Outbrain Click Prediction

Girish Limaye
glimaye@stanford.edu

Prashant Jaiswal
prjaiswa@stanford.edu

Vijayaraj Gopinath
vgopinat@stanford.edu

Introduction:
The internet is incredibly rich in the information that the people need on day to day basis. Every day we access hundreds of documents regarding various topics like travel, news, recipes and other subjects. While the users surf on their favorite sites, they are presented with lot of content which they might be interested in. Outbrain is one such content discovery platform that delivers these moments while the users surf their favorite sites.

Currently, Outbrain pairs relevant content with curious readers in about 250 billion personalized recommendations every month across many thousands of sites. In this project, we aim to predict which pieces of content its global base of users are likely to click on. Improving Outbrain's recommendation algorithm will mean more users uncover stories that satisfy their individual tastes.

Background and related work:
Predicting ad click–through rates (CTR) is a popular learning problem that is central to the multi-billion dollar online advertising industry. While we focus on CTR prediction for display advertising, we consider the problem as a recommendation problem, where ads must be recommended for appropriate users. It is also very similar to CTR for sponsored search that has been studied extensively.

Large amount of unbalanced training data (% click is way lower than 50%) and data sparsity are key challenges for CTR prediction. Owing to its easy implementation, immediate prediction and acceptable performance, logistic regression or generalized linear models have been widely applied for ad CTR prediction (Lee et al. 2012; Yan et al. 2014; Chapelle et al. 2014)

Display advertising involves features regarding contexts (publishers) as well as users and ads (advertisers). This implies more complicated interactions between these features, which should be taken into consideration when CTR is predicted. However, traditional approaches, such as linear models are weak in modeling these complicated interactions between context, users and ads. To address this problem, two approaches have been used to learn the effect of feature conjunction.

The first approach[1] is the obvious one of using degree-n polynomial features and learn weights for each of these polynomial feature either linearly or using SVM kernel trick. (Chang et al. 2010, Kudo et al. 2003)

But these approaches don't work well in scenario with sparse data. For example, data we used for this project has over ~84 million ad impressions.

- 99.96% of Users that appear less than 10 times in these 84 million ad impressions.
- 61.74% of Ads that appear less than 10 times in these 84 million ad impressions.[2]

In such scenarios, feature combinations (feature *a* and feature *b* coexisting) are rare and thus provide relatively little value.

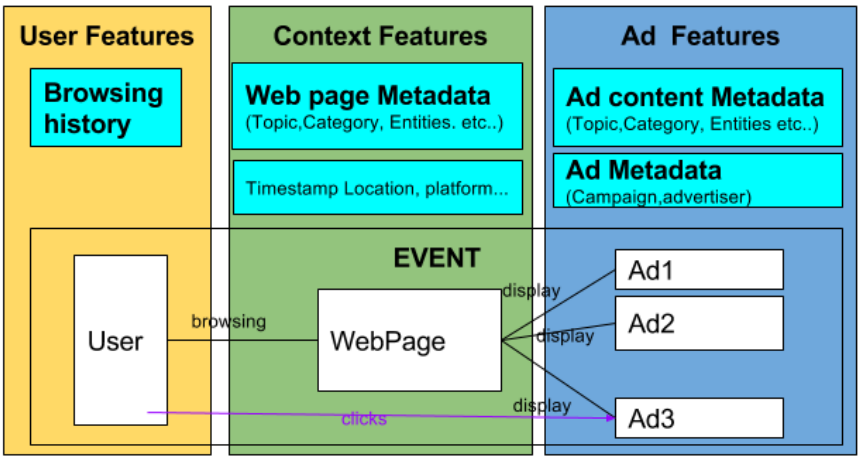
The second approach involves factorization models which provide a powerful technique to make use of explicit data to overcome the sparsity of the implicit data. These models involve representing the features for users and ads by a fixed length vector, or a latent factor. Popular latent factor models based on a matrix for collaborative filtering have been successfully used to address the recommendation problem (Koren et al. 2009; Chen et al. 2012) and ad CTR prediction for sponsored search (Wu et al. 2012).

