Matrix Factorization Techniques for Top-N Recommender Systems

Ernesto Diaz-Aviles
<diaz@L3S.de>

Web Science 2013
L3S Research Center. Hannover, Germany
Learning to Rank
Learning to Rank: Supervised Machine Learning Approach for Ranking
Summary from Last Lecture

- Learning to Rank: Supervised Machine Learning Approach for Ranking
- Approaches
Learning to Rank: Supervised Machine Learning Approach for Ranking
Approaches: Pointwise
Learning to Rank: Supervised Machine Learning Approach for Ranking

Approaches: Pointwise, Pairwise
Learning to Rank: Supervised Machine Learning Approach for Ranking

Approaches: Pointwise, Pairwise, and Listwise Approaches.
Learning to Rank: Supervised Machine Learning Approach for Ranking

Approaches: Pointwise, Pairwise, and Listwise Approaches.

Huge impact: Web Search, E-Business, Government, etc.
Algorithm 1 Stochastic Pairwise Descent (SPD). This generic framework reduces the pairwise learning to rank problem to the learning a binary classifier via stochastic gradient descent. The CreateIndex and GetRandomPair functions, instantiated below, enable efficient sampling from $P$.

1: $D_{index} \leftarrow \text{CreateIndex}(D)$
2: $w_0 \leftarrow \emptyset$
3: for $i = 1$ to $t$ do
4:    $((a, y_a, q), (b, y_b, q)) \leftarrow \text{GetRandomPair}(D_{index})$
5:    $x \leftarrow (a - b)$
6:    $y \leftarrow \text{sign}(y_a - y_b)$
7:    $w_i \leftarrow \text{StochasticGradientStep}(w_{i-1}, x, y, i)$
8: end for
9: return $w_t$
Matrix Factorization Techniques for Top-N Recommender Systems
(Personalized Ranking)
Recommender Systems Tasks

1 Rating Prediction
1. Rating Prediction
2. Top-N Recommendation
1. Rating Prediction

2. Top-N Recommendation
   (Item Prediction — Personalized Ranking)
User-based Neighborhood Method

## User-Item Matrix

<table>
<thead>
<tr>
<th>User/Item</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>$I_4$</th>
<th>$I_5$</th>
<th>$I_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$U_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$U_3$</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$U_4$</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$U_5$</td>
<td></td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Matrix Factorization

Feedback Matrix: 
\[ X = [U] \times [i] \]

User Latent Factors 
\[ W = [U] \times [k] \]

Item Latent Factors 
\[ H = [k] \times [i] \]

\[ X_{ij} \]
MF and Latent Factors

[Ernesto Diaz-Aviles]


Cover Feature

COMPUTER  44

vector $q_i \in \mathbb{R}^f$, and each user $u$ is associated with a vector $p_u \in \mathbb{R}^f$. For a given item $i$, the vector $q_i$ measures the extent to which the item possesses those factors, and $p_u$ captures the interaction between user $u$ and $i$, which the item possesses those factors, and $p_u$ captures the interaction between user $u$ and $i$, which the item possesses those factors, and $p_u$ captures the interaction between user $u$ and $i$. The projection of the interaction onto $q_i$ is $\hat{r}_{ui} = p_u^T q_i$, which is denoted by $\hat{r}_{ui}$.

Here, $r_{ui}$ is the set of the $(u, i)$ pairs for which observed ratings are known. The system learns the model by fitting the previously observed ratings in a way that predicts future, unknown ratings. However, the goal is to generalize those works.

Earlier systems relied on imputation to fill in missing data, which is high-quality representing items of interest. The most convenient data input is low-quality.

Explicit feedback is not available, recommender systems can infer user preferences using implicit feedback usually denotes the presence or absence of an event, so it is typically represented by a densely filled matrix.

Explicit feedback is given directly by the user, such as a rating on a scale from 1 to 5. Implicit feedback is gathered indirectly, such as a user clicking on an item or watching a video.

