DSA5101
Introduction to Big Data for Industry

Recommendation Systems

LX Zhang
Department of Mathematics
National University of Singapore

1. Recommendation Systems

- Recommendation system is an extensive class of Web applications that involves predicting user responses to options.

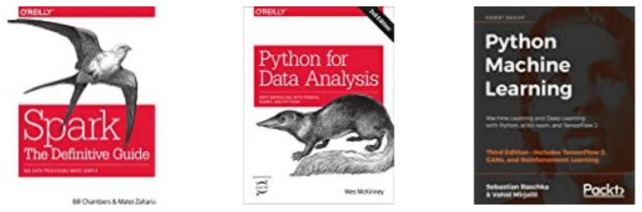
  --- Product Recommendations

  Amazon provides each customer with some suggestions of books that they might like to buy

"Customers who viewed (bought) this item also viewed (bought)"

Spark The Definitive Guide
Data Analytics with Hadoop
Mining of Massive Datasets
Python for Data Analysis

"Products related to this item"



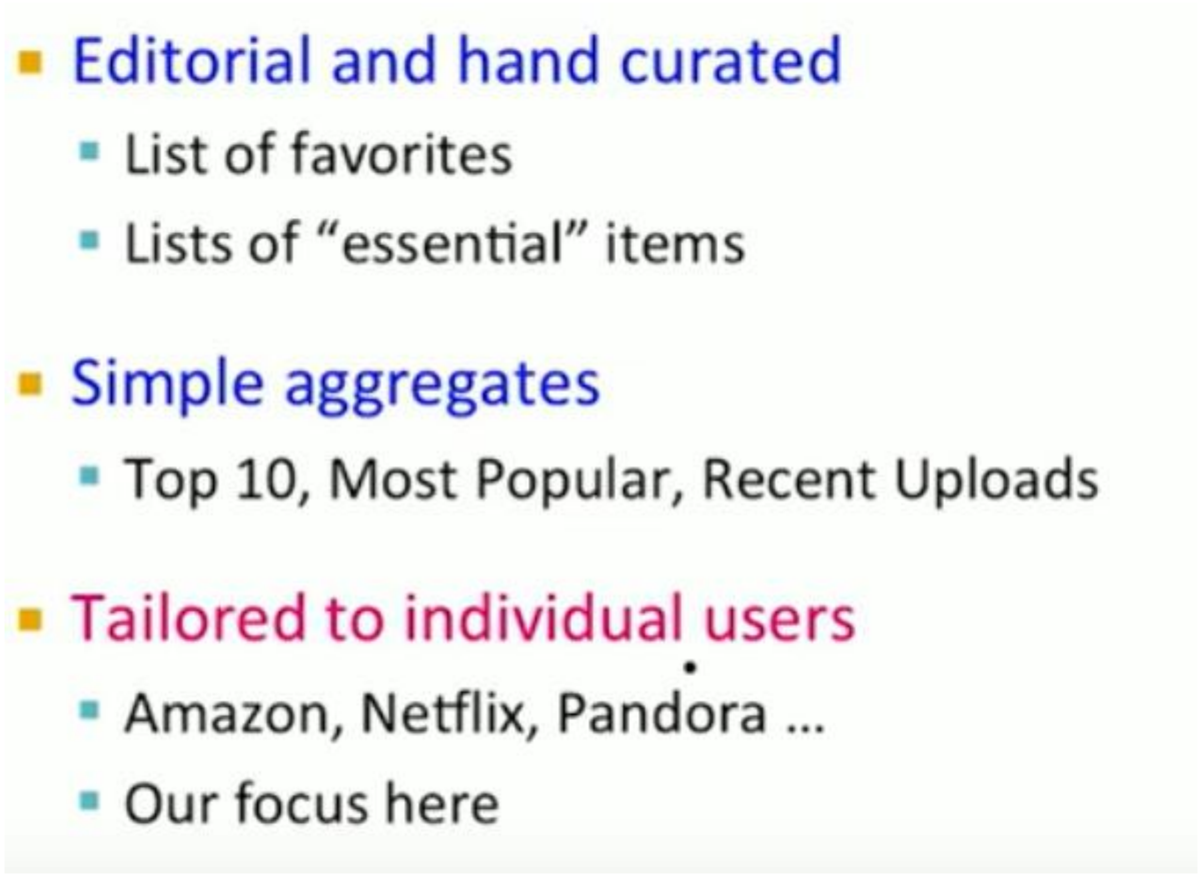
-- Movie Recommendations

Netflix offers its customers recommendations of movies they might like to watch, which are based on users' ratings

-- News Articles

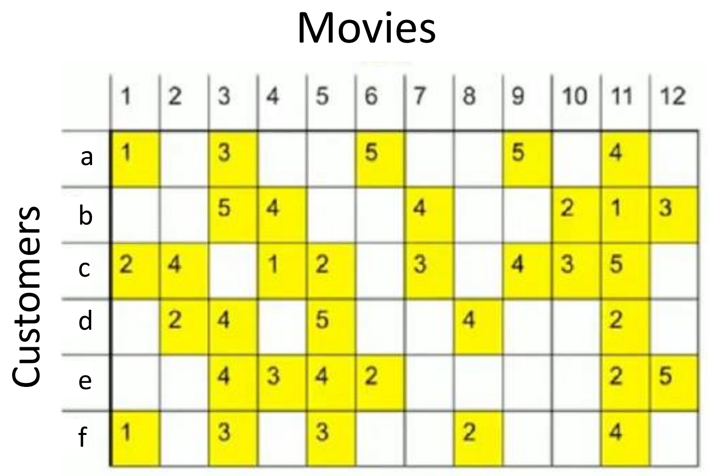
News services have attempted to identify articles of interest to readers, based on the articles that they have read in the past.

2. The Recommendation Problem



- Editorial and hand curated
  - List of favorites
  - Lists of “essential” items
- Simple aggregates
  - Top 10, Most Popular, Recent Uploads
- Tailored to individual users
  - Amazon, Netflix, Pandora …
  - Our focus here

- Recommendation Systems involves two identities: Customers (C) and Items (S)
- The relationship between users and items are presented by a matrix, called the utility matrix U, which is usually sparse.
  It is a partial function f: $S \times C \rightarrow R$


Movies
Customers

	1	2	3	4	5	6	7	8	9	10	11	12
a	1		3			5			5		4	
b			5	4			4			2	1	3
c	2	4		1	2		3		4	3	5	
d		2	4		5			4			2	
e			4	3	4	2					2	5
f	1		3		3			2			4	

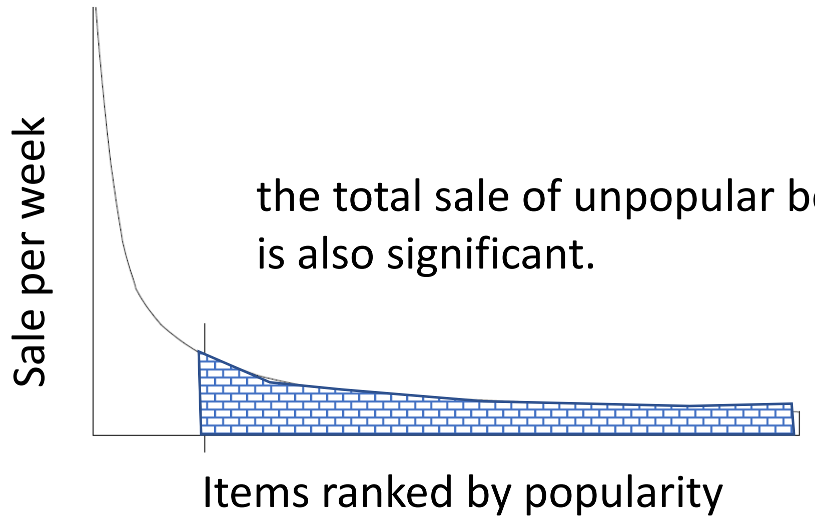
- Problem: Estimate the unknown values or unknown high values in the matrix
- In generally, the problem is called matrix imputation (completion)

