Position-and-Depth Normalized Click-Through-Rate Prediction in Search Advertising

PATRICK KLAUSBAGINSKI*

CORNELL UNIVERSITY – CORNELL TECH – Contact: pkb42@cornell.edu

April, 30th, 2017 – Using the KDD Cup 2012 Track 2 Dataset from Tencent modified under Apache Spark 2.1.0 (PySpark) to build an accurate classification model for the Search Advertising sector

ABSTRACT: This paper covers ways to apply machine learning to very large data sets in the Search Advertising sector using Apache Spark 2.1.0 on a high-performance Amazon AWS Elastic-Cloud-Compute 2 cluster. It describes comprehensively the data scientific process including Extract-Transform-Load, feature engineering and machine learning pipeline preparation. It explores the concept of normalization of Click-Through-Rates based on Depth and Position factors and then proceeds to conduct a machine learning experiment with the Kaggle KDD Cup 2012 Track 2 dataset from Tencent.

1 Introduction

The Tencent KDD Cup 2012 data set is a large data set consisting of a training set and test set, the first with more than 155 million instances and the second with over 20 million instances. It has been used as a basis for various other research papers in the machine learning space [1:6]. The Kaggle description prompts the competitors to predict the Click-Through-Rate used for ranking of ad's in search advertising. Accurate ranking and determination of ad positioning is one of the most important revenue drivers for companies such as Google, Facebook and Baidu. Despite its age, the problem stated in the KDD Cup 2012 prompt remains an area for research even in present time. Each instance in the data sets contains information about the User, Ad, Query (what was searched for by the user who had been shown the ad), the amount of times and ad was shown in a session (# of Impressions), how many times it was clicked (# of Clicks) and where it was positioned (Position) amongst the other ads shown in the session (Depth). Additionally, there are adjacent data sets containing information about the exact ad text, the advertising company, the website it links to and the user's age and gender. Overall, the training set contains 11 features and one output variable (clicks). Further information on the competition and the data set is available here [7]. The measure of performance for the competition is the area under the ROC curve (AUC) and this paper adopts this practice. Although AUC is a measure known from the space of classification and not prediction [8], it can also be beneficially applied to a prediction problem. This is done by dividing each instance of the training set into positive samples (clicked ad) and negative samples (impressed but not clicked ad) [2]. During the original competition, high-ranked teams demonstrated prediction performances above 0.80 AUC. After a few years have passed and with the help of AWS EC2 clusters and Apache Spark, this paper shows how the improvements in tools and performance can benefit this old problem.

This paper is structured based on the typical data science process as suggested by Farcaster [9]: The second section describes the data processing of the Tencent data, the third section describes how in search advertising click-through-rates should be normalized according to both, the positioning of the ad and the number of ads shown (depth). The fourth section uses exploratory analysis to gain a rudimentary understanding of the data, the fifth section describe the experiment setup for the machine learning model and the sixth section describes the outcomes and results of the experiment.

2 Processing the training and test set

To be able to handle the amount of data contained in the KDD Cup 2012, the first step is to set-up appropriate infrastructure for storage, execution and exporting of results. In the case at hand the selection was to set up an Apache Spark 2.1.0 (scala-11, auto-updating) cluster using AWS EC2 Instances and

AWS S3 Storage. Attached to this infrastructure is the Databricks notebook service. To set-up, all that is required is to enter the AWS Console, load the files onto the S3 bucket and launch the EC2 instances. In the next step a notebook is opened in the Databricks workspace. The code then specifies all encrypted log-in and access information regarding AWS. Lastly, the cluster with the EC2 instances is attached to the Databricks notebook and the S3 bucket is mounted. For this exercise, eight instances were used: Memory-optimized EC2 r3.xlarge's with each 4 cores and 30.5 GB of memory split into one Driver and seven On-Demand Nodes.

Once the training and test set data is loaded, the amount of data becomes imminent. The full training data set contains over 150 million instances, the test set over 20 million instances. The next step is to format and process the training set and test set.

