Progress Report: Predicting Which Recommended Content Users Click

Stanley Jacob, Lingjie Kong

Machine learning models can be used to predict which recommended content users will click on a given website. The given dataset contains millions of samples that map some feature about an ad or web page to a number. We reduced this dataset to a more manageable size to minimize computation time, and we extract features based on this reduced set. The features we extracted are based on the advertisers and campaigns associated with the advertisements in the dataset. We initially built models based on Naive Bayes and logistic regression. We also built a model based on the support vector machine (SVM) using hinge loss. In addition, we constructed a neural network using the multilayer perceptron model to capture the non-linearity of features in order to obtain a better prediction score. The best result we obtained was using SVMs, and it yielded an accuracy of 0.46. The result can be scaled up to the complete dataset by optimizing our implementation using parallel computing.

1 Introduction

The problem we seek to solve is to use minimal information about an advertisement and a web page to predict which ad a user is likely to click under a given web page. Several Internet companies strive to solve this problem to maximize profits. Outbrain is one such company. It has provided a dataset composed of web pages with some set of advertisements, one of which a user has clicked. One of the goals of an advertisement company like Outbrain is to target advertisements with minimal information about the user. They need to link sparse information such as the topic of the article in order to narrow down which advertisements to present to the user. In addition, they would likely want to know which advertisements perform better than others in order to attract more clicks in the future.

The goal of our project is to use the sparse information provided by Outbrain to accurately determine which advertisement the user will click from a given set. The input to our model is the web page id, the set of ad ids associated with suggested content on that web page, and the advertiser and the campaign ids associated with each ad. The output of our model is one ad id that is clicked on a given web page by the user. Given a page of some set of advertisements, we seek to classify which advertisement has the highest probability of getting clicked. The models that we use for prediction include Naive Bayes, logistic regression, support vector machines (SVMs), and the multilayer perceptron.

2 Related Work

During our research, we found several groups that sought to use machine learning techniques in order to determine whether an advertisement got clicked. A team from Google published an article describing how they present advertisements to users based on search queries [1]. They describe their use of probabilistic methods to reduce the number of features in order to save memory. In addition, they report best results for the logistic regression model with regularization along with the use of modified gradient descent algorithms that allow them to save memory while maximizing prediction accuracy. A key difference between our project and Google's task is that we do not have a large amount of information about the users in our dataset, and we have to run more feature extraction techniques in order to obtain features that we can predict on.

A team from Facebook wrote a similar paper to that of the Google team, but they discuss combining the results from logistic regression and boosted decision trees for a more accurate solution [2]. The Facebook team also has more access to user history and ad history, and they use real-time data to update their model daily. They also described how setting the learning rate to a formula they refer to as the per-coordinate learning rate resulted in a better accuracy when they ran gradient descent.

Another team from USC also worked on a project similar to ours using a dataset from Kaggle that sought to determine whether an advertisement is clicked or not [3]. This team faced an issue similar to ours regarding how to reduce the number of feature used in their models and choosing which features to select. They reported that logistic regression with L2 regularization gave the best results.

Two teams from Microsoft described a logistic model and a Bayesian model used to display ads on the Bing search engine. One team describes a problem of determining optimal ordering of advertisements, and they use click-through rate (CTR) as

their metric of determining if an ad is more likely to get clicked [4]. They solve their problem using logistic regression and textual information about the ads. In addition, they noted that an ad that appears first is more likely to get clicked. For our dataset, the positioning of the ads are not given, and we also do not have textual information about the ads. Another team used CTR with Naive Bayes, and they describe how they prune their model when they determine certain features are not as important [5]. We use CTR in our model to determine which advertisers and campaigns are more likely to be clicked.

3 Dataset and Features

The data we obtained is from Outbrain, a company that seeks to target advertisements for users visiting a given website. The dataset contains millions of sets of advertisements, where each set refers to a web page that has around five advertisements [6]. Additionally, Outbrain provided several files that together form a relational database that links each user, website, advertisement, and advertiser together. However, having several separate files provided a pre-processing challenge. We created several dictionaries connecting the files together and exported them to json files in order to quickly access them later.

Much of the given dataset is categorical in nature. The categories are all numbers since Outbrain wanted to anonymize the data. Thus, we do not have information regarding the actual text or image of the advertisements or the web pages themselves. The data does not reveal anything about the user aside from their location and the time stamp when they visited the particular web page and clicked on one of the ads on that web page. Also, users in the training data set most of the time will not show on the testing set. Thus, our initial process was to select features that directly relate to the advertisement. We first sought to determine if certain advertisers and campaigns of an advertisement resulted in more clicks. We then included the web page's topic, source, and category as feature as well. Figure 1 below shows the hierarchy of how these features given from the dataset are related.