1M Netflex Prize (2009)

- **Movie Recommendation System**
- **Training data**
  -- 100 million ratings
  -- 17,770 movies and 480,000 users
  -- collected from 2000 to 2005.
- **Test data**
  -- a few ratings of each user after 2005
  -- 2.8 millions in total
- **Criteria**:
  -- 10% improvement in accuracy

3. Why Recommendation?

- The long tail phenomenon that makes recommendation systems necessary for online retailers
  -- Shelf space is a scarce commodity for traditional retailers. As such, a street-side bookstore can only display only popular books
  -- A online bookstore can literally sale any book that has ever been published.
- Recommenders can create demand for an obscure title
  -- a few successful story on Amazon



4. Issues of Recommendation System

- How to gather “known” ratings?

  -- Explicit.

  Ask customers to rate items.

  Not scalable: Not many people rate and reviews

  -- Implicit

  Learning ratings from user actions

  Biased to high rating.

  -- Combination of both explicit and implicit.

- How to estimate unknown ratings from known ones

  -- Key problem: little information;
  new items have no ratings;
  new customers have no rating history

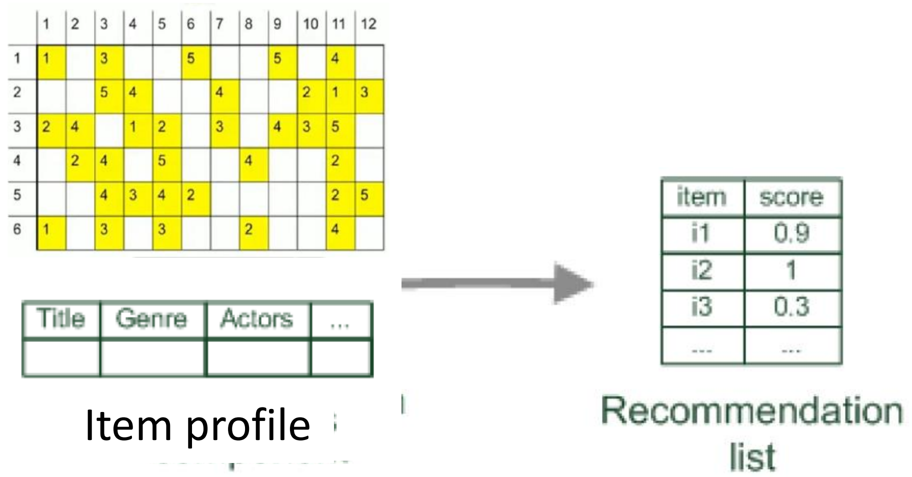
  -- Three approaches

  1) Context based
     Show me more similar to what I liked.
     This requires a metric to measure the similarity of items.
  2) Collaborative filtering
     What’s popular among my peers.
  3) Latent factor based
     Solve an optimization problem
     How to measure the accuracy of a recommendion method

5. Content-based recommendation

- Goal recommend to a user those items that are similar to previous items rated highly by the user.

- Examples:
  --- Recommend movies with same actor(s), director, genre…
  --- Recommend articles with the similar content



	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Title	Genre	Actors	...

Item profile

item	score
i1	0.9
i2	1
i3	0.3
---	---

Recommendation list

Item Profile: Given in another matrix P where rows are items and columns are features. Feature values can be Boolean or real-valued.

	a_1	a_2	a_3	…	a_k
i_1	1	0	1	0	0
i_2	1	0	1	0	1
i_3	1	1	0	1	1

Feature examples

--- Movies: producer, actor(s), director,…

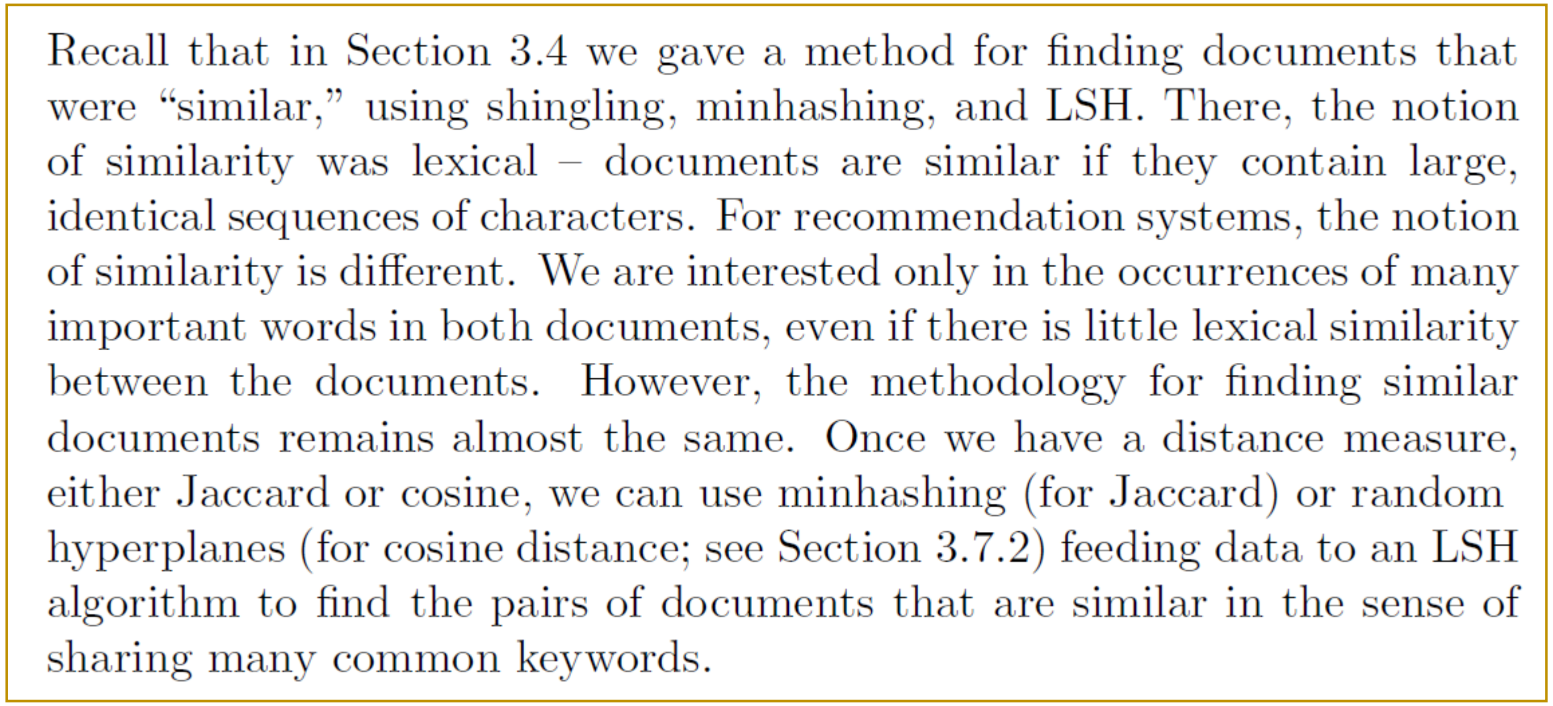
Challenges: How to pick important explainable features?

