

---

**FACTORIZATION MEETS THE NEIGHBORHOOD  
: A MULTIFACETED COLLABORATIVE FILTERING MODEL**

**MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS**

---

2022-07-05    KyuHyeok Seo

# CONTENTS

- I Intro
- I Preliminaries
- I Neighborhood Model
- I Latent Model
- I Evaluation through a Top-K Recommender
- I Integrated Model
- I Discussion
- I Implement

# INTRO

without requiring the creation of explicit profiles

## Contents-Based Filtering



## Collaborative Filtering

creates a profile for each user or product  
to characterize its nature.

require gathering external information that  
might not be available or easy to collect

analyzes relationships  
users - interdependencies  
among products to identify  
new user-item associations.

# INTRO

## Collaborative Filtering

### Neighborhood Model



centered on computing the **relationships between items**  
OR **between users**

effective at detecting very **localized relationships**

**unable** to capture the **totality of weak signals**  
encompassed in all of a user's ratings.

**Item - oriented approach**

based on ratings of **similar items by the same user**

**User - oriented approach**

based on ratings of **similar Users by the same items**

Better **scalability** and **improved accuracy**

**reasoning behind predictions**

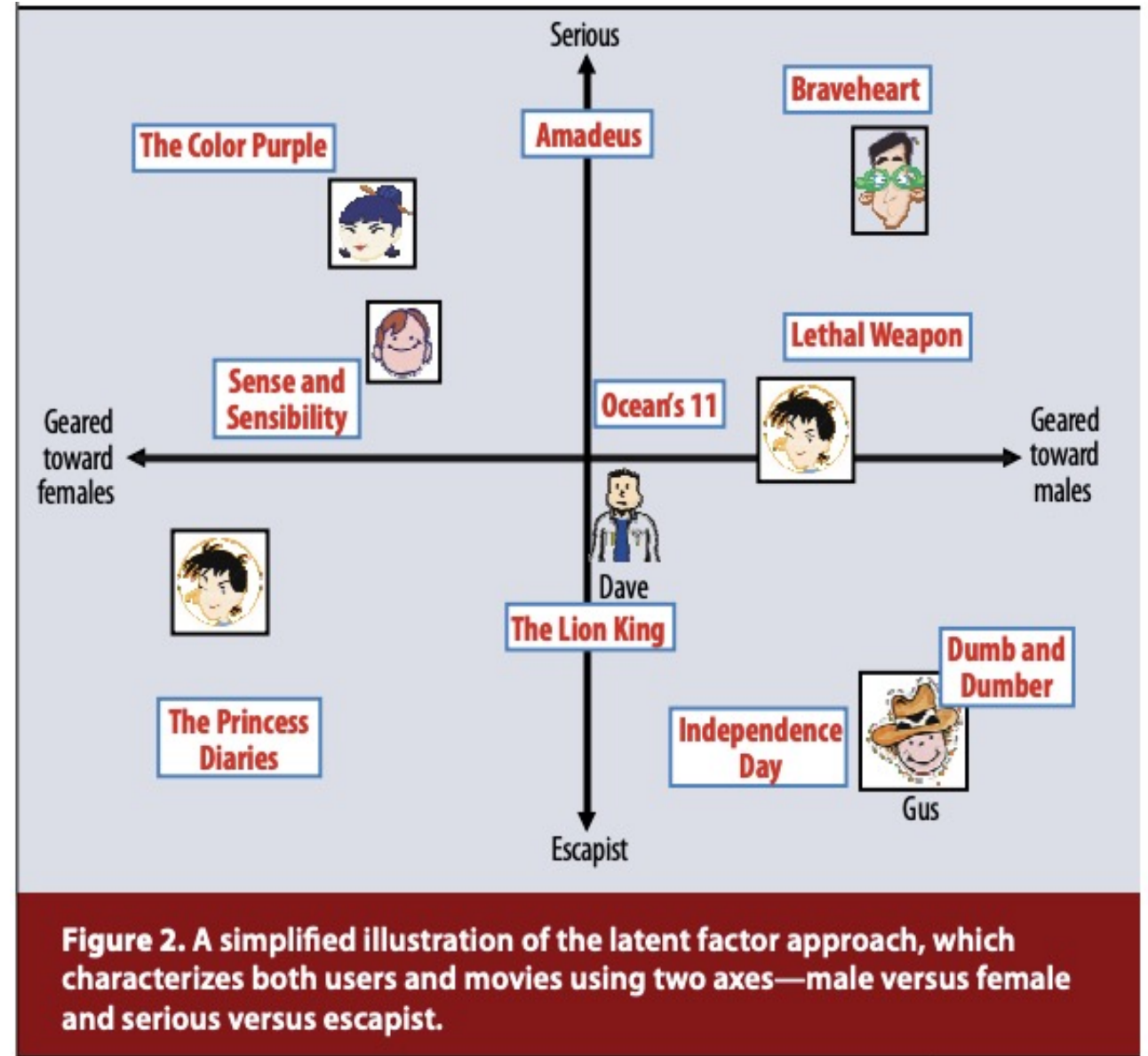
# INTRO

## Latent Model

transforming both items and users  
to the **same latent factor space**

**effective** at estimating **overall structure**  
that relates simultaneously to **most items**

**poor** at detecting **strong associations**  
among a **small set of related items**



# INTRO

## Different Forms of USER INPUT

### EXPLICIT DATA

**explicit input** by users  
regarding their interest in products

Netflix - star ratings for movies  
TiVo - hitting thumbs-up/down buttons

### IMPLICIT DATA

**indirectly** reflect opinion through  
**observing user behavior**

purchase history, browsing history, search  
patterns, or even mouse movements

# PRELIMINARIES - 01. Baseline estimates

Effect based on user and item

$$b_{ui} = \mu + b_u + b_i$$

Objective function with regularizing term

