KDD Cup 2012 Track 2:

Ensemble of Collaborative Filtering and Feature Engineered Models for Click Through Rate Prediction

—Methods of Opera Solutions
The Dream Team

- Jeong-Yoon Lee
- Jingjing (Bruce) Deng
- Hang Zhang
- Jacob Spoelstra
- Andreas Töscher
- Michael Jahrer
The Task

• Predicting the click-through rate (CTR) a search advertisement receives from a querying user
  – Search advertising has been one of the major revenue sources of the internet industry
  – Predicting CTR correctly helps search providers to rank/price ads correctly
  – Important to user experience improvements and revenue growth
  – Widely applicable to searching engines, online stores, online finance services, etc.
  – Evaluation metric: Area Under ROC Curve (AUC)
Preparing the data for learning

- We do some basic checks
- Decide to use random 3% of train as valid
  - Split 1.5% to Valid1
  - Split 1.5% to Valid2

- Main data table

... 150M records !! - 10Gig raw csv file + keywords + userProfiles
Opera’s Approaches

• Individual models
  – Collaborative filtering (Bias model, Factor models)
  – Naïve Bayesian classifiers (NBC)
  – Feature engineering and advanced statistical models

• Blending (mix the individuals)
  – Weighted sum (linear)
  – Neural network
Collaborative filtering

- Sparse matrix
- What is the matrix?
- What is the target?

<table>
<thead>
<tr>
<th>clicks</th>
<th>impr</th>
<th>adUrlID</th>
<th>adID</th>
<th>adverID</th>
<th>depth</th>
<th>pos</th>
<th>queryID</th>
<th>keyWId</th>
<th>titleID</th>
<th>descrID</th>
<th>userID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>12673870462623600000</td>
<td>4242983</td>
<td>26519</td>
<td>2</td>
<td>1</td>
<td>47350</td>
<td>812</td>
<td>8842</td>
<td>25537</td>
<td>6023881</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>63990246178566700000</td>
<td>21299603</td>
<td>36491</td>
<td>2</td>
<td>2</td>
<td>546</td>
<td>113</td>
<td>3225</td>
<td>121</td>
<td>6023881</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>12673870462623600000</td>
<td>4242983</td>
<td>26519</td>
<td>1</td>
<td>1</td>
<td>47350</td>
<td>812</td>
<td>9164</td>
<td>7625</td>
<td>6023881</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>68775161343899990000</td>
<td>20053263</td>
<td>2332</td>
<td>2</td>
<td>1</td>
<td>23447</td>
<td>476</td>
<td>3547</td>
<td>3397</td>
<td>2583834</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>63608090048063600000</td>
<td>10164628</td>
<td>18209</td>
<td>2</td>
<td>2</td>
<td>4035850</td>
<td>947</td>
<td>74709</td>
<td>37226</td>
<td>2583834</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>116593736142415000000</td>
<td>20934246</td>
<td>34882</td>
<td>2</td>
<td>1</td>
<td>4035850</td>
<td>592434</td>
<td>1507528</td>
<td>4127</td>
<td>2583834</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>17697127834378000000</td>
<td>10484162</td>
<td>29135</td>
<td>2</td>
<td>1</td>
<td>1788197</td>
<td>147838</td>
<td>971553</td>
<td>628205</td>
<td>2583834</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>46603877359288400000</td>
<td>21313239</td>
<td>36540</td>
<td>1</td>
<td>1</td>
<td>6600</td>
<td>11342</td>
<td>10208</td>
<td>1785</td>
<td>4019508</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>26709527232789000000</td>
<td>20172874</td>
<td>23805</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>35</td>
<td>16</td>
<td>4019508</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>77718844412582700000</td>
<td>20108617</td>
<td>32367</td>
<td>1</td>
<td>1</td>
<td>1315</td>
<td>316</td>
<td>177</td>
<td>64</td>
<td>4019508</td>
</tr>
</tbody>
</table>