Factorization machines (Rendle 2010) combines polynomial regression and the concept of latent factors to create a general purpose approach which along with its variant (Juan et al. 2015 Field aware factorization machine) is considered the state of the art approach in general purpose CTR

[1] Our results (& approach) have significantly improved from what we had in poster.
[2] This project is active Kaggle competition https://www.kaggle.com/c/outbrain-click-prediction

prediction. In terms of efficiency of learning, factorization machines can be compared to support vector machines (SVMs) with a polynomial kernel. However, SVMs known weaknesses with respect to data sparsity that is been addressed by factorization machines.
Lastly, combining **decision trees** with other traditional approaches have been shown to derive good results (He et al. 2014).

Problem Formulation:

Input	Training set from Kaggle provides labeled data about multiple sets of Ads where exactly one Ad in each set is clicked. Learning is done based on the following data 1. User Features (page views of the user potentially providing an insight into users interests and preferences) 2. Context Features (Web page metadata, date time, platform, location etc.) 3. Ad Features (CampaignID and Advertiser id , Ad Metadata)  **Content Metadata :** Note that details like topic, category and entity attributes related to the content are standard across User, Context and Ad. For e.g. the category describing the ad, web page and browsing history come from the same category types.
Output	A test set provides data similar to training set with multiple sets of Ads but doesn't say which one ad is clicked. The expected output is to provide list of ads for each set sorted by the probability of it being clicked Set1 : ad1, ad2, ad3 Set2: ad4, ad5, ad6 where P(ad1 is clicked) >P(ad2 is clicked) > P(ad2 is clicked) and so on
Evaluation	Mean average precision MAP@12 is used to measure the performance of our prediction.

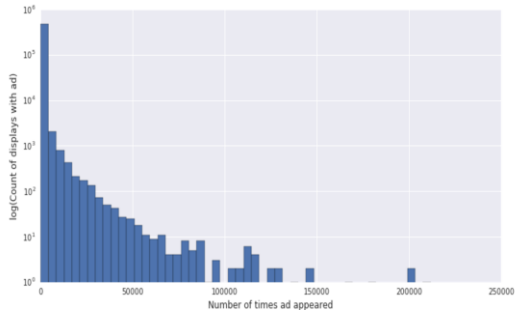
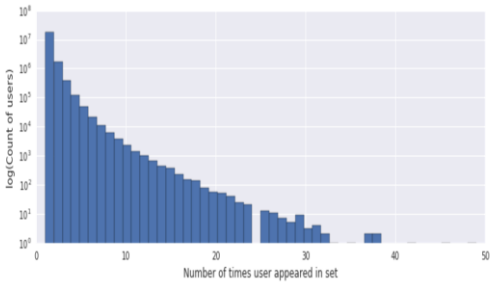
Note: We only used fraction of the data due to computational cost.

Data Analysis:
Before getting our hands dirty with some ML algorithms, we analyzed the data thoroughly to get some intuition on the problem. We'll talk about only part of our analysis due to space constraints.

- We looked at how many ads were repeated within the train set. We found out that almost 23% of the ads in the train set were unique and less than 62% data appeared less than 10 times

(Figure 1). So developing a model just based on past ads will not help unless we link with other information such as advertiser and campaign data.

- Similarly we wanted to check the how users are distributed within the events table. We found out this is very similar to ads distribution, almost 88% users in the events table are unique. (Figure 2) So building a user profile exclusively based on the user's click rate will be very ineffective to solve this problem.
- We also have topics and categories which are metadata about documents. We found out that there are 300 unique topics and 97 unique categories. There are some topics and categories which appears very often.

 Figure 1	Ads that appear less than 2 times: 22.69% Ads that appear less than 10 times: 61.74% Ads that appear less than 50 times: 86.93% Ads that appear less than 100 times: 91.04% Ads that appear less than 1000 times: 97.45%
 Figure 2	Users that appear less than 2 times: 88.42% Users that appear less than 5 times: 99.51% Users that appear less than 10 times: 99.96%

Based on our analysis we conclude that data sparsity, imbalance of data (clicked vs not clicked) and capturing feature interactions across User, Ad and Context are the key challenges we will have to address in our solution

Approach 1:

Step1: Baseline classification algorithm

We built a basic working prototype with the goal of learning more about the problem.