Recommender systems rely on different types of input by users regarding their interest in products. For example, users indicate their preferences for TV shows by pressing thumbs-up and thumbs-down buttons. We refer to explicit feedback as user feedback as $r_{ui}$.

Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting. Thus, the system should avoid overfitting the accuracy. In addition, they offer much flexibility for model computation can be very expensive as it significantly increases computation.

To learn the factor vectors $p_u$ and $q_i$, we minimize the regularized squared error on the set of previously known ratings:

$$\min_{p_u, q_i} \sum_{(u, i) \in \text{training set}} (r_{ui} - \hat{r}_{ui})^2 + \lambda (||p_u||^2 + ||q_i||^2)$$

where $\lambda$ is a parameter controlling the smoothness of the factors.

Once the recommender system has estimated the factor vectors, it can easily estimate the rating of item $i$ for user $u$ by computing $\hat{r}_{ui} = p_u^T q_i$.

The major challenge is computing the map $q_i$ and $p_u$. After the recommender system completes this mapping, it can easily estimate the rating of item $i$ for user $u$ by computing $\hat{r}_{ui} = p_u^T q_i$.

Figure 2.

The Color Purple

The Princess Diaries

Sense and Sensibility

Amadeus

Lethal Weapon

Braveheart

Ocean’s 11

Dave

The Lion King

Gus

Independence Day

Dumb and Dumber

Geared toward males

Geared toward females

Serious

Escapist

[www.L3S.de]
Matrix Factorization

Feedback Matrix:
\[ X = \begin{bmatrix} U \end{bmatrix} x i \]

User Latent Factors
\[ W = \begin{bmatrix} U \end{bmatrix} x k \]

Item Latent Factors
\[ H = \begin{bmatrix} k \end{bmatrix} x i i \]

Feedback Matrix:
\[ X = \begin{bmatrix} U \end{bmatrix} x i i \]
CF Based on Matrix Factorization

- MF estimates $\hat{\mathbf{X}} : U \times I$ by the product of two low-rank matrices $\mathbf{W} : |U| \times k$ and $\mathbf{H} : |I| \times k$ as follows:
- $\hat{\mathbf{X}} := \mathbf{W} \mathbf{H}^\top$, where $k$ is a parameter corresponding to the rank of the approximation.
General Stochastic Gradient Descent Procedure for MF

**SGD for MF**

**Input:**
- Training data $S$
- Regularization parameters $\lambda_W$ and $\lambda_H$
- Learning rate $\eta_0$
- Learning rate schedule $\alpha$
- Number of iterations $T$

**Output:** $\theta = (W, H)$

1. initialize $W_0$ and $H_0$
2. for $t = 1$ to $T$ do
3.   $(u, i, x_{ui}) \leftarrow \text{randomExample}(S)$
4.   $w_u \leftarrow w_u - \eta \frac{\partial}{\partial w_u} \ell(x_{ui}, \langle w_u, h_i \rangle) - \eta \lambda_W w_u$
5.   $h_i \leftarrow h_i - \eta \frac{\partial}{\partial h_i} \ell(x_{ui}, \langle w_u, h_i \rangle) - \eta \lambda_H h_i$
6.   $\eta = \alpha \cdot \eta$
7. end for
8. return $\theta_T = (W_T, H_T)$
Stochastic Gradient Descent
Learning Algorithms: Standard Framework†

- Assumption: examples are drawn independently from an unknown probability distribution $P(x, y)$ that represents the rules of Nature
- Expected Risk: $E(f) = \int \ell(f(x), y) \, dP(x, y)$
- Empirical Risk: $E_n(f) = \frac{1}{n} \sum_n \ell(f(x_i), y_i)$
- We would like $f^*$ that minimizes $E(f)$ among all functions
- In general $f^* \notin \mathcal{F}$
- The best we can have is $f^*_{\mathcal{F}}$ that minimizes $E(f)$ inside $\mathcal{F}$
- But $P(x, y)$ is unknown by definition
- Instead we compute $f_n \in \mathcal{F}$ that minimized $E_n(f)$

Vapnik-Chervonenkis theory tells us when this can work

† Léon Bottou: The Tradeoffs of Large Scale Learning, NIPS tutorials, Vancouver, 2007
Computing $f_n = \arg\min_{f \in \mathcal{F}} E_n(f)$ is often costly

Since we already make lots of approximations, why should we compute $f_n$ exactly?

