mooney. Transfer learning by mapping with minimal target data. In *AAAI-08 Workshop on Transfer Learning for Complex Tasks*, 2008.

[21] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[22] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost. Machine learning for targeted display advertising: Transfer learning in action. *Machine learning*, 95(1):103–127, 2014.

[23] M. Regelson and D. Fain. Predicting click-through rate using keyword clusters. In *Proceedings of the Second Workshop on Sponsored Search Auctions*, volume 9623, 2006.

[24] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[25] P. Zhao and T. Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *International Conference on Machine Learning*, pages 1–9, 2015.

[26] Y. Zhu, Y. Chen, Z. Lu, S. J. Pan, G. R. Xue, Y. Yu, and Q. Yang. Heterogeneous transfer learning for image classification. In *AAAI Conference on Artificial Intelligence*, pages 1304–1309, 2011.