To predict which of the ads will be clicked on a document, we considered the features such as publisher of the ads, features that summarize the document (such as document category, entity it refers to, topic it is talking about and other metadata), user whom the ads are being served etc. We start with stochastic gradient descent with Logistic loss with basic independent features. Stochastic gradient descent takes one example at a time so we don't have to process the whole training data every time and thus can be used in online learning scenarios. Also Logistic loss has a smooth probability output as opposed to others (such as hinge loss) and the output probability can be more

meaningfully used. We started with a baseline model that gives 80% MAP@12 precision on training data. The takeaway from our baseline model was that we need to add more features regarding the summary of the document on which the ads is served, the clickability of the ad in general and the relationship of the ad to the document on which it is served.

Step 2: One hot encoding

In our problem, Document attributes of 'category' and 'topic' are central component of the training data. There are 90 categories and 300 topics in the universe of documents provided and each document has a one to many to each document. One hot encoding transforms categorical features to a format that works better with our classification algorithms. Note that this is not needed for random forest algorithms as they handle categorical features natively.

So for of the 3 category columns (Ad-Category, PageView-Category, Context-Category) we generate 90 features with a binary value for each of the 90 categories. We use a similar technique for topics.

These new binary features are then passed along to our stochastic gradient descent algorithm with log loss which provides weight for each of these features. Note that this method considers only first order features and no interaction between features are captured so far.

Number of features considered: 12 (from base classification) + 300*3 +90*3 = 1182

Step 3: Feature engineering for conjugate features

In this step, we start to exploit the pairwise interaction between the categorical features that are shared across User, Context and Ad. We first note that since the topics and categories are shared across all 3 feature-groups, a match on any of these features is intuitively an important piece of information that we need to provide to the learning algorithm. As an example, when browsing a page about Fast Cars (Context-Category=Cars) an Ad about sports Car (Ad-Category=Cars) will likely be more relevant than another category like politics. To capture the interaction between matching categories and topics we use the well-researched Inverse Document Frequency Measure

CategoryMatchScore = Log (Total Documents/ Documents with given category)

In our case this gives a score between 0 and 12 that we use in our learning algorithm.

We further want to capture interaction across different categories and topics for each of the 3 feature groups. We do this by pre calculating the number of co-occurrences of each interaction and storing it in a lookup table. We then use the Bayesian prior probability of the co-occurrence of the interaction and use them as features in the test set. This incurs a penalty of a one-time scan of the training data at the time of training which is *O(m*n)* and a linear lookup time at the time of prediction.

Note that the above method capture pairwise interaction between features and do increase the MAP score as compared to linear features. But the drawback of the above method is that the small number of co-occurrences of topics and categories reduce the amount of learning possible. We look at the next method of Factorization to use these features but to overcome the sparsity drawback.

Number of features considered: 1182 (from previous) + 4 + 12 = 1198

Step 4: Factorization model

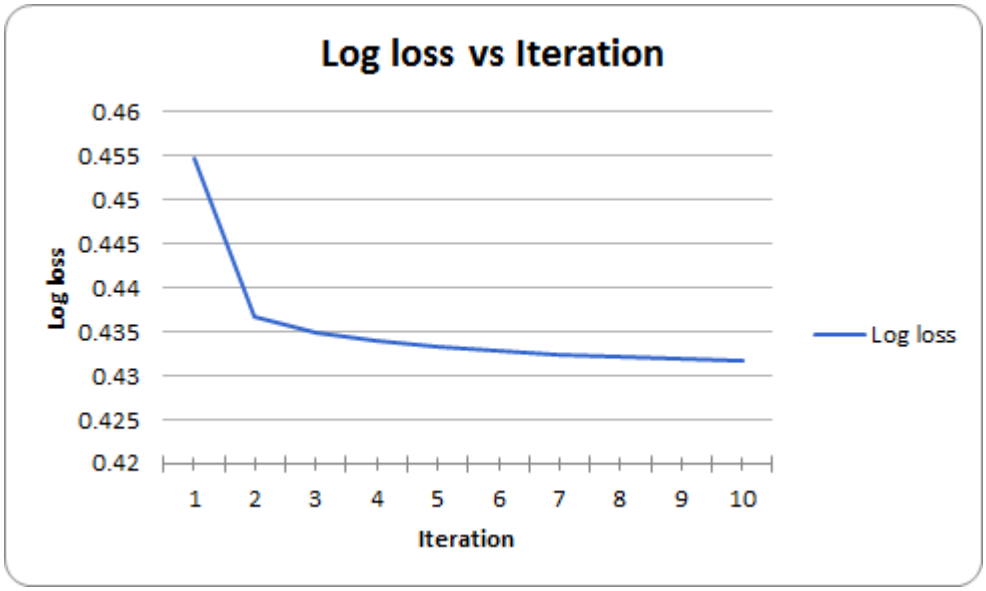
Factorization Machines is considered state-of-art method for generalized recommendation and CTR prediction problem in sparse scenarios. FMs are a general predictor working with any real valued feature vector. In our case they use the one hot encoded features as well as the additional features we engineered. FMs model all interactions between variables using factorized parameters.