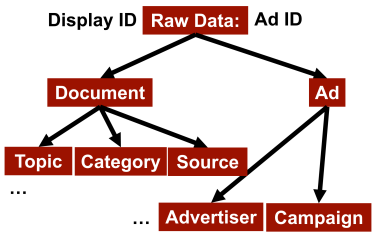

Figure 1: Data Visualization

We needed to find ways to reduce the number of categorical variables to a more manageable number. Based on the events.csv file, the total number of unique advertisers in the dataset is 4385 and the total number of unique campaigns is 34675. Additionally, there are 559583 unique ads in the dataset, and each ad id itself can be mapped to a categorical variable. Trying to extract features for each individual ad, advertiser, and campaign result in an infeasible number of features to train on.

In order to reduce the number of features, we decided to transform the original problem to a simpler version where the complete dataset is the first $1,000,000$ set of ads, which corresponds to the first display ids in the train.csv file. This reduced dataset corresponds to a subset of ads, advertisers, campaigns, and web pages. We then had indicator functions for a subset of the total advertiser and campaign id. Thus, the total dataset was reduced down to this smaller model in order to allow us to try implementing richer features and for models involving decision trees or neural nets to have access to most of the information for a given combination of display id and ad id.

4 Methods

Naive Bayes

We implemented the Naive Bayes algorithm with the assumption that the conditional probabilities associated with our model are independent. To be more specific, each advertisement getting clicked given the campaign and the advertiser of the given ad is conditional independent. We also implemented Laplace Smoothing in order to incorporate advertisements that we have never seen before when running our Naive Bayes model on the test set.

The equation for Naive Bayes is shown in (1) in which X is advertiser and campaign feature. Y = 1 means a ad is clicked and Y = 0 means a ad is not clicked

$$P(Y|X) \propto \prod_{i=1}^{n} P(X^i|Y)P(Y) \tag{1}$$

Under the condition that we know every time that one of multiple specific advertisements will be clicked, we do not really have the prior probability P(Y). In other words, our Naive Bayes equation will end up being simplified to P(X|Y) only. This is the major reason why Naive Bayes might not be a good model here.

The modified Naive Bayes equation (2) without prior probability is

$$P(Y|X) \propto \prod_{i=1}^{n} P(X^i|Y) \tag{2}$$

Logistic Regression

We additionally implemented logistic regression using features related to the advertiser and campaign that ad is provided by as well as topic, source, and category that website is related to as the feature. The features that we used are essentially indicator functions for each advertiser that returned 1 if the given ad was created by that advertiser. Similarly, we created indicator functions for each of the possible campaigns as well. By running stochastic gradient descent on the training data, we determined appropriate values for the weights of the features. In addition, we plan to use L2-norm regularization to prevent the overfitting of the training dataset, though the result we describe below does not include it.

The loss function (3) we used for the logistic regression is

$$J(\theta) = -(1/m)\sum_{i=1}^{m} \log(h_\theta(y^i x^i)) \tag{3}$$

h is the sigmoid function (4)

$$h_\theta(y^i x^i) = 1/(1+\exp(-y^i \theta^T x^i)) \tag{4}$$

As for the stochastic gradient descent in equation (5), we will update based on

$$\theta^i := \theta^i - \eta \nabla \log(-h_\theta(y^i \phi(x)^i)) \tag{5}$$

Support Vector Machines

We additionally sought to classify the data using SVM with a hinge loss function. We sought to do so since we initially thought that the loss function associated with the logistic regression was incorrect and thus led to a lower accuracy than desired. Also, SVM will fit the data well enough and stop. In other words, SVM should be able to prevent potential overfitting compared to logistic regression. The overall procedure is the same as logistic regression except we use the hinge loss in equation (6) function rather than the sigmoid.

$$J(\theta) = (1/m)\sum_{i=1}^{m} \max(0, 1 - y^i \theta^T x^i) \tag{6}$$

Multilayer Perceptron

The multilayer perceptron (MLP) model is utilized in order to learn an optimal combination of weights for the sparse features that we used from the dataset. The motivation for using the MLP model is to capture non-linearity relationships between the features that we captured. Our goal is to use a large number of hidden layers to learn the mathematical relationship between the clicked advertisement and the associated campaign and advertiser ids. We use the tanh function as the mapping to the hidden layers and take the output as a linear combination of the hidden layers.

The algorithm of Multilayer Perceptron should be similar to most methods that we tried above. The goal is to find the weights w that capture the input output mapping implicitly in a dataset of the samples. The algorithm is briefly shown below.

