Recommender Systems: *Latent-Factors Based*

Mining Massive Datasets

Materials provided by Prof. Carlos Castillo — https://chato.cl/teach

Instructor: Dr. Teodora Sandra Buda — https://tbuda.github.io/

Sources

- Data Mining, The Textbook (2015) by Charu Aggarwal (Section 18.5) – slides by Lijun Zhang
- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. (Chapter 9) - slides A, B

Key idea

- Summarize the correlations across rows and columns in the form of lower dimensional vectors, or **latent** factors
- These latent factors become **hidden** variables that encode the correlations in the data matrix in a concise way and can be used to make **predictions**
- Estimation of the k-dimensional dominant latent factors is often possible even from **incompletely** specified data

Modeling

- *n* users: $\overline{U_1}, \ldots, \overline{U_n} \in \mathbb{R}^k$
- *d* items: $\overline{I_1}, \ldots, \overline{I_d} \in \mathbb{R}^k$
- Approximate rating r_{ij} by

$$r_{ij} \approx \left\langle \overline{U_i}, \overline{I_j} \right\rangle = \overline{U_i}^T \overline{I_j} = \overline{I_j}^T \overline{U_i}$$

- Approximate rating matrix $D = [r_{ij}]_{n \times d}$

$$D \approx F_{\text{user}} F_{\text{item}}^T$$

$$F_{\text{user}} \in \mathbb{R}^{n \times k}$$
$$F_{\text{item}} \in \mathbb{R}^{d \times k}$$

Matrix factorization

- Factorizing D into U and V

$$D \approx UV^T$$

- Objective when D is fully observed

$$\min \left\| D - UV^T \right\|_F^2$$

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$$

- Objective when D is partially observed

$$\min \sum_{(i,j)\in\Omega} \left(D_{ij} - \overline{U_i}^T \overline{V_j} \right)^2$$

  - Ω is the set

Non-negative, regularized matrix factorization

- Matrix factorization $D \approx UV^T$
- Objective:

$$\min \sum_{(i,j)\in\Omega} \left(D_{ij} - \overline{U_i}^T \overline{V_j}\right)^2 + \lambda \left(\|U\|_F^2 + \|V\|_F^2\right)$$

  - Ω is the set of observed cells in the matrix
  - U ≥ 0, V ≥ 0

Example: grocery shopping

Example: grocery shopping

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

- This purchase history indicates the number of time each person has purchased an item
- For clarity we're dealing with categories of items, but they can be the items themselves

This example (2018) by Piotr Gabrys

In Python

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

Python code

```
V = np.array(
          [[0,1,0,1,2,2],
           [2,3,1,1,2,2],
           [1,1,1,0,1,1],
           [0,2,3,4,1,1],
           [0,0,0,0,1,0]])

V = pd.DataFrame(V, columns=['John', 'Alice',
'Mary', 'Greg', 'Peter', 'Jennifer'])

V.index = ['Vegetables', 'Fruits', 'Sweets',
'Bread', 'Coffee']
```

This example (2018) by Piotr Gabrys

Matrix factorization (V ≃ WH)

Matrix W (items x factors) with possible names for each factor added for legibility

	Fruits pickers	Bread eaters	Veggies
Vegetables	0.00	0.04	2.74
Fruits	1.93	0.15	0.47
Sweets	0.97	0.00	0.00
Bread	0.00	2.66	1.18
Coffee	0.00	0.00	0.59

This example (2018) by Piotr Gabrys

Python code

```
from sklearn.decomposition import NMF
nmf = NMF(3)
nmf.fit(V)

H = pd.DataFrame(np.round(nmf.components_,2), columns=V.columns)
H.index = ['Fruits pickers', 'Bread eaters',  'Veggies']

W = pd.DataFrame(np.round(nmf.transform(V),2), columns=H.index)
W.index = V.index
```

Matrix W (items x factors)

	Fruits pickers	Bread eaters	Veggies
Vegetables	0.00	0.04	2.74
Fruits	1.93	0.15	0.47
Sweets	0.97	0.00	0.00
Bread	0.00	2.66	1.18
Coffee	0.00	0.00	0.59

Possible names for each factor added for legibility: these names are **not needed for the method to work**

Matrix H (factors x people)

	John	Alice	Mary	Greg	Peter	Jennifer
Fruits pickers	1.04	1.34	0.55	0.26	0.89	0.90
Bread eaters	0.00	0.60	1.12	1.36	0.03	0.07
Veggies	0.00	0.35	0.00	0.34	0.77	0.69

