# Semester Project

Jivitesh Poojary[1]*, Shreyas Rewagad[2]@

**Abstract**
Predicting ad click through rates (CTR) is a massive-scale learning problem that is central to the multi-billion dollar online advertising industry. Users are provided with personalized recommendations across thousands of sites. In our work we are challenged to predict the order of the ads in an descreasing order of probability of being clicked by the user. The size of the data presents to us a unique challenge and also provides an excellent opportunity to experiment with infrastructure which handles large scale data. We try to model the data with some traditional data mining techniques. We also came across techiques which have been used in the industry, which provide us glimse into the real world of data mining.

**Keywords**
Online Advertising — Data Mining — Ad prediction — Maximum Liklihood — Smoothing — MAP — FTRL

[1] *Data Science, School of Informatics and Computing, Indiana University, Bloomington, IN, USA*
[2] *Data Science, School of Informatics and Computing, Indiana University, Bloomington, IN, USA*
***Corresponding author**: jpoojary@iu.edu
**@Corresponding author**: srewagad@iu.edu

## Contents



## Introduction

Online advertising is a multi-billion dollar industry that has served as one of the great success stories for machine learning. Sponsored search advertising, contextual advertising, display advertising, and real-time bidding auctions have all relied heavily on the ability of learned models to predict ad click through rates accurately, quickly, and reliably. This problem setting has also pushed the field to address issues of scale that even a decade ago would have been almost inconceivable. A typical industrial model may provide predictions on billions of events per day, using a correspondingly large feature space, and then learn from the resulting mass of data. [5]

This domain has some overlapping concepts with information retrieval, that help us to better understand it and also consider cross domain tools and methodologies for solving the problem.

The purpose of any recommending system is often summarized as an activity of helping the users find relevant items, and the predominant operationalization of this goal has been to focus on the ability to numerically estimate the users' preferences for unseen items or to provide users with item lists ranked in accordance to the estimated preferences. This dominant, albeit narrow, view of the recommendation problem has been tremendously helpful in advancing research in different ways, e.g., through the establishment of standardized evaluation procedures and metrics. In reality, recommender systems can serve a variety of purposes from the point of view of both consumers and providers. [6]

## 1. Background

The dataset for this challenge contains a sample of users' page views and clicks, as observed on multiple publisher sites in the United States between 14-June-2016 and 28-June-2016. Each viewed page or clicked recommendation is further accompanied by some semantic attributes of those documents. [7]

### 1.1 Variable Analysis and observation

We are having 11 files in total to perform the analysis, the table below gives more information about the files, number of

records and presence of data: Reference to table 1, 2 and 3

**Table 1.** Table of File Size

| File Name | File Size |
|---|---|
| clicks train.zip | 389.75 MB |
| clicks test.zip | 135.43 MB |
| documents meta.zip | 15.51 MB |
| documents categories.zip | 32.34 MB |
| documents entities.zip | 125.67 MB |
| documents topics.zip | 120.91 MB |
| promoted content.zip | 2.52 MB |
| events.zip | 477.74 MB |
| page views.zip | 29.71 GB |
| page views sample.zip | 148.51 MB |
| sample submission.zip | 99.57 MB |

**Table 2.** Table of Records

| File Name | No. of Records |
|---|---|
| clicks train.zip | 87141732 |
| clicks test.zip | 32225163 |
| documents meta.zip | 2999335 |
| documents categories.zip | 5481476 |
| documents entities.zip | 5537553 |
| documents topics.zip | 11325961 |
| promoted content.zip | 559584 |
| events.zip | 23120127 |
| page views sample.zip | 10000000 |
| sample submission.zip | 6245534 |

The page views file was too large and hence we were not able to get the number of rows here.

### 1.2 Data Definition

We have provided a definition of the attributes across various files, they provide some insights into how we can go about in merging the documents and thus try to get some more information for our prediction model. Reference to table 4.

### 1.3 Previous work

Logistic regression and decision trees are popular models used in the computational advertising literature. Applying decision trees to display advertising, however, has additional challenges due to having categorical features with very large cardinality and the sparse nature of the data[1]

From our observation at various forums we can conclude that regularlized logistic regression work best for such problems, If we were to use it with stochastic gradient descent we can get good results.

