Machine Learning Engineer Nanodegree

Capstone Proposal

Armen Donigian

November 22nd, 2016

Proposal

Domain Background

The following proposal is a machine learning project to build a predictive model in response to the **Kaggle Outbrain Click Prediction**.

> Every day we stumble on news stories relevant to our communities or experience the serendipity of finding an article covering our next travel destination. Outbrain, the web's leading content discovery platform, delivers these moments while we surf our favorite sites.

The competition began on October 5, 2016 while the final submission must be made by Jan 18th, 2017. Only two submissions per day are allowed. The top 3 of the leaderboard are:

Competitor	Score
Tick Tock	0.69466
xiaohaoxx	0.69128
clustifier	0.68755

This project is of special interest to me due to it's relevance with user content consumption and advertising. Any insights as well as lessons learned would be directly transferrable to a wide variety of problems across several business verticals. Additionaly, Kaggle competitions offer a great opportunity to solve real world problems in a collaborative, competitive and open learning envionrment. Doing well on Kaggle competitions helps enhance my personal experience and online brand.

Problem Statement

Currently, Outbrain pairs relevant content with curious readers in about 250 billion personalized recommendations every month across many thousands of sites. In this competition, Kagglers are challenged 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.

The task is to rank the recommendations in each group by decreasing predicted likelihood of being clicked. This is a **supervized learning** problem which will require **a multi-class classifier** to be implemented. The predictors (features) are attributes of page views, user clicks, promoted content, and web page content. The labels (target) are the id ads. See example submission below for details.

Datasets and Inputs

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. For full details, see data specifications below.

The dataset contains numerous sets of content recommendations served to a specific user in a specific context. Each context (i.e. a set of recommendations) is given a display_id. In each such set, the user has clicked on at least one recommendation. The identities of the clicked recommendations in the test set are not revealed. The task is to rank the recommendations in each group by decreasing predicted likelihood of being clicked.

Data Fields

Each user in the dataset is represented by a unique id (uuid). A person can view a document (document*id), which is simply a web page with content (e.g. a news article). On each document, a set of ads (ad*id) are displayed. Each ad belongs to a campaign (campaign*id) run by an advertiser (advertiser*id). You are also provided metadata about the document, such as which entities are mentioned, a taxonomy of categories, the topics mentioned, and the publisher.

File	Dimensions (rows, columns)	Text or Numeric
clicks_train	87141731, 3	Numeric
clicks_test	32225162, 2	Numeric
documents_categories	5481475, 3	Numeric
documents_entities	5537552, 3	Alpha Numeric
documents_meta	2999334, 4	Alpha Numeric
documents_topics	11325960, 3	Numeric
events	23120126, 6	Alpha Numeric
promoted_content	559583, 4	Numeric
page*views*sample	9999999, 6	Alpha Numeric

Pre-processing

There are several pre-processing steps which will need to be performed, including but not limited to: + join several segments (tables of data) with specified join keys + identify any gaps of referential integrity (missing rows from any segments during join) + apply log transformations for feature scaling + parse geo-location field into seperate features + convert string date representations to date variables (will eventually add features for day, month, year as part of feature engineering) + identify null & missing data, apply a mnemonic or drop it (depending on size & impact relative to other training data)

File Descriptions

page_views.csv is a the log of users visiting documents. To save disk space, the timestamps in the entire dataset are relative to the first time in the dataset. If you wish to recover the actual epoch time of the visit, add 1465876799998 to the timestamp.

```
uuid
document_id
timestamp (ms since 1970-01-01 - 1465876799998)
platform (desktop = 1, mobile = 2, tablet =3)
geo_location (country>state>DMA)
traffic_source (internal = 1, search = 2, social = 3)
```

clicks_train.csv is the training set, showing which of a set of ads was clicked.

```
display_id
ad_id
clicked (1 if clicked, 0 otherwise)
```

clicks_test.csv is the same as clicks*train.csv, except it does not have the clicked ad. This is the file you should use to predict. Each display*id has only one clicked ad. Note that test set contains display_ids from the entire dataset timeframe. Additionally, the public/private sampling for the competition is uniformly random, not based on time. These sampling choices were intentional, in spite of the possibility that participants can look ahead in time.

sample_submission.csv shows the correct submission format.

events.csv provides information on the display_id context. It covers both the train and test set.

```
display_id
uuid
document_id
timestamp
platform
geo_location
```

promoted_content.csv provides details on the ads.

```
ad_id
document_id
campaign_id
advertiser_id
```

documents_meta.csv provides details on the documents.

```
document_id
source_id (the part of the site on which the document is displayed, e.g. edition.cnn.
com)
publisher_id
publish_time
```

Solution Statement

A potential solution to the problem would look something like the following:

```
display_id,ad_id
16874594,66758 150083 162754 170392 172888 180797
16874595,8846 30609 143982
16874596,11430 57197 132820 153260 173005 288385 289122 289915
etc.
```

Let's break this down:

- For each display*id in the test set, I will predict a space-delimited list of ad*ids, ordered by decreasing likelihood of being clicked
- The candidate ad*ids for each display*id are provided in clicks_test.csv.

Benchmark Model

Given this is a Kaggle competition, the benchmark model will be the score of my model on the private leaderboard (data I don't have access to). Additionally, I plan to create an initial benchmark model after generating the training data set without doing any feature engineering or hyper-parameter tuning just to have a point of reference for the rest of the model training process.

My goal is to target the top 20% of the public leaderboard.

Evaluation Metrics

Given this is a Kaggle competition, the evaluation metric has been specified as the **[Mean Average Precision @12]**.

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

The mean average precision metric intutively is suitable for advertising since we are interested in how precise our ad predictions are given historical data collected from users.

Project Design

The first step to any Machine Learning project is to understand the data as well as the problem (what is the target). The dataset contains numerous sets of content recommendations served to a specific user in a specific context. The task is to rank the recommendations in each group by decreasing predicted likelihood of being clicked.

Exploratory Data Analysis

I will perform Exploratory Data Analysis (EDA) to explore and better undertand the data. EDA will include the following:

- examine data types, handle incorrect / non-conforming
- ensure proper join keys exist to join all relevant data into training data set (tabular form)
- examine missing values
- examine outliers
- create correlation plots of different dimensions relative to one another
- examine feature distributions against target (guassian, few unique, heavy-tailed)
- examine target variable
- create visualizations of insights discovered from analysis

Feature Transformation & Engineering

- does data need to be standardized?
- convert categorical features to numerical
- engineer new features from existing features & accordingly to domain knowledge

Next, I would do principle component analysis to better understand the correlation of the data. For example, say you have 100 features, is it really 100 dimensions or fewer. Look at your eigenvalues in order (biplot) and see if it decays fast or slowly. If the dimensions drop off pretty quickly then it implies there's a lot of structure in your data and that you need less dimensions. On the other hand, if it decays more slowly then it's harder to find structure.

Since this is a supervised learning problem, decision trees & variations of tree models are worthwhile. The types of algorithms to try are:

- Decision Trees
- Random Forrests
- Gradient Boosted Machines
- XGBoost
- Deep Learning Model using TensorFlow

Consider spending more time tuning the hyper-parameters for better results. I would try deep learning method in the event there's a non-linear structure which decision trees weren't able to pick up on.