Winning model documentation

Name: Changsheng Gu

Location: Beijing, China

Email: gzs_iceberg@outlook.com

Competition: [Avito Context Ad Clicks]

1. Summary

This document describes the 2th prize solution to the Avito Context Ad Clicks. The solution uses FFM, FM, XGBoost as the basic models, then a NN model is used for model ensembling.

2. Features Selection/Extraction

We test new features in first 10M records of trainSearchStream.tsv. Here're features we used.

- Raw features: AdID, CategoryID, HistCTR, IPID, IsUserLoggedOn, Params, Position, Price, SearchParams, SearchQuery, UserAgentFamilyID, UserAgentID, UserAgentOSID, UserDeviceID, UserID, searchCategoryID, searchLocationID, Title
- Artificial Features：

Feature name	Description
adid_cnt	count of the ad id in stream data
af_3h_cnt	count of records in 3 hours after the search session
af_cnt	count of records after the search session
bf_3h_cnt	count of records in 3 hours before the search session
bf_cnt	count of records before the search session
bf_clk_cnt	count of click records before the search session
bf_ctr	click through rate before the search session
ca_match	matched category id
ca_pid_match	matched parent category id
clk_cnt	historic click count of user on some ad.
hl_lcnt	count of highlighted ads below the position in search session
hl_ucnt	count of highlighted ads above the position in search session
ot*_cnt	count of different object type in search session
pos_ot_type	hash value of object type tuple in search session
pos_type	hash value of position tuple in search session
price_pos	rank by price in search session
price_ratio	divide price by average price in search session
qe_ng_cnt	count of matched 2-ngram words between query and title
qe_ng_min_pos	earliest position of matched 2-ngram words between query and title
qe_ng_ratio	ratio of matched words 2-ngram between query and title
qe_w_cnt	count of matched words between query and title
qe_w_pos	earliest position of matched words between query and title

qe_w_ratio	ratio of matched words between query and title
record_cnt	count of records in search session
show_cnt	historic impression count of user on some ad.
t_cnt	total search count of user
t_match	check if query is in the title.
t_show_cnt	total impression count of user on some ad.
title_len	the length of title
u_aid_ctr	historic click through rate of user on some ad.

3.Modeling Techniques and Training

3.1 Validation

How to generate validation set.

1. take the 6 days before the end time as start time
2. take all last sessions for each user after start time

We choose 6 because of ratio of new user ids, it's more closer to test set.

3.2 Training

Because of data size and class imbalance, we use negative down sampling. Sample rate 0.1 reduce the data to 20M lines and 10GB, then we can train model in memory.

First, train different models with different settings by using tools below.

- FFM(basically follow the code of 3 idiots' winning solution to the Criteo competition)
- FM(basically follow the libfm)
- XGBoost

Second, train a neural network on validation set to combine all predictions of basic models.

4. Code Description

The implementation is organised in the following three parts.

- start point
    - run.py
    - run.sh
- data generator
    - ins/
    - ins2/
    - ins3/
    - ins5/
    - ins20/
    - ins_bag/
    - xgb5/
- model
    - IceLR/ (FFM and FM)
    - xgb.py (XGBoost)
    - ensemble/ (lasagne)

5. Dependencies

- Python 2.7
- pypy 2.1.0
- XGBoost
- scons
- numpy 1.9.2
- sklearn 0.15.2
- theano 0.7.0
- lasagne 0.1.dev0
- nolearn
- g++ (with C++11, OpenMp, gflags and protobuf)

6. How To Generate the Solution

- move all tsv files into ./data folder
- Compile the C++ code by

```
cd IceLR
scons
```

- run script to generate data and basic models

```
bash run.sh
```

- generate final submission by

```
pypy run.py --type ensemble --method nn > nn_log.log
```

7. Additional Comments and Observations

- Negative down sample tends to make model converge quickly, but only a little impact in accuracy.
- The context features are very important.

8. Simple Features and Methods

Train a FFM model by following command.

```
IceLR/ffm --passes 10 --sr 0.1 --nthread 20 --l2 1e-5 --alpha 0.25 --train_path data/all.ins5 --validate_path data/te.ins5 --b 22 --shuffle --seed 9
```

it will give you **0.04073** in private leaderboard and **0.04063** in public leaderboard.