Avazu
Click-Through Rate Prediction

Xiaocong Zhou and Peng Yan

March 9, 2015

Problem description

- Predict the click-through rate of impressions on mobile device
- Dataset
  - Raw features:
    - hour,click,C1,banner pos,site id,...,app id,...,device id,...,C14,...,C21
    - Hour is in format "20141021",
    - The other known features are hashed as e.g."1fbe01fe".
  - Click(label) of day 21 to 30 is given, day 31 to predict.
  - #Train: 40M records
  - #Test: 4.6M records

Evaluation

2-class Logarithmic Loss

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i logp_i + (1 - y_i)\log(1 - p_i))$$

where N is the number of instances, y_i is the true label and p_i is the predicted probability.

- Feature Engineering
- Models: FTRL and FFM
- Ensemble
- Calibration

Preliminary analysis

- We make the inference that C14 is ad id, C17 is ad group id and C21 is ad sponsor id by analyzing the hierarchy of the unknown features.
- We identify the user with device id if it is not null and device ip + device model for others.

Feature Engineering

- Rare features: The features which appear less than 10 times are converted to “rare”
- 8 Additional numerical features:
  - 4 features: Number of impressions to the user for the ad id/ad group id in the hour/day
  - 1 feature: Number of impressions to the user in the day
  - 1 feature: Number of impressions to the user for the app id/site id in the day
  - 1 feature: Time interval from the last visit
  - 1 feature: Number of days the user appeared

  Most of these features are cut off by 10

Feature Engineering

- LSA feature
  - We take the site cate and app cate as words of each device ip which is not rare and calculate the tf-idf vector. Then we perform truncated SVD(LSA) to reduce the dimensionality to 16.
  - The index with the max value is added to features in FTRL.

Feature Engineering

- GBDT features
  - The gradient boosting tree model takes the 9 numerical features(8 additional features and the number of impressions of the device ip) as input and the indices of the trees are the output. 19 trees with depth 5 are used and the 19 generated features are included both in FTRL and FFM.

This approach is proposed by Xinran He et al. at Facebook[1]

and used by 3 idiots in Criteo's competition[3].

Models-FTRL

- Logistic regression

$$argmin_w \frac{\lambda}{2}||w||^2 + \sum_i \log(1 + \exp(-y_i \emptyset(w, x_i))) \quad (1)$$

$$where\ \emptyset(w, x) = w^T x\ for\ linear\ models$$

- Follow-the-Regularized-Leader

FTRL uses the weight update

$$w_{t+1} = argmin_w (g_{1:t} w + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \left|\left|w - w_s\right|\right|^2 + \lambda ||w||)$$

where σ_s is defined in terms of the learning rate schedule such that $\sigma_{1:t} = \frac{1}{\eta_t}$

This approach is proposed by H. Brendan McMahan et al. at Google[2].

Models-FTRL

- 21 original features + 8 additional features + 1 LSA feature + 19 gbdt features are included in FTRL.
- 82 selected interactions are included, mostly with site id or app id.
- All features are one-hot encoded to a space of 2^26.
- 3 epochs are used with the learning rate of 0.05.

Models-FFM

- Field aware Factorization Machine
  - The object function is similar to (1) but with

$$\emptyset(w,x) = \sum_{j_1,j_2 \in c} < W_{j_1,f_2}, W_{j_2,f_1} > x_{j_1}\, x_{j_2}$$

where f_1 and f_2 are the corresponding fields of j_1 and j_2 respectively

- Besides feature interactions, first order features are also used in our FFM model

$$\emptyset(w,x) = \sum_{j_1,j_2 \in c} < W_{j_1,f_2}, W_{j_2,f_1} > x_{j_1}\, x_{j_2} + \sum_{j \in C} \sum_{k=1}^{K} w_{jk} x_j$$

Models-FFM

- 21 original features + 8 additional features + 19 gbdt features are included in FFM.
- The number of latent factors is 8 and 5 epochs are used.
- This approach was proposed by Michael Jahrer et al. in KDD Cup 2012 Track 2 and used by 3 idiots in Criteo's competition.