The training data give us information about occurrence or non-occurrence of event, event in this case the ad being clicked. We cannot employ metrics like precision or accuracy since all we have to predict is the probability of the ad being clicked rather than. We will not be able to truly evaluate our system as we do not have the real values of the occurrence or non occurrence of the click event in the test data.

**Table 3.** Table of Missing data

| File Name | Missing Data |
|---|---|
| clicks train.zip | FALSE |
| clicks test.zip | FALSE |
| documents meta.zip | FALSE |
| documents categories.zip | FALSE |
| documents entities.zip | FALSE |
| documents topics.zip | FALSE |
| promoted content.zip | FALSE |
| events.zip | FALSE |
| page views sample.zip | FALSE |
| sample submission.zip | FALSE |

## 2. Algorithm and Methodology

### 2.1 Naive Maximum Probability Estimation

This estimation approach assumes each observation to be an independent event and the model parameters to be constants. Anything we know about the model parameters a priori is ignored, in this case the association of a display id associated with a click event. While it is tempting to assume that we know nothing and let the model do the work, feeding prior knowledge into the model helps to minimize the generalization error.

One consequence of naive MLE is a tendency to overfit, i.e., exaggerate relatively small fluctuations in observed data. This can be mitigated by techniques such as regularization, but this leads to mis calibrated probabilities, which require additional adjustment. It is also important to remember that the probability of success is a random variable and is influenced by various exogenous factors not included in the model. For example, the basic model excludes randomness in usage patterns that may have occurred over time. MLE only gives an average point estimate of this random variable. While this estimate is a useful measure of the central tendency, we cannot be certain that it is representative of the entire distribution for prediction purposes. [2]

In our approach caution must be taken as the ad id's present in the trainData may not be present in our testData. Here in order to solve the problem we to associate a probability of mean global clickthrough rate. In this way we do not assign a lower probability of occurance to unseen data.

### 2.2 Probability smoothing

This approach is an extension of the Naive Maximum likelihood estimation process discussed earlier. The differentiating factor in this technique is the way we handle low occurance clicks and how we differentiate between negative instances with different number of occurances. One of the issue with using per-ad target rate encoding is that ads with very small number of views can get inflated click probabilities. We need

**Table 4.** Table of Attribute description

| Attribute name | Definition |
|---|---|
| display id | Each context (i.e. a set of recommendations) is given a display id |
| uuid | Each user in the dataset is represented by a unique id |
| document id | A web page with content (e.g. a news article) |
| ad id | On each document, a set of ads are displayed |
| timestamp | If you wish to recover the actual epoch time of the visit, add 1465876799998 to the timestamp |
| platform | desktop = 1, mobile = 2, tablet =3 |
| geo location | country>state>DMA |
| traffic source | internal = 1, search = 2, social = 3 |
| clicked | 1 if clicked, 0 otherwise |
| source id | The part of the site on which the document is displayed, e.g. google.com |
| campaign id | Unique Identifier for a campaign |
| advertiser id | Unique Identifier for a advertiser |
| publisher id | Unique Identifier for a publisher |
| publish time | Time associated with ad publishing |
| topic id | Unique Identifier for a topic |

to add a term that penalizes ads with small amounts of data, therefore making it prefer an ad with large amounts of training data and a reliable probability. We can do this by using a smoothing technique which will reduce the probabilities of all ads, but will reduce the probabilities of ads with fewer records drastically. In a way this approach is helpful in making level playing field for predicting as probabilities.[3]

**Requirement** For many applications, it is important to not only estimate the CTR of the ad, but also to quantify the expected accuracy of the prediction. In particular, such estimates can be used to measure and control explore/exploit tradeoffs: in order to make accurate predictions, the system must sometimes show ads for which it has little data, but this should be balanced against the advantages of showing ads which are known to be good