- Let’s assume our optimizer returns $\hat{f}_n$ such that
  
  $$E_n(\hat{f}_n) < E_n(f_n) + \varepsilon$$

  For instance, one could stop an iterative optimization algorithm long before its convergence
Small-scale vs. Large-scale Learning†

\[ \hat{f}_n \approx \arg\min_\Theta = w \frac{1}{n} \sum_n \ell(f(x_i), y_i) \]

Simple parametric setup

- \( \mathcal{F} \) is fixed: functions \( f_w(x) \) linearly parametrized by \( w \in \mathbb{R}^d \)

Subject to budget constraints:

- Maximal number of examples \( n \) (Small-Scale Learning)
- Maximal computing time \( T \) (Large-Scale Learning)

† Léon Bottou: The Tradeoffs of Large Scale Learning, NIPS tutorials, Vancouver, 2007
Gradient Descent

Iterate

\[ w_{t+1} \leftarrow w_t - \eta \frac{\partial E_n(f_{w_t})}{\partial w} = w_t - \eta \sum_n \ell'(f_w(x_i), y_i) \]
Stochastic Gradient Descent

Iterate

- Draw Random Example \((x_t, y_t)\)
- \(w_{t+1} \leftarrow w_t - \eta_t \frac{\partial}{\partial w} \ell'(f_{w_t}(x_t), y_t)\)

More info. at L. Bottou webpage: http://leon.bottou.org/
SGD for MF

Input:
  Training data $S$; Regularization parameters $\lambda_W$ and $\lambda_H$;
  Learning rate $\eta_0$; Learning rate schedule $\alpha$; Number of
  iterations $T$

Output: $\theta = (W, H)$

1: initialize $W_0$ and $H_0$
2: for $t = 1$ to $T$ do
3:   $(u, i, x_{ui}) \leftarrow \text{randomExample}(S)$
4:   $w_u \leftarrow w_u - \eta \frac{\partial}{\partial w_u} \ell(x_{ui}, \langle w_u, h_i \rangle) - \eta \lambda_W w_u$
5:   $h_i \leftarrow h_i - \eta \frac{\partial}{\partial h_i} \ell(x_{ui}, \langle w_u, h_i \rangle) - \eta \lambda_H h_i$
6:   $\eta = \alpha \cdot \eta$
7: end for
8: return $\theta_T = (W_T, H_T)$
E.g, Pairwise Preferences for Alice:

#EHEC > {#Hannover, #HUS, #RKI}
#Hannover > {#HUS, #RKI}
Ranking Loss (pairwise):

\[
L(P, W, H) = \frac{1}{|P|} \sum_{p \in P} h(y_{uij} \cdot \langle w_u, h_i - h_j \rangle),
\]

where

\[
h(z) = \max(0, 1 - z)
\]

is the hinge-loss;
Ranking Loss (pairwise):

\[ L(P, W, H) = \frac{1}{|P|} \sum_{p \in P} h(y_{uij} \cdot \langle w_u, h_i - h_j \rangle) \quad , \quad (1) \]

\[
\arg\min_{\theta=(W,H)} L(P, W, H) + \frac{\lambda W}{2} \|W\|_2^2 + \frac{\lambda H}{2} \|H\|_2^2 . \quad (2)
\]

where

- \( h(z) = \max(0, 1 - z) \) is the hinge-loss;
- \( y_{uij} = \text{sign}(x_{ui} - x_{uj}) \) is the \( \text{sign}(z) \) function, which returns +1 if \( z > 0 \), i.e., \( x_{ui} > x_{uj} \), and \(-1\) if \( z < 0 \).
CF Based on Matrix Factorization