Say for news articles, images

Test Features

- Profile Set of “important” words in a document
- How to pick important words?

  TF-IDF: Term Frequency × Inverse Doc Frequency



Recall that in Section 3.4 we gave a method for finding documents that were “similar,” using shingling, minhashing, and LSH. There, the notion of similarity was lexical – documents are similar if they contain large, identical sequences of characters. For recommendation systems, the notion of similarity is different. We are interested only in the occurrences of many important words in both documents, even if there is little lexical similarity between the documents. However, the methodology for finding similar documents remains almost the same. Once we have a distance measure, either Jaccard or cosine, we can use minhashing (for Jaccard) or random hyperplanes (for cosine distance; see Section 3.7.2) feeding data to an LSH algorithm to find the pairs of documents that are similar in the sense of sharing many common keywords.

Definition of TF-IDF

Features: words
Items: documents

TF_{ij} (normalized term frequency) = $\frac{\text{Frequency } f_{ij} \text{of word } i \text{ in doc } j}{\max_k f_{kj}}$

$IDF_i = \log \frac{\text{No. of doc in which word } i \text{ apepars}}{\text{Total no. of doc}}$

TF-IDF score for word i and document j = $TF_{ij} \times IDF_i$

Recommendation Algorithm

Inputs: Item profile matrix P ("content") and utility matrix U
Output: Preference for unrated items for user x

Step 1: Computer a vector v to describe the features of rated items
is called the user profile.

Step 2: Predict preference of x for items which the user has not rated
-- Compute the similarity between v and p for each item p.
-- Use the similarity values to make a recommendation.

	a_1	a_2	a_3	…	a_k
i_1	1	0	1	0	0
i_2	1	0	1	0	1
i_3	1	1	0	1	1
i_4	1	1	0	1	1
i_5	1	1	0	0	1

v = (1, 2/3, 1/3, …, 1)

Sim(v, item1) = $\cos(v, i_1) = 0.55$
Sim(v, item5) = $\cos(v, i_2) = 0.9$

Rrecommend item5, not item 1.

Step 1: Computer the user profile v

- (Assumption) The profile of the all k items rated by the user are $g_1, g_2, \dots, g_k$
- Method 1:
  (weighted) average of rated item profiles $g_1, g_2, \dots, g_k$
- Method 2:
  Normalized weights using mean rating of a user.

	Feature A	Feature B	Feature C
Movie1	1	0	1
Movie2	0	1	0
Movie3	1	1	0
Movie4	1	1	1
Movie5	0	1	1
Movie6	1	0	1
Movie7	1	1	0

Let a person have watched the first 4 movies.

By method 1. Average profile.

$$v = \frac{1}{4}[(1,0,1) + (0,1.0) + (1,1,0) + (1,1,1)]$$
$$= \left(\frac{3}{4}, \frac{3}{4}, \frac{2}{4}\right)$$

Assume the user rated the 4 movies with score 1, 2, 3, 5.

Weight average profile:

$$\frac{1}{4}[\,1\cdot(1,0,1) + 2\cdot(0,1.0) + 3\cdot(1,1,0) + 5\cdot(1,1,1)]$$

Step 1: Computer the user profile v

- (Assumption) The profile of the all k items rated by the user are $g_1, g_2, \dots, g_k$
- Method 1:
  (weighted) average of rated item profiles $g_1, g_2, \dots, g_k$
- Method 2:
  Normalized weights using mean rating of a user.

	Feature A	Feature B	Feature C
Movie1	1	0	1
Movie2	0	1	0
Movie3	1	1	0
Movie4	1	1	1
Movie5	0	1	1
Movie6	1	0	1
Movie7	1	1	0

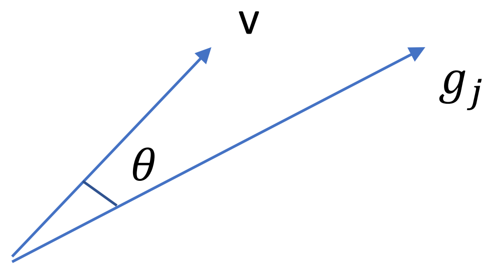
Let a person have watched the first 4 movies.

By method 2:

Assume the user rated the 4 movies with score
1, 2, 3, 5. and mean rating is 2.
Weight average profile:
$\frac{1}{5}[\,(1-2)\cdot(1,0,1)+0\cdot(0,1.0)+1\cdot(1,1,0)+3\cdot(1,1,1)]$

Step 2: Make predictions

- (Assumption) User profile v
- For each item j not rated by the user
  -- compute cosine similarity value
  $$\mathrm{Sim}(v, g_j) = \frac{v \cdot g_j}{|v||g_j\}} = \cos\theta$$
- Recommend item with large similarity.



Pros

-- No need for data on other users.

-- Able to recommend to users with unique tastes

-- Able to recommend new & unpopular items

-- Can explain recommended items by listing the content-features

Cons

-- Finding the appropriate features is hard, e.g., images, movies, music

-- Cold start problem new users. How to build a user profile?

-- Overspecialization

  -- Never recommend items outside user's content profile

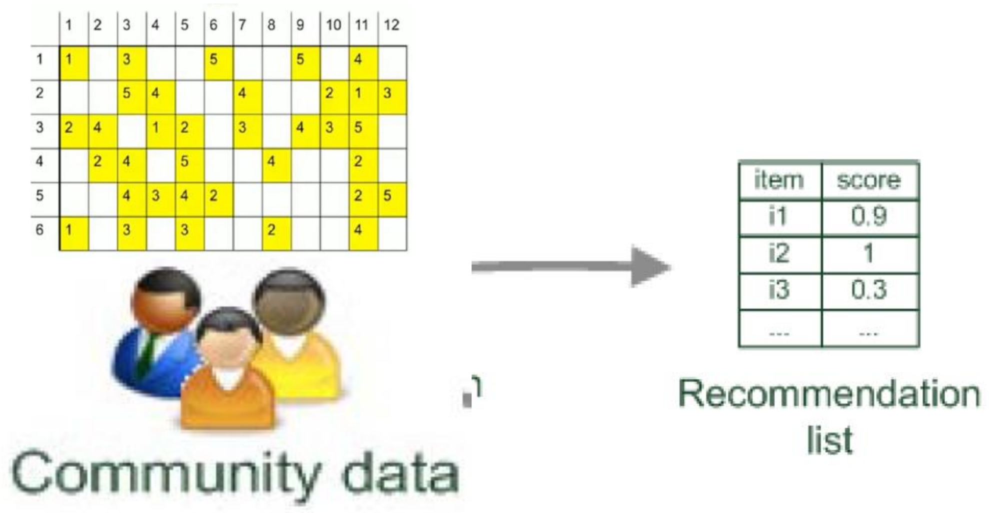
  -- People might have multiple interests

-- Unable to exploit quality judgments of other users

6. Collaborative filtering

- Goal: Identifying similar users and recommending what similar users like.