- We have 10 ID sources (adUrlID, adID, advertiserID, depth, pos, queryID, keyWId, titleID, descrID, userID)
- userID x adUrlID?
- userID x adID?
- userID x advertiserID?
- …
- …
- …
- Target: clicks/impressions

45 combinations
Bias model

- Biases for every unique ID
  - approx. 50M biases
- Prediction is sum of $M=10$ biases

\[
\hat{p}_i = \sum_{m=1}^{M} b_{m_k}^m
\]

where $k = d_{mi}^m$

Value of column=$m$ and row=$i$ in data

- Training with stochastic gradient descent
  - Minimizing MSE
  - Small learning rate, L2 regularization (both optimized)
  - Public Leaderboard AUC: 0.76461
Bias model improved #1

- Same model
  \[ \hat{p}_i = \sum_{m=1}^{M} b_{mk}^m \text{ where } k = d_{mi}^m \]

- Separate learning rates \( \eta_m \) and regularizations \( \lambda_m \) for each of the 10 ID sources

<table>
<thead>
<tr>
<th>ID NAME</th>
<th>( \eta )</th>
<th>( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADURLID</td>
<td>0.000013</td>
<td>0.01</td>
</tr>
<tr>
<td>ADID</td>
<td>0.0001</td>
<td>0.0135</td>
</tr>
<tr>
<td>ADVERTISERID</td>
<td>0.0001</td>
<td>0.0379</td>
</tr>
<tr>
<td>DEPTH</td>
<td>0.000013</td>
<td>0.0379</td>
</tr>
<tr>
<td>POSITION</td>
<td>0.009</td>
<td>0.002</td>
</tr>
<tr>
<td>QUERYID</td>
<td>0.0025</td>
<td>0.0379</td>
</tr>
<tr>
<td>KEYWORDID</td>
<td>0.0001</td>
<td>0.002</td>
</tr>
<tr>
<td>TITLEID</td>
<td>0.0001</td>
<td>0.0135</td>
</tr>
<tr>
<td>DESCRIPTIONID</td>
<td>0.0001</td>
<td>0.137</td>
</tr>
<tr>
<td>USERID</td>
<td>0.0025</td>
<td>0.0075</td>
</tr>
</tbody>
</table>

- Training with stochastic gradient descent
  - Minimizing MSE
  - Public Leaderboard AUC: 0.77336
Bias model improved #2

- Same model
  \[ \hat{p}_i = \sum_{m=1}^{M} b_{k}^m \text{ where } k = d_{i}^m \]

- Separate learning rates \( \eta_m \) and regularizations \( \lambda_m \) for each of the 10 ID sources

- Training with pairwise stochastic gradient descent
  - Minimizing MSE on pairs – related to AUC maximization directly
  - Public Leaderboard AUC: 0.788

FOR \( e = 1...\text{maxEpochs} \)

FOR \( n = 1...N \) (all samples, e.g. N=150M for train set)
- Select a sample: \( a \)=index to positive sample
- Select b sample: \( b \)=index to negative sample
  \[ \hat{p}_a = \sum_{m=1}^{M} b_{d_{a}^m}^m \quad \text{a sample prediction} \]
  \[ \hat{p}_b = \sum_{m=1}^{M} b_{d_{b}^m}^m \quad \text{b sample prediction} \]
  \[ \Delta_{\text{pred}} = \hat{p}_a - \hat{p}_b \quad \text{difference of predictions} \]
  \[ \Delta_{\text{target}} = t_a - t_b \quad \text{difference of targets} \]
  \[ \text{error} = \Delta_{\text{pred}} - \Delta_{\text{target}} \quad \text{the error} \]

FOR \( m = 1...M \) (all 10 ID sources)
  \[ k_a = d_{a}^m \quad k_b = d_{b}^m \]
  \[ b_{k_a}^m = b_{k_a}^m - \eta_m \cdot (\text{error} + \lambda_m \cdot b_{k_a}^m) \quad \text{update the a and b sample biases} \]
  \[ b_{k_b}^m = b_{k_b}^m - \eta_m \cdot (-\text{error} + \lambda_m \cdot b_{k_b}^m) \]
Bias model improved #3

