RTB ad price predictions

Shay Bozo (ID. 301166278), Edan Hauon (ID. 305249187)

Submitted as final project report for Deep Learning, IDC, 19/09/2019

1 Introduction

1.1 Fyber - Business background

- I work at Fyber (Shay) as an Server team lead and choose to do the project on something that is close to my day to day work
- Fyber is a company that its main product act as an ad exchange platform
- Our users are publishers that own the free content app and their business model is running ads
- The publishers install our SDK and when they want to present an ad they activate a function in our SDK that sends an 'ad request' to our servers. The request contains various information about the user, device, app, and physical location.
- The ad service performs the following actions upon receiving the ad request
  - Enrich the request with data from external services
    - MaxMind
    - WURFL
    - 42Matters
    - Please see 'Features explained' appendix
  - Sends bid request with most of the information aggregated and with a minimum floor price to the relevant ad networks
  - Selects the highest bidder from the ad networks that responded with a bid above the floor price
  - Return to the app response with the winning ad

1.2 My idea for this project - What do I try to predict?

- When an ad request comes from the app and before we send our requests to the ad networks I will use the daily trained model that was uploaded by the service in order to predict what will be the response from the ad network. I can use this prediction to a number of issues (will be decided in the future)
  - Ad request filtering - since we spend a lot of money on network costs if the prediction is very low we might choose not to send a request.
  - Changing the floor price - there is one major parameter that we can change in order to get the overall best revenue
    - Best revenue is comprised of overall performance by day of: fill rate and winning bid price
    - This use case is most relevant for cases where we used a non-dynamic floor price

1.3 Related Works

I used couple of internal works that were done inside the company but since it's proprietary I cannot reference it here.

2 Solution

2.1 Main flows

- During the run of the Viper web service collect all data and send it to Kafka
- Upload the raw data to S3
- Transform encode and filter the raw data and save it
- Train and test a DNN model
- Export and save the model in Pmml format into S3
- Load the model and use it in run time in Viper

2.2 Detailed Design

- Since we have a lot of traffic I will do this process to only percent of the traffic
- Get the dataset from existing data that is collected during the auction in real-time -¿ this step proved to be much more than I intercepted and most of my time went into enriching the data send to Kafka

- Create a new daily job in data bricks that will consist of 2 main parts
  - Data transformation that will be run in Scala DataBricks on spark so I will be able to use parallelism (we can't do it in the DNN training and testing phase)
    - Get all raw data from S3
    - Remove outliers
    - Filter and transform the data into our wanted features
    - Collect winning and losing bids
    - One hot encode the features that behave like labels
    - Encode the date-time as a number of new features (day of the week, day of the month...)
    - Tag the Y columns
    - Split into train and test data sets to 70
    - Save the prepared data into a new bucket in S3
  - Data training and testing using Databricks 'Deep Learning Pipelines' that uses mainly TensorFlow that is written in python contain karas. The downside is that I cannot work as a cluster (as spark) and this is a one master one worker architecture (no parallelism)
    - Load the prepared data
    - Feed the data into the TensorFlow DNN
    - Train and test the DNN
      - Since I have a lot of data the learning rate can be low (0.10)
      - Feed the transformed data into the network
      - The layout of the neural network
      - MLP neural network type
      - 2 hidden layers of 100 neural nodes each
      - I will use a sigmoid function as the activation function
      - I will divide the train data to 100 epochs
    - Export and save the model in Pmml format into a new bucket in S3
  - In Viper (the web service that will use the prediction in real-time) - due to lack of time I skipped this part for now
    - Load the model data in Pmml format
    - When a request arrives create an input record from the request and calculated data
    - Get the prediction from the model
    - Act according to the prediction
- More technical notes

  - Data is not an issue for us (there are billions of transactions hourly) we get it daily from the ad networks
  - Every day the model will be rebuilt based on the data of the previous 14 days. The first data transformation phase and then train and test
  - Important issue: since our predictions have a big impact on the actual revenues, in order to prevent a model that perpetuates himself use the model for 95

2.3 My tech stack

- For the Data preparation and transformation I used AWS dataBricks on Spark cluster. I used a lot of SQL manipulations at first since I am working with structured data. I broke the given data into winning and losing bids and than reassembled them into a single flat data schema. than I used Spark to one hot encode all the category features and than wrote the entire raw data to S3 storage. please see starting schema and pre encoding schema in the appendices. This code was written in Scala.
- For the training and testing phase I used a DataBricks cluster that is design to work with Python and deep neural networks. The problem is that unlike Spark it's not distributed so the train and test took about 2 hours (with a very strong cluster with GPU - 5.5 LTS ML (includes Apache Spark 2.4.3, GPU)). I used most of the python ml libraries like TensorFlow, sklearn, pandas and very good one called pyspark (Pipeline in it).

3 Experimental results

- Since the project uses internal private information I cannot provide real data to be run with
- Please see the report results with tables and figures in the appendixes for much more details

3.1 Overview

At first the data didn't make any since and it was needed to transform and rebuild it into a work ready data set (please see starting schema and pre encoding schema in the appendixes). After playing with the DNN a bit and not getting to a working solution I started to add more features (ended up with over 40), the most important ones that were added were the time features (minute, hour, day of week, etc) - apparently that in this business there is a lot of importance to when does we request an ad. I also had some success with adding the device features of screen size and running OS.

3.2 Experimental settings and alternatives

- At first I started with processing 30 days back with a cluster of a single node - that prove to be too long and than I split the job into 2 parts so I can use parallelism computing for some extent for the first data transformation. The data transformation uses Spark and written in Scala and the second training and testing uses Python with tools more like the one we were shown in class. Now the first part runs for around 1h and the second for 1/2h
- The data was splited into 0.7 train and 0.3 test - I didn't play with it much the results were fine
- The data was splited into 100 epoch
- The error rates were significantly reduced after adding the date features (see appendix)

4 Discussion

The project was a great way for me to get in quickly into the deep learning world. From analyzing the data I also learned a great deal about the business in my company. I learned that today there are great advanced tools and most of the difficulties are in getting a relevant useful data and transform it to a way it can be used for machine learning. Most of my time was spent on adding some features work on the schema and one hot encode it. Once the data was ready the insertion into the model training and testing it wasn't very hard. I must mentioned that this is not a production grade project and it needs a lot of extra work in making it money decisions taking ready. I found out that the machine learning ecosystem is not fully connected yet and the last part of the project of saving the model into a format that can be loaded in real time into a Scala web service is exterminate hard and unreliably (that is way I didn't finish this part). Also that there is no existing solution of transforming the model with it's encoding attributes in a way that I can in real time create a bit vector from an incoming request and use the model. It was a hard and great experience and now I believe that given the extra time I can create a real working DNN based machine learning complete flow in a production environment.

5 Code

In the Below link there is the final project GitHub repository that contain the 2 phases of data preparation and training and testing. Also please see all the documents such as data sachems, validation error graphs, etc.

5.1 GitHub repo

https://github.com/shaybozo/RtbAdNetworkBidPricePredictions2