Recommender systems

Hung-Hsuan Chen

Many are taken from Robert Bell and S.-D. Lin

Netflix prize

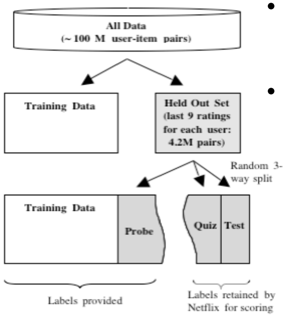
Netflix prize

- On 2006/10/2, Netflix initiate a competition
- Challenge: drop the RMSE by 10%
- Prize:
  - $1M for the first team that completes the challenge
  - $0.5M for best result each year

Data summary

- Training data
  - ~1M ratings
  - 480,000 users
  - 17,770 items
  - Rating scale: [1, 2, 3, 4, 5]
- Test data
  - Last few ratings of each user
  - Further divided into 3 parts
    - Probe, Quiz, and Test

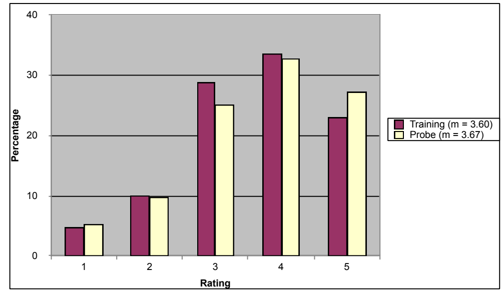
The last 9 ratings split into 3 parts



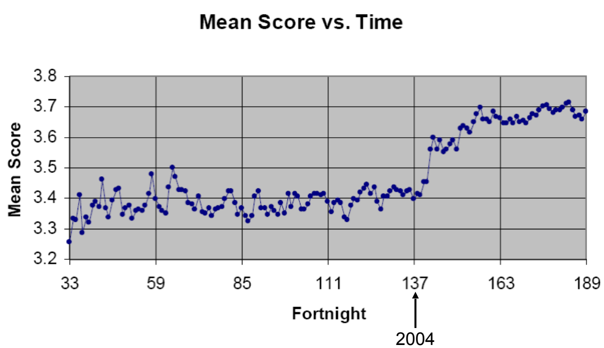
- Probe
  - Part of the last ratings are released
- Quiz and Test
  - Participants don't know how Quiz and Test are split
  - Participants submit the prediction of the combined Quiz and Test
  - Netflix responses the RMSE on the Quiz
  - Test is used to decide the winner

Training vs probe data

- Probe data (later ratings) indeed differ systematically from the training data



Mean score vs. time



Something happened in 2004, although we don't know what it is

Some stats of the movies

es

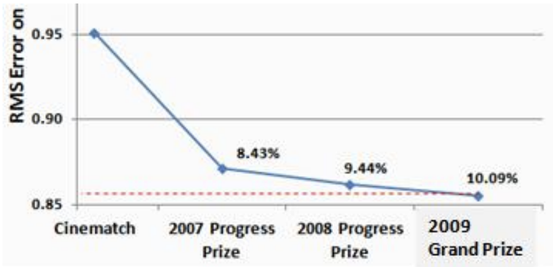
Highest Variance
The Royal Tenenbaums
Lost In Translation
Pearl Harbor
Miss Congeniality
Napolean Dynamite
Fahrenheit 9/11

Most active users

User ID	# Ratings	Mean Rating
305344	17,651	1.90
387418	17,432	1.81
2439493	16,560	1.22
1664010	15,811	4.26
2118461	14,829	4.08
1461435	9,820	1.37
1639792	9,764	1.33
1314869	9,739	2.95

Rate 5000+ movies everyday

Progress over the years



- The winner's approach is a blending of over 800 models
- It is too complex that Netflix had never used it

Lessons learned from Netflix

- Factorization-based approaches
- Identify useful features for rating prediction
  - Implicit feedback
  - Temporal effect
  - Neighborhood effect
- Regularization is important
- We will cover some of the these topics in the following

Recommender system techniques

Types of recommender systems

- Content-based
  - Recommendation based on contents
- Collaborative filtering
  - Recommendation based on users' collective behavior

Content-based

- Users' information
  - E.g., users' profile, interest, gender, etc.
- Items' information
  - E.g., movie title, genre, actors, actresses, director, content description, etc.
- Compare the similarity between user profiles and items
- Compare the similarity between users' unseen items with the items they liked
- Disadvantage: user and item information is not always clean or available

Collaborative filtering (CF)

- A very successful type of method
  - Amazon, Netflix, etc.
- Cross domain
- No content information is required
- Types
  - Memory based
    - User-based CF
    - Item-based CF
  - Model based
    - Matrix factorization (a.k.a., SVD, latent factor model)

Math form of CF

- Given: some ratings
- Predict: unknown ratings
- Netflix prize!