- Same model
  \[ \hat{p}_i = \sum_{m=1}^{M} b_{ik} \] where \( k = d_{mi} \)

- Unroll the training set based on impressionCnt
  - From 150M to 235M training samples (+56% more training samples)
  - Use only 1 (+) or 0 (-) as targets

- Gives also improvement
  - Unfortunately, we have no detailed notes

\[ \begin{array}{ccccccccccc}
\text{clicks} & \text{impr} & \text{adUrlID} & \text{adID} & \text{advertID} & \text{depth} & \text{pos} & \text{queryID} & \text{keyWordID} & \text{titleID} & \text{descrId} & \text{userID} \\
0 & 1 & 1267387046262360000 & 4242983 & 26519 & 2 & 1 & 47350 & 812 & 8842 & 25537 & 6023881 \\
0 & 1 & 6399024617856670000 & 21299603 & 36491 & 2 & 1 & 546 & 113 & 3225 & 121 & 6023881 \\
1 & 1 & 1267387046262360000 & 4242983 & 26519 & 1 & 1 & 47350 & 812 & 9164 & 7625 & 6023881 \\
0 & 1 & 1267387046262360000 & 4242983 & 26519 & 1 & 1 & 47350 & 812 & 9164 & 7625 & 6023881 \\
0 & 1 & 6877516134389999000 & 20053263 & 2332 & 2 & 1 & 23447 & 476 & 3547 & 3397 & 2583834 \\
0 & 1 & 6877516134389999000 & 20053263 & 2332 & 2 & 1 & 23447 & 476 & 3547 & 3397 & 2583834 \\
0 & 4 & 5836080900480636000 & 10164628 & 18209 & 2 & 2 & 4035850 & 947 & 74709 & 37226 & 2583834 \\
0 & 4 & 5836080900480636000 & 10164628 & 18209 & 2 & 2 & 4035850 & 947 & 74709 & 37226 & 2583834 \\
1 & 1 & 1769712783433780000 & 10484162 & 29135 & 2 & 1 & 1788197 & 147838 & 971553 & 628205 & 2583834 \\
0 & 2 & 4660387735928840000 & 21313239 & 36540 & 1 & 1 & 6600 & 11342 & 10208 & 1785 & 4019508 \\
1 & 1 & 2670962723278900000 & 20172874 & 23805 & 2 & 2 & 5 & 3 & 35 & 16 & 4019508 \\
0 & 1 & 7771884441258270000 & 20108617 & 32367 & 1 & 1 & 1315 & 316 & 177 & 64 & 4019508 \\
\end{array} \]

E.g. if impressionCnt=4
- unroll 1 data sample to 4 +/- samples
Factorized model

- Again, $d^m_i$ is the value of the data at
  - $m$ = the sourceID (1...10)
  - $i$ = the sampleID (1...150M)
- The prediction is a sum of all dot products!

$$\hat{p}_i = \sum_{m=1}^{M} \sum_{n=m+1}^{M} p^c (d^m_i)^T \cdot q^c (d^n_i)$$

$$\hat{p}_i = p^T_0 q_0 + p^T_1 q_1 + p^T_2 q_2 + \ldots + p^T_{44} q_{44}$$

45 dot products

- On every cell we have a feature matrix: $F \times |d^m_\star|$
  - $F$ = number of features
  - e.g. $P_0 = F \times 26272$  $P_1 = F \times 641706$
  - Huge number of features!
Factorized model #2

• Very HUGE memory consumption
  – We were only able to train models with F=2 features

• Problems with overfitting
  – Error is minimal after 1 epoch of training!
  – High L2-regularization does not help
  – Too less time to do careful analysis

• Training with pairwise stochastic gradient descent
  – Minimizing pairwise MSE
  – Small learning rate, L2 regularization (both optimized)
  – **Public Leaderboard AUC: 0.7913**
Factorized model #3