**Ranking Loss (pairwise):**

\[
L(P, W, H) = \frac{1}{|P|} \sum_{p \in P} h(y_{uij} \cdot \langle w_u, h_i - h_j \rangle),
\]

(1)

\[
\argmin_{\theta=(W,H)} L(P, W, H) + \frac{\lambda_W}{2} \| W \|^2 + \frac{\lambda_H}{2} \| H \|^2.
\]

(2)

where

- \( h(z) = \max(0, 1 - z) \) is the hinge-loss;
- \( y_{uij} = \text{sign}(x_{ui} - x_{uj}) \) is the \( \text{sign}(z) \) function, which returns +1 if \( z > 0 \), i.e., \( x_{ui} > x_{uj} \), and −1 if \( z < 0 \).
- The prediction function \( \langle w_u, h_i - h_j \rangle = \langle w_u, h_i \rangle - \langle w_u, h_j \rangle \) corresponds to the difference of predictor values \( \hat{x}_{ui} - \hat{x}_{uj} \).
CF Based on Matrix Factorization

**Ranking Loss (pairwise):**

\[
L(P, W, H) = \frac{1}{|P|} \sum_{p \in P} h(\text{sign}(x_{ui} - x_{uj}) \cdot \langle w_u, h_i - h_j \rangle),
\]

(1)

\[
\text{argmin}_{\theta=(W,H)} L(P, W, H) + \frac{\lambda_W}{2} \| W \|^2_2 + \frac{\lambda_H}{2} \| H \|^2_2.
\]

(2)

where

- \( h(z) = \max(0, 1 - z) \) is the hinge-loss;
- \( y_{uij} = \text{sign}(x_{ui} - x_{uj}) \) is the \( \text{sign}(z) \) function, which returns +1 if \( z > 0 \), i.e., \( x_{ui} > x_{uj} \), and −1 if \( z < 0 \).
- The prediction function \( \langle w_u, h_i - h_j \rangle = \langle w_u, h_i \rangle - \langle w_u, h_j \rangle \) corresponds to the difference of predictor values \( \hat{x}_{ui} - \hat{x}_{uj} \).
- Gradient

\[
-\nabla h(p_r, \theta_t) = \begin{cases} 
  y_{uij} \cdot (h_i - h_j) & \text{if } \theta_r = w_u, \\
  y_{uij} \cdot w_u & \text{if } \theta_t = h_i, \\
  y_{uij} \cdot (-w_u) & \text{if } \theta_t = h_j, \\
  0 & \text{otherwise}.
\end{cases}
\]
Input: Stream representative sample at time $t$: $S_t$; Regularization parameters $\lambda_W$, $\lambda_{H+}$, and $\lambda_{H-}$; Learning rate $\eta_0$; Learning rate schedule $\alpha$; Number of iterations $T_\theta$

Output: $\theta = (W, H)$

1. **procedure** UPDATEMODEL($S_t, \lambda_W, \lambda_{H+}, \lambda_{H-}, \eta_0, \alpha, T_\theta$)
2.  **for** $t = 1$ **to** $T_\theta$ **do**
3.     $((u, i), (u, j)) \leftarrow$ randomPair($S_t$) $\in P$
4.     $y_{uij} \leftarrow \text{sign}(x_{ui} - x_{uj})$
5.     $w_u \leftarrow w_u + \eta y_{uij} (h_i - h_j) - \eta \lambda_W w_u$
6.     $h_i \leftarrow h_i + \eta y_{uij} w_u - \eta \lambda_{H+} h_i$
7.     $h_j \leftarrow h_j + \eta y_{uij} (-w_u) - \eta \lambda_{H-} h_j$
8.     $\eta = \alpha \cdot \eta$
9.   **end for**
10. **return** $\theta = (W_{T_\theta}, H_{T_\theta})$
11. **end procedure**
Scenario
Towards Real-time Collaborative Filtering for Big Fast Data
Scenario: \#-tags Recommendation in Twitter