**Overview** Probability smoothing is a modeling technique that assigns some non-zero probability to events that were unseen in the training data. This has the effect that the probability mass is divided over more events, hence the probability distribution becomes more smooth. Smoothing overcomes the so-called sparse data problem, that is, many events that are plausible in reality are not found in the data used to estimate probabilities. When using maximum likelihood estimates, unseen events are assigned zero probability. The sparse data problem is the reason that it is hard for information retrieval systems to obtain high recall values without degrading values for precision, and smoothing is a means to increase recall (possibly degrading precision in the process. A smoothing method

**Algorithm 1** Naive Maximum Probability Estimation

```
INPUT (clickTrainFile, clickTestFile)
OUTPUT (submissionFile)
procedure NMPE
    train ← train data from clickTrainFile
    test ← test data from clickTestFile
    nmpe[total] ← total of each train[ad]
    nmpe[sum] ← sum of each train[ad]
    mean ← sum of all train[ad]/total of each train[ad]
    nmpe[likelihood] ← nmpe[sum]/ nmpe[total]
    if any.nmpe[likelihood] = NA then
        nmpe[likelihood].ad = Mean.
    nmpe[likelihood] ← sort by the likelihood column
    submissionFile ← group by ad id nmpe[]
    create csv of submissionFile
```

used may be a simple one like Laplace smoothing which adds an extra count to every possible count (ads in this case) or it can be complicated as Linear interpolation smoothing or Dirichlet smoothing.[4]

Smoothing is required in our case as we are having a very sparse data when it come to the event of an ad being clicked. These techniques help in making a better likelihood estimation in this scenarios. The equation of Dirichlet smoothing is given below:

$$p_\mu(w;d) = \frac{c(w;d) + \mu p(w|C)}{\sum_{w_1} c(w_1,d) + \mu} \tag{1}$$

**Algorithm 2** Probability smoothing

```
INPUT (clickTrainFile, clickTestFile)
OUTPUT (submissionFile)
procedure SMOOTHING(constant)
    train ← train data from clickTrainFile
    test ← test data from clickTestFile
    μ ← constant
    smt[total] ← total of each train[ad]
    smt[sum] ← sum of each train[ad]
    mean ← sum of all train[ad]/total of each train[ad]
    smt[prob] ← (smt[sum] + Mean* μ)/ (smt[total] + μ)
    smt[prob] ← sort by the probability column
    submissionFile ← group by ad id smt[]
    create csv of submissionFile
```

Here our objective becomes to get the most appropriate value of $\mu$ so that we get the best possible likelihood prediction.

**Example:** Suppose we represent the ads an its occurrence as vectors, the $a_1 : \{0,0,0,1,0\}$, $a_2 : \{0,0,0,0,0\}$, $a_3 : \{0,0,1\}$,$a_4 : \{0,0,0,0\}$, $a_5 : \{1,1,0,0,0\}$ and $a_6 : \{1,1,1,0,0,0\}$, with $\mu$ = 4
$p(w|C)$ = *4 / 23* = 0.174 Then,

$P(a_1)$ = *(1 + 4*0.174) / (5 + 4)* = 0.1884
$P(a_2)$ = *(0 + 4*0.174) / (5 + 4)* = 0.0773
$P(a_3)$ = *(1 + 4*0.174) / (3 + 4)* = 0.2423
$P(a_4)$ = *(0 + 4*0.174) / (4 + 4)* = 0.0870
$P(a_5)$ = *(2 + 4*0.174) / (5 + 4)* = 0.1566
$P(a_6)$ = *(3 + 4*0.174) / (6 + 4)* = 0.2088

From the example we can conclude that the probabilities help in resolving the bias towards records which do not have a click associated with them, in this case if we compare two ads which have not been clicked, the ad with a lower occurrence also has a lower negative output. Hence it should have a higher probability of occurrence.

### 2.3 Follow-the-Regularized-Leader [FTRL-Proximal]

Methods such as regularized logistic regression are a natural fit for this problem setting. It is necessary to make predictions many billions of times per day and to quickly update the model as new clicks and non-clicks are observed. Of course, this data rate means that training data sets are enormous and the prediction thus challenging.

FTRL-Proximal [5] is an algorithm based on the seminal paper 'Ad Click Prediction: a View from the Trenches'. This algorithm along with Field-aware Factorization machines (FFM) and Vowpal Wabbit are a go to algorithms when it comes to ad prediction, in this project we have made an attempt to implement the FTRL-Proximal algorithm. The reason being they aim at creating parse data which is then applied on a regression model, with an update performing stochastic gradient boosting.

