Outbrain Click Prediction

Naveenkumar Ramaraju
Indiana Univeristy, Bloomington
nramaraj@iu.edu

Saketh Pilli
Indiana Univeristy, Bloomington
sakpilli@indiana.edu

Keshav Sridhar
Indiana Univeristy, Bloomington
ksridhar@iu.edu

December 15, 2016
All the work herein is solely ours.

Abstract

Currently, we live in an information age where digital advertising is inevitable for businesses who want to reach out to the masses. One of the key forms of digital advertising is online advertising wherein advertisements are posted on various articles and documents on the internet. It's a cost efficient way to reach out to a specific target audience by advertising on a specific type of online content.

Personalized advertising is very common today where advertisements are tailored to meet the interests of individual users. By displaying the advertisement that an user is interested in and likely to click on, online content providers can increase their profit. In this project, we are trying to predict which advertisement is most likely to be clicked using the Outbrain dataset from Kaggle.

1 Introduction

In this digital information era where Internet of Things is ubiquitous and a vast majority people use online platforms for almost everything from reading news, entertainment and connecting to friends, the entities whose core business is providing online content or such a platform, run their business either through donations, subscription charges, advertisements or a combination of these. Using subscription charges could reduce the user base of the website and is not an ideal approach. Donations sound a reasonable approach for non profit organizations but is not suited for other kinds of entities. Thus advertisements are the most common way to make a business profitable by having a large user base and gaining more popularity.

Online advertising are done in two ways. First one is traditional "pay per display" where advertisers are charged based on number of users to which the advertisement is displayed. The most popular approach is "pay per click" where advertisers are charged based on number of times the advertisement was clicked. Advertisers don't have to pay if the users did not click their advertisements and have to pay only for successful advertisements. In order to maximize the profit, online content providers have to ensure that more number of advertisements are clicked by the users. Personal recommendation system to the rescue.

Personalized advertisement is the approach of displaying advertisements that would interest users and is more likely to be clicked. Personalized advertisement is an win-win scenario as users are happy that they are clicking the advertisements of their interest, advertisers are happy as they have to pay only for the successful advertisements and the success rate is more with ideal personalizations. And finally the online platform that hosts the content also wins by making more profit resulting out of increased number of clicks due to personalization.

In this project we used data provided by Outbrain, the web's leading content discovery platform, which delivers the advertisements while we surf our favorite sites. In order for an online content provider to gain more profit, they have to advertise the advert that is most likely to be clicked. To find the likelihood of an advertisement to be clicked, we can use contextual information like, web page where advertisement is displayed, user details like location and device used to access the content.

Since we are trying to find the likelihood of an advertisement to be clicked, probabilistic approaches are natural choice for this problem. Also our model needs to take into account the various relationships that exists between different features involved in the problem, for instance relationship between advertiser and publisher, user and location etc. Bayesian networks or tree augmented naive Bayes will be an ideal choice for this scenario. In this project we used tree augmented naive Bayes approach to predict the probability of an advertisement to be clicked.

2 Background

2.1 Data

Outbrain click prediction data has anonymized information about users, page views and clicks, as observed on multiple publisher sites in the United States between 14-June-2016 and 28-June-2016. The dataset is huge with total size around 100GB. Data is provided in a relational data model. A brief overview of various key attributes present in data is described below.

Table 1: page_veiws

Attribute	Description
uuid	unique user id
document_id	Document in which ad was viewed
timestamp	Time of event
platform	Desktop/Mobile/tablet
geo_location	Country/State/DMA
traffic_source	Internal/Search/Social media

Table 2: clicks_train

Attribute	Description
display_id	Unique id of each page visit
ad_id	Ads that were displayed for display_id
clicked	Whether ad was clicked or not

Table 3: clicks_test

Attribute	Description
display_id	Unique id of each page visit
ad_id	Ads that were displayed for display_id

Table 4: events

Attribute	Description
display_id	Unique id of each page visit
uuid	unique user id
document_id	Document in which display occured
timestamp	Time of event
platform	Desktop/Mobile/tablet
geo_location	Country/State/DMA