- Top-$N$ Recommendations
Scenario: #-tags Recommendation in Twitter

- Top-$N$ Recommendations
- Recommendation of Interesting Topics $\Rightarrow$ #-tags
Scenario: #-tags Recommendation in Twitter

- Top-$N$ Recommendations
- Recommendation of Interesting Topics $\Rightarrow$ #-tags
- Online Collaborative Filtering
Pairwise Learning

E.g, Pairwise Preferences for Alice:
#EHEC > {#Hannover, #HUS, #RKI}
#Hannover > {#HUS, #RKI}
RMFO: Main Steps

Social Media Stream, e.g., Twitter

Sample the Stream

Online Matrix Factorization: Pairwise Approach for Personalized Rank Learning

Real-time Personalized Recommendations
RMFO: Sampling Strategies

Social Media Stream, e.g., Twitter

Sample the Stream

Online Matrix Factorization: Pairwise Approach for Personalized Rank Learning

Real-time Personalized Recommendations

Sampling Strategies
- RMFO-SP: Single Pass
- RMFO-UB: User Buffer
- RMFO-RSV: Reservoir Sampling
RMFO-SP: Single Pass

Twitter Monitor

Online Collaborative Filtering Pairwise MF

Model Update Using Single Pair at the Time

Real-time Personalized Recommendations
RMFO–UB: User Buffer

Individual User Buffers of Limited Size

Online Collaborative Filtering Pairwise MF

Real-time Personalized Recommendations
RMFO-RSV: Reservoir Sampling

Twitter Monitor

Reservoir Sampling

Online Collaborative Filtering Pairwise MF

Real-time Personalized Recommendations

Ernesto Diaz-Aviles <diaz@L3S.de>
Our three **RMFO** variations: RMFO-SP, RMFO-UB, and RMFO-RSV

---

1. **WRMF**: Collaborative Filtering for Implicit Feedback Datasets Authors: Hu, Y.; Koren, Y.; Volinsky, C. IEEE International Conference on Data Mining (ICDM 2008)

MyMediaLite Implementation
Our three **RMFO** variations: RMFO-SP, RMFO-UB, and RMFO-RSV

**SNAP@Stanford 476 million Twitter tweets Dataset**

Dataset statistics (5-core):
- Events (tweets): 35,350,508
- Users: 413,987
- Items (#-tags): 37,297

---

**WRMF**: Collaborative Filtering for Implicit Feedback Datasets Authors: Hu, Y.; Koren, Y.; Volinsky, C. IEEE International Conference on Data Mining (ICDM 2008)

MyMediaLite Implementation
Our three **RMFO** variations: RMFO-SP, RMFO-UB, and RMFO-RSV

**SNAP@Stanford 476 million Twitter tweets Dataset**

**Dataset statistics (5-core):**
- Events (tweets): 35,350,508
- Users: 413,987
- Items (♯-tags): 37,297

**Leave-One-Out Protocol with a time sensitive split of dataset**

---

1. **WRMF**: Collaborative Filtering for Implicit Feedback Datasets Authors: Hu, Y.; Koren, Y.; Volinsky, C. IEEE International Conference on Data Mining (ICDM 2008)

MyMediaLite Implementation
Our three **RMFO** variations: RMFO-SP, RMFO-UB, and RMFO-RSV

**SNAP@Stanford 476 million Twitter tweets Dataset**

Dataset statistics (5-core):
- Events (tweets): 35,350,508
- Users: 413,987
- Items (#-tags): 37,297

**Leave-One-Out Protocol with a time sensitive split of dataset**

**Metric recall** also known as **hit-rate**:

\[
\text{recall} := \frac{\sum_{u \in U_{test}} \mathbb{1}[i_u \in \text{Top-N}_u]}{|U_{test}|}
\]

---

1. **WRMF**: Collaborative Filtering for Implicit Feedback Datasets Authors: Hu, Y.; Koren, Y.; Volinsky, C. IEEE International Conference on Data Mining (ICDM 2008)