In recent years, online gradient descent and stochastic gradient descent (its batch analogue) have proven themselves to be excellent algorithms for large-scale machine learning. In the simplest case FTRL-Proximal is identical, but when $L_1$ or other non smooth regularization is needed, FTRL-Proximal significantly outperforms FOBOS, and can outperform RDA as well. Since the implementations of FTRL-Proximal and RDA only differ by a few lines of code, we recommend trying both and picking the one with the best performance in practice.

The features used in our system are drawn from a variety of sources, including the promotion content, the event, and various ad-related metadata. Data tends to be extremely sparse, with typically only a tiny fraction of nonzero feature values per example.

The FTRL-Proximal online learning algorithm is like Stochastic Gradient Descent, but much sparser models. In our case we have kept the regularization to be zero and non-zero dummy values. The key for the better performance of this model is re-expressing gradient descent implicitly.

$$w_{t+1} = (g_{1+t}.w + \frac{1}{2}\sum_{s=1}\sigma_s||w - w_s||_2^2 + \lambda_1||w||_1) \quad (2)$$

**Algorithm 3** Per-Coordinate FTRL-Proximal with $L_1$ and $L_2$ Regularization for Logistic Regression

*With per-coordinate learning rate of Eq. (3)*
**INPUT** parameter $\alpha, \beta, \lambda_1, \lambda_2$
$(\forall i\ 1, ..., d)$, *initialize* $z_i$=0 and $n_i = 0$
**for** *t = 1 to T* **do**
  Recieve feature vector $x_t$ and let I =$\{i|x_i \neq 0\}$

$$w_{t,i} = \begin{cases} 0 & \text{if} |z_i| \leq \lambda_1 \\ -(\frac{\beta+\sqrt{n_i}}{\alpha} + \lambda_2)^{-1}(z_i - sgn(z_i)\lambda_1) & otherwise \end{cases}$$

  **for** $i \in I$ **do**
    $g_i = (p_t - y_t)x_i$ ....gradient of loss w.r.t. $w_i$
    $\sigma_i = \frac{1}{\alpha}(\sqrt{n_i + g_i^2} - \sqrt{n_i})$.
    $z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$
    $n_i \leftarrow n_i + g_i^2$

$$\eta_{t,i} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t} g_{s,i}^2}} \quad (3)$$

## 3. Experiments and Results

### 3.1 Tools and infrastructure used

We were successfully able to implement it on the Sharks (SoIC server) and the Karst (IUB University Server). Reference to table 5 for more details.

**Table 5.** Table of tools and infrastructure

| Name | Type | Specification |
|---|---|---|
| Karst | Server | 16 Cores x86 64 architecture |
| Sharks | Server | 32 Cores x86 64 architecture |
| Python | Platform | 2.7.3 64-bit version |
| Spyder | IDE | Version 2.3.9 |
| Linux | OS | Red Hat Enterprise Linux Server release 7.3 (Maipo) |
| PuTTY | SSH Tool | Version 0.67 |
| WinSCP | FTP tool | Version 5.9.3 |

### 3.2 Evaluation Technique

The evaluation was performed on the Kaggle server once we provided the required input derived from our model. The technique used in this case was MAP@12, it is a mean average precision method which consider the first 12 significant prediction. This technique was set by the competition organizer. As the real values for the click predictions of the test data are available only with the organization, we wont be able to evaluate our model unless we submit it on the portal. We tried to obtain different results by using different values. Please

find the equation below used for evaluation. [7]

$$MAP@12 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{min(12,n)} P(k) \tag{4}$$

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

P(k) means the precision at cut-off k in the item list, i.e., the ratio of number of recommended nodes followed, up to the position k, over the number k; P(k) equals 0 when the k-th item is not followed upon recommendation.Note this means that order matters. But it depends. Order matters only if there is at least one incorrect prediction. In other words, if all predictions are correct, it doesn't matter in which order they are given. So, it is better to submit more certain recommendations first. AP score reflects this.