- Examples: Amazon.
    -



	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

item	score
i1	0.9
i2	1
i3	0.3
---	---

Algorithm

Inputs: Utility matrix U, a user x
Output: User preferences for unrated items

Step 1: Compute a set N of users (whose ratings)
that are "similar" to x

Step 2: Estimate x's ratings based on ratings of users in N

Finding “Similar” Users with the Utility matrix

- Let r(x) be the vector of user x’s ratings
- Jaccard similarity measure

  -- Consider r(x) as a subset of items

  -- Problem: Ignores the value of the rating
- Cosine similarity measure;

  -- treat missing ratings as 0 (negative).
- Pearson correlation coefficient (Normalized Cosine similarity measure)

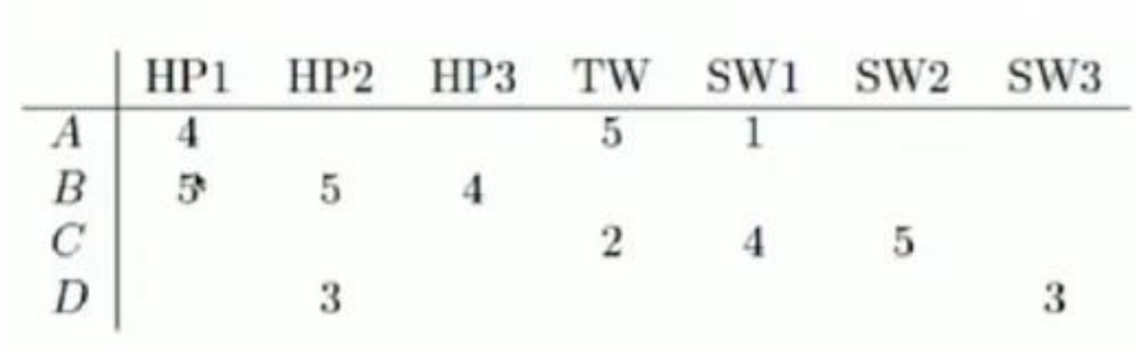
r(x) = [*, _, _, *, ***]
r(y) = [*, _, **, **, _]

r(x), r(y) as subsets:
  r(x) = {1, 4, 5}
  r(y)= {1, 3, 4}
Jaccard similarity = 2/4

r(x), r(y) as vectors:
  r(x) = (1, 0, 0, 1, 3)
  r(y) = (1, 0, 2, 2, 0)

Cosine similarity = $\frac{3}{\sqrt{11}\,\sqrt{9}} = 1/\sqrt{11}$

Example 1



	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Observation: User A is more similar to User B than User C.

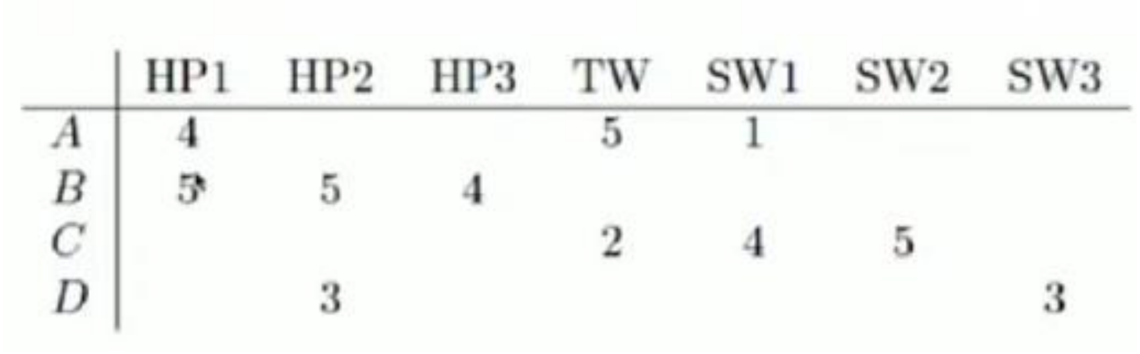
Jaccard similarity:

JS(A, B)= 1/5 < JS(A, C)=2/4

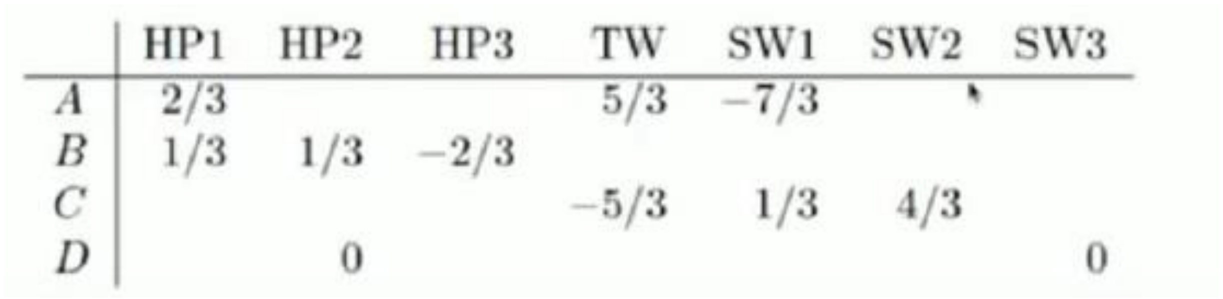
Cosine similarity:

CS(A, B)= 0.38 > CS(A, C) = 0.32

Normalize by subtracting the row mean



	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3



	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	−7/3		
B	1/3	1/3	−2/3				
C				−5/3	1/3	4/3	
D		0					0

- sim(A,B) = cos(r_A, r_B) = 0.09; sim(A,C) = -0.56
  - sim(A,B) > sim(A,C)
- Captures intuition better
  - Missing ratings treated as "average"

Prediction of Ratings

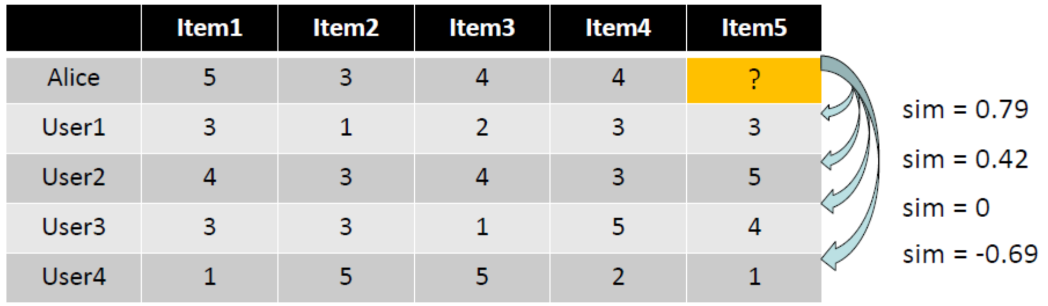
- Use average of their ratings, N is the set of identified users similar to x.

$$r(x,t) = \frac{1}{|N|}\sum_{y \in N} r(y,t)$$

- Or some weighted measures

$$r(x,t) = \frac{1}{\sum_{y \in N} sim(y,x)}\sum_{y \in N} sim(y,x) r(y,t)$$

- Example, x=Alice and N={user1, user2}



	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0.79

sim = 0.42

sim = 0

sim = -0.69

Item-based Collaborative Algorithm

Inputs: Utility matrix U, a user x
Output: User preferences for unrated items

Step 1: For item *t*, find a set M of similar items
(that has been rated by user x)

Step 2: Estimate rating for item *t* based on ratings (of the user) for similar items

$$r(x,t) = \frac{1}{\sum_{s \in M} sim(s,t)} \sum_{y \in M} sim(s,t) r(x,s)$$

Step 3: Use same similarity metrics and prediction functions as in user-based algorithm

Item –based vs User-based

- In theory, they are dual method, no different
- In practice, item-based method usually outperforms user-based method

- (Why?) Items have more significant features; their similarity is more meaningful than user similarity.