Table 5: promoted_content

Attribute	Description
ad_id	Unique id of each ad
document_id	A document in which ad displayed
campaign_id	Campaign in which ad got released
advertiser_id	Advertiser who released the ad

Table 6: documents_meta

Attribute	Description
document_id	Unique identifier of document
source_id	Website in which the ad was displayed
publisher_id	Publisher who posted the document

The core data is events, which refers details from parent tables like promoted_content and documents_meta. Unique identifier of events is display_id. About 70% of events are given as train and rest as test. For the train data we were provided with ad that was clicked in each data and our task in the test data is to predict which ad got clicked among the given ads for a display_id. Train and test split was done uniformly across time and not randomly. In addition to that all the events data on 27-June-2016 and 28-June-2016 are given as test data.

2.2 Algorithm

To predict the probability of an ad clicked in the dataset, we utilise a **Tree Augmented Naive Bayes model**.

We initially planned to implement a Naive Bayes model to the data.However, there are some attributes which are not independent of each other, so we implemented a Tree Augmented Naive Bayes method. A brief description of the model is explaned below-

A Naive Bayes model assumes that given a class label, attributes are independent of each other which rarely occurs in real life. So, taking into account the correlations between variables can help in improving the classification accuracy.

A Tree Augmented Naiye Bayes (TAN)[1] improves a tree structure of a Naive Bayes model by limiting interactions among variables to a single level.

2.3 Implementation

The tree structure in a TAN model is built by using a procedure described by Chow and Liu and involves "reducing the problem of constructing a maximum likelihood tree to a maximum weighted spanning tree in a graph"[2].

We want to construct a maximal weight spanning tree such that we have $N - 1$ edges for N nodes, where N represents the attributes of the model. In this spanning tree, the edge weights correspond to the mutual information. This mutual information can be represented as:

$I_p(X;Y) = \sum_{x,y} P(x,y) log(\frac{(P(x,y)}{(P(x)P(y)}),$
where y is the class label and x represent attributes

The algorithm used to construct a tree is written below-

1. Compute Ip(Xi ; Xj — C) between each pair of attributes i : j.

2. Build a complete undirected graph in which the vertices are the attributes X1,...,Xn. And annotate the weight of an edge connecting Xi to Xj by Ip(Xi ; Xj — C).

3. Build a maximum weighted spanning tree.

4. Transform the resulting undirected tree to a directed one by randomly choosing a root variable and setting the direction of all the edges outward from the root[2].

5. Construct the tree, calculate conditional probabilities of each attribute based on parent and class label.

The formula for TAN classification is as follows:

$$P(C|X_1, ...X_n) = \\ P(C).P(X_{root}|C)\Pi_i P(X_i|C, X_{parent})$$

where C represents class label and X represent the attributes (X_{parent}, X_{root} represent dependant attributes)

A TAN model for our dataset is shown in Figure 1. In the figure, probability of an ad being clicked is the class label. (Country,State,DMA) /(Advertiser Campaign)/ (Publisher, Source, Document) exhibit a heirarchical relationship naturally. Hence, these were treated as dependant features and their conditional joint probabilities were used.

Other attributes such as Time and Platform were considered as independent variables.

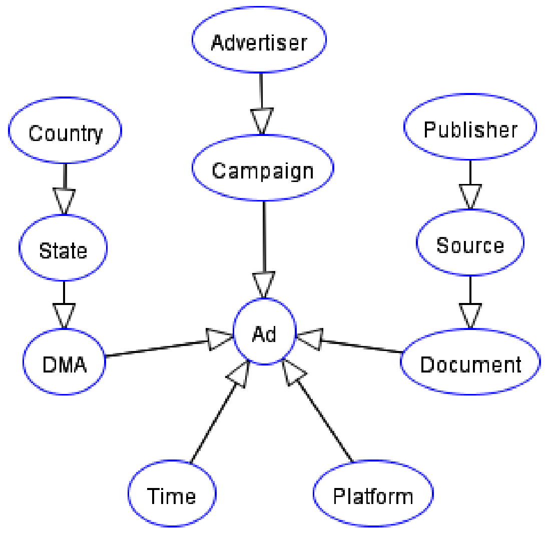

Figure 1: Bayesian network for click prediction

2.4 Evaluation metric

The evaluation metric considered for this problem is mean average precision(MAP) @ 12. It is given as follows,