	I1	I2	I3	I4
U1	3	?	1	?
U2	?	4	?	3
U3	1	?	?	?
U4	?	?	5	2

- This may look different from what we've learned in class
  - Target variables are explicit, but where are the features?
  - It turns out that popular techniques to solve the problem are very similar to what we've learned

User-based CF

- How to recommend items to a user *u*?
- Find users that are similar to *u* based on rated items
  - $$\mathrm{Sim}(u, v) = \frac{\sum_{i \in R(u,v)} r_{ui} r_{vi}}{\sqrt{\sum_{i \in R(u,v)} r_{ui}^2} \sqrt{\sum_{i \in R(u,v)} r_{vi}^2}}$$
    - $R(u, v)$: items rated by both *u* and *v*
- Recommend items that are liked by the similar users but haven't been watched by *u*
  - $$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N(u)} \mathrm{Sim}(u, v)(r_{vi} - \bar{r}_v)}{\sum_{v \in N(u)} \mathrm{Sim}(u, v)}$$
- Problem
  - Users may have very few ratings. Thus, similarity between users might be unstable

	M1	M2	M3	M4
Ann	3	0	3	3
Bob	5	4	0	2
Chloe	1	2	4	2
Dave	3	?	1	0
Elli	2	2	0	1

- $\mathrm{Sim}(\mathrm{Dave},\ \mathrm{Ann}) = \frac{3 \cdot 3 + 3 \cdot 1 + 3 \cdot 0}{\sqrt{3^2+3^2+3^2}\sqrt{3^2+1^2+0^2}} = 0.73$
- $\mathrm{Sim}(\mathrm{Dave},\ \mathrm{Bob}) = \frac{5 \cdot 3 + 0 \cdot 1 + 2 \cdot 0}{\sqrt{5^2+0^2+2^2}\sqrt{3^2+1^2+0^2}} = 0.88$
- $\mathrm{Sim}(\mathrm{Dave},\ \mathrm{Chloe}) = \frac{1 \cdot 3 + 4 \cdot 1 + 2 \cdot 0}{\sqrt{1^2+4^2+2^2}\sqrt{3^2+1^2+0^2}} = 0.48$
- $\mathrm{Sim}(\mathrm{Dave},\ \mathrm{Elli}) = \frac{2 \cdot 3 + 0 \cdot 1 + 1 \cdot 0}{\sqrt{2^2+0^2+1^2}\sqrt{3^2+1^2+0^2}} = 0.85$

- $\bar{r}_{\mathrm{Ann}} = \frac{3+0+3+3}{4} = 2.25$
- $\bar{r}_{\mathrm{Bob}} = \frac{5+4+0+2}{4} = 2.75$
- $\bar{r}_{\mathrm{Chloe}} = \frac{1+2+4+2}{4} = 2.25$
- $\bar{r}_{\mathrm{Dave}} = \frac{3+1+0}{3} = 1.33$
- $\bar{r}_{\mathrm{Elli}} = \frac{2+2+0+1}{4} = 1.25$

- $\hat{r}_{\mathrm{Dave,M2}} = 1.33 + \frac{0.88(4-2.75) + 0.85(2-1.25)}{0.88+0.85} = 2.33$
  - Neighborhood size = 2

Item-based CF

- How to recommend items to a user *u*?
- Find items that are similar to item *i* based on known ratings
  - $$\mathrm{Sim}(i, j) = \frac{\sum_{u \in R'(i,j)} r_{ui} r_{uj}}{\sqrt{\sum_{u \in R'(i,j)} r_{ui}^2} \sqrt{\sum_{u \in R'(i,j)} r_{uj}^2}}$$
    - $R'(i, j)$: users who rated both item *i* and item *j*
- Recommend items that are similar to the items liked by *u*
  - $$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in N(i)} \mathrm{Sim}(i, j)\left(r_{uj} - \bar{r}_j\right)}{\sum_{j \in N(i)} \mathrm{Sim}(i, j)}$$
- Why item-based might be better than user-based?
  - Items usually receive more ratings; similarity between items are more stable

	M1	M2	M3	M4
Ann	3	0	3	3
Bob	5	4	0	2
Chloe	1	2	4	2
Dave	3	?	1	0
Elli	2	2	0	1

- $\mathrm{Sim(M2, M1)} = \frac{3 \cdot 0 + 5 \cdot 4 + 1 \cdot 2 + 2 \cdot 2}{\sqrt{3^2+5^2+1^2+2^2}\sqrt{0^2+4^2+2^2+2^2}} = 0.85$
- $\mathrm{Sim(M2, M3)} = \frac{3 \cdot 0 + 0 \cdot 4 + 4 \cdot 2 + 0 \cdot 2}{\sqrt{3^2+0^2+4^2+0^2}\sqrt{0^2+4^2+2^2+2^2}} = 0.33$
- $\mathrm{Sim(M2, M4)} = \frac{3 \cdot 0 + 2 \cdot 4 + 2 \cdot 2 + 1 \cdot 2}{\sqrt{3^2+2^2+2^2+1^2}\sqrt{0^2+4^2+2^2+2^2}} = 0.67$

