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?

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

<table>
<thead>
<tr>
<th>clicks</th>
<th>impr</th>
<th>adUrlID</th>
<th>adID</th>
<th>advertiserID</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>1267387046262360000</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>6399024617856670000</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>1</td>
<td>1</td>
<td>1267387046262360000</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>6877516133438990000</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>6360809000480636000</td>
<td>101164628</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>1165937361424150000</td>
<td>20934246</td>
<td>34882</td>
<td>2</td>
<td>1</td>
<td>4035850</td>
<td>5924344</td>
<td>1507528</td>
<td>4127</td>
<td>2583834</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1769712783433780000</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>4660387735928840000</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>2670952723278900000</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>777188444415282700000</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>
Bias model

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

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

where $k = d_{m,i}$

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

```markdown
FOR e = 1...maxEpochs
  FOR n = 1...N (all samples, e.g. N=150M for train set)
    Select a sample: \(a=\text{index to positive sample}\)
    Select b sample: \(b=\text{index to negative sample}\)
    \(\hat{p}_a = \sum_{m=1}^{M} b_{d_a}^{m}\) \(a\) sample prediction
    \(\hat{p}_b = \sum_{m=1}^{M} b_{d_b}^{m}\) \(b\) sample prediction
    \(\Delta_{\text{pred}} = \hat{p}_a - \hat{p}_b\) difference of predictions
    \(\Delta_{\text{target}} = t_a - t_b\) difference of targets
    \(\text{error} = \Delta_{\text{pred}} - \Delta_{\text{target}}\) the error
  
  FOR m = 1...M (all 10 ID sources)
    \(k_a = d_{a}^{m}\) \(k_b = d_{b}^{m}\)
    \(b_{k_a}^{m} = b_{k_a}^{m} - \eta_m \cdot (\text{error} + \lambda_m \cdot b_{k_a}^{m})\) update the \(a\) and \(b\) sample biases
    \(b_{k_b}^{m} = b_{k_b}^{m} - \eta_m \cdot (-\text{error} + \lambda_m \cdot b_{k_b}^{m})\)
```

<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>
Bias model improved #3

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

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

<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>keyWordID</th>
<th>titleID</th>
<th>descrlD</th>
<th>userID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1267387046262363000000</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>639902461785657600000</td>
<td>21299603</td>
<td>36491</td>
<td>2</td>
<td>1</td>
<td>546</td>
<td>113</td>
<td>3225</td>
<td>121</td>
<td>6023881</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>126738704626236300000</td>
<td>4242983</td>
<td>26519</td>
<td>1</td>
<td>1</td>
<td>47350</td>
<td>812</td>
<td>9164</td>
<td>7626</td>
<td>6023881</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>687751613436999300000</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>1</td>
<td>127691278343378000000</td>
<td>10484162</td>
<td>2913</td>
<td>2</td>
<td>1</td>
<td>178519</td>
<td>178519</td>
<td>9243</td>
<td>628205</td>
<td>2583834</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>466038773592884000000</td>
<td>21313239</td>
<td>36540</td>
<td>1</td>
<td>1</td>
<td>600</td>
<td>11342</td>
<td>10247</td>
<td>1785</td>
<td>4019508</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>267095272327890000000</td>
<td>20172847</td>
<td>32805</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4019508</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>777188444125827000000</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>

  e.g. if impressionCnt=4
  -> unroll 1 data sample to 4 +/- samples

• Gives also improvement
  – Unfortunately, we have no detailed notes
## Factorized model

- Again, $d_i^m$ is the value of the data at:
  - $m =$ sourceID (1...10)
  - $i =$ sampleID (1...150M)

- The prediction is a sum of all dot products:

$$\hat{p}_i = \sum_{m=1}^{M} \sum_{n=m+1}^{M} p_i^{(m)}^T \cdot q_i^{(n)}$$

45 dot products

- On every cell we have a feature matrix: $F \times |d_i^m|$
  - $F =$ number of features
  - e.g. $P_0=F \times 26272 \quad P_1=F \times 641706$
  - 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)

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

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

Pub. Leaderboard AUC’s
FactorModel: 0.795
BiasModel: 0.788
AFM: 0.74
TokenOverlapStat: 0.57
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 probabilitly 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 $t_{(+)}=1$
- Select a negative sample: $b$=index to negative sample $t_{(-)}=0$

\[ \begin{align*}
\hat{p}_{(+)} &= \sum_{j=1}^{P} w_j p_{aj} & \text{(+ sample prediction)} \\
\hat{p}_{(-)} &= \sum_{j=1}^{P} w_j p_{bj} & \text{(- sample prediction)} \\
\Delta_{\text{pred}} &= \hat{p}_{(+)} - \hat{p}_{(-)} & \text{difference of predictions} \\
\Delta_{\text{target}} &= t_{(+)} - t_{(-)} & \text{difference of targets} \\
\text{error} &= \Delta_{\text{pred}} - \Delta_{\text{target}} & \text{the error}
\end{align*} \]

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 $p_{nj}$
  - Targets as a vector $t$
  - Features (pos, gender, age, tokenOverlaps, supports)

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

- **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 $t_{(-)}=0$

  $p_{(+)} = \text{calcNN}(p_{aj})$  (+) sample prediction
  $p_{(-)} = \text{calcNN}(p_{bj})$  (-) sample prediction
  $\Delta_{\text{pred}} = p_{(+)} - p_{(-)}$  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>

<table>
<thead>
<tr>
<th>Ensemble methods</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!