This example (2018) by Piotr Gabrys

Reconstruction

Original matrix (V)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

Reconstructed matrix (W H)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0.00	0.98	0.04	0.99	2.11	1.89
Fruits	2.01	2.84	1.23	0.87	2.08	2.07
Sweets	1.01	1.30	0.53	0.25	0.86	0.87
Bread	0.00	2.01	2.98	4.02	0.99	1.00
Coffee	0.00	0.21	0.00	0.20	0.45	0.41

```
reconstructed = pd.DataFrame(np.round(np.dot(W,H),2), columns=V.columns)
reconstructed.index = V.index
```

This example (2018) by Piotr Gabrys

Recommendation

Original matrix (V)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

Reconstructed matrix (W H)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0.00	0.98	0.04	0.99	2.11	1.89
Fruits	2.01	2.84	1.23	0.87	2.08	2.07
Sweets	1.01	1.30	0.53	0.25	0.86	0.87
Bread	0.00	2.01	2.98	4.02	0.99	1.00
Coffee	0.00	0.21	0.00	0.20	0.45	0.41

If you were to recommend one product to someone, what would you recommend and to whom?

This example (2018) by Piotr Gabrys

Evaluation

Direct evaluation

- Randomized controlled experiment
  - Renamed A/B testing for … reasons
  - People are split randomly in control/experimental
  - Control group: receives one type of recommendation
  - Experimental group: receives another type
- Metrics such as CTR, retention, etc.
- Requires infrastructure, users, policies

