Improved Neighborhood-Based Algorithms for Large-Scale Recommender Systems

A.Töscher and M.Jahrer and R.Legenstein

commendo research & consulting GmbH
University of Technology Graz

24^{th} August, 2008

Collaborative Filtering

Successful algorithms for large scale recommender systems

- Neighborhood information (KNN)
- Rating matrix factorization (SVD)

We will present:

- How to learn similarities with gradient decent
- How to integrate neighborhood information into a matrix factorization

Collaborative Filtering

Successful algorithms for large scale recommender systems

- Neighborhood information (KNN)
- Rating matrix factorization (SVD)

We will present:

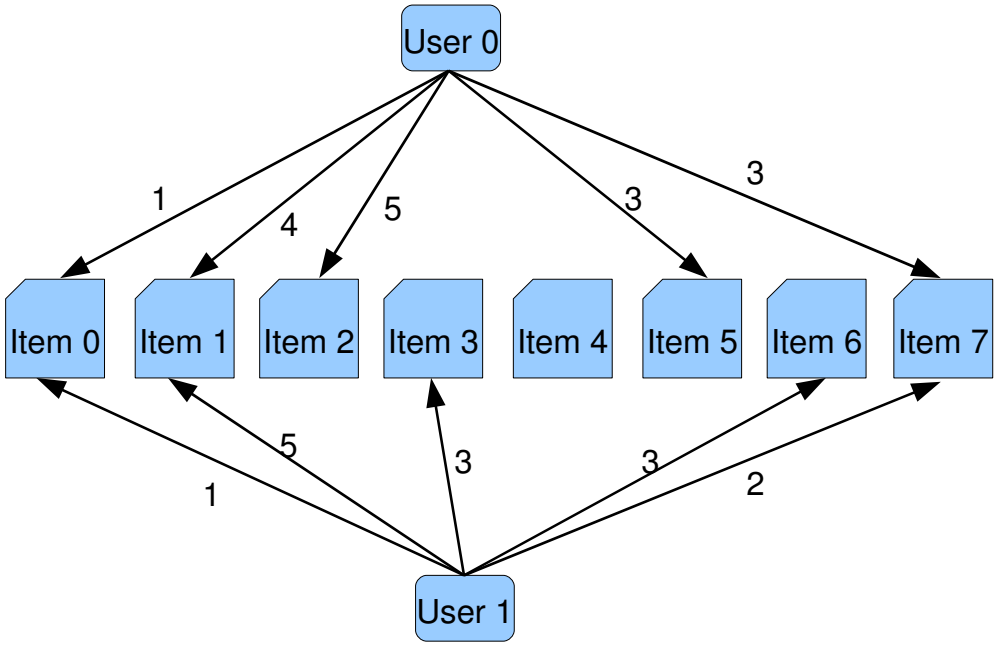
- How to learn similarities with gradient decent
- How to integrate neighborhood information into a matrix factorization

Neighborhood Model

How to define similarity

- Item-Item
  - Defined over common user votes
- User-User
  - Defined over common voted items

Similarity between users



- Common items: $0, 1, 7$
- $\mathbf{U}_0 = [0, 1, 2, 5, 7]$
- $\mathbf{U}_1 = [0, 1, 3, 6, 7]$
- Rating vectors of common items: $\mathbf{x} = [1, 4, 3], \mathbf{y} = [1, 5, 2]$

Define correlation measure

We have now two vectors $\mathbf{x} = [1, 4, 3], \mathbf{y} = [1, 5, 2]$

- Different measures
  - Pearson: $\frac{cov(\mathbf{x},\mathbf{y})}{\sigma_\mathbf{x}\sigma_\mathbf{y}}$
  - Support (no rating info): $\frac{\mathbf{U}_0 \cap \mathbf{U}_1}{min(|\mathbf{U}_0|,|\mathbf{U}_1|)}$
  - Cosine similarity (angle): $arccos \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$
  - Rank correlation

Neighborhood model (KNN)

- The quality of a correlation couldn't be quantified directly
- Only indirect by the RMSE of a KNN model