Result: Calculate weights for each nodes
Model the objective function specify size of hidden layer, node, activation function, and other hyperparameters
for *each iteration t = 1, ... T:* **do**
  for *each weight w:* **do**
    Back-prop learns the weights through gradient descent for each node at each hidden layer
    if *convergence* **then**
      Apply early stop
    end
  end
end

Algorithm 1: Multilayer Perceptron

5 Experiment

We will describe two approaches that we took in solving this problem. Both involve reducing the dataset with the assumption that our model will scale for the complete dataset. The first approach involved writing our own Naive Bayes, logistic regression, and SVM models. This approach involves computing the gradient of the loss functions for each model and using stochastic gradient descent to determine appropriate weights. For the test set, we used the weights to compute the loss and the smallest loss is associated with the clicked ad. Additionally, for this first approach, we selected features related to the advertiser and campaign id being clicked over the entire train set. To reduce the number of features in our model, we initially limited the features to be indicator functions related to the first 50 most clicked advertisers and campaigns. With further testing, we noticed

that including all of the possible advertisers and campaigns improved the overall accuracy. Our second approach involved using SKLearn's library and using their function for predicting probabilities that a given ad belongs to the clicked class. We then selected the ad in a given set that had the highest probability for being clicked.

For both approaches, we used a similar training and testing method involving cross-validation, and we selected features that maximized the train and test accuracy. Before we ran our experiments, we reduced the dataset from millions of samples to one million. We then split up this reduced dataset into 30 percent set aside for determining the test accuracy and 70 percent set aside for the training. For the first approach described previously, we tuned hyperparameters related to the learning rate and regularization term using our train set. For the second approach, we split the training portion again into 80 percent and 20 percent. The 80 percent of the train set is passed into our modifications of SKLearn's models in order to compute probabilities associated with each display ad pair. We then chose the ad within each display which has the maximum probability associated with the clicked class. We tuned hyperparameters related to the learning rate, regularization parameters, and optimization algorithm in order to maximize our accuracy on the 20 percent set aside from train. Finally, once the hyperparameters were tuned, we took the result and used it on the test set to compute the test accuracy.

Results

Figure 2 and Table 1 below contain the results of our first few experiments in which we built our own Naive Bayes model, Logistic Regression model, and SVM model based on the advertiser and campaign ids associated with each clicked ad. The metric that we used to determine accuracy is defined by the fraction of times our model correctly chose which ad was clicked by the user.

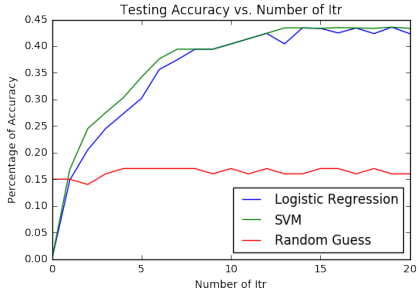

Figure 2: Testing Accuracy vs. Number of Iteration

Table 1: 1st Experiment Result

Method	Train accuracy	Test accuracy
Naive Bayes	0.45	0.45
Logistic Regression	0.51	0.46
SVM	0.52	0.48

As for the second experiment, we use SKLearn's library to validate our results from the first experiment through built-in optimizers and models in SKLearn library. We first compress the data into 2D by using PCA, which allows us to visualize how well are the features we selected based on given dataset.

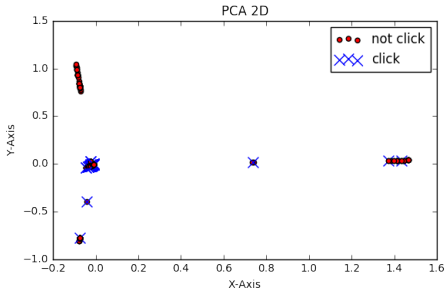

Figure 3: PCA

As shown in the PCA result in Figure 3 above, the clicked and non-clicked examples are mixed together and therefore cannot be easily shattered by a simple hypothesis. The PCA plot also shows insight into why we obtained a low accuracy from the first experiment.

Below are the results of our second set of experiments in which we used SKLearn's library to build models for Naive Bayes, Logistic Regression, SVM, and Multilayer Perceptron. The result from our both experiments match. This again proves that our own codes in the first experiment are correct.

Table 2: 2nd Experiment Result

Method	Train accuracy	Test accuracy
Naive Bayes	0.45	0.45
Logistic Regression	0.48	0.46
SVM	0.49	0.46
Multilayer Perceptron	0.38	0.32

Discussion

The accuracy of our models was limited mainly due to the minimal features available from the original dataset. We did not have access to textual information about the advertisement, the advertiser, the campaign, or the user. Instead, we had numbers that we had to transform to indicator functions. However, this method results in thousands of features that exceed the number of samples we took from the original dataset. We found that extracting each ad id, campaign id, and advertiser id did not significantly improve the accuracy. Additionally, we noticed that using PCA to compress the features does not improve

the accuracy and made it difficult to understand what feature mainly led to user clicks.