Pros and Cons of Collaborative Recommendation

Pros

- Work for any kind of item

  Use U only. No feature selection needed

Cons

- Cold Start

  -- Need enough users in the system to find a match

- Sparsity

  -- Hard to find users that have rated the same items

- Popularity bias

  -- Tend to recommend popular items

7. Latent Factor Based

- Assume the user-item rating relationship is a simple function. Then, the utility matrix U is likely of low rank and can be rewritten as a product of two simple matrices:
  -

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

- If there are enough known entries in U, we can compute the two matrix factors on the right-handed sided using known ratings and then use the two matrices to rate unrated items.

- In reality, We may assume that $U_{m\times n}$ is approximately equal to a product $S_{m\times d}T_{d\times n}$, where d is a small integer.

- In this way, the computation of S and T is formulated into the following optimization problem:

$$\min \sum_{known\ rating\ entry\ (i,j)} |u_{ij} - \sum_{1\le k\le d} s_{ik}t_{kj}|^2$$

(matrix factorization)

or

$$\min \sum_{known\ rating\ entry\ (i,j)} |u_{ij} - \sum_{1\le k\le d} s_{ik}t_{kj}|^2$$

subject to $s_{ik} \ge 0, t_{kj} \ge 0\ \ for\ all\ i, k, j.$

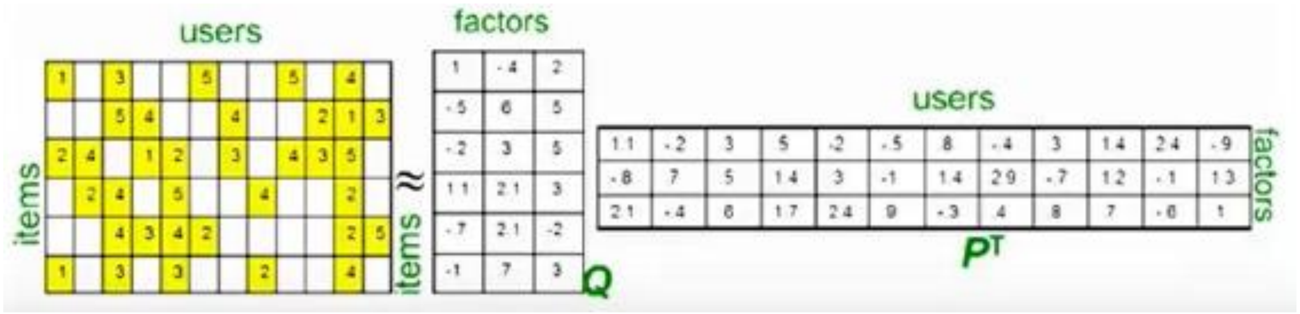
(Non-negative matrix factorization)

Matrix-Factorization Algorithm

Inputs: Utility matrix U

Step 1: Factor U into the product PQ of two low rank matrices P and Q by solving the optimization problem

Step 2: Estimate the rating at position (j, k) as the inner product of the row j of P and the column k of Q.

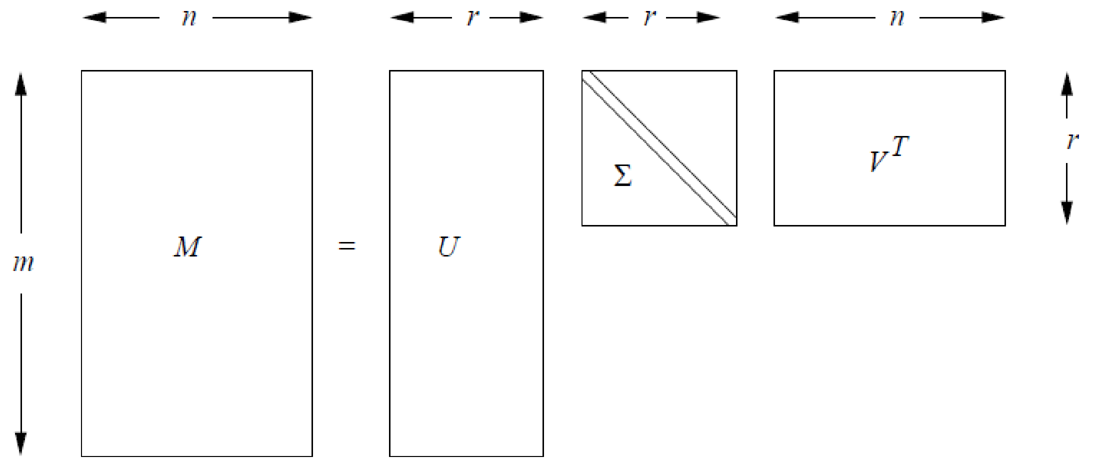


1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

1	-.4	2
-.5	6	5
-.2	3	5
1.1	2.1	3
-.7	2.1	-2
-1	7	3

1.1	-.2	3	5	-2	-.5	8	-.4	3	1.4	2.4	-.9
-.8	7	5	1.4	3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	6	1.7	2.4	9	-.3	4	8	7	-.6	1

The latent factor approach has root in CVD for matrix.



- **Accuracy assessment:**

  (Root-Mean-Square Error)

sqrt($(\frac{1}{N}) \sum_{testing\ rating\ entry\ (i,j)} |u_{ij} - \sum_{1 \le k \le d} s_{ik} t_{kj}|^2$),

where N is the number of the terms in the sum.

Deal with overfitting issue

$$\min \sum_{known\ rating\ entry\ (i,j)} \underbrace{|u_{ij} - \textstyle\sum_{1\le k\le d} s_{ik}t_{kj}|^2}_{\text{Error term}} + \underbrace{\lambda[\textstyle\sum_k s_{ik}^2 + \sum_k t_{kj}^2\ ]}_{\text{Model complexity}}$$

How to solve the optimization problem?

Gradient Descent for matrix factorization formulation
Iterative methods for non-negative matrix formulation