• Added an 11th ID based on token overlap
  – # same tokens per instance: queryTokens -> \{keywordTokens, titleTokens, descriptionTokens\}
  – Public Leaderboard AUC: 0.7945

• Tried 12th ID based on
  – # pairs in tokens: hurts the model (but inside ensemble)
Other Collaborative filtering models tried

- **KNN**
  - Tried a few tweaks, but didn’t help

- **AFM**
  - Uses features in „test set“ to learn!
  - Helps a little (0.0001 in blend)
  - Bad performance itself (public leaderboard AUC 0.74xx)

The prediction of a sample $i$ was

$$\hat{r}_i = p^T q$$

$p = \{\text{sum of 7 features of sample } i\}$

$q = q_u + \sum_{j \in N(u)} \sum_{k=1}^7 \{\text{sum of 7 features of sample } j\}$

7 features are:
- adUrlID
- adID
- advertiserID
- queryID
- keyWordID
- titleID
- descriptionID
ROC curves comparisons

Pub. Leaderboard AUC’s
FactorModel: 0.795
BiasModel: 0.788
AFM: 0.74
TokenOverlapStat: 0.57

For classifiers, this is the important region -> operating point
But for Track2 unimportant, just area under the curve
CF observations and model tweaks

• Construct a 11th ID
  – tokenMatchID
  – Use it in bias model and factor model

• >50% of userIDs in the test set are unknown
  – Bad for user-based models

• Never clip predictions to 0...1
  – Can hurt in the final blend

• Every model is re-trained on the whole data before making predictions on the testset

• Use the tokenIDs in factor models
  – queryTokens, keywordTokens, titleTokens, descriptionTokens
  – Very small improvements in the blend

• Use gender and age codes
  – Very small improvements in the blend, if all
  – Hurts if we add this as new ID source in factor models

• We have problems with overfitting in the factor model, even if regularization is high
  – Back to F=1 features
Engineered Features

- **Risk Features**
  - 1D: conditional probability of click given an ID was present in a record.
    \[
    Pr(Y = 1|ID_i) = \frac{\sum_{j=1}^{n} (c_j + N_1) \times I(ID_i \in R_j)}{\sum_{j=1}^{n} (n_j + N_2) \times I(ID_i \in R_j)}
    \]
  - 2D: conditional probability of click given two IDs were present in a record.
  - 8 1D-risk features for adUrlID, adID, advertiserID, depth, position, userID, gender, age
  - 8 2D-risk features for \{adID, advertiserID, depth, position\} x \{gender, position\}

- **Similarity Features**
  - Overlap between tokens of queryID (ID1) and keywordID/titleID/descriptionID (ID2).
    - The proportion of the tokens in ID1 that are present in ID2 tokens.
    - The proportion of the 2-consecutive tokens in ID1 that are present in ID2.
    - If there exist common tokens between ID1 and ID2, their earliest position in ID2.
    - If there exist common 2-consecutive tokens between ID1 and ID2, their earliest position in ID2
  - 12 similarity features.
Feature Engineered Models

• Built on the engineered features

• Gradient Boosting Machine (GBM)
  – “gbm“ package in R was used.
  – Number of trees, shrinkage, and depth were chosen based on the validation errors.
  – AUC: 0.757

• Support Vector Machine (SVM)
  – SVM_perf was used.
  – AUC loss function, linear kernel, c = 500.
  – AUC: 0.764

• Neural Network (NN)
  – NN with AUC optimization was implemented in C.
  – Single hidden layer.
  – Other parameters were chosen based on the validation errors.
  – AUC: 0.765
Blending with a linear model

• Inputs
  – P Predictors (models) as a matrix with elements \( p_{nj} \)
  – Targets as a vector \( t \)
  – Features (pos, gender, age, tokenOverlaps, supports)

• Model
  – Weights \( w_j \)
    \[ \hat{p}_i = \sum_{j=1}^{p} w_j p_{nj} + w_0 \] (\( w_0 = 0 \), because of pairwise ranking)

• Training
  – Gradient descent on pairs of samples
    – Public Leaderboard AUC: 0.8030