- $\hat{r}_{\mathrm{Dave,M2}} = 2 + \frac{0.85(3-2.8) + 0.67(0-1.6)}{0.85+0.67} = 1.41$
  - Neighborhood size = 2

- $\bar{r}_{\mathrm{M1}} = \frac{3+5+1+3+2}{5} = 2.8$
- $\bar{r}_{\mathrm{M2}} = \frac{0+4+2+2}{4} = 2$
- $\bar{r}_{\mathrm{M3}} = \frac{3+0+4+1+0}{5} = 1.6$
- $\bar{r}_{\mathrm{M4}} = \frac{3+2+2+0+1}{5} = 1.6$

Quiz

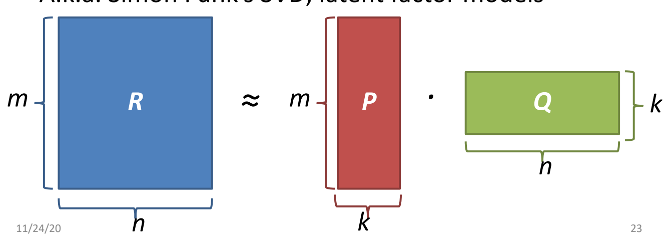
- Do you feel familiar with the User-based CF and the item-based CF?

KNN

Model-based CF

Matrix factorization

- Assume *m* users and *n* items
- $\boldsymbol{R} \approx \boldsymbol{P} \cdot \boldsymbol{Q}$, many r_{ij}'s are unknown
  - $k \ll \min\{m, n\}$, k: number of latent factors
- A.k.a. Simon Funk's SVD; latent factor models



What are latent factors?

- Each latent factor represents certain property of the users and the items
  - However, we don't really know the meaning of each latent factor

	M1	M2	M3	M4
Ann	3	0	3	3
Bob	5	4	0	2
Chloe	1	2	4	2
Dave	3	?	1	0
Elli	2	2	0	1

$\approx$

	f1	f2
Ann	p_{11}	p_{12}
Bob	p_{21}	p_{22}
Chloe	p_{31}	p_{32}
Dave	p_{41}	p_{42}
Elli	p_{51}	p_{52}

$\cdot$

	M1	M2	M3	M4
f1	q_{11}	q_{12}	q_{13}	q_{14}
f2	q_{21}	q_{22}	q_{23}	q_{24}

	M1	M2	M3	M4
Ann	3	0	3	3
Bob	5	4	0	2
Chloe	1	2	4	2
Dave	3	?	1	0
Elli	2	2	0	1

$\approx$

	f1	f2
Ann	p_{11}	p_{12}
Bob	p_{21}	p_{22}
Chloe	p_{31}	p_{32}
Dave	p_{41}	p_{42}
Elli	p_{51}	p_{52}

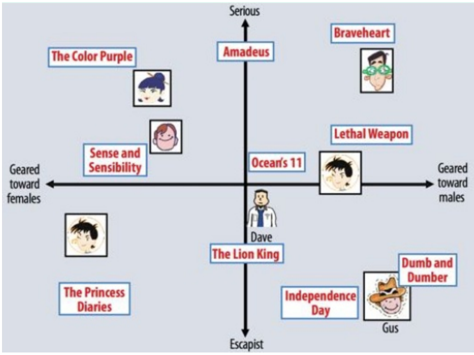
$\cdot$

	M1	M2	M3	M4
f1	q_{11}	q_{12}	q_{13}	q_{14}
f2	q_{21}	q_{22}	q_{23}	q_{24}

- $\hat{r}_{ij} = p_{i1}q_{1j} + p_{i2}q_{2j} = \sum_k p_{ik}q_{kj}$
- $(\boldsymbol{P},\ \boldsymbol{Q}) = \underset{\boldsymbol{P},\boldsymbol{Q}}{\operatorname{argmin}} \sum_{\forall (i,j) \in \tilde{K}} \left(r_{ij} - \hat{r}_{ij}\right)^2$
  - $\tilde{K}$: all **rated** (i, j) pairs (e.g., $r_{\text{Dave,M2}}$ is not included)
  - All the entries in ***P*** and ***Q*** are parameters to learn
  - (Stochastic) gradient descent!
- Prediction: $\hat{r}_{\text{Dave, M2}} = p_{41}q_{12} + p_{42}q_{22}$