8. Case Study: Drug Response Prediction

Drug response prediction is a recommendation problem

Drug Response Prediction Problem

Input: A drug response matrix $R_{n \times m}$ on *n* cell lines and *m* drugs

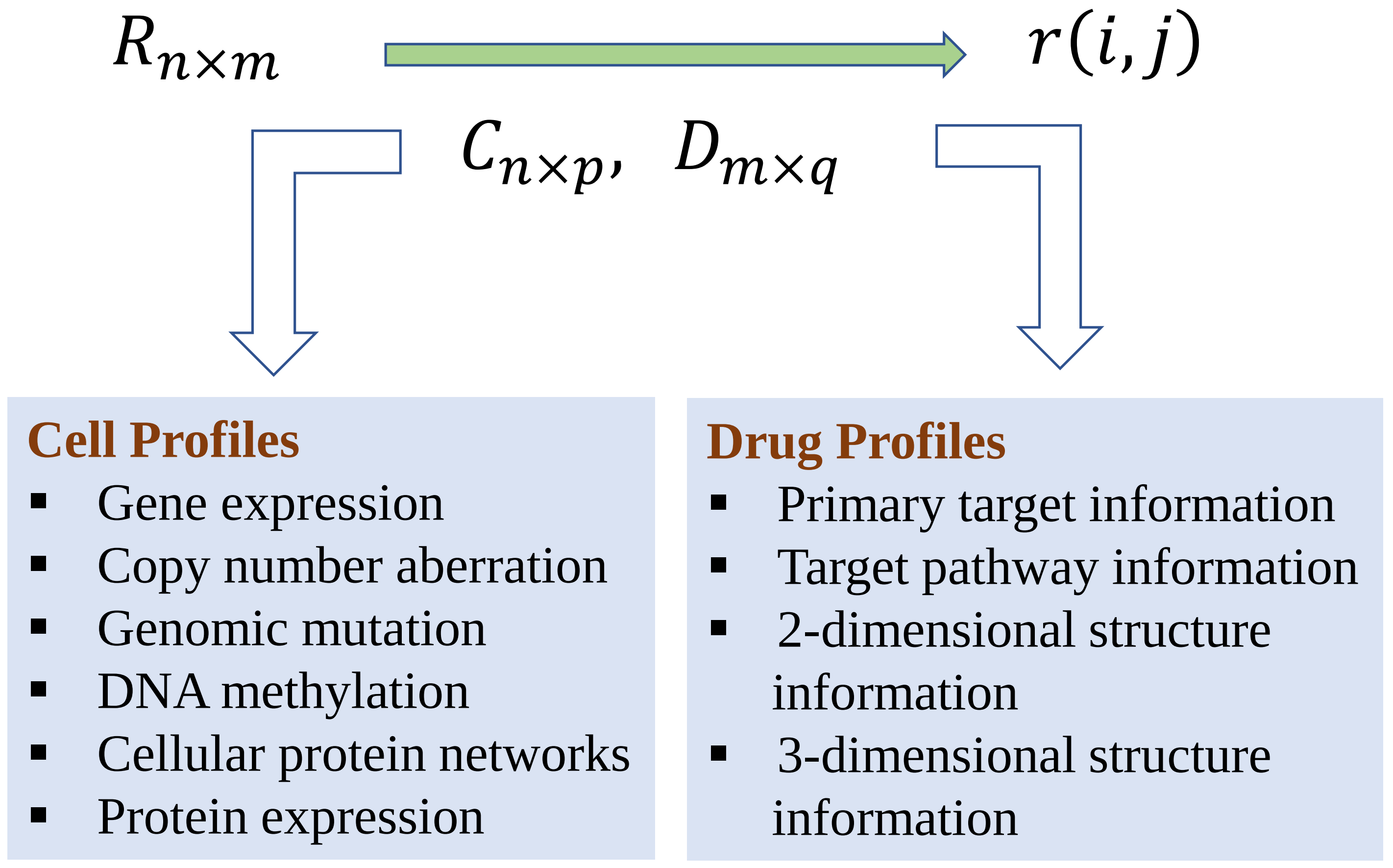
Output: A drug response function *r*(*x*, *y*) of cell line *x* and drug *y*

Goal: *r*(*x*, *y*) can be used to predict the response missing in R.

Hypothesis

- The response to similar drugs are highly related, and
- Similar cell lines also show similar drug responses.

J Chen, L Zhang
Briefings in bioinformatics 22 (1), 232-246; 2021


$R_{n\times m}$
$r(i,j)$
$C_{n\times p}$, $D_{m\times q}$
Cell Profiles
Gene expression
Copy number aberration
Genomic mutation
DNA methylation
Cellular protein networks
Protein expression
Drug Profiles
Primary target information
Target pathway information
2-dimensional structure information
3-dimensional structure information

6.1 17 Prediction Methods Published Since 2014

- **Linear regression models (with or without kernel extension)**
  - -- Elastic net method
  - -- Generalized elastic net
  - -- Response-weighted elastic net
  - -- Kernel ridge regression
  - -- Random forest-enhanced methods
  - -- Multi-view multi-task linear regression
  - -- The MERGE method
  - -- Pairwise multiple kernel learning

- **Matrix-factorization based Bayesian inferences**
  - -- Bayesian multi-task multi-kernel learning
  - -- Component-wise kernel Bayesian matrix factorization
  - -- The MACAU method

- **Matrix factorization via optimization methods**
  - -- Cancer drug response using a recommender system
  - -- Dual-layer integrated cell line-drug network model
  - -- Our method SRMF

- **Other methods**
  - -- Pairwise multiple kernel learning
  - -- CDRScan, a deep learning method
  - -- Kernel rank learning
  - -- The TANDEM method

Method 1: Linear regression methods: ENet and generalizations

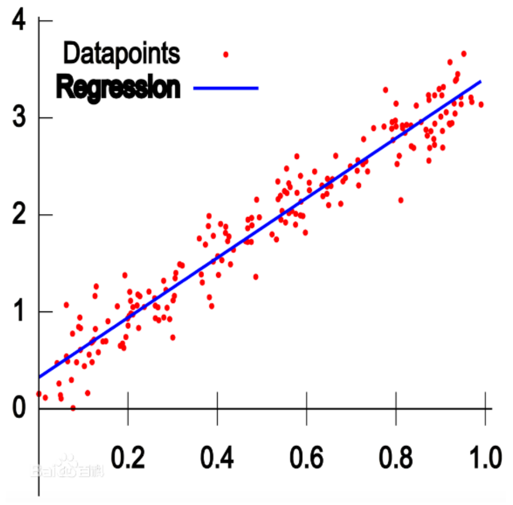
Input $R_{n\times 1}$ (responsese to a drug), $C_{n\times p}$ (cell line profile)

Assumption $R_{n\times 1} \approx C_{n\times p} \cdot [w_1, w_2, \cdots, w_p].$

Algorithm

$$Min_{\boldsymbol{w}} \left\|R_{n\times 1} - C_{n\times p} \cdot \boldsymbol{w}\right\|_2^2 + \lambda_1 \|\boldsymbol{w}\|_1 + \lambda_2 \|\boldsymbol{w}\|_2^2$$ (ENet)

$$Min_{\boldsymbol{w}} \left\|B_{n\times n}(R_{n\times 1} - C_{n\times p} \cdot \boldsymbol{w})\right\|_2^2 + \lambda_1 \|\boldsymbol{w}\|_1 + \lambda_2 \boldsymbol{w} M \boldsymbol{w}^T$$ (Generalization)



Method 2: Bayesian multi-task multi-kernel learning

Assumptions

$$R_{n\times m} = (R_1, R_2, \cdots, R_m)$$

$$R_j \approx \sum_{k=1}^{p} e_k Y_j^{(k)} + b_j [1, 1, \cdots, 1]^T$$

$$\approx \sum_{k=1}^{p} e_k Z_j^{(k)} [a_{1j},\ a_{2j}, \cdots a_{nj}]^T + b_j [1, 1, \cdots, 1]^T$$

Each e_k follows the same normal distribution $\mathrm{N}(0, 1/\mu_e)$.

Each b_j follows the same normal distribution $\mathrm{N}(0, 1/\mu_b)$.

Each a_{ij} follows a normal distribution $\mathrm{N}(0, 1/\mu_{ij})$.