Ensemble

- Our final model is an ensemble of 4 models. For FTRL and FFM with gbdt features, we both train the model on the whole data and the data separated by sites and apps
- We ensemble by weighted average of the inverse logit of CTRs and then do the calibration

Calibration

- The observed average CTR on test set c_a and our predicted average CTR c_p is slightly different
- We define:

inverse_logit(x) = log$\frac{x}{1-x}$

logit(x)=$\frac{1}{1+\exp(-x)}$

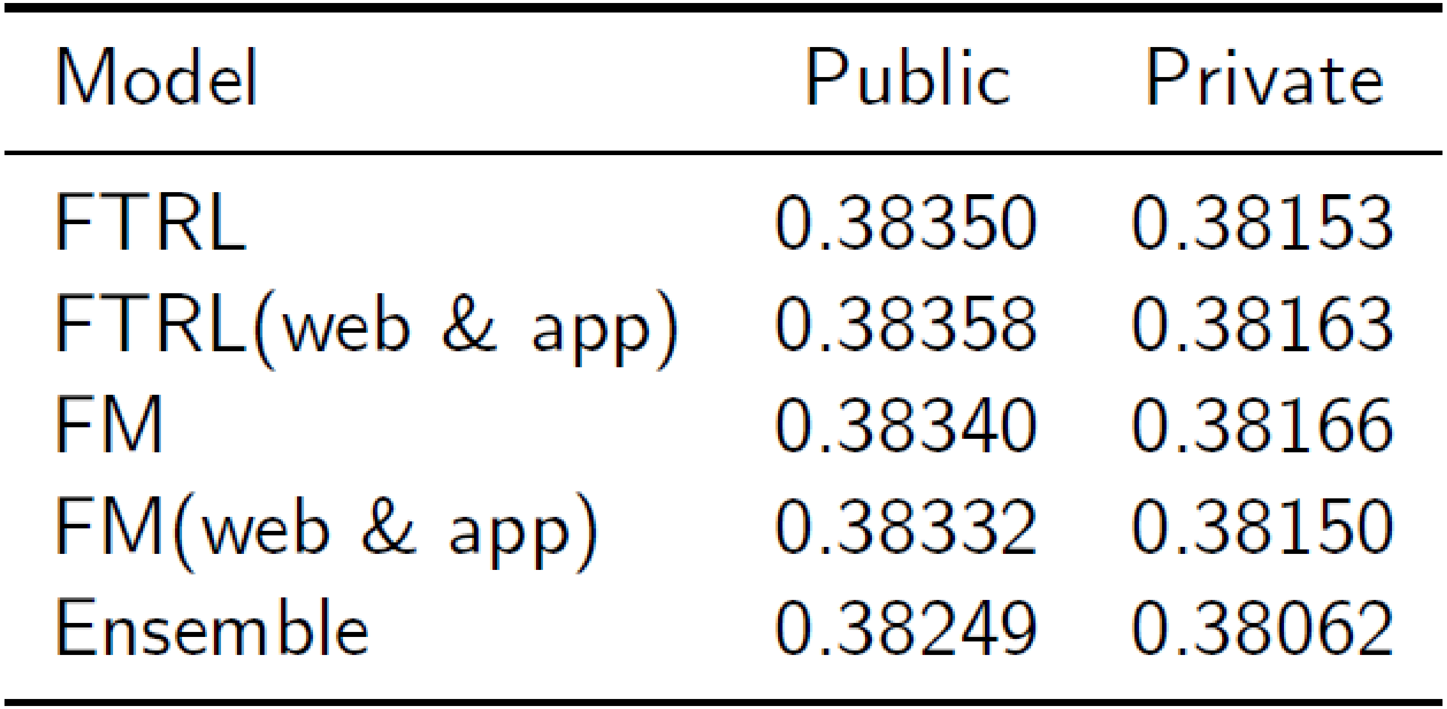
The calibration is as follows:

intercept = inverse_logit(c_p) – inverse_logit(c_a)

p = logit(inverse_logit(p)-intercept)

where p is the predicted CTR for each record.

Result



Model	Public	Private
FTRL	0.38350	0.38153
FTRL(web & app)	0.38358	0.38163
FM	0.38340	0.38166
FM(web & app)	0.38332	0.38150
Ensemble	0.38249	0.38062

Reference

- Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quionero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. (ADKDD'14)
- H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. (KDD '13)
- 3 idiots' Approach for Display Advertising Challenge
- Tinrtgu and 3 idiots' code