Below is the equation for Factorization Machine for 2nd order predicting *y(x)* from feature set *X*

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

Note that the above learning captures the intercept (first term), the first order feature (second term in the equation) as well as the second order features (third term in the equation). Note that the weight learnt on the second order terms is expressed by dot product of vectors *Vi* and *Vj* which are individual vectors for *Xi* and the *Xj*. Thus we are able to capture interaction in features even in sparse scenarios. And since Factorization machines are generic learners we use all the features we devised in our previous steps.

The graph and the table below shows our experiments with different values of *k* (factors for *Vi* and *Vj*) and the number of epochs.



# of factors	# of epochs	MAP Score
2	2	0.64900
2	5	0.65070
2	10	0.65110
2	2	0.64900
3	3	0.64900
5	2	0.65090
10	10	0.65110

Number of features considered: All 2nd order combinations of all 1198 features in the previous step pt (e.g. For features a, b, c factorization machine considers ab, bc and ac but not sqr(a), sqr(b), sqr(c))

Approach 2: Boosted Trees Model

In parallel with the above approach, we also explored one more model i.e. Boosted Trees Model to solve this problem. Boosted Trees is very powerful model which can learn highly non-linear relationships as additional trees in boosting increases the function space that the algorithm considers for finding the fit to the data. In addition to this, trees handle missing values very naturally by considering them to be one of the values attained by the columns of the data. We used Microsoft's internal tool to implement Boosted Trees Model.

In Boosted Trees Model [7], trees are grown sequentially. Each tree is grown using the information from the previously grown trees. In each step b, a tree f^b is fit to the training data and the learning algorithm f is updated by adding the shrunken version of the new tree.

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

We update the residuals so that subsequent tree can be fit on the residual.

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

Finally all the trees are combined to give the learning model using the learning rate λ.

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

To solve this problem, we consider the following features – document features like document categories, topics, publisher and source etc. for the document on which the ads are triggered; ad features like advertiser, ad_topic, ad_cateogry etc; we also add other secondary features like click probability for the ad, click probability for publisher of the ads etc. In the charts below, we show the effect of various parameters of the Boosted Trees Model on MAP score. We found that this model works very well for this problem and gives **MAP@12 score of 0.666**.