MyMediaLite Implementation
Our three **RMFO** variations: RMFO-SP, RMFO-UB, and RMFO-RSV

**SNAP@Stanford 476 million Twitter tweets Dataset**

Dataset statistics (5-core):
- Events (tweets): 35,350,508
- Users: 413,987
- Items (#-tags): 37,297

**Leave-One-Out Protocol with a time sensitive split of dataset**

**Metric recall** also known as *hit-rate*:

\[
\text{recall} := \frac{\sum_{u \in U_{test}} \mathbb{1}[i_u \in \text{Top-N}_u]}{|U_{test}|}
\]

**Baselines:** (1) Trending Topics (TT) and (2) Weighted Regularized Matrix Factorization (WRMF)\(^1\)

\(^1\)WRMF: Collaborative Filtering for Implicit Feedback Datasets Authors: Hu, Y.; Koren, Y.; Volinsky, C. IEEE International Conference on Data Mining (ICDM 2008) MyMediaLite Implementation
<table>
<thead>
<tr>
<th>Method</th>
<th>Recall@1</th>
<th>Recall@5</th>
<th>Recall@10</th>
<th>Recall@15</th>
<th>Recall@20</th>
<th>Recall@30</th>
<th>Recall@40</th>
<th>Recall@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT (previous month)</td>
<td>0.0226</td>
<td>0.0885</td>
<td>0.1040</td>
<td>0.0885</td>
<td>0.1040</td>
<td>0.1132</td>
<td>0.1210</td>
<td>0.1304</td>
</tr>
<tr>
<td>WRMF</td>
<td>0.0522</td>
<td>0.1896</td>
<td>0.2458</td>
<td>0.2458</td>
<td>0.2458</td>
<td>0.2655</td>
<td>0.2955</td>
<td>0.3258</td>
</tr>
<tr>
<td>RankMF-SP</td>
<td>0.0780</td>
<td>0.2573</td>
<td>0.3215</td>
<td>0.3215</td>
<td>0.3215</td>
<td>0.3537</td>
<td>0.3955</td>
<td>0.4360</td>
</tr>
<tr>
<td>RankMF-UB</td>
<td>0.0929</td>
<td>0.3045</td>
<td>0.3694</td>
<td>0.3694</td>
<td>0.3694</td>
<td>0.4087</td>
<td>0.4562</td>
<td>0.5247</td>
</tr>
<tr>
<td>RankMF-RSV</td>
<td>0.1061</td>
<td>0.3403</td>
<td>0.4048</td>
<td>0.4048</td>
<td>0.4048</td>
<td>0.4515</td>
<td>0.5037</td>
<td>0.5722</td>
</tr>
</tbody>
</table>

**Recommendation Performance**

![Graph showing recommendation performance](image)

- **recall @ N**
- **recall**
- **Top-1**
- **Top-5**
- **Top-10**
- **Top-15**
- **Top-20**
- **Top-30**
- **Top-40**
- **Top-50**

Legend:
- TT (previous month)
- WRMF
- RMFO-SP
- RMFO-UB
- RMFO-RSV
Recommendation Performance for Different Reservoir Sizes

### Top-10: recall vs Reservoir Size

![Graph showing recall vs Reservoir Size](chart.png)

- **RMFO-RSV**
- **RMFO-UB-512**
- **WRMF (Batch)**
- **Trending Topics (previous month)**

<table>
<thead>
<tr>
<th>Reservoir Size (Millions)</th>
<th>Recall@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.0621339</td>
</tr>
<tr>
<td>1</td>
<td>0.1143407</td>
</tr>
<tr>
<td>2</td>
<td>0.184501</td>
</tr>
<tr>
<td>4</td>
<td>0.261145</td>
</tr>
<tr>
<td>8</td>
<td>0.321516</td>
</tr>
</tbody>
</table>

Recall@10: Test Set Size (= n test users; leave one out)