It is difficult to extract features with an extremely large dataset, so cutting down the dataset allowed us for greater flexibility for computation. However, we found that reducing the dataset in certain ways could lead to biases in the train and test accuracies. For example, one initial experiment we attempted was to focus on the sets of ads shown to different user for only one particular web page. However, several web pages have a small number of ad sets associated with them, so training and testing on them resulted in high variance. Sampling large numbers of web pages and associated sets of ads greater than 10,000 examples from the dataset resulted in the accuracies obtained in the previous tables.

The best model that we have determined so far is SVM with hinge loss using our own code rather than SKLearn's library. The Naive Bayes model and the logistic regression model yields a similar test accuracy, but the SVM model outperformed these other two models in determining an optimal boundary. The reason why our own code works better is because we directly compute the loss for the click rate associated with each advertiser and each campaign id for an ad given a display id, and then we decide that the ad with the least loss under that display will be clicked. While the SKLearn's library makes prediction on each ad without utilizing the information from display. Considering we have biased data on the non-clicked ads compared to the clicked ads. Obviously, none of the ads will be clicked. Therefore, we need to check the probabilities for each ad under a display to see which ad will most likely to be clicked. However, we are unsure how probabilities are calculated in SKLearn library. Our better accuracy implies that trying to use probabilities is not as reliable and is subject to statistical errors due to being so similar.

We ran experiments on several other models that tended to overfit the data most likely due to the large number of features used. Although we expected the MLP model to result in a better accuracy since it captures more non-linearity, it did not yield a better train accuracy most likely due to overfitting or the lack of hidden layers or nodes. We additionally tested the second approach with Random Forests and the XGBoost library and found that the probabilities associated with classification for these models also could not be used reliably to predict which ad got clicked. Improved results may be obtain by using another metric to choose which ad is clicked.

6 Future Work

The accuracy of click prediction can be improved by clustering the users and by optimizing our code using parallel computing in order to scale our model for the complete dataset. A modified k-means clustering algorithm can be used to group users based on their location and time stamp, and this clustering can be used as an additional feature. Another model that could be implemented is the Field-aware Factorization Machine, which can be used to vectorize the campaign ids and advertiser ids so that the calculation between overlapping combinations of campaign ids and advertiser ids can be more accurately calculated [7]. By leveraging cloud computing, we can split the csv files into smaller files that we can build models for in parallel. These models could be combined for a more accurate model.

The most common ad click prediction approach is implemented based on user information, such as previous click history. However, due to the lack of detailed user information in the dataset, our approach yields a low accuracy so far. The next step would be to find patterns in the information in the dataset relating to the user's location and the timestamp the user visited the web page. We expect that by combining the user information and advertisement features could help yield a higher accuracy.

7 Conclusion

We built an SVM model for ad click prediction, and we achieve an accuracy of 0.46 for our test set. We also obtain similar accuracies using Naive Bayes and logistic regression. The optimization algorithm we used is gradient descent in order to learn the model weights based on a sample of a large dataset. The difficulty of building a model for click prediction lies in feature extraction, and we selected features based on the advertisers and campaigns associated with the clicked ads. We also built models that used SKLearn to determine the highest probability associated with the clicked class in order to choose which ad got clicked. By scaling the models to the entire dataset, the accuracy of click prediction can be increased.

8 References

[1] McMahan, Brendan et al. "Ad Click Prediction: a View from the Trenches." Available: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41159.pdf.

[2] He, Xinran et al. "Practical Lessons from Predicting Clicks on Ads at Facebook." Available: https://pdfs.semanticscholar.org/daf9/ed5dc6c6bad5367d7fd8561527da30e9b8dd.pdf.

[3] Kondakindi, Gouthami et al. "A Logistic Regression Approach to Ad Click Prediction." Available: http://www-scf.usc.edu/ vparakh/ MachinLearning-Project.pdf.

[4] Richardson, Matthew et al. "Predicting Clicks: Estimating the Click-through Rate for New Ads." Available: http://dl.acm.org/citation.cfm?id=1242643.

[5] Graepel, Thore et al. "Web-scale Bayesian Click-through Rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine." Available: http://quinonero.net/Publications/AdPredictor ICML2010-final.pdf.

[6] "Outbrain Click Prediction." Available: https://www.kaggle.com/c/outbrain-click-prediction.

[7] "Field-aware Factorization Machines for CTR Prediction". Available: http://www.csie.ntu.edu .tw/ cjlin/papers/ffm.pdf.