The first step in processing the data set is to identify column names and formats. In the present case, we identify 11 feature columns and 1 output column (clicks). The initial dataset comes without annotation on which column refers to which feature, however it is expected that the data description on the Kaggle competition lists the contents of the dataset in order. This also goes in hand with the winning solution for the Kaggle competition [2]. The next step is to create a SparkSQL view for further processing of the data sets. The data sets contain different identifiers of which one is UserID. However, both the training set and test set contain many instances where the UserID is "0", equivalent for cases where the user was not identifiable. To handle this for future modeling purposes, the data sets are split into two sets each, one covering the instances with identified Users and one covering instances where the User has not been identified. This is a means of controlling for bias in the data set given that the click-through-rates (going forward, "CTR") can largely differ between those two types, as has also been observed in other team's work [2]. Another point is to condense the data by grouping it by a selection of identifiers. It has to be mentioned here however that the condensing is not the "unique" combination of identifiers for the dataset [3]. The unused identifier to create uniqueness is "DescriptionID" which is considered a part of setting (Depth, Position, DescriptionID) of an AdID, queried by the UserID. In the present case, we continue grouping the data sets for condensing purposes based on the assignment request. This grouping is done in the following way:

1. Select identifiers (UserID, AdID, QueryID, Position, Depth)
2. Sum Impressions and Clicks
3. Aggregate the data set using the select statement

This step however introduces selection bias into the data set because some features are dropped, such as additional information that can be pulled for the advertiser, description, keywords and title. The decision to proceed without these features is based on the intuition that the paper wants to classify good ads from bad ads purely based on their core-information and filtering out the skills and performance of certain advertiser (who constantly optimize their ad's) or user (who constantly optimize their search terms).

In former applications of the problem, the data set has been further reduced due to performance constraints. Although in the present case it has been decided to work on the full data sets, the following operations have to be followed to further reduce data set size and select the top 25.000 instances based on AdID and QueryID:

1. Group the training set based on AdID and QueryID
2. Sum up impressions as a measure of frequency (Alternately, count the occurrence of AdID/QueryID groups)
3. Select the instances from the top AdID/QueryID groups until 25.000 instances are reached.

As mentioned before this approach has been disregarded. The main reasons for this is that the combinations of AdID/QueryID in themselves contain so many instances, that merely one single combination would be chosen to work with the top 25.000 instances. That seems counter-intuitive and not appropriate given the prior introduction of selection bias to reduce the data set size. This effect holds true for both of above interpretations of "frequency".

3 Computing the normalized click-through-rates

The last step in preparing the data sets for later usage in the ML pipeline is to compute the CTR's. The following process is applied to the data sets:

1. CTR is computed for all identifiers present and combinations of those identifiers. This leads to a total number of single-identifiers of five (UserID, AdID, QueryID, Depth, Position) and conjunction identifiers of five (Combinations of the above including a Depth-Position combination).

2. Compute # of Clicks divided by # of Impressions for each of the above identifiers, known as CTR
3. Compute the normalization on all of the above calculated CTR's.
4. Annotate each of the normalized CTR's to the training set and to the test set.

One of the conjoined identifiers is "Depth-Position". This has been used by other researchers in the field [2]. The formula used to compute this Depth-Position feature is:

$$DepthPosition = \frac{(\text{Depth} - \text{Position})}{\text{Depth}}$$

The normalization of CTR's requested in the assignment is based on depth and position. In other research papers, it has been stated that it is common practice in the industry to normalize for position [4]. This also seems to be intuitive as an Ad that is placed higher should accumulate more clicks than one that is lower. Generally, the way that CTR is normalized based on position in the area is based on the following [5]:

$$norm.CTR = \frac{\text{Clicks}}{\text{Searches} * \text{Position effect}}$$

Where "position effect" is equal to the rank of an ad and its position. Rank can be interpreted as a measure of frequency and quality of the ad. Following in this logic it is possible to compute a depth and position normalized CTR, as also stated in other research [4] that suggests such a depiction can be used as an indicative feature for machine learning purposes. When using "Depth" as a measure for frequency/quality of the Ad, the formula becomes:

$$norm.CTR = \frac{\text{Clicks}}{\text{Searches} * (\frac{\text{Position}}{\text{Depth}})}$$