Classical approach

- Try various similarity measures and select the best

Our approach

- Train the similarities to minimize the RMSE

Neighborhood model (KNN)

- The quality of a correlation couldn't be quantified directly
- Only indirect by the RMSE of a KNN model

Classical approach

- Try various similarity measures and select the best

Our approach

- Train the similarities to minimize the RMSE

Neighborhood model (KNN)

- The quality of a correlation couldn't be quantified directly
- Only indirect by the RMSE of a KNN model

Classical approach

- Try various similarity measures and select the best

Our approach

- Train the similarities to minimize the RMSE

14 Global Effects

Data normalization is important

Nb.	Side	Effect	probe RMSE
10	both	Previous effects	0.9659
11	item	average movie year	0.9635
12	user	movie production year	0.9623
13	user	STD of movie ratings	0.9611
14	item	STD of user ratings	0.9604

detailed description of global effects

Y. Koren and R. Bell, *"Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights"*, KDD-Cup07

Extract similarities from data

- We want to minimize[1]: $E(\mathbf{C}, \mathcal{L}) = \sum_{(u,i)\in\mathcal{L}} (\hat{r}_{ui} - r_{ui})^2$
- Prediction is a weighted sum of all neighbors: $\hat{r}_{ui} = \frac{\sum_{j\in N(u,i)} c_{ij}\, r_{uj}}{\sum_{j\in N(u,i)} |c_{ij}|}$
- Trainable parameters: c_{ij}
  - c_{ij} can be directly interpreted as linear correlation
- For item-item correlations we have $\approx 158 Mio.$ c_{ij}'s (upper triangle of symmetric correlation matrix)

[1] $\mathcal{L}$ is training list, $\mathcal{L} = \{(u_1, i_1), \ldots, (u_L, i_L)\}$

The parameter training

- New error function, which penalizes large correlations
  $E(\mathbf{C}, \mathcal{L}) = \sum_{(u,i)\in\mathcal{L}} (\hat{r}_{ui} - r_{ui})^2 + \gamma \sum_{j<k} c_{jk}^2$
- Correlation update is done with a sign gradient descent
  $$c_{ij}^{new} = c_{ij}^{old} - \eta \cdot \text{sign}\left((\hat{r}_{ui} - r_{ui}) \frac{\partial \hat{r}_{ui}}{\partial c_{ij}} \right) - \eta\,\gamma\, c_{ij}^{old}$$
- In one epoch there are $L \cdot s_M$ parameter updates. [1]
- Learnrate η and regularization γ are both set to 0.01
- Problem with overfitting
  - Huge amount of trainable parameters
  - In many cases: 1 epoch suffices

[1] $L = \#\mathcal{L}$, $s_M = \#\text{userVotes}$ ($\approx$200)

Similarity Training: The results

- Training without probeset to determine break criteria
- For submission we train with the whole data
- Works much better on raw data (Pearson-kNN: RMSE=0.99)
- Train time around 1 hour per epoch

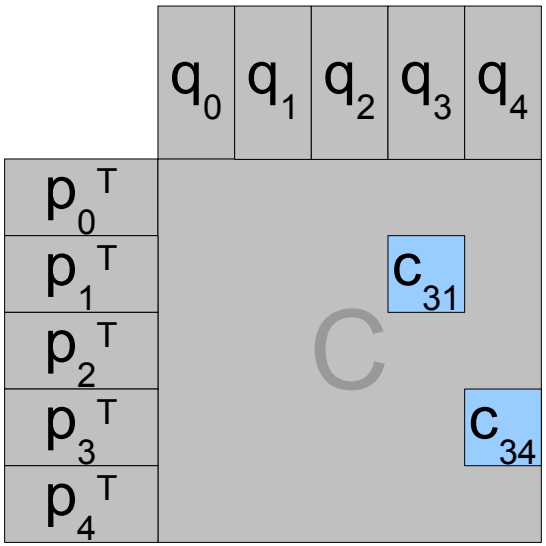
Preprocessing	probe RMSE	qual. RMSE
Raw data	0.9574	0.9487
1GE	0.9458	0.9384
2GE	0.9459	0.9384
6GE	0.9372	0.9281
10GE	0.9349	0.9256
14GE	0.9331	0.9239

Factorize the item-item similarity matrix

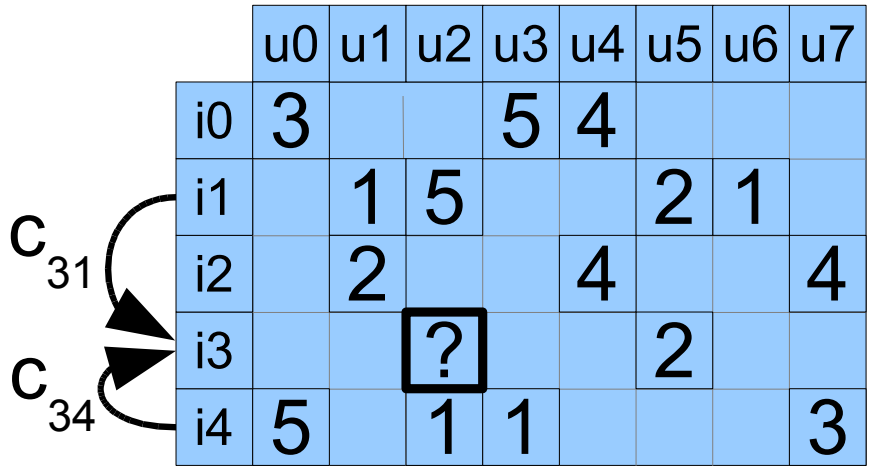
- The idea is to reduce the number of parameters
- $\mathbf{C} = \mathbf{P}^T\mathbf{Q}$
  - $\mathbf{C}$ is a L x L matrix (L=17770, number of items)
  - $\mathbf{P}$ is a K x L matrix
  - $\mathbf{Q}$ is a K x L matrix
- $\mathbf{P}$ and $\mathbf{Q}$ are different, because $\mathbf{C}$ must not be positive semidefinite

Rating prediction

Item Similarity Matrix $C \sim P^T Q$



Rating Matrix R



query: i=3 user=2

$$c_{31} = p_1^T \cdot q_3$$
$$c_{34} = p_3^T \cdot q_4$$

$$r = \frac{5 \cdot c_{31} + 1 \cdot c_{34}}{|c_{31}| + |c_{34}|}$$

Training: factorized item-item similarity matrix

- Use stochastic gradient descent as described before to train the matrices $\mathbf{P}$ and $\mathbf{Q}$.
- In one epoch there are $L \cdot s_M \cdot K$ parameter updates.
- Training requires around 5-10 epochs (3 hours per epoch)

Very memory efficient

Similarities can be easily held in main memory, even with millions of items

Preprocessing	K	probe RMSE
10GE	80	0.9324
14GE	100	0.9313

Factorize the user-user similarity matrix

- Full User-User correlations are hard to precompute (around 1TB HDD storage)
- A factorized version can be easily held in RAM (fast access)
- Same training as in factorized item-item, but mirrored

Preprocessing	K	probe RMSE
Raw	10	0.9951
2GE	10	0.9539
6GE	10	0.9469
14GE	20	0.9371

Neighborhood Aware Matrix Factorization

Idea behind NAMF

- Combination of matrix factorization and user/item neighborhood models
- Neighborhood models work best with good correlations
- The ratings of the best correlating users/items are generally not known
- Use predicted ratings for the unknown ratings