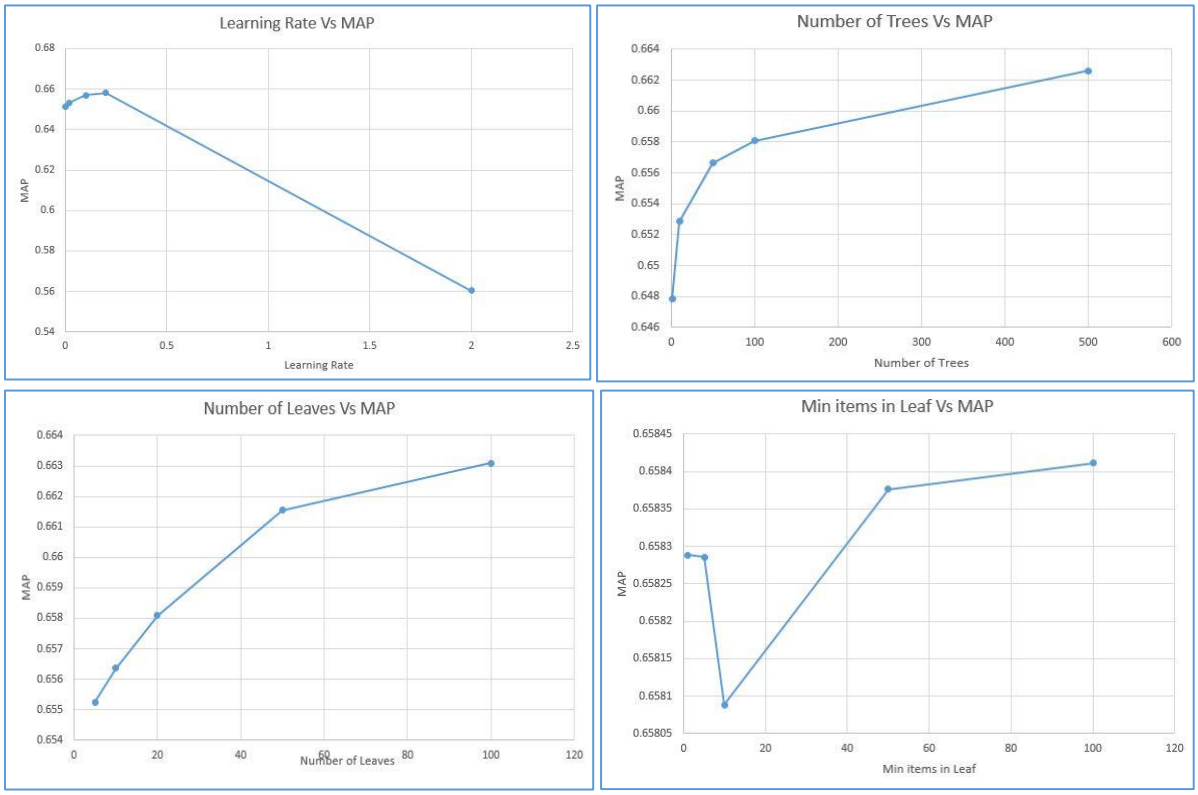


Fig. *Effect of various parameters of the Boosted Trees model on MAP@12*

Results:
Mean average precision MAP@12 is used to measure the performance of our prediction.

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

Where **|*U*|** is the number of display_ids, ***P(k)*** is the precision at cutoff ***k***, ***n*** is the number of predicted ad_ids.

Our training set contained around 4M rows and test set contained around 1M rows. We used only subset of the data available due to computational cost.

	Approach	**Maximum MAP Score on Test**
1	Base model with linear features	0.613
2	With one hot encoding	0.634
3	With engineered conjugate features	0.642
4	With Factorization Machines	0.651
5	Boosted Trees Model	0.666
**Kaggle competition top score*		0.695

In terms of training time, we use Factorization machines instead of standard second order polynomial models. The model equation of FMs can be calculated in linear time .This makes our approach practical in scenarios like ours with high number of features.

Future work:
While we currently consider pairwise features in our model, in future we look to consider ternary features (Ad-Context-User). Current implementations of factorization machines tend to be unstable for any orders above 2. But intuitively ternary features are important for learning algorithm (E.g. A user interested in automobiles, browsing Auto magazine is likely to click on a BMW Ad) and that will be our next direction.
Another interesting direction is to explore the user browsing history data further. Our dataset makes 2 billion page view impressions available. Different intelligent ways of clustering users based on their page view behavior seems to have lot of potential for CTR prediction
Lastly, in most of the CTR scenarios, the features are grouped into fields - Ad, User and Context in our example. Field Aware Factorization machines are variants to FM's that use this additional information about the field. This is another direction we look to explore in future.

References

[1] Y. W. Chang, C. J. Hsieh, K. W. Chang, M. Ringgaard, and C.-J. Lin, “Training and testing low-degree polynomial data mappings via linear SVM,” Journal of Machine Learning Research, vol. 11, pp. 1471–1490, 2010.
[2] T. Kudo and Y. Matsumoto, “Fast methods for kernel-based text analysis,” in Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL), 2003
[3] S. Rendle, “Factorization machines,” in Proceedings of IEEE International Conference on Data Mining (ICDM), pp. 995–1000, 2010.
[4] B. Mcmahan, G. Holt, D. Scully , “Ad Click Prediction: a View from the Trenches”
[5] J. Pan, O. Jin, T. Xu, “Practical Lessons from Predicting Clicks on Ads at Facebook”
[6] Y. Juan, Y. Xhuang, W, Chin, “Field-aware Factorization Machines for CTR Prediction”
[7] G. James, D. Witten, T. Hastie, R. Tibshirani, “An Introduction to Statistical Learning”, 2013.