Evaluating with existing data

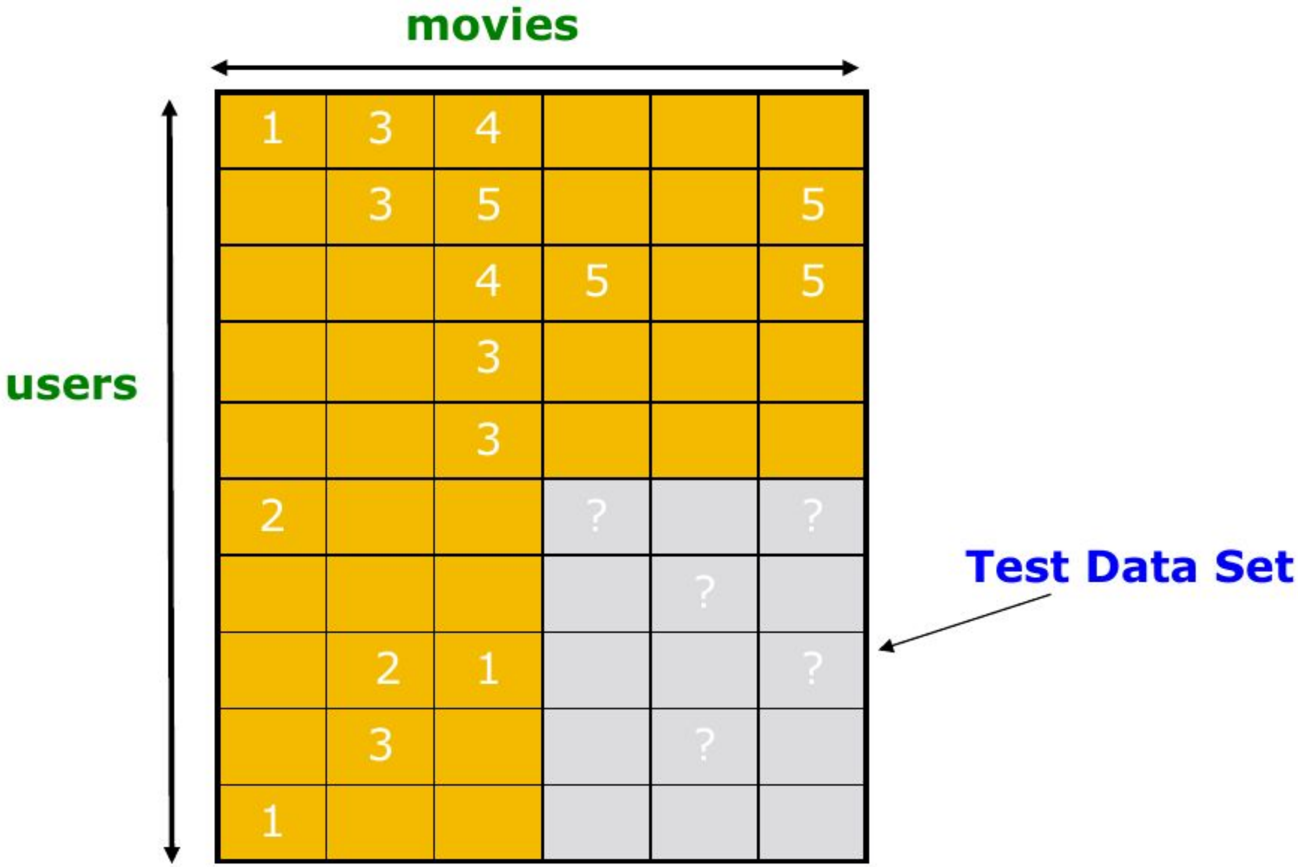
movies

users

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. ([Chapter 9]) - slides [A], [B]

Evaluating with existing data



Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. (Chapter 9) - slides A, B

Evaluation metrics

- RMSE (root of mean of squared errors)

$$\sqrt{E[(x-\hat{x})^2]}$$

- Precision @ k
  - % of recommendations that are correct among those in the top k positions
- Rank correlation
  - Spearman’s correlation between system and user

Evaluating is hard

- Accuracy is not all
- We also want diversity
- We want to be contextually sensitive
- The order of predictions matters
- RMSE might penalize a method that does well for high ratings but bad for others

Summary

Things to remember

- Interaction-based recommendations
  - Latent factors based
- Evaluation methods

Exercises for TT16-TT18

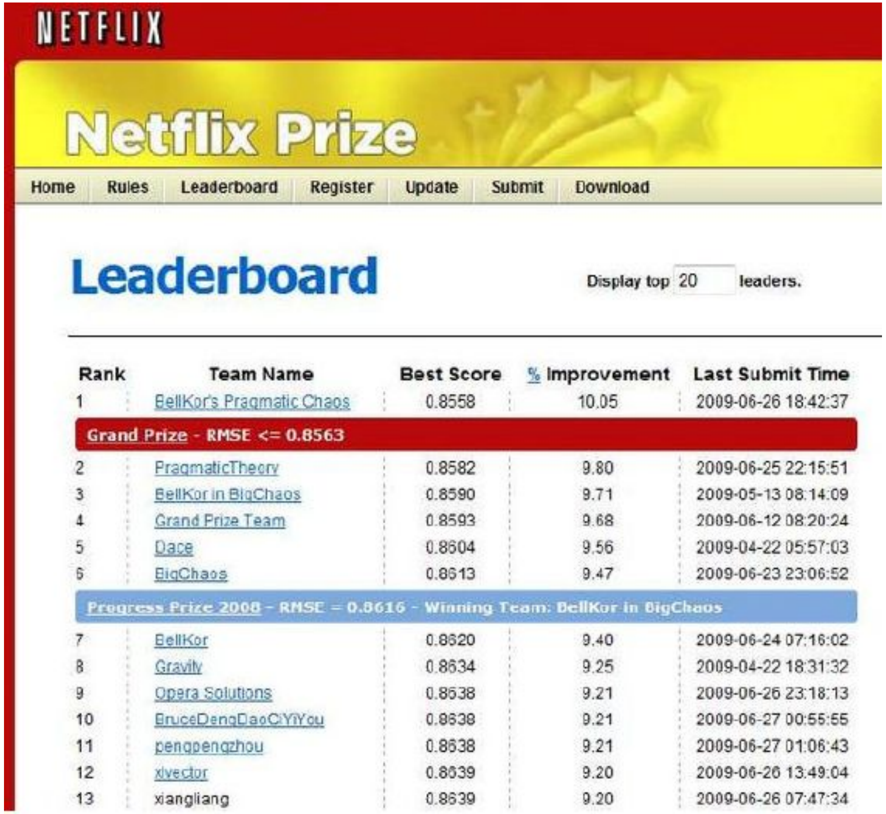
- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. Note that some exercises cover advanced concepts:
  - Exercises 9.2.8
  - Exercises 9.3.4
  - Exercises 9.4.6

Additional contents
(not included in exams)

Example 2: Netflix prize

Example 2: Netflix prize (2009)

- Netflix offered $1,000,000 to anyone beating their algorithm by 10% in **RMSE**
- Provided 100M (user,movie) ratings for training
- Held a testing set and allowed one guess/day on the testing set to create a leader board