$$\min_{b_*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \left( \sum_u b_u^2 + \sum_i b_i^2 \right)$$

**Regularizing term** : avoid **overfitting** by penalizing the magnitude of the parameters

# PRELIMINARIES - 02. Neighborhood Model : CorNgbr

Measure **Similarity** item  $i, j$

$$s_{ij} \stackrel{\text{def}}{=} \frac{n_{ij}}{n_{ij} + \lambda_2} \rho_{ij}$$

As  $n$  increases,  $s$  also increases

**Predict Value** based on  $k$  neighbors

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij} (r_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}}$$

Using **K-neighbors**, reflect **bias (r-b)**



# PRELIMINARIES - 02. Neighborhood Model : WgtNgbr

## *Concerns about CorNgbr*

Question **suitability of a similarity measure**  
: relations between two items  
: w/o analyzing the interactions full data

neighborhood information is **absent**  
-> rely on **baseline estimates**.

**Predict Value** based on k neighbors with weights

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in R(u)} (r_{uj} - b_{uj}) w_{ij}$$

global weights independent of a specific user

# PRELIMINARIES - 03. Latent Model : SVD, SVD w bias

Latent model by **Singular Value Decomposition(SVD)**

$$\hat{r}_{ui} = q_i^T p_u.$$

$$\min_{q^*, p^*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - q_i^T p_u)^2 + \lambda (\|q_i\|^2 + \|p_u\|^2)$$

Latent model by **SVD with Base Estimate**

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

$$\min_{p_*, q_*, b_*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda_3 (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

# PRELIMINARIES - 04. RMSE & 05. Implicit Feedback

Evaluation using **RMSE**

$$(\text{RMSE}): \sqrt{\sum_{(u,i) \in \text{TestSet}} (r_{ui} - \hat{r}_{ui})^2 / |\text{TestSet}|}.$$

How to reflect **Implicit Feedback**

Consider **whether movies users rate or not**, regardless of how they rated these movies

user implicitly show their preferences : **choosing and rating**

# PRELIMINARIES - 06. Optimizing Algorithm

## Stochastic Gradient Descent (SGD)

*gradient descent optimization*

*ease with a relatively fast running time.*

- $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$
- $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$