### 3.3 Naive Maximum Probability Estimation

**Findings:** As mentioned in the algorithm this was a very basic and naive approach for probability prediction. Surprisingly this approach seems to give a good result. We were able to obtain a score of 0.63529 on our submission on the Kaggle portal. The overal time for execution was 1108.00900412 seconds.

**Conclusion:** One important conclusion that we drew from this was the role of the train data file in our predictions. Considering the only attributes present in the file were the display id, ad id and the boolean variable of click event occurrence. And we only used last two attributes in our model.

**Limitations:** As we have named it this approach is too naive. Here we not only ignore the impact of other attributes like promoted content or event by but also fail to differentiate between two ads with same probabilities but different occurrence counts.

### 3.4 Probability smoothing

**Findings:** As the evaluation method was based on the first 12 significant prediction, we first tried setting the coefficient $\mu$ to 12. After that we tried to vary the value of $\mu$ to observe how the scores vary with change in the $\mu$ value. We varied the value of the parameter form 8 to 15 (considering 12 as our baseline) Our observation was that as we increased the value of $\mu$ the score showed minor improvements. However this was a substantial improvement of when compared to the Naive maximum likelihood estimation. Our best scenario with $\mu$ as 15 garnered us a score of 0.63718 which was a 0.00189 or 0.3% increase in prediction. We are considering it as a significant improvement as the range of scores are in the deviation of 7-8%. Please Refer to Figure 1.

**Conclusions:** From our findings we concluded that adjusting probabilities by making use of Dirichlet smoothing we are getting better results. Some of the key takeaways from this implementation were the effect of changing parameter $\mu$ on the final score.

**Limitations:** Like the Naive Maximum Probability Estimation here too we fail to consider the role of other attributes in different files in our model.

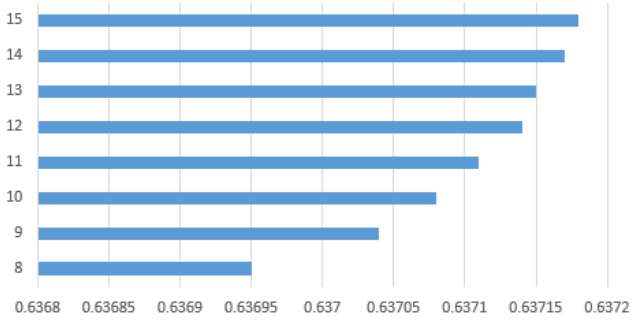


**Figure 1.** Score variation with changing value of $\mu$

### 3.5 Follow-the-Regularized-Leader [FTRL-Proximal]

**Findings:** Our implementation was inspired by some approaches discussed on the Kaggle forum [8] [9] . We implemented the FTRL-Proximal algorithm both in its regularized form and non-regularized form. The FTRL-Proximal in its non-regularized form was a successful implementation. While attempting the regularized version the execution did not stop for more than 10 hours, after which we had to terminate the process. Due to the time constraint we also tried to implement it as a job on the Karst server provided by Indiana University. However, it failed on each of our 18 attempts.

The files that were used in the implementation were promoted content, event, click train and click test. A join was performed between the promoted content and train using the ad id's. Further this file was joined with the event data using the display id as the common element. Finally we added a term which showed the clicked label of the record. At each step we were hashing the data for producing a sparse matrix at the end. Making the data sparse helps in getting better results from the model. The overal time for execution was 4203.087458002 seconds. In all our implementations we obtained a score of 0.64174 which was an improvement of 0.00456 translating into a significant 0.715 Some of the parameters that were set for non-regularized approach were (Please refer to the algorithm section above to better understand the parameter): D = $2^{20}$, $\alpha = 0.1$,$\beta = 0.0$, $L_1$ = 0.0 and $L_2$ = 0.0.

**Conclusions:** FTRL-Proximal with no regularization was the model that provided the best output. It reconfirmed that our approach was in the right direction.

**Limitations:** FTRL-Proximal in its non-regularized form is almost equivalent to stochastic gradient descent. With our only successful implementation being this we were not able to fully leverage the aspects of logistic regression that we were aiming at. This method is a complex model and is definitely a non intuitive one.