NETFLIX

Netflix Prize

Home | Rules | Leaderboard | Register | Update | Submit | Download

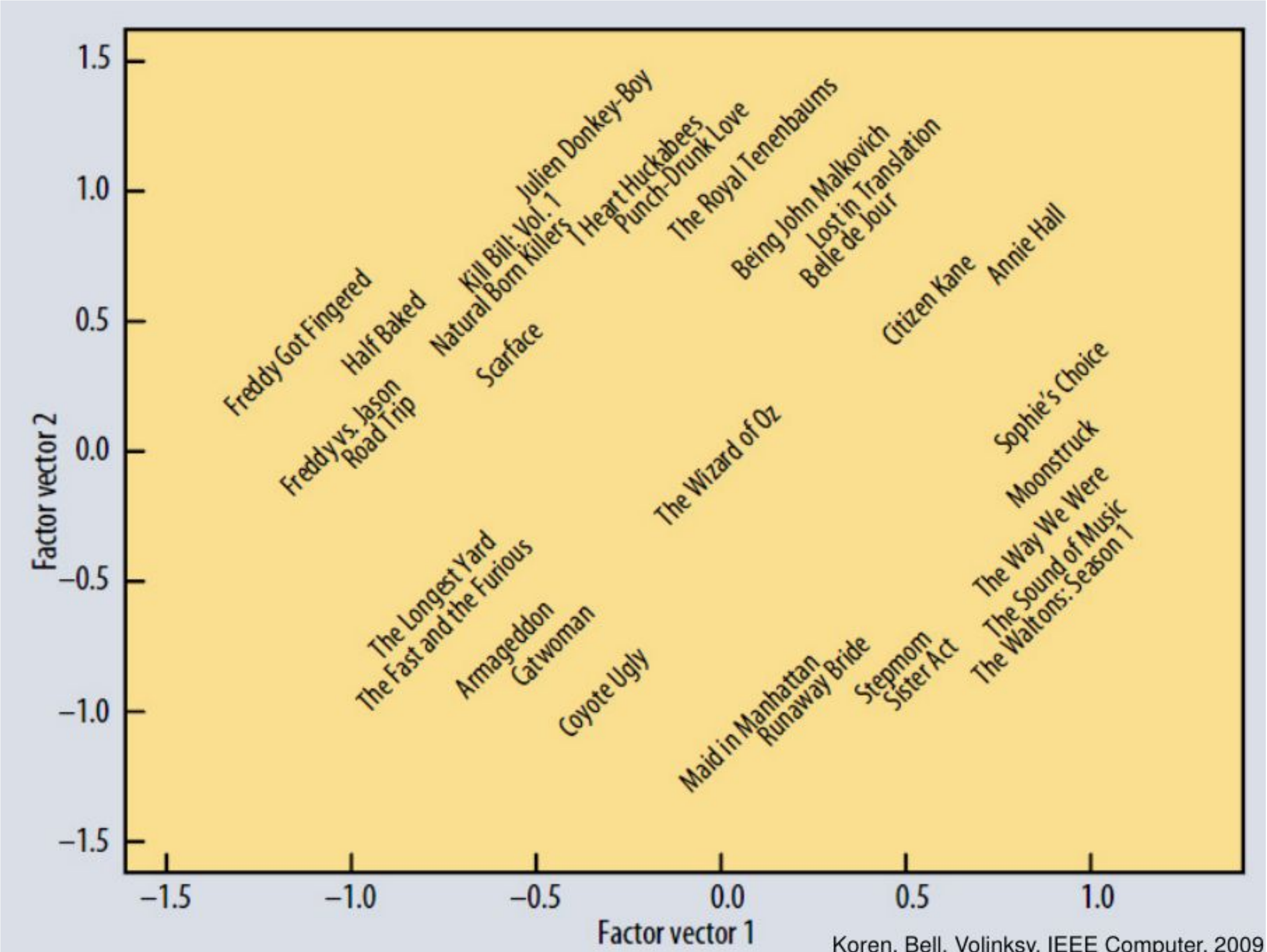
Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
7	BellKor	0.8620	9.40	2009-06-24 07:16:02
8	Gravity	0.8634	9.25	2009-04-22 18:31:32
9	Opera Solutions	0.8638	9.21	2009-06-26 23:18:13
10	BruceDengDaoCiYiYou	0.8638	9.21	2009-06-27 00:55:55
11	pengpengzhou	0.8638	9.21	2009-06-27 01:06:43
12	xivector	0.8639	9.20	2009-06-26 13:49:04
13	xiangliang	0.8639	9.20	2009-06-26 07:47:34

Latent factors

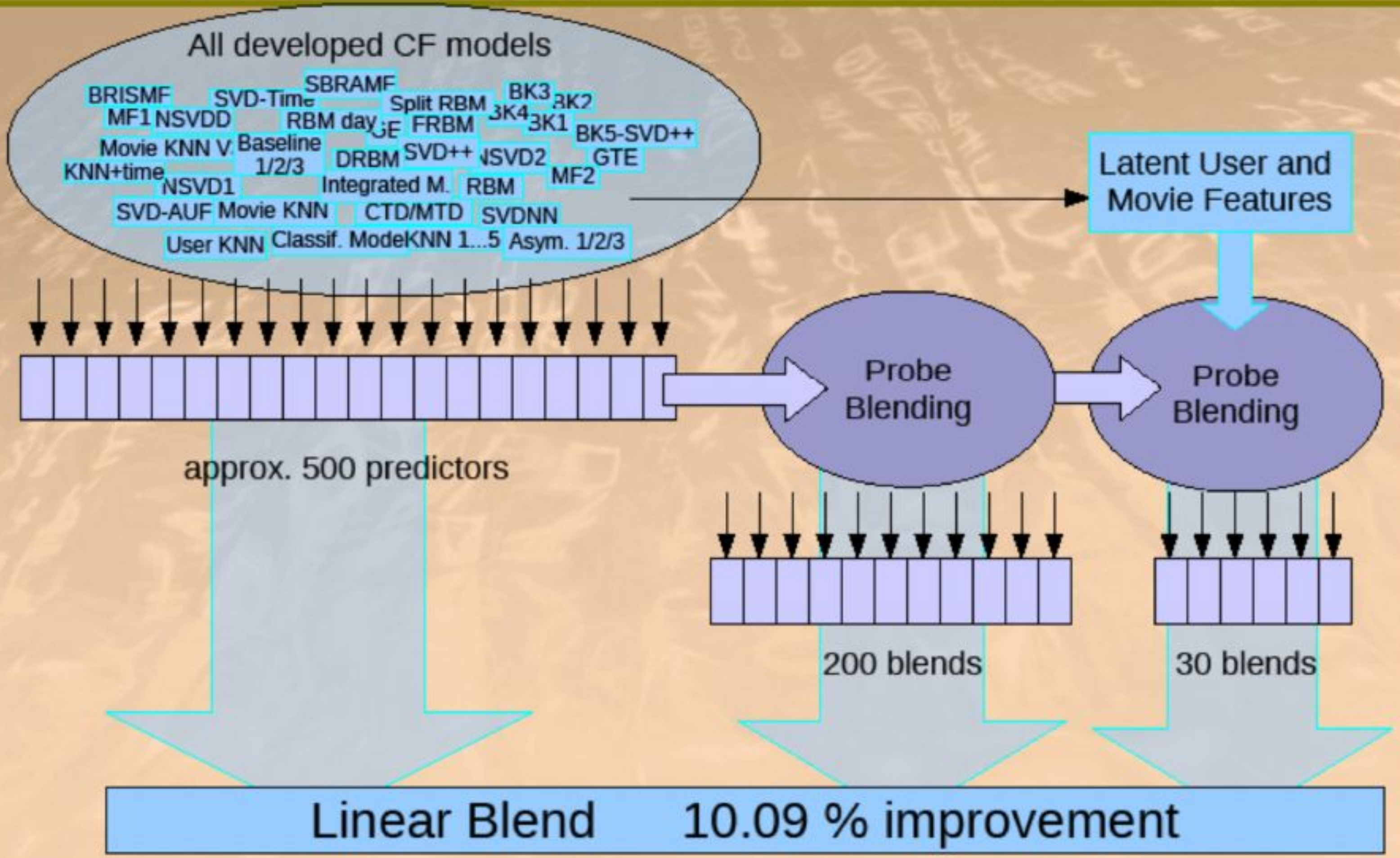
In latent factor space, similar movies are mapped to similar points



Koren, Bell, Volinksy, IEEE Computer, 2009

Shortly before deadline ...

The big picture
Solution of BellKor's Pragmatic Chaos



Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	[illegible]	[illegible]	[illegible]
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	
6	PragmaticTheory	0.8594	9.77	

26 July 2009.- Bellkor team submits 40 minutes before the deadline, "The Ensemble" team made of a mix of other teams submitted 20 minutes before the deadline.

Bellkor team wins one million dollars