## Alternating Least Squares (ALS)

Fix  $p$ , Recompute  $q$  by **solving a least-squares problem only for  $q$**

**favorable in two cases**

system can use parallelization

systems centered on implicit data

*computes each  $q$  independently of other parameters*

*the training set cannot be considered sparse*

*loop over each training case : not practical for SGD*

# Neighborhood Model - Integrating Implicit feedback

## Reflect Implicit Feedback

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in R(u)} (r_{uj} - b_{uj})w_{ij} + \sum_{j \in N(u)} c_{ij}$$

Formed not only by **what he rated**, but also by what **he did not rate**

## Reflect Implicit Feedback with penalizing input size

$$\hat{r}_{ui} = \mu + b_u + b_i + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj})w_{ij} + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} c_{ij}$$

Prevent to overemphasize the dichotomy between **heavy raters** and **those that rarely rate**

# Neighborhood Model - Integrating Implicit feedback

## Pruning parameters

$$\hat{r}_{ui} = \mu + b_u + b_i + |\mathcal{R}^k(i; u)|^{-\frac{1}{2}} \sum_{j \in \mathcal{R}^k(i; u)} (r_{uj} - b_{uj}) w_{ij} + |\mathcal{N}^k(i; u)|^{-\frac{1}{2}} \sum_{j \in \mathcal{N}^k(i; u)} c_{ij}$$

Trade-off  $k$  : accuracy and computational Cost

## New NEIGHBORHOOD MODEL

$$\begin{aligned} \min_{b_*, w_*, c_*} \sum_{(u, i) \in \mathcal{K}} & \left( r_{ui} - \mu - b_u - b_i - |\mathcal{N}^k(i; u)|^{-\frac{1}{2}} \sum_{j \in \mathcal{N}^k(i; u)} c_{ij} - |\mathcal{R}^k(i; u)|^{-\frac{1}{2}} \sum_{j \in \mathcal{R}^k(i; u)} (r_{uj} - b_{uj}) w_{ij} \right)^2 \\ & + \lambda_4 \left( b_u^2 + b_i^2 + \sum_{j \in \mathcal{R}^k(i; u)} w_{ij}^2 + \sum_{j \in \mathcal{N}^k(i; u)} c_{ij}^2 \right) \end{aligned} \quad (11)$$

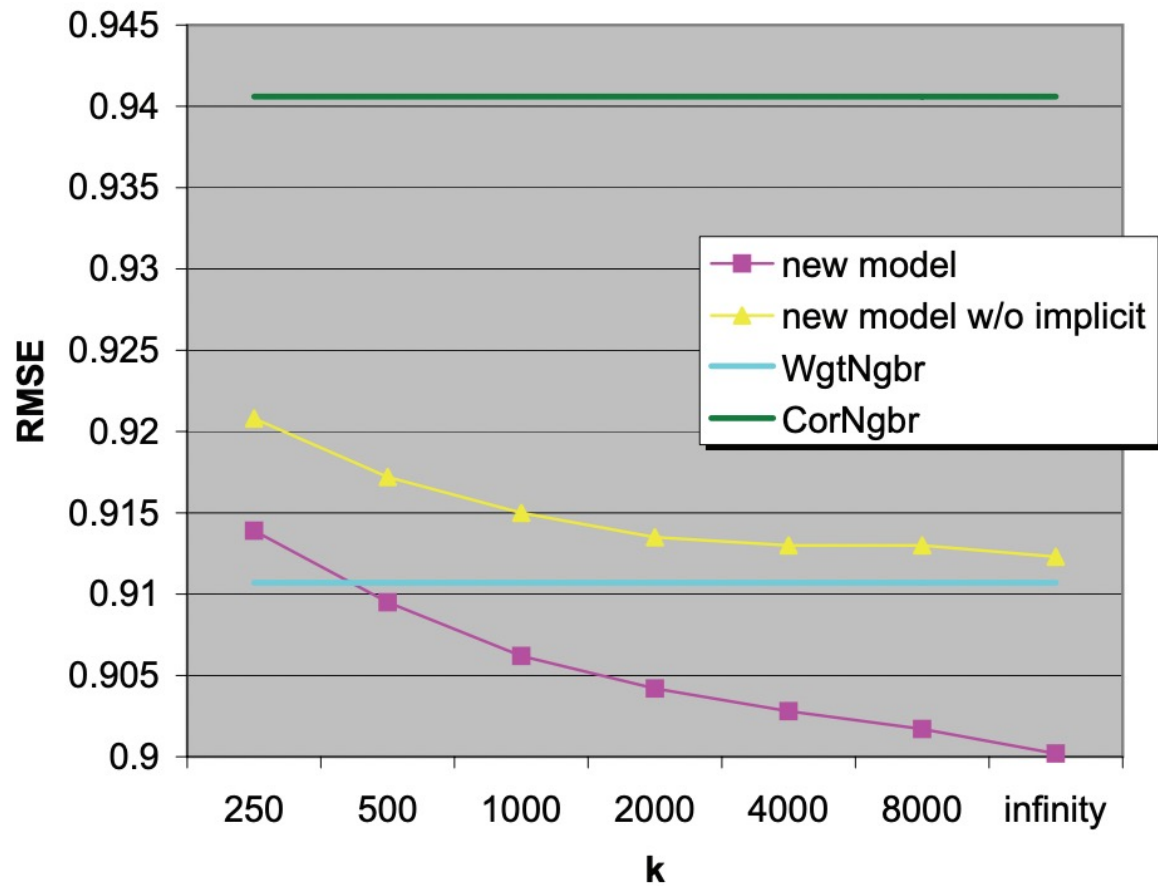
# Neighborhood Model - Optimization

Optimized by modifying the parameters  
by moving in the **opposite direction of the gradient**

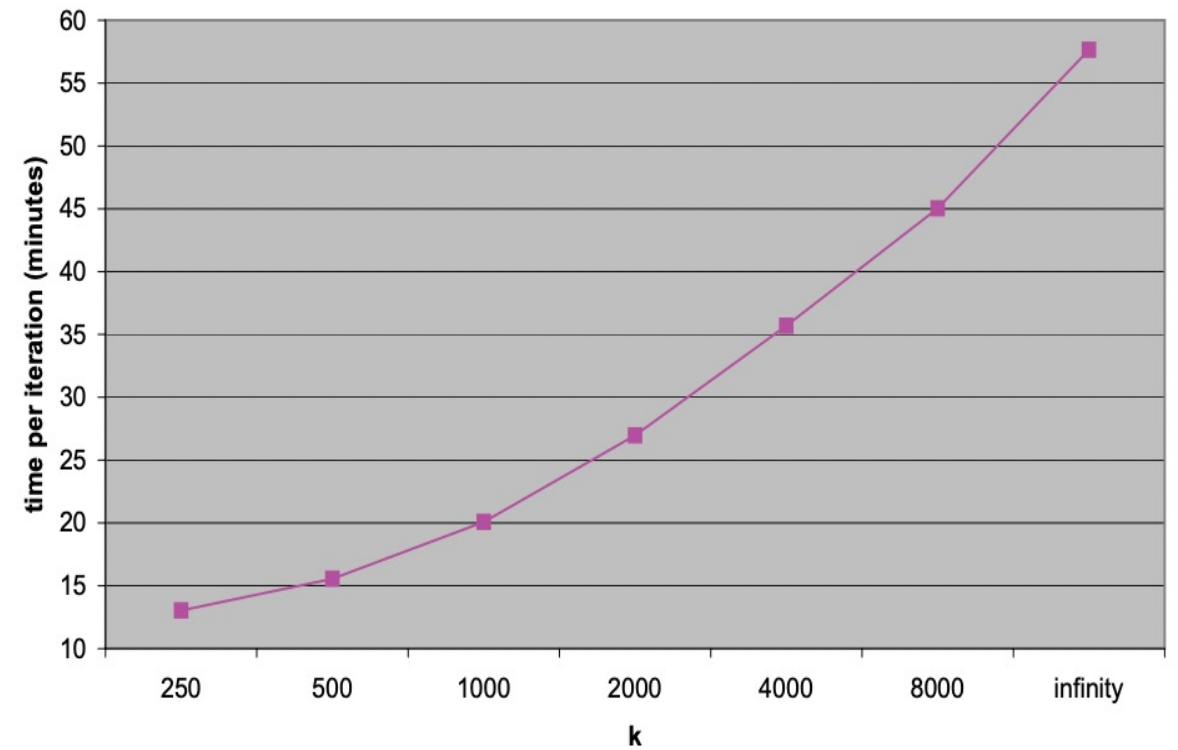
- $b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda_4 \cdot b_u)$
- $b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda_4 \cdot b_i)$
- $\forall j \in \mathcal{R}^k(i; u) :$   
 $w_{ij} \leftarrow w_{ij} + \gamma \cdot \left( |\mathcal{R}^k(i; u)|^{-\frac{1}{2}} \cdot e_{ui} \cdot (r_{uj} - b_{uj}) - \lambda_4 \cdot w_{ij} \right)$
- $\forall j \in \mathcal{N}^k(i; u) :$   
 $c_{ij} \leftarrow c_{ij} + \gamma \cdot \left( |\mathcal{N}^k(i; u)|^{-\frac{1}{2}} \cdot e_{ui} - \lambda_4 \cdot c_{ij} \right)$

# Neighborhood Model - Experimental results

## Comparison of neighborhood-based models



## Running times per iteration





# Latent Model - Asymmetric SVD

Extend SVD model by considering implicit information

$$\hat{r}_{ui} = b_{ui} + q_i^T \left( |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

Replace previous user factor  $p$

Rated by user  $u$  : ( Rated Item by user  $u$  ) ( Rated item  $\rightarrow$  factors )

Implicit feedback by user  $u$  : ( Implicit feedback by user  $u \rightarrow$  factors )

# Latent Model - Asymmetric SVD

## Benefits

$$\hat{r}_{ui} = b_{ui} + q_i^T \left( |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

<b>Fewer parameters</b>	the number of users is much larger than the number of products
<b>New users</b>	handle new users without needing to re-train the model
<b>Explainability</b>	abstracting users via an intermediate layer of user factors complicates explanations
<b>Efficient integration of implicit feedback</b>	Prevent to reflect more implicit feedback compare to importance of explicit feedback

# Latent Model - SVD++

Question : **predictive accuracy** OR **benefits of Asymmetric-SVD**

**SVD++ model**

$$\hat{r}_{ui} = b_{ui} + q_i^T \left( p_u + |\mathcal{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathcal{N}(u)} y_j \right)$$

Reflect **Implicit Feedback** on basic SVD model

# Latent Model - Comparison

## Comparison of SVD-based Model ( **SVD, Asymmetric-SVD, SVD++** )

Model	50 factors	100 factors	200 factors
SVD	0.9046	0.9025	0.9009
Asymmetric-SVD	0.9037	0.9013	0.9000
SVD++	0.8952	0.8924	0.8911

**Table 1: Comparison of SVD-based models: prediction accuracy is measured by RMSE on the Netflix test set for varying number of factors ( $f$ ). Asymmetric-SVD offers practical advantages over the known SVD model, while slightly improving accuracy. Best accuracy is achieved by SVD++, which directly incorporates implicit feedback into the SVD model.**

# Latent Model - Other models

## Temporal Dynamics

$$\hat{r}_{ui}(t) = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

an item's popularity might change over time

users to change their baseline ratings over time.

## Inputs with Varying Confidence Levels

$$\min_{p^*, q^*, b^*} \sum_{(u,i) \in K} c_{ui} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

a system might face adversarial users that try to tilt the ratings of certain items.

# Integrated Model

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left( p_u + |\mathbf{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathbf{N}(u)} y_j \right) + |\mathbf{R}^k(i; u)|^{-\frac{1}{2}} \sum_{j \in \mathbf{R}^k(i; u)} (r_{uj} - b_{uj}) w_{ij} + |\mathbf{N}^k(i; u)|^{-\frac{1}{2}} \sum_{j \in \mathbf{N}^k(i; u)} c_{ij}$$

general properties of the item and the user

the interaction between the user profile and the item profile

neighborhood tier

# Integrated Model - Optimization & Experiment

## Optimization by Gradient Descent

- $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_u)$
- $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_6 \cdot b_i)$
- $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot (p_u + |\mathcal{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathcal{N}(u)} y_j) - \lambda_7 \cdot q_i)$
- $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_7 \cdot p_u)$
- $\forall j \in \mathcal{N}(u) :$   
 $y_j \leftarrow y_j + \gamma_2 \cdot (e_{ui} \cdot |\mathcal{N}(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_7 \cdot y_j)$
- $\forall j \in \mathcal{R}^k(i; u) :$   
 $w_{ij} \leftarrow w_{ij} + \gamma_3 \cdot (|\mathcal{R}^k(i; u)|^{-\frac{1}{2}} \cdot e_{ui} \cdot (r_{uj} - b_{uj}) - \lambda_8 \cdot w_{ij})$
- $\forall j \in \mathcal{N}^k(i; u) :$   
 $c_{ij} \leftarrow c_{ij} + \gamma_3 \cdot (|\mathcal{N}^k(i; u)|^{-\frac{1}{2}} \cdot e_{ui} - \lambda_8 \cdot c_{ij})$

## Performance of the INTEGRATED MODEL

	50 factors	100 factors	200 factors
RMSE	0.8877	0.8870	0.8868
time/iteration	17min	20min	25min

**Table 2: Performance of the integrated model. Prediction accuracy is improved by combining the complementing neighborhood and latent factor models. Increasing the number of factors contributes to accuracy, but also adds to running time.**

# Evaluation through a Top-K Recommender

A solution with a **slightly better RMSE** will lead to  
**completely different and better recommendations?**

common case : providing **“top K recommendations”**

**Our goal is to find the relative place of these “interesting movies”**

for each such movie  $i$ , rated 5-stars by user  $u$ , select 1000 additional random movies

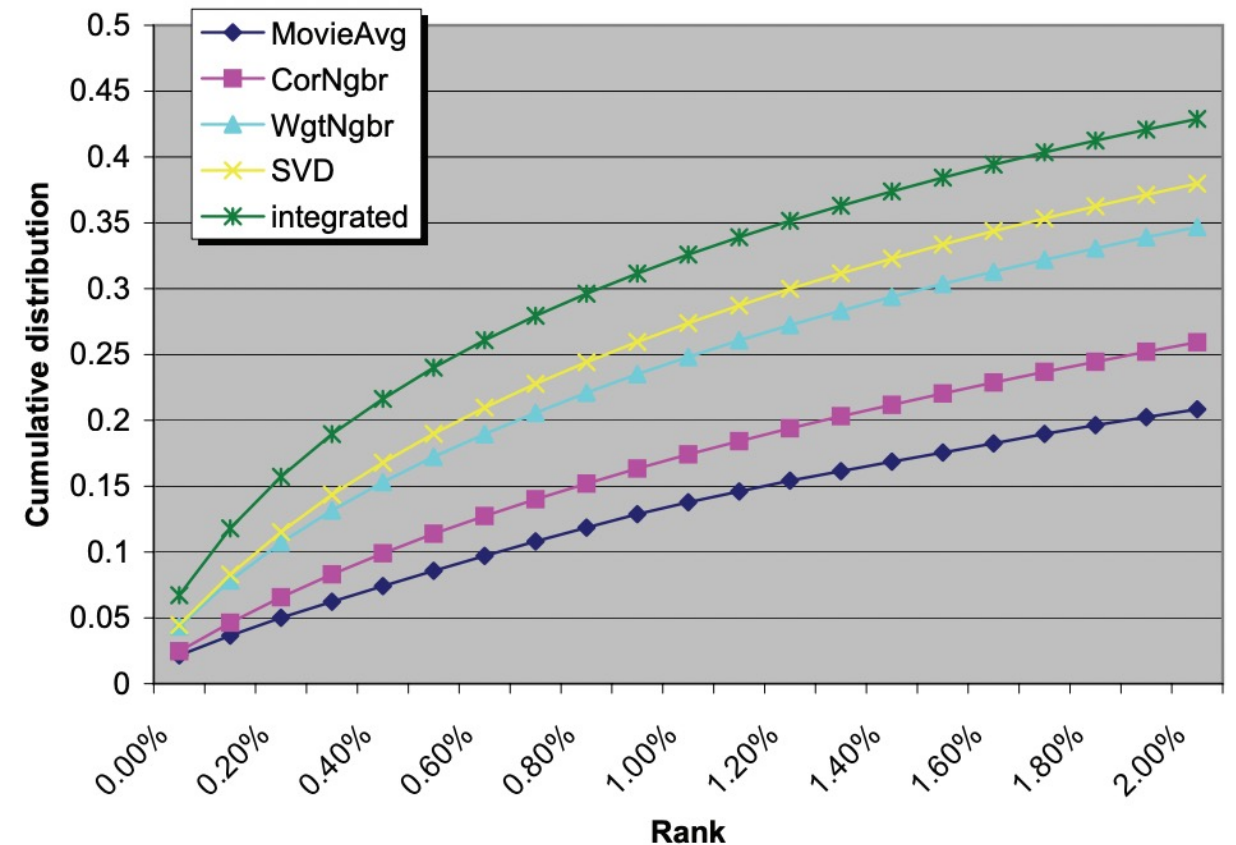
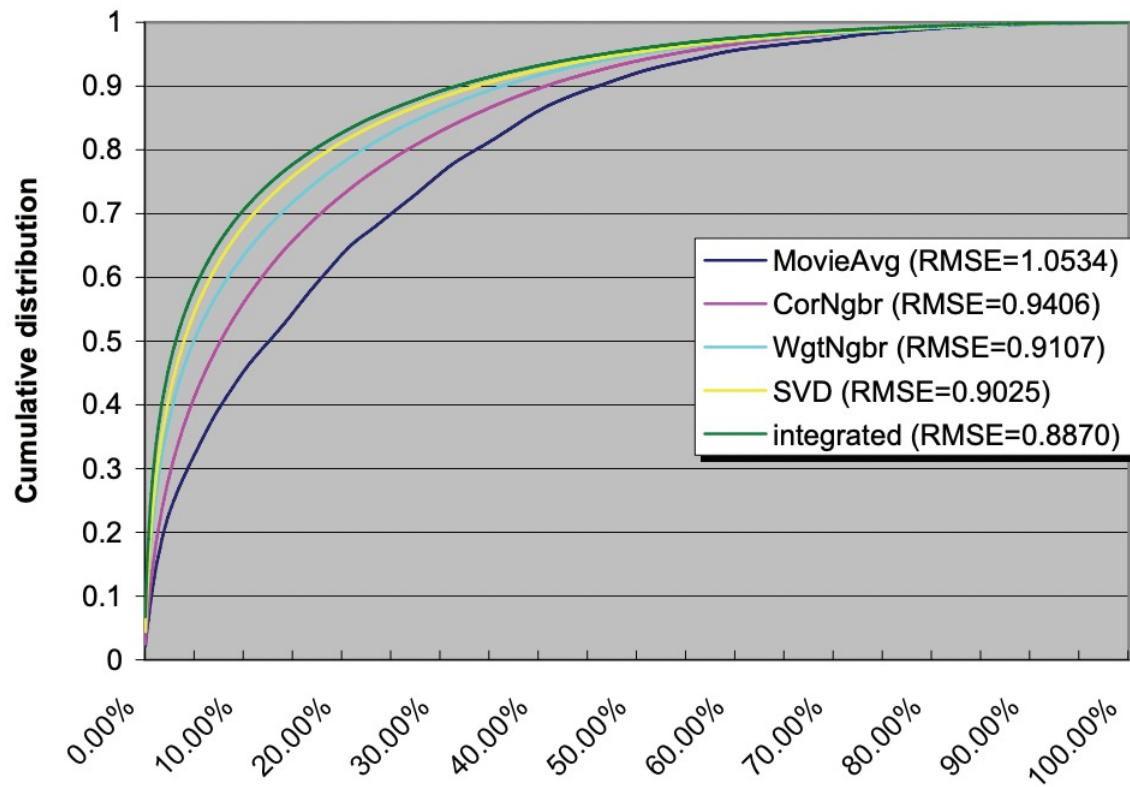
predict the ratings by  $u$  for  $i$  and for the other 1000 movies

order the 1001 movies based on their predicted rating, in a decreasing order



# Evaluation through a Top-K Recommender

Comparing the performance of 5 methods on a top-K recommendation



# DISCUSSION

**Propose IMPROVEMENTS to two of the most popular approaches**

**New Neighborhood Based Model**

Improve prediction accuracy + maintain explainability, ability to handle new users

**Integrated Model ( neighborhood model & SVD++ model )**

Improve prediction accuracy, integrating implicit feedback

**Evaluating top-K recommenders sharpens the differences between the methods**

**Further experimentation is needed with better sources of implicit feedback**

# Implement

## Modeling

```
def predict_r_ui(mat, u, i, mu, Bu, Bi, Rk_iu, W, Nk_iu, C, baseline_bu, baseline_bi, Q, P, N_u, Y):  
  
    p = P[u] + Y[N_u].sum(0)/np.sqrt(len(N_u))  
    Factor_part = np.dot(p, Q[i].T)  
  
    if len(Rk_iu)>0 :  
        bias_uj = mat[u, Rk_iu] - (mu + Bu[u] + Bi[Rk_iu])  
        Neigh_part = np.dot(bias_uj, W[i, Rk_iu]) / np.sqrt(len(Rk_iu))  
    else :  
        bias_uj = 0  
        Neigh_part = 0  
  
    if len(Nk_iu)>0 :  
        Neigh_part = Neigh_part + np.sum(C[i, Nk_iu]) / np.sqrt(len(Nk_iu))  
  
    return mu + Bu[u] + Bi[i] + Factor_part + Neigh_part, bias_uj
```

# Implement

## Gradient

```
for it in range(n_iter):
    start = time.time() # 시간측정 시작
    for u, i, v in zip(cx.row, cx.col, cx.data):
        N_u = bi_index[u]
        Nk_iu = np.flip(np.argsort(S[i,:].toarray()))[:k].ravel()
        Rk_iu = Nk_iu

        e_ui, bias = compute_e_ui(mat, u, i, mu, Bu, Bi, Rk_iu, W, Nk_iu, C, baseline_bu, baseline_bi, Q, P, N_u, Y)

        Bu[u] = Bu[u] + gamma1 * (e_ui - l_reg6 * Bu[u])
        Bi[i] = Bi[i] + gamma1 * (e_ui - l_reg6 * Bi[i])

        Q[i] = Q[i] + gamma2 * (e_ui * (P[u] + (np.sum(Y[N_u], axis=0))/np.sqrt(len(N_u))) - l_reg7 * Q[i])
        P[u] = P[u] + gamma2 * (e_ui * Q[i] - l_reg7 * P[u])

        Y[N_u] = Y[N_u] + gamma2 * ((e_ui * Q[i])/np.sqrt(len(N_u))).reshape(1,-1) - l_reg7 * Y[N_u]

    if len(Rk_iu)>0:
        W[i][Rk_iu] = W[i][Rk_iu] + gamma3 * (e_ui * bias / np.sqrt(len(Rk_iu)) - l_reg8 * W[i][Rk_iu])
    if len(Nk_iu)>0:
        C[i][Nk_iu] = C[i][Nk_iu] + gamma3 * (e_ui / np.sqrt(len(Nk_iu)) - l_reg8 * C[i][Nk_iu])

gamma1 *= 0.9
gamma2 *= 0.9
gamma3 *= 0.9
```

# Implement

## Experiment

### Comparison of SVD-based Model

	25_factors	50_factors	100_factors
SVD	0.8380	0.7054	0.5602
SVD++	0.8182	0.6969	0.5434
Integrated	0.8074	0.6745	0.5277

### Performance of Integrated Model

Integrated	50_factors	100_factors	200_factors
RMSE	0.6745	0.5277	0.4380
Time / Iter	1781.0ms	1789.0ms	1804.2ms