- 260246
- 128 factors

Ernesto Diaz-Aviles <diaz@L3S.de>
Top-10: Recall vs Number of Factors

- RMFO-RSV
- WRMF-128
- TT

Recall@1 ... 20 32 44 56 68 80 92 104 116 128

0.0468 0.0624 0.0793 0.1040
Recall@5 0.1529 0.1830 0.2118 0.2458
0.2289 0.2627 0.2912 0.3076
Recall@10 0.2811 0.3146 0.3407 0.3076
0.3207 0.3529 0.3771 0.3570
Recall@15 0.3787 0.4080 0.4297 0.4048
0.4216 0.4483 0.4683 0.4431
Recall@20 0.4557 0.4803 0.4988 0.4804
0.4988 0.4804 0.4988 0.4804
Recall@30 0.4787 0.4803 0.4988 0.4804
0.4988 0.4804 0.4988 0.4804
Recall@40 0.4557 0.4803 0.4988 0.4804
0.4988 0.4804 0.4988 0.4804
Recall@50 0.4321 0.4803 0.4988 0.4804
0.4988 0.4804 0.4988 0.4804

Ernesto Diaz-Aviles <diaz@L3S.de>  www.L3S.de | 36/39
# Time and Space Savings vs Recommendation Quality

<table>
<thead>
<tr>
<th>Method (128 factors)</th>
<th>Time (seconds)</th>
<th>recall@10</th>
<th>Space in speed</th>
<th>Gain in speed</th>
<th>Gain in recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRMF (Baseline)</td>
<td>23127.34</td>
<td>0.2573</td>
<td>100.00%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>RMFO-RSV 0.5 M</td>
<td>47.97</td>
<td>0.0621</td>
<td>1.41%</td>
<td>482.16</td>
<td>-75.85%</td>
</tr>
<tr>
<td>RMFO-RSV 1 M</td>
<td>89.15</td>
<td>0.1143</td>
<td>2.83%</td>
<td>259.42</td>
<td>-55.56%</td>
</tr>
<tr>
<td>RMFO-RSV 2 M</td>
<td>171.18</td>
<td>0.1845</td>
<td>5.66%</td>
<td>135.11</td>
<td>-28.30%</td>
</tr>
<tr>
<td>RMFO-RSV 4 M</td>
<td>329.60</td>
<td>0.2611</td>
<td>11.32%</td>
<td>70.17</td>
<td>+1.49%</td>
</tr>
<tr>
<td>RMFO-RSV 8 M</td>
<td>633.85</td>
<td>0.3215</td>
<td>22.63%</td>
<td>36.49</td>
<td>+24.95%</td>
</tr>
<tr>
<td>RMFO-RSV INF</td>
<td>1654.52</td>
<td>0.3521</td>
<td>100.00%</td>
<td>13.98</td>
<td>+36.84%</td>
</tr>
</tbody>
</table>
Matrix Factorization: Powerful Machine Learning approach for Top-N Recommendation

RMFO approach for recommending topics to users in presence of streaming data.
RMFO achieves state-of-the-art performance in terms of recommendation quality with significant improvements in speed and space efficiency.
Conclusions

- Matrix Factorization: Powerful Machine Learning approach for Top-N Recommendation
- SGD for MF: Large Scale Datasets
Conclusions

- **Matrix Factorization**: Powerful Machine Learning approach for Top-N Recommendation
- **SGD for MF**: Large Scale Datasets
- **RMFO** approach for recommending topics to users in presence of streaming data
Conclusions

- **Matrix Factorization**: Powerful Machine Learning approach for Top-N Recommendation
- **SGD for MF**: Large Scale Datasets
- **RMFO** approach for recommending topics to users in presence of streaming data
- **RMFO** achieves state-of-the-art performance in terms of recommendation quality with significant improvements in speed and space efficiency
Thank you!

Your Software Project and Thesis at L3S Research Center :)  

Ernesto Diaz-Aviles

diaz@L3S.de
More Info