Basic steps in training

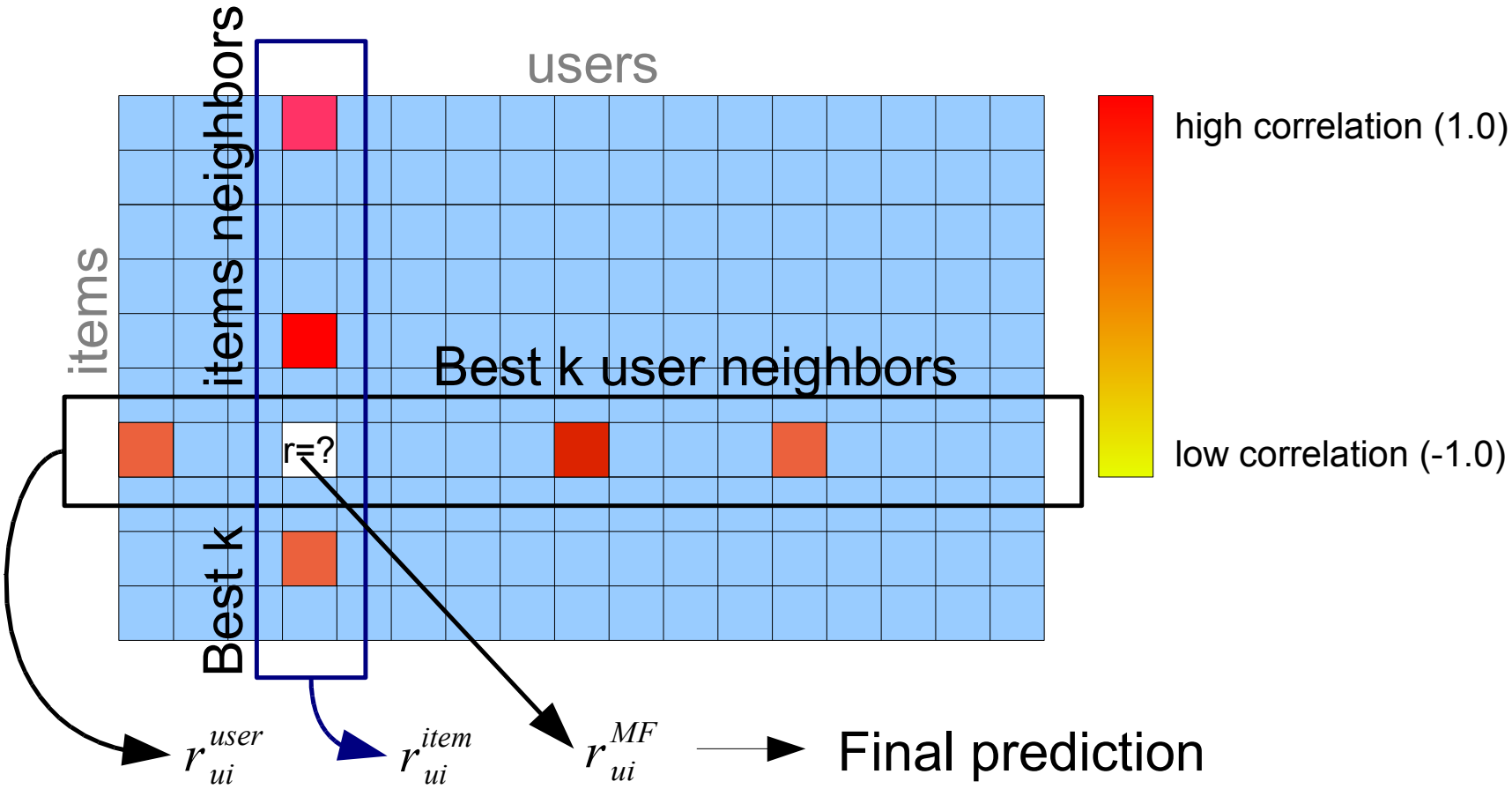
- Precompute J-best item and J-best user neighbors for every item/user
- Train a matrix factorization (RMF)
- Rating prediction r_{ui} with NAMF
  - Predict r_{ui} directly by trained RMF
  - Predict $U_J(u)$ (J-best user neighbors)
  - Predict $I_J(i)$ (J-best item neighbors)
  - Mix the predictions to get the final prediction for r_{ui}

Regularized Matrix Factorization

- aka SVD
- RMF computes a rank K approximation $\mathbf{R}' = \mathbf{A}\mathbf{B}^T$ of the rating matrix $\mathbf{R}$
- User and Item feature matrix $\mathbf{A} \in \mathbb{R}^{m \times K}$ and $\mathbf{B} \in \mathbb{R}^{n \times K}$
- $E(\mathbf{A}, \mathbf{B}, \mathcal{L}) = \sum_{(u,i) \in \mathcal{L}} (r_{ui} - \hat{r}_{ui}^{MF})^2 + \frac{\lambda}{2}(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2)$ [1]

[1] $\mathcal{L}$ is training list, $\mathcal{L} = \{(u_1, i_1), \ldots, (u_L, i_L)\}$

Rating prediction



Item neighborhood model

- $\hat{r}_{ui}^{item} = \frac{\sum_{j \in I_J(i)} c_{ij}^{item} r_{uj}}{\sum_{j \in I_J(i)} c_{ij}^{item}}$
- $c_{ij}^{item} = \left(\sigma\left(s^{item} \rho_{ij}^{item} - b^{item}\right)\right)^{\gamma^{item}}$
- $\rho_{ij}^{item} = \frac{s_{ij} \tilde{\rho}_{ij}^{item}}{s_{ij} + \alpha^{item}}$

- $\tilde{\rho}_{ij}^{item}$ ... correlation between items
- s_{ij} ... support of the correlation
- $I_J(i)$ ... the J best correlating items for item i
- Adjustable Parameters: α^{item}, γ^{item}, s^{item}, b^{item}
- $\sigma(x) = 1/\left(1 + \exp\left(-x\right)\right)$

User neighborhood model

- $\hat{r}_{ui}^{user} = \frac{\sum_{v \in U_J(u)} c_{uv}^{user} r_{vi}}{\sum_{v \in U_J(u)} c_{uv}^{user}}$
- $c_{uv}^{user} = \left(\sigma\left(s^{user} \rho_{uv}^{user} - b^{user}\right)\right)^{\gamma^{user}}$
- $\rho_{uv}^{user} = \frac{s_{uv} \tilde{\rho}_{uv}^{user}}{s_{uv} + \alpha^{user}}$

- $\tilde{\rho}_{uv}^{user}$ ... correlation between users
- s_{uv} ... support of the correlation
- $U_J(u)$ ... the J best correlating users for user u
- Adjustable Parameters: α^{user}, γ^{user}, s^{user}, b^{user}
- $\sigma(x) = 1/\left(1 + \exp\left(-x\right)\right)$

Rating prediction

$$\hat{r}_{ui} = \frac{\tilde{S}(u,i)^{\delta} \cdot \hat{r}_{ui}^{MF} + \hat{\beta}\hat{S}(u,i)^{\hat{\delta}} \cdot \hat{r}_{ui}^{user} + \bar{\beta}\bar{S}(u,i)^{\bar{\delta}} \cdot \hat{r}_{ui}^{item}}{\tilde{S}(u,i)^{\delta} + \hat{\beta}\hat{S}(u,i)^{\hat{\delta}} + \bar{\beta}\bar{S}(u,i)^{\bar{\delta}}}$$

- $\tilde{S}(u,i)$ Support (minimum of user/item votes)
- $\bar{S}(u,i)$ the number of items $I_J(i)$ which are rated by user u
- $\hat{S}(u,i)$ the number of users $U_J(u)$ which rated the item i
- Adjustable Parameters: δ, $\hat{\beta}$, $\hat{\delta}$, $\bar{\beta}$, $\bar{\delta}$

Results

- Improved accuracy
- Very fast rating prediction

Features K of the RMF	probe RMSE
10	0.9175
50	0.9069
100	0.9056
300	0.9046
600	0.9042

with Pearson correlation and 50 best neighbors

Integration of neighbor information

- Memory efficient trainable neighborhood model
- Factorized User-User correlations can easily be held in RAM
- Combine SVD prediction with K-best correlating neighbors
- A linear blend of the results in a 6.25% improvement over Cinematch
- Memory usage scales linearly with number of items and users
  - Therefore scaleable for large scale problems

Thank you for your attention