This translates to:

$$norm.CTR = \frac{\text{Clicks} * \text{Depth}}{\text{Searches} * \text{Position}}$$

This normalization method is then applied to every CTR feature in the data sets and this process is called annotation. The final product is then a training set and a test set consisting of the following:

1. Identifiers (UserID, AdID, QueryID, Position, Depth, Depth-Position).
2. Normalized Click-Through Rates (All IDs, UserAd, AdQuery, UserQuery, User, Ad, Query, Position, Depth, Depth-Position).

4 Final shaping of the training data and exploratory analysis

Before proceeding to the experiment setup, the final shape and form of the data sets is chosen and analyzed using summary statistics, with special attention paid to the variance of the features in the data set.

The final shape of the training data is set by applying three trade-offs:

1. Training of the model will be done using instances with clicks only and identifiable users only.
2. Each instance on this data set is then split into positive instances (# Clicks) and negative instances (# Impressions – # Clicks).

The first trade-off increases the bias of the CTR prediction. Essentially, the experiment analyzes the performance of click-through-rate prediction and ads based on a sample of users who have been actively engaged with ads and who have been identified in gender and age by Tencent. A much larger population of inactive and unidentified users exists, however for the purpose of AUC optimization the analysis of the unidentified user population is subject to other research [2]. An important requirement for such analysis is the use of time-stamped data, which in the case of the Tencent data set can only be assumed, as further elaborated on in section 5. The exploratory analysis of the data set shows that overall there are over 7 million such instances to be trained on, after splitting the training data into 90% model training set and 10% validation data set. The choice of this kind of split is based on research in [2] showing that different splits have not shown better performance in the models. The summary statistics reveal a representative split between positive and negative samples and normalized summary statistics. The variance of the features is moderate. This problem is further addressed in section 5. Overall, the training set shows that 5.69 million Ad's were clicked and 1.95 million times impressed but not clicked. This can be explained given that the training set only considers instances with clicks.

Summary	Clicks	Impressions	label	nctr_all
count	7641254	7641254	7641254	7641254
mean	1.4789	5.4588	0.7452	1.1217
stddev	4.6349	40.9909	0.4357	0.7432
min	1.00	1.00	0	0
max	299	1906	1	3

5 Machine learning experiment setup

In the present case, the training and validation set are largely described by the following characteristics:

1. Large scale
2. 16 Features of which:
3. 6 categorical features (identifiers) and
4. 10 continuous features (normalized click-through-rates)
5. Selection bias is present due to prior reduction of data set based on identifiers
6. Bias due to focus on instances with identifiable users as addressed in section 4

Referring to vast research and experiments done in this area, the choice of model for this assignment will fall onto a linear model with few parametrization such as Logistic Regression or Naïve Bayes. In both cases, research has shown the strong performance of these models when using AUC and ROC as evaluation metrics [2]. Other research has shown that similar models are equally well performing [6]. The main reasoning behind choosing this type of model is due to the Bias-Variance trade-off. Although linear models introduce a more biased approach to the training process, they tend to generalize well across the data set. Particularly in large data sets like the one present, where variance of features is high, this is a well counter-balancing way of handling the data set. To shape the data for this type of model, the use of unique combinations as row values (UserID, AdID, QueryID, Position, Depth, Position-Depth) and the use of the normalized CTR's as vector values is conducted. The dependent variable to be predicted will be "clicks" based on the independent CTR's.

It is suggested to use a logistic regression based on the L-BFGS solver and using L2 regularization as a means to counter-act the moderate variance of the features. However, in the present experiment we use PySpark ML classification package "LogisticRegression", which comes without the regularization term and uses SGD as a basis. However, to balance this out, the logistic regression will be wrapped into a cross-validation model using 5-fold cross validation and iterating through the logistic regression parameters as shown below. The reason for this general setup is to balance the high variance in the data set. Through the application of AWS EC2 and Apache Spark, this set up on such a large data set has become efficient and economical for businesses to use, as in 2012 5-fold cross-validation and parameter-screening wasn't as economic. The models use the following parameters for the exercise: regParam = 0.5 and maxIter = 5.

The modeling exercise is repeated twice to show the impact of the Bias-Variance trade-off. The first model trained will be using less features. In this first model only the normalized click-through-rates are used for classification, which introduces a highly-biased approach. In the second version of the model ID features are added as numerical features: UserID, AdID, QueryID. The reason for this is that it seems these identifiers contain some information regarding the timing of the data set and instances. Additional numerical features are added: Depth, Position and Depth-Position. Lastly, the features user age and user gender are added. Altogether, this selection of features proves to be the most complete approach in terms of understanding the data set and reducing bias.

6 Experiment outcomes and model performances based on AUC

Logistic regression and other parametric, linear models are often used by machine learning researchers due to their good generalization of the data at hand. The great performance of linear models is often only shadowed by their biases. The experiment present is another demonstration of this. The performance based on AUC of both the first and second model is higher than that of the winning team of the KDD Cup 2012 competition. This is due to two factors, one being the different approach on the data preparation the other being the increased performance due to the usage of Spark and AWS. The first model setup is slightly over-generalized given the moderate-variance input of the features. In the second model this slightly changes through the addition of new features that have less variance, however generally adding features is a means of deviating bias. Full accounting for variance would be to use regularization terms. The error analysis of the first model shows there have been several instances classified incorrectly. The second model improves this significantly. To conclude, in using Apache Spark and AWS clusters, the model performance can be increased significantly as well as training time can be reduced significantly. The model training time on this experiment was about 20 minutes on the described cluster.

Version 1 Biased-Model	AUC	True Positives	False Positives
Training data	0.931703	5,125,600	393,946
Validation data	0.932085	568,656	43,373

Version 2 - Added Features	AUC	True Positives	False Positives
Training data	0.92972	5,125,980	396,480
Validation data	0.93021	568,276	43,819

ASSOCIATED CONTENT

In addition to this document there are three more files. One is a working PySpark code example conducting the above analysis. The other ones are two .csv files, one a snippet of the computed training set and one of the computed test set. For the purpose of publicizing the code it has been uploaded to a github repository whose link is below.

REFERENCES

[1] Yui, Makoto. KDDCup 2012 track 2 CTR prediction dataset. In *github*, 2014.

[2] National Taiwan University. A Two-Stage Ensemble of Diverse Models for Advertisement Ranking in KDD Cup 2012. In *University Press*, 2012

[3] User *Mingot*. Data aggregation and primary key. In *Kaggle KDD Cup 2012 discusson forum*, 2012

[4] Rodnitzky, David. Is Click-Through-Rate normalized by position? Of course it is!. In *3QDigital Blog*, 2011

[5] Lahaie, Sebastien and McAfee, Preston. Efficient Ranking in Sponsored Search. In *Private Website of the Authors*, Unknown

[6] Purpura, Stephen and others. Private release in personal website, 2012

[7] Tencent. Kaggle Competition Description, in *KDD Cup 2012 Track 2*, 2012

[8] Fawcett, Tom. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. In *HP Research Own-Release*, 2003

[9] Farcaster. What is Data Science and What Does a Data Scientist Do?. In *KDnuggets,* 2017

Github Repository with PySpark Code Example:

https://github.com/pbaginski-cornelltech/CS5304---Data-Science-in-the-Wild/blob/master/Position-Depth-Normalized%20Click-Through-Rate