$Z_j^{(k)}$ is a similarity matrix computed from a cell profile submatrix

Algorithm

Apply the Bayes theorem to infer the nm + p+ n parameters

Method 3: A matrix factorization-based method

Assumption $R_{n\times m} \approx U_{n\times k} V^{\mathrm{T}}_{\mathrm{m}\times k}$,

$U_{n\times k}$: a compressed cell line similarity, i.e., $U_{n\times k}U^{\mathrm{T}}_{n\times k} \approx C_{n\times p}\,\mathrm{C}^{\mathrm{T}}_{\mathrm{n}\times p}$

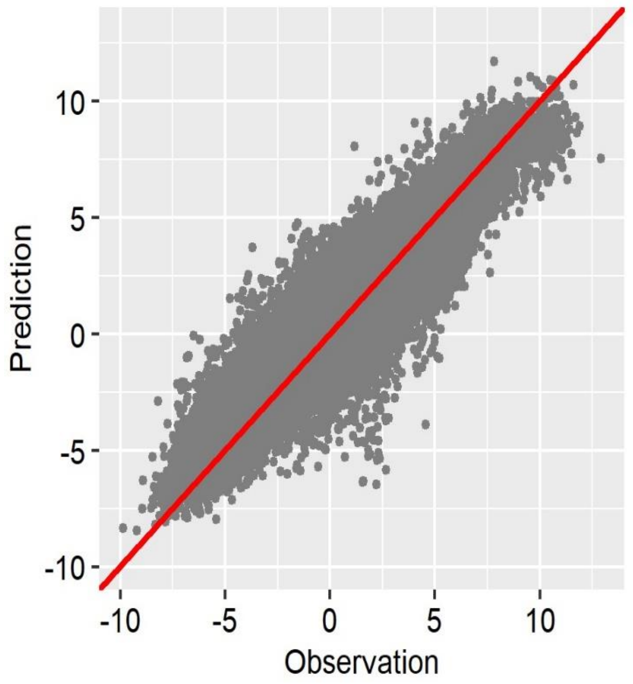
$V_{m\times k}$: a compressed drug similarity, i.e., $V_{m\times k}V^{\mathrm{T}}_{m\times k} \approx D_{m\times q}D^{\mathrm{T}}_{\mathrm{m}\times q}$

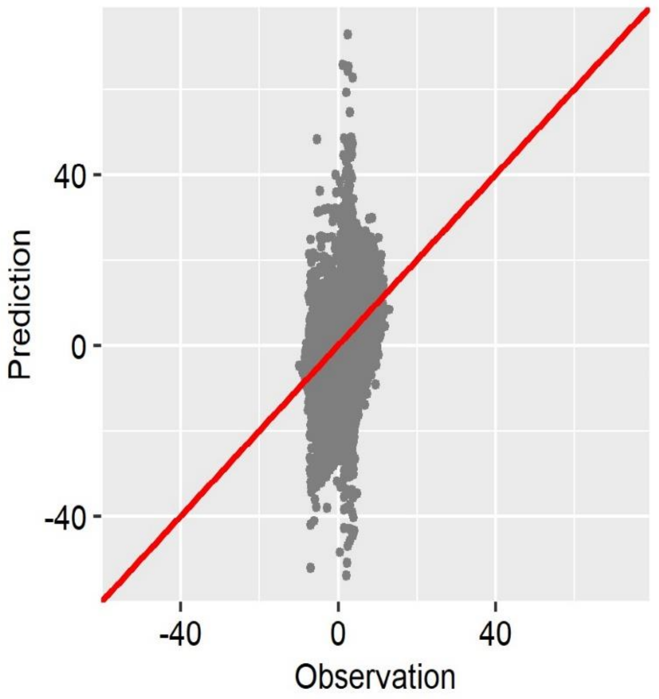
Algorithm

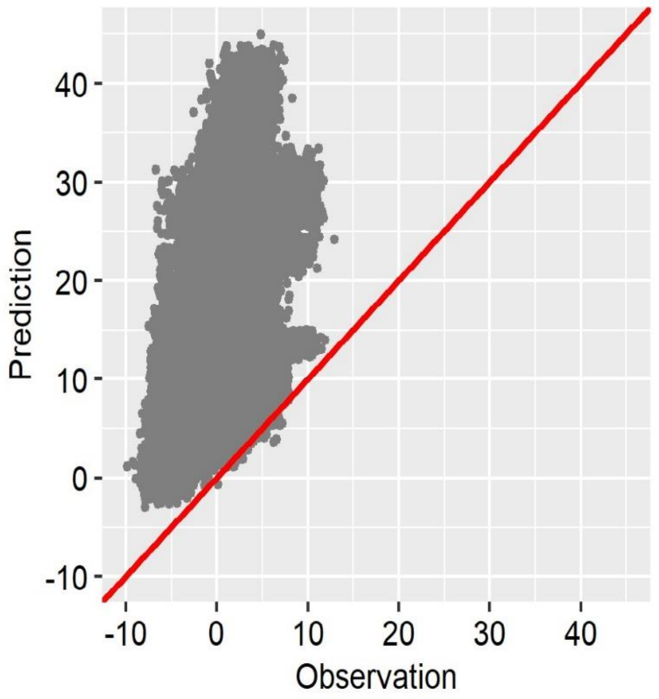
$$Min_{U,V}\ \{\|W \circ (R - UV^T)\|_F^2$$
$$+\lambda_1\left\|C_{n\times p}C^{\mathrm{T}}_{\mathrm{n}\times p} - UU^T\right\|_F^2 + \lambda_2\left\|D_{m\times q}D^{\mathrm{T}}_{\mathrm{m}\times q} - VV^T\right\|_F^2$$
$$+\lambda_3(\|U\|_F^2 + \|V\|_F^2)\}$$

Wang et al., 2018

6.2 Assessment of Methods: Accuracy







1. Four public drug responses--cell lines datasets

- The Genomics of Drug Sensitivity in Cancer (GDSC) :
  -- Drug responses measured in IC50 and AUC for 200 drugs and 734 cell lines
- The Cancer Cell Line Encyclopedia (CCLE),
  -- IC50 and AUC responses for 23 drugs and 285 cell lines
- The US National Cancer Institute NCI 60 Tumor Cells Screen Database (NCI-60)
  -- GI50 drug responses for 215 drugs and 59 cell lines
- The Cancer Therapeutics Response Portal (CTRP)
  -- AUC drug responses for 63 drugs on 720 cell lines

2. Trained and tested on the same dataset

3. Trained on a dataset and tested on another dataset

4. Trained and tested on the dataset involving same drug groups or same cell line types

Eight evaluation metrics

- Measures for performance on full response range

  -- Root-Mean-Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{y}, \widehat{\mathbf{y}}) = \sqrt{\frac{\sum_i (y_i - \widehat{y}_i)^2}{n}}.$$