## 4. Summary and Conclusions

We investigated methods for prediction using Naive Maximum likelihood Estimation, Likelihood estimation using Probability smoothing approach and FTRL-Proximal, which is type of logistic regression with stochastic gradient descent. From our implementation the FTRL-Proximal permed the best. Please refer to figure 2 for understanding the score improvment.

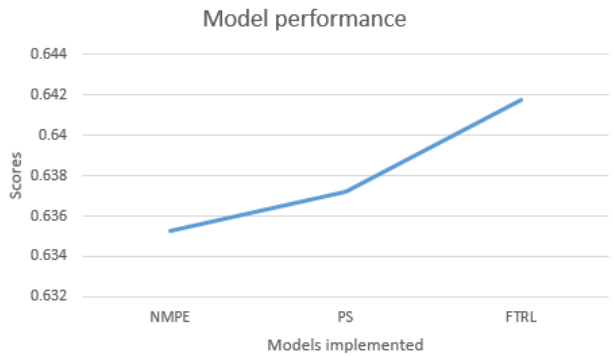


**Figure 2.** Score improvment chart

The main challenge that we faced in the project was to find ways to efficiently handle huge data with limited resources. We were however able to leverage the Karst (Indiana University server) and Sharks (SoIC server) The main advantage of using FTRL-Proximal [5] was that we could do computation by reading the data one line at a time.

Using interaction variables in FTRL Proximal helps improve the prediction many times. Interaction variables are combination of two or more variables. Similarly methods like Field-aware Factorization machines (FFM) and Vowpal Wabbit have shown to better predictive models when it comes to sparse data like ad click through rate. We plan to try implementing these in the future.

Similarly we also recommend to look at the given problem as a time series analysis challenge as the the data is spread out across a duration. For instance if there is a sale or a festival season during a period certain ads may receive a better click through rate.

Another way of looking at the problem would be to approach it form a users point of view. If a user is accustomed to click on ads frequently, if presented with a new ad the probability that he/she would click on the ad is more than the average predicted click through rate on the ad.

## Acknowledgments

The authors wishes to thank Prof. Mehmet Dalklic for numerous helpful discussions and comments. We also wish to thank Hasan Kurban, lead AI for his guidance and suggestions during the project.

We would like to thank Indiana University Bloomington, School of Informatics and Computing and UITS for providing excellent infrastructure like the Sharks and Karst servers which encouraged us to implement these algorithms. Such implementations are extremely difficult and time consuming on personal machines it would not have been possible without their help. Finally we also want to extend thanks to the Kaggle community for their varied analysis and approaches that inspired us to solve the problem.

## References

[1] Simple and scalable response prediction for display advertising
Olivier Chapelle - Criteo, Eren Manavoglu-Microsoft, Romer Rosales-LinkedIn, ACM Trans. Intell. Syst. Technol. V, N, Article A (January YYYY)

[2] Predicting Ad Click-Through Rate: Strategies For Incorporating State Dependence and Random Effects
https://www.kochava.com/predicting-ad-click-rate-strategies-incorporating-state-dependence-random-effects/

[3] A Study of Smoothing Methods for Language Models Applied to Information Retrieval,
Chengxiang Zhai and John Lafferty, Carnegie Mellon University, ACM Transactions on Information Systems, Vol. 22, No. 2

[4] Probability Smoothing
Djoerd Hiemstra, University of Twente, http://www.cs.utwente.nl/hiemstra

[5] Ad Click Prediction: A View from the Trenches,
McMahan, H. Brendan and Holt, Gary and Sculley, et al., Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

[6] Recommendations with a Purpose,
Jannach, Dietmar and Adomavicius, Gediminas, Proceedings of the 10th ACM Conference on Recommender Systems

[7] Outbrain click prediction,
https://www.kaggle.com/c/outbrain-click-prediction

[8] FTRL Starter (with leakage vars)
https://www.kaggle.com/sudalairajkumar/outbrain-click-prediction/ftrl-starter-with-leakage-vars/discussion

[9] FTRL starter code
https://www.kaggle.com/jiweiliu/springleaf-marketing-response/ftrl-starter-code