Winning model documentation

Name: Owen Zhang
Location: NJ, USA
Email: zhonghua.zhang2006@gmail.com
Competition: Click-through rate prediction

1. Summary

The final solution is a manually tuned blend (based on PB feedback) of 4 different models (RandomForest/sklearn, GBDT/xgboost, OnlineSGD/Vowpal Wabbit, Factorization machine/3 idiots). This solution is largely based on 3 idiot's winning solution to the Criteo competition (https://github.com/guestwalk/kaggle-2014-criteo) with moderate amount of feature engineering and manual tuning.

2. Feature Selection / Extraction

   A few different approaches were utilized for feature engineering:

   2a. Combining site/app based features. These features are complementary (in the sense of if one is missing the other one is not), so combining them will at least save space.

   2b. Prior day mean(y) encoding for categorical features. These are done in both univerate and multivariate approaches

   2c. Counts and sequence of device_ip. Device_ip seems to be a reasonable proxy of user identity.

   2d. Factorization Machine based predictions using raw features and counts/sequences.

   2e. GBDT predicted leaf node using raw features, counts/sequences, and prior day mean(y) encoded categorical features.

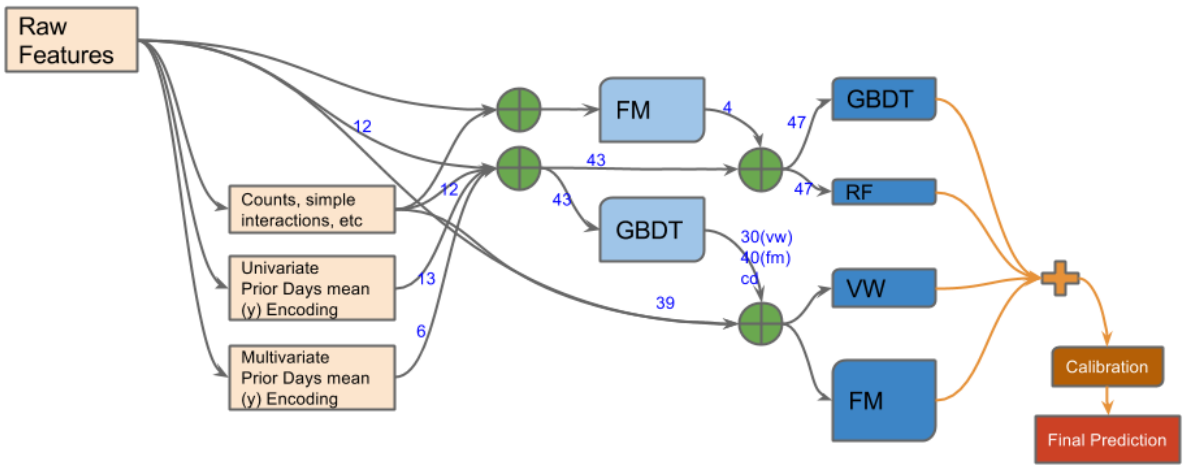
   2f. Some manual interactions, especially app_site_id * C14-21

3. Modeling techniques and Training
   a. I used day 30 as validation and had very stable (and comparable in score movement) results throughout the competition.
   b. 3 idiots' factorization machine turns to be extremely effective in this problem. FM based models are the best individual models in this solution. It is worth noting that they outperform VW with manually built 2 way interactions.
      i. Best FM with GBDT features get ~.3830 on public LB
   c. I spent fair amount of time tuning VW (vowpal wabbit) models, espcially around interactions. I defined name space by feature type (C14-21, device, app/site, device id/ip, GBDT prediction) and tested two way interactions through ad-hoc (almost a step-wise) process.
      i. I tried VW built-in FTRL optimization but cannot get it to perform better than the default adaptive procedure.

ii. A small L2 penalty 1e-7 seems to produce optimal results
iii. Tried cubic interaction and low rank proximation but they only make model worse.

d. RF and GBDT (xgboost) models produce competitve models ONCE we add FM output (built using only raw + count + sequence features) as features.
   i. FM using raw + count + sequence alone get ~.384x on public LB
   ii. Feed them back with raw + count + sequency + prior days mean(y) will get ~.383x
e. Calibration:
   i. Overall actual response rate for public LB is about .161 so the predictions are adjusted to be at this level
   ii. The new site_id on day 31. A new site_id (17d1b03f) appears on day 31 and account for about 20% of all observations (800K). There is no way of knowing the true nature of this new batch of data. The average response is about .13 (from public LB), so the predictions for these observations are always adjusted to be .13.
   iii. It is interesting to see that the blended model has prediction very close to .13 for these observations, even without calibration.

High Level Model Structure



f.

4. Code Decription
   a. Code is packaged in several python files and a shell script to run them
   b. _0_run_me.sh -- shell script to execute the rest
   c. utils.py -- some utility functions and all parameters such as path to data and external excutables.
   d. _1_encode_cat_features.py -- generate prior days mean(y) encoding + counts/sequences

    e. _2b_generate_dataset_for_vw_fm.py -- convert raw values into categorical representations, for processing speed in later steps. Aslo adding a few interactions, such as site_app_id * C14-21.
    f. _2c_generate_fm_features.py -- run FM on prior days data for each day. Output of these will feed into RF/GBDT.
    g. _3a_rf.py -- RandomForest models
    h. _3b_gbdt.py -- GBDT (xgboost) models
    i. _3c_vw.py -- online SGD (vw) models
    j. _3d_fm.py -- Factorization machine models
    k. _4_post_processing.py -- blend all results and generate submission file.
5. Dependencies
    a. Python 2.7.x + numpy + pandas + scikit-learn
    b. Vowpal Wabbit https://github.com/JohnLangford/vowpal_wabbit/wiki
    c. 3 idiot's FM: https://github.com/guestwalk/kaggle-2014-criteo
    d. xgboost: https://github.com/tqchen/xgboost
6. How to generate the solution
    a. Please see the READ.me file in this github repo https://github.com/owenzhang/kaggle-avazu
7. Additional comments and observations
    a. The random sampling seems to render time meaningless.
    b. Observations appear to have been random shuffled within each hour.
        i. This prevents a potential leakage from accurate sequencing of events.
8. Simple Features and Methods
    a. As mentioned earlier, raw features + counts + sequences + FM will get about .384x, good enough for #6 on the leaderboard.
9. Figures
    a. Please see above
10. References
    a. Please see dependencies.