- Measures for performance for low, high or both ends of responses

  -- Left-RMSE,

  -- Right-RMSE,

  -- Left and right-RMSE

- Other metrics

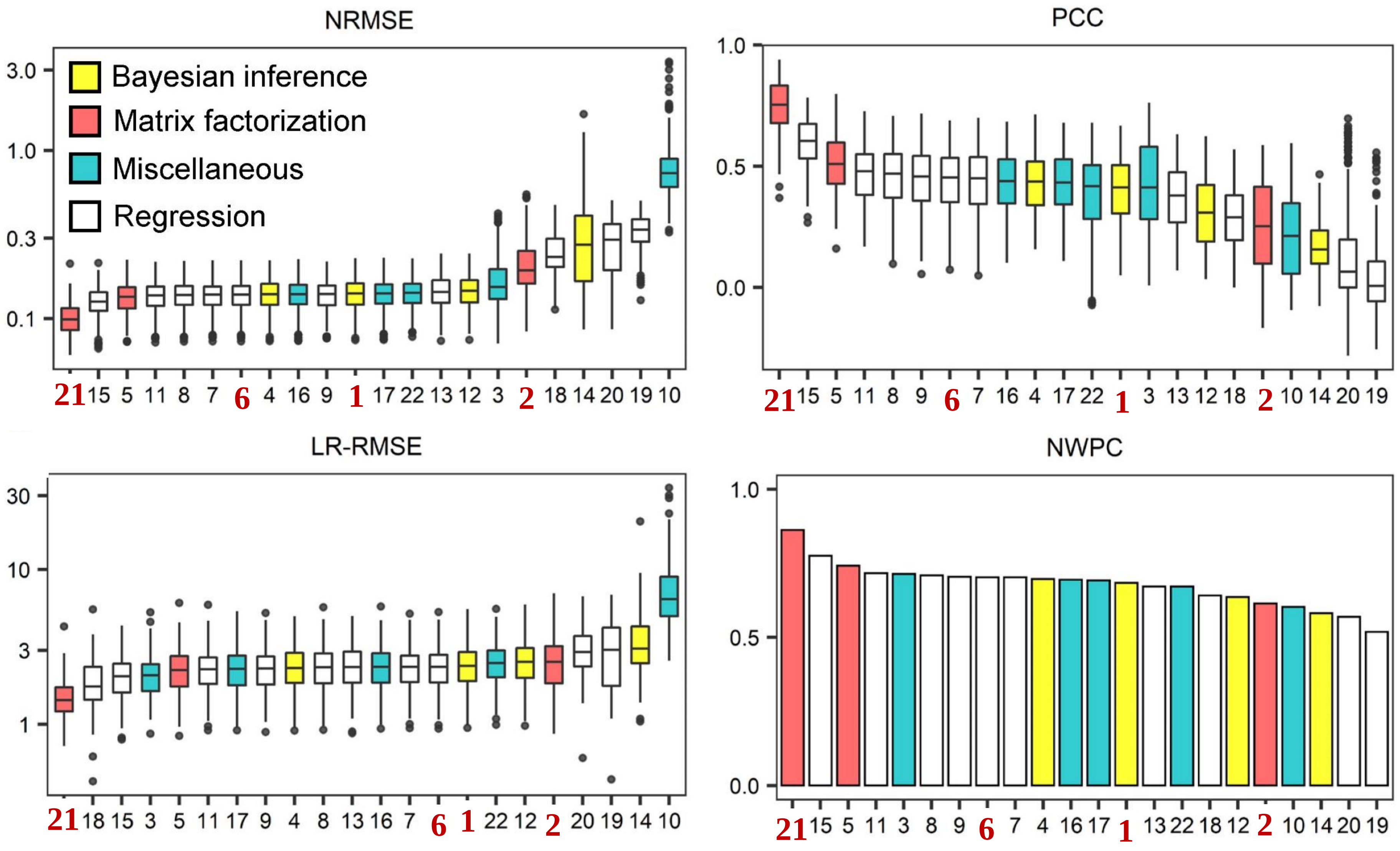
  -- Pearson correlation coefficient: $\mathrm{PCC}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{\sum_i \left(\widehat{y}_i - \mu(\widehat{\mathbf{y}})\right)\left(y_i - \mu(\mathbf{y})\right)}{\sqrt{\sum_i \left(\widehat{y}_i - \mu(\widehat{\mathbf{y}})\right)^2}\sqrt{\sum_i \left(y_i - \mu(\mathbf{y})\right)^2}}.$

  -- Spearman correlation coefficien

  -- Normalized discounted cumulatıve gaın

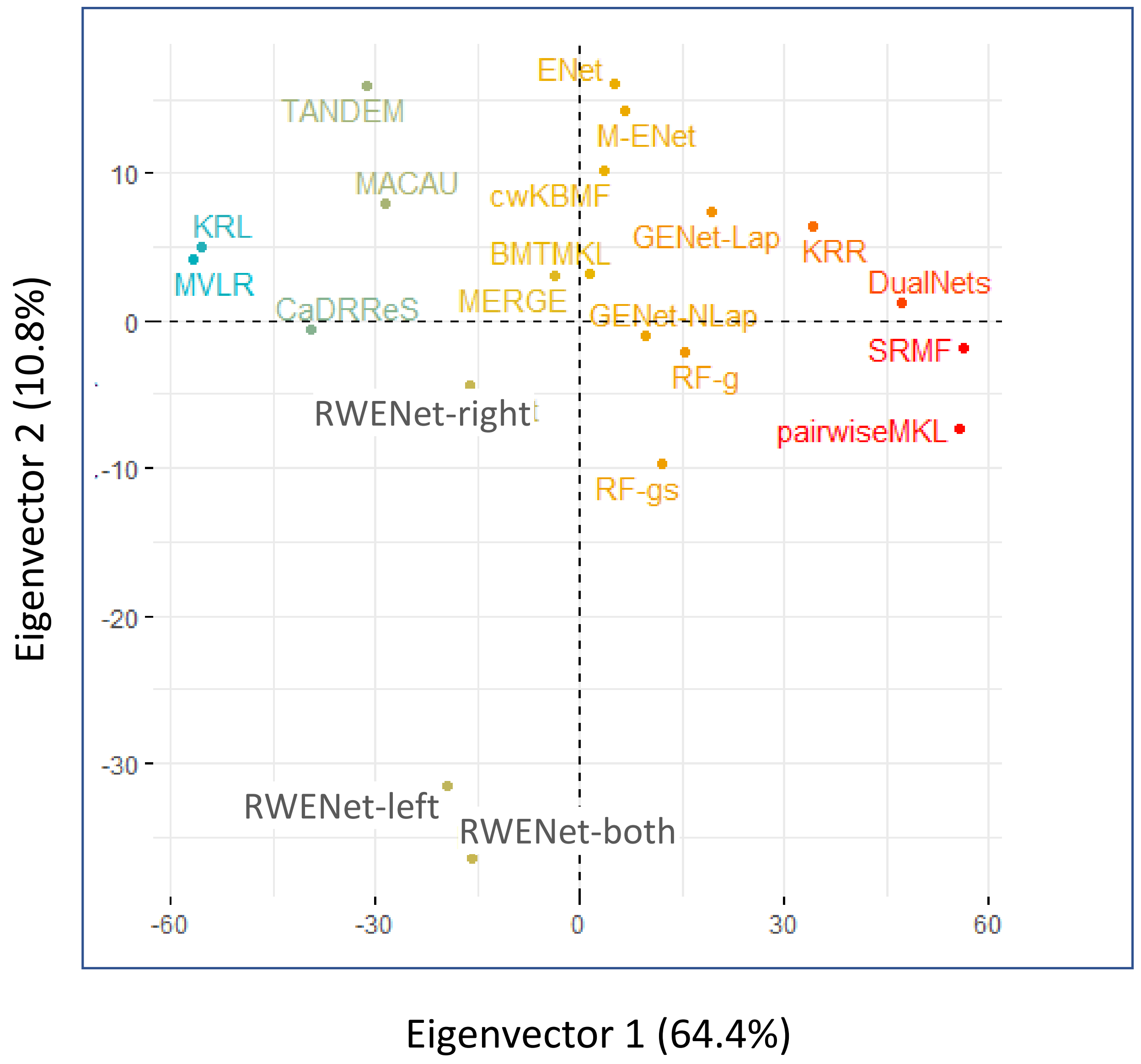
  -- Scaled weighted probabilistic c-index

Rankings of the 17 Methods on the GDSC in Different Metrics



1: BMTMKL
2: CDRScan
6: ENet
21: Our method

Overall PCA Ranking of the 17 Methods



2.4 Observations and future research problems

- The matrix factorization via optimization approach outperformed linear regression models, whereas the linear regression models outperformed different Bayesian inference methods
- Performance of a method varies under different measurements
- Overfitting Issue
- Deep neural network are likely powerful for drug response prediction