Example

Factor 2



Factor 1

In practice, the meaning of each factor is unknown

Summary of MF

- Given the ratings $\boldsymbol{R} \in \mathbb{R}^{m \times n}$, find two matrices $\boldsymbol{P} \in \mathbb{R}^{m \times k}$ and $\boldsymbol{Q} \in \mathbb{R}^{k \times n}$ such that $\boldsymbol{R} \approx \boldsymbol{P} \cdot \boldsymbol{Q}$, where
  - $k \ll \min(m,\ n)$
- If two users share similar latent factors, they give similar ratings to most items
- If two items share similar latent factors, they receive similar ratings from most users
- MF is sometimes called
  - Latent factor model
  - Singular value decomposition (SVD)
    - In fact, the model is different from the SVD in linear algebra (although they share many similarities)

MF – including the regularization terms

- $(\boldsymbol{P},\ \boldsymbol{Q}) = \underset{\boldsymbol{P},\boldsymbol{Q}}{\operatorname{argmin}} \left[\sum_{\forall (i,j) \in \tilde{K}} \left(r_{ij} - \hat{r}_{ij}\right)^2 + \frac{\lambda_{\boldsymbol{P}}}{2}\|\boldsymbol{P}\|^2 + \frac{\lambda_{\boldsymbol{Q}}}{2}\|\boldsymbol{Q}\|^2 \right]$
  - $\hat{r}_{ij} = (\boldsymbol{P} \cdot \boldsymbol{Q})_{ij} = \boldsymbol{p}_i \boldsymbol{q}_j$
  - $\sum_{\forall (i,j) \in \tilde{K}} \left(r_{ij} - \hat{r}_{ij}\right)^2$: training error
  - $\frac{\lambda_{\boldsymbol{P}}}{2}\|\boldsymbol{P}\|^2 + \frac{\lambda_{\boldsymbol{Q}}}{2}\|\boldsymbol{Q}\|^2$: regularization

simple SVD

SVD

full version

This is different from the SVD in linear algebra

- $(\boldsymbol{P}, \boldsymbol{Q}) = \underset{\boldsymbol{P},\boldsymbol{Q}}{\operatorname{argmin}} \left[\sum_{\forall (i,j) \in \tilde{K}} \left(r_{ij} - \hat{r}_{ij} \right)^2 + \frac{\lambda_P}{2} \|\boldsymbol{P}\|^2 + \frac{\lambda_Q}{2} \|\boldsymbol{Q}\|^2 + \frac{\lambda_b}{2} \|\boldsymbol{b}\|^2 + \frac{\lambda_c}{2} \|\boldsymbol{c}\|^2 \right]$
  - $\hat{r}_{ij} = \mu + b_i + c_j + \boldsymbol{p}_i \boldsymbol{q}_j$ → learn
    - μ: mean of all ratings
    - $\boldsymbol{b}$: vector of rating bias for users
      - Some users may consistently rate higher or lower scores
    - $\boldsymbol{c}$: vector of rating bias for items
      - Some items may consistently receive higher or lower ratings

ask 2 ?

learn

random asign b, c, p, q

SVD training procedure

HW

- Loss function
  - $L(\Theta) = \frac{1}{2} \sum_{\forall (i,j) \in \tilde{k}} \left(r_{ij} - \hat{r}_{ij} \right)^2 + \frac{\lambda}{2} \|\Theta\|^2$
    - , where $\hat{r}_{ij} = \mu + b_i + c_j + \boldsymbol{p}_i \cdot \boldsymbol{q}_j$
- Let $d_{ij} = r_{ij} - \hat{r}_{ij}$, the gradients are
  - $\nabla_{b_i} = -d_{ij} + \lambda b_i$
  - $\nabla_{c_j} = -d_{ij} + \lambda c_j$
  - $\nabla_{\boldsymbol{p}_i} = -d_{ij} \boldsymbol{q}_j + \lambda \boldsymbol{p}_i$
  - $\nabla_{\boldsymbol{q}_j} = -d_{ij} \boldsymbol{p}_i + \lambda \boldsymbol{q}_j$
- Update rule of SGD
  - $\theta^{(k+1)} = \theta^{(k)} - \eta \nabla_{\theta^{(k)}}$

ask 1 i r_{ij} ?

Summary (1/2)

- A large branch of studies on recommender systems aims at predicting users' ratings on items based on the known ratings
- Although the problem looks different from most supervised learning problems (no features), it can be solved by some techniques we learned in class
  - *k*NN
  - (Stochastic) gradient descent
- If you can model your task as a optimization problem, there's a good chance that gradient descent might be able to help you

Summary (2/2)

- Adv of MF
  - No need to label to item and user features
  - Support online learning
- Disadv of MF
  - Cold start
  - Difficult to integrate item features and user features, