Machine Learning Engineer Nanodegree

Capstone Project

Bárbara Barbosa April 7st, 2019

I. Definition

(approx. 1-2 pages)

Project Overview

Display advertising is a billion dollar effort and one of the central uses of machine learning on the Internet. However, its data and methods are usually kept under lock and key.

In this project, I develop a model to predict ad click-through rate (CTR). Given a user and the page he is visiting, what is the probability that he will click on a given ad? The application uses online learning with some parallel computing techniques to solve the problem. Inspired by this challenge at kaggle.

Problem Statement

The goal is to create a model to predict when a user will click on a given ad; the tasks involved are the following:

- Download the dataset presented at CriteoLabs.
- Use Dask to understand the anonimized data
- Create a pre-processing for the numerical data
- Create a pre-processing for the categorical data
- Create a classification model to deal with a dataset of 11.1 Gb
- Submit the solution at the Kaggle competition using the text data (that has 1.5 Gb)

  The final model is expected to get a similar result to the ones presented as winners for the competition.

Metrics

The evaluation metric is log-loss. This metric is frequently used for classification models where the final result is a probability. Log loss increases as the predicted probability diverges from the actual label.

The logloss is defined by the equation bellow from. Except that in scikit-learn the equation is negative, so the greater the better (as other metrics).

-log P(yt|yp) = -(yt log(yp) + (1 - yt) log(1 - yp))

where yt is the true label and yp is the predicted probability.

II. Analysis

(approx. 2-4 pages)

Data Exploration

The full dataset exploratory data analysis can be found at DataExploration.ipynb.

The dataset can be obtained at (http://labs.criteo.com/2014/02/download-kaggle-display-advertising-challenge-dataset/) and consists of a weak of data from Criteo, which has:

- 6 data centers on 3 continents
- More than 7000 servers and a few tens of Petabytes of storage on our HPC cluster
- 30B HTTP requests and 3B unique banners displayed per day
- Peak traffic of 800K HTTP requests per second
- Respond to bids in 80ms or less, 24/7

The datasets and file descriptions are obtained from kaggle: https://www.kaggle.com/c/criteo-display-ad-challenge/data

Data fields Label - Target variable that indicates if an ad was clicked (1) or not (0). I1-I13 - A total of 13 columns of integer features (mostly count features). C1-C26 - A total of 26 columns of categorical features. The values of these features have been hashed onto 32 bits for anonymization purposes. The semantic of the features is undisclosed.

When a value is missing, the field is empty.

Since the dataset is anonymized, it's hard to obtain domain knowledge from it. Because of that all the analysis are some simple statistics like mean, standard deviation and median.

categorical11	0.00
categorical10	0.00
categorical9	0.00
categorical8	0.00
categorical7	0.00
categorical14	0.00
categorical5	0.00
categorical2	0.00
categorical1	0.00
categorical15	0.00
categorical18	0.00
categorical23	0.00
interger2	0.00
categorical17	0.00
categorical13	0.00
interger8	0.00
interger5	0.02
categorical16	0.03
categorical12	0.03
categorical3	0.03
categorical4	0.03
categorical21	0.03
categorical24	0.03
interger11	0.04
interger9	0.04
interger7	0.04
categorical6	0.12
interger3	0.21
interger13	0.21
interger4	0.21
interger6	0.22
categorical19	0.44
categorical20	0.44
categorical25	0.44
categorical26	0.44
interger10	0.45
interger1	0.45
categorical22	0.76
interger12	0.76

As we can see from the table above, there are variables as interger12 , categorical22, interger1, interger10, categorical26, categorical25, categorical20, categorical19 that have more than 30% of nulls. Those variables will be disconsidered.

Because of the anonimization, an analysis of levels was made to verify which feature engineering would be the better.

The analysis showed that some categories have 10131226 levels and some others have just 3. These huge differece between categories are really important to understand later strategies of feature engineering.

As we can see from the distributions from the integer data, there are a lot of variables with outliers, and the mean is far from the median

for a lot of those variables.

Algorithms and Techniques

The classifier is a logistic regression training using an online learning approach. Since I have more data than memory available in my computer, I had to find a manner to train the algorithm, and online learning is very suitable for large datasets. Online machine learning is a method of machine learning in which data becomes available in a sequential order and is used to update our best predictor for future data at each step, as opposed to batch learning techniques which generate the best predictor by learning on the entire training data set at once.

The following parameters can be tuned to optimize the classifier:

- penalty (regularization)
- max_iter
- learning_rate
- alpha
- Preprocessing parameters (see the Data Preprocessing section)

The train dataset was chunked into 327433 rows. This was made to allow to model to run offline at my notebook.

Benchmark

Our benchmark model will be the third place winner of the competition. That used a logistic regression with a quadratic/polynomial feature generalization. In it's paper (file:///home/barbarabarbosa/Downloads/Display-ad-challenge-Song.pdf) the winner is not very specific in what features are combined to generate that combinations.

The benchmark model has a log-loss of 0.44610 but since this is a kaggle competition, and the main objective is to win the score by super minor changes at the leaderboard, we will focus on obtaining at least a model with 0.50 of log-loss. Which will give me a position of 500th at the leaderboard.

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

Here we are only considering interger2, interger8, interger5,interger11, interger9, interger7 as continuous variables and categorical8, categorical14, categorical5, categorical12, categorical11, categorical10, categorical7, categorical1, categorical15, categorical18, categorical13, categorical16 as categorical variables. Those were chosen because the low number of nulls and amount of levels at the category.

All the integer nulls were replaced by zeros and all the categorical variables were replaced by a string 'NULL'.

The first iteration is the fit for the z-score, obtaining the mean and the standard deviation of the continuous data. For the all the next iterations, this first data is used to calculate de z-score for other batches. For the categorical data, a feature hashing was used. All the features use the same hash size: 1000. The hash function is the same, so we don't need to do anything in different iterations.

An other approach was to use PCA for dimensionality reduction and keep only 7 features.

Implementation

The implementation process can be split into two main stages: 1. The classifier training stage 2. Testing the application at the kaggle competition

After the pre processing phase the only thing to do was train the model. The first iteration was the fit and the iterations after that were a partial_fit, that uses information from the previous fit to keep improving the model.

The test phase was done using a for loop and using chunks as well. For the kaggle competition a .csv is need to be made, and the evaluation was made using the prediction of the algorithm and the dataset indexes.

The implementations are made using the SGDCatTrain.py and PCATrain.py to train a model using categorical variables and a model that uses PCA respectively.

Refinement

Differently from the benchmark solution, we tried to make a dimensionality reduction using PCA after the regular pre processing. Because of the size of the data, all solutions had to be made in a python file, because the jupyter notebook kernel keeps dying.

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

The original approach, without PCA, had better results than the PCA one.

Since the logloss is better as it grows smaller, we can see that the PCA had a little lower performance than the original one.

The images bellow show the results presented at the kaggle competition.

[!original.png](Resultados da abordagem original)

[!pca.png](Resultados da abordagem com PCA)

Justification

As said in the proposal, those kaggle competitions are optimized to reach the best ranking. Perhaps using all the variables and a better replacement of null data could give better results. Although that, the model could had a better performance using differente PCAs variables.

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

Since the data is really huge, I wasn't able to find a visualization library to plot the data and used Dask to generate statistics on the data.

Reflection

The main challenge in this project was handling the big data that didn't fit my machine's memory. At first I tried to use Dask, but I had too much trouble finding the correct model and dealing with the errors. The only approach that worked was using online machine learning. The next challenge was to find an ideal chunk size, to maximize the size and don't end up with a memory error.

The next challenge was the pre-processing phase, since we have lot's of levels at the categorical data, and we couldn't process the whole dataset in memory, I had to study techniques that could handle this problem. The only knowledge from the course was the logistic regression, all the rest had to be studied just for this challenge.

Improvement

I believe that a more robust classifier could be made with a better pre processing phase. Perhaps using the median to replace nulls, using different hash sizes for different categorical variables. Also a non linear approach could also be great, but it wasn't possible to achieve using online learning.