FOR e = 1...maxEpochs

FOR n = 1...N (all samples, e.g. N=3,430,641 for upsampled Valid1)
  Select a positive sample: \( a=\)index to positive sample \( t_\left(+\right)=1 \)
  Select a negative sample: \( b=\)index to negative sample \( t_\left(-\right)=0 \)
  \[ \hat{p}_{\left(+\right)} = \sum_{j=1}^{p} w_j p_{aj} \] (+) sample prediction
  \[ \hat{p}_{\left(-\right)} = \sum_{j=1}^{p} w_j p_{bj} \] (-) sample prediction
  \[ \Delta_{\text{pred}} = \hat{p}_{\left(+\right)} - \hat{p}_{\left(-\right)} \] difference of predictions
  \[ \Delta_{\text{target}} = t_\left(+\right) - t_\left(-\right) \] difference of targets
  \[ \text{error} = \Delta_{\text{pred}} - \Delta_{\text{target}} \] the error

FOR j = 1...P (all predictors, e.g. P=57)
  \[ w_j = w_j - \eta \cdot (\text{error} \cdot (p_{aj} - p_{bj}) + \lambda \cdot w_j) \] update the weights
**Blending with a neural network**

- **Inputs**
  - P Predictors (models) as a matrix with elements $p_{n,j}$
  - Targets as a vector $t$
  - Features (pos, gender, age, tokenOverlaps, supports)

- **Model**
  - A single neural network, 1 hidden layer, $K=20$ units
  - $\hat{p}_i = \text{calcNN}(p_{n,i})$

- **Training**
  - Normalization of inputs to -1...1
  - Gradient descent on pairs of samples
  - **Public Leaderboard AUC**: approx. 0.80524 (0.80824 on private)

```
FOR e = 1...maxEpochs
  FOR n = 1...N (all samples, e.g. N=3,430,641 for upsampled Valid1)
    Select a positive sample: $a$=index to positive sample $t_{(+)}=1$
    Select a negative sample: $b$=index to negative sample $t_{(-)}=0$
    $\hat{p}_{(+)} = \text{calcNN}(p_{aj})$ (+) sample prediction
    $\hat{p}_{(-)} = \text{calcNN}(p_{bj})$ (-) sample prediction
    $\Delta_{\text{pred}} = \hat{p}_{(+)} - \hat{p}_{(-)}$ difference of predictions
    $\Delta_{\text{target}} = t_{(+)} - t_{(-)}$ difference of targets
    error = $\Delta_{\text{pred}} - \Delta_{\text{target}}$ the error
  
  Update the NN with both (+) and (-) sample
Using backprob rule
```

+0.002 AUC improvement to linear blending
## Summary of Results

<table>
<thead>
<tr>
<th>Model name</th>
<th>Performance on public leaderboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias model (rank optimization)</td>
<td>0.788</td>
</tr>
<tr>
<td>Factor model (rank optimization)</td>
<td>0.795</td>
</tr>
<tr>
<td>AFM</td>
<td>0.745</td>
</tr>
<tr>
<td>NBC</td>
<td>0.77847</td>
</tr>
<tr>
<td>ANN optimizing AUC on feature metrics</td>
<td>0.76535</td>
</tr>
</tbody>
</table>

### Ensemble methods

<table>
<thead>
<tr>
<th>Model name</th>
<th>Performance on public leaderboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Network rank blend (1x20 neurons)</td>
<td>0.80524</td>
</tr>
<tr>
<td>Linear rank blend</td>
<td>0.803</td>
</tr>
</tbody>
</table>

It was very close on the private leaderboard!
Conclusions

- Was a challenge to handle this HUGE dataset
- Collaborative filtering methods (for sparse data)
  - Pairwise-rank training
  - Unroll the data (150M -> 235M +/- samples)
- Feature engineering + supervised models
- Blending (mix models) is the key for accuracy
  - Pairwise rank SGD -> optimized the AUC
  - Neural network perform better than linear models
Thank you for the attention!