$$MAP@12 = \frac{\mathbf{1}}{|\mathbf{U}|} \sum_{\mathbf{u=1}}^{|\mathbf{U}|} \sum_{\mathbf{k=1}}^{\mathbf{min(12,n)}} \mathbf{P(k)}$$

where $|U|$ is the number of display ids, $P(k)$ is the precision at cutoff k and n is the number of predicted ad ids.

Precision at cut of k for clicked ad is given as below,

$$P(k) = \frac{\mathbf{1}}{\mathbf{c}}, \text{ and}$$

Precision at cut of k for not clicked ad is given as below,

$$P(k) = \frac{\mathbf{0}}{\mathbf{c}}$$

where c is the position of the clicked advertisement and k is minimum of 12 or number of ad ids in the specific example.

`Example:`
Consider there are only three display ids in the example case, and had displayed two, three and four

advertisements in them respectively. Also in all the cases only one advertisement was clicked and it need to be placed at the front of the list. Farther the predicted ads location in the list, lower the value of $P(k)$.
Let's say the actual order of preference of the ads for three display id is $(a, b), (c, d, e), (f, g, h, i)$ displaying the only clicked ad at first position. And let's say the predicted order of ads is $(a, b), (d, c, e), (g, h, f, i)$. Now $P(k)$ for the display id with expected ad order (a, b) and predicted ad order (a, b) has $k = 2$ and $c = 1$ is
$P(2) = \frac{\mathbf{1}}{\mathbf{1}}$, since there are two ads and correct ad is at first position.
$P(k)$ for the display id with expected ad order (c, d, e) and predicted ad order (d, c, e) has $k = 3$ and $c = 2$ is
$P(3) = \frac{\mathbf{1}}{\mathbf{2}}$, since there are three ads and correct ad is at second position.
$P(k)$ for the display id with expected ad order (f, g, h, i) and predicted ad order (g, h, f, i) has $k = 4$ and $c = 3$ is
$P(2) = \frac{\mathbf{1}}{\mathbf{3}}$, since there are four ads and correct ad is at third position. The maximum value that c or k can take is 12 as $MAP@12$ is used.
Now $MAP@12$ is just the mean $P(k)$ of all display ids. It can be computed as follows for this case with three examples and hence $|U| = 3$

$$MAP@12 = \frac{\mathbf{1}}{\mathbf{3}}\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3}\right) = \frac{11}{18} = 0.6111$$

3 Experiments

The following are the software components involved in this project effort:

1. Python v3.5.2
2. Python packages: numpy
3. MySQL 5.7.16
4. OS specs: Mac - Sierra, RAM - 16 GB
5. Code: `https://github.com/naveenkumar2703/ClickPrediction/tree/master`

3.1 Data preprocessing

Oubrain click prediction data has over 2 billion rows of data. In order to handle the problem efficiently we need to do some data preprocessing. Following steps were done as part of preprocessing.

1. Data analysis - First we analyzed the distribution of various features and their relationships. We found that the number of times an ad was posted and number of times it got clicked are exponentially distributed. Also about 85% of uuids are unique, which means that id of user cannot help us much in terms of prediction.

2. Data cleaning - We did several data cleaning steps. We reformatted the geo location data into a hierarchical structure of country, state and dma for each events. Also we converted the time into event day, and hour window, ie the day in which the event occurred and the hour at which the event occurred.

3. Data reduction - We employed the following data reduction technique to reduce 100Gb data to 5GB of data. The page_views table contained over 2 billion records of web pages visited by all the users. It did not contain any information about the ads displayed or clicked by the user. The potential information in page_views table was traffic source of each users and their location. Also the page_views had information about users and documents which are not part of train and test data. So, we discarded the data that are not part of train or test data. This reduced the data size tremendously in the ratio of 20 to 1.

3.2 Feature engineering

The main objective of the problem is to find the probability of an ad being clicked. As with most of the probability estimation techniques, we need to know the count of each events and their combinations. To count and store the probabilities data, we created following tables:
- ad_country_probabilities
- ad_state_probabilities
- ad_dma_probabilities
- ad_document_probabilities
- ad_publisher_probabilities
- ad_source_probabilities
- ad_day_probabilities
- ad_hour_probabilities
- ad_platform_probabilities
- ad_advertiser_probabilities
- ad_campaign_probabilities

Above tables contained the click probability of respective feature over 500000 different ads along with the number of times they were adjusted and number of times they were clicked. Click probabilities were computed after applying Laplace correction. We used the average number of ads displayed per page, which is 5, in the number of possible values to estimate the Laplace corrected probability.

3.3 Click Probability Estimation

We used Bayesian network described in section 2.2 to estimate the joint probabilities of tree augmented naive Bayes. We retrieved the advertised counts and clicked counts for the respective joint condition by querying the relational data base. Once the probabilities are computed using laplacian correction, the click probability of each ad was estimated as the product of probabilities of the child features described in the section 2.2. The predicted ad probabilities are sorted and the ads returned as the output in the decreasing order of the probability of click.

3.4 Cross validation approach

In order to get a better predicting result, we took 15 % of train data as cross validation data and used it for testing. We marked all display ids that end with zero and all data that belong to day 12, which is 25-June-2016 as cross validation data. We ran our code on a sampled data at the size of 20000 from cross validation data for quick cross validation. We estimated all our probabilities from train data excluding cross validation data for initial validation and fine tuning of better attributes. We fine tuned the algorithm by including a cut off number of displayed ads for each feature and experimenting them on cross validation data. Based on the best result, we again computed the probabilities over all the train data and used it to predict the test data.

4 Results

The MAP@12 score for 20000 random points from cross validation data for various combination of features is displayed in the table. We were able to achieve the highest score when using all the features with some cut-off. Also we could clearly see that the MAP@12 value dropped when that cut off was not used. This indicates that the result was influenced by including noisy occurrences and does not generalize well. In order to improve the performance, we included the use of probabilities of features only if they were clicked at least 10 times to avoid impact of random or noisy data points. Summary of the result is given below.

Probabilities used	**MAP@12 with cap on number of ads clicked**	**MAP@12 without cap on number of ads clicked**
ad click	0.5942	0.5973
ad click with advertiser and campaign	0.6314	0.6351
ad click with advertiser, campaign, location and time	0.6387	0.6401
ad click with advertiser, campaign, location, time, publisher and source	0.6412	0.6480

Note: Our Kaggle score during submission of this report was 0.64796 and Rank was at 96.

5 Conclusion

The following are the conclusions inferred from this project effort:
• The prediction of ad clicks in a sparse data environment is challenging one.
• By using a very simple probability of ad click we were able to get a decent MAP.
• Adding more features into the model improved the results by .05 in MAP@12 scale.

6 Future Work

In addition to what has been done, we feel we can improve upon our results by taking a few more steps as follows:
• Try a non bayesian approach like association analysis and see if we could find any patterns among the relationship between ads that were clicked and ads that were not clicked.
• Instead of using tree augmented naive Bayes, use a dynamic Bayesian network and see how it impacts

the results.

7 Acknowledgements

We would like to thank Prof. Mehmet Dalkilic for giving us an idea about the quintessential aspects of data mining as we have found that to have a major impact on the outcome of the model. Also we would like to thank the team of associate instructors for taking time to guide us and clarify our questions over the course of the semester.

References

[1] Harini Padmanaban:Comparative Analysis of Naive Bayes and Tree Augmented Naive Bayes Models
`http://http://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1350&context=etd_projects`

[2] Chow, C.K. and C.N. Liu (1968). Approximating discrete probability distributions with dependence trees. IEEE Trans. on Info. Theory, 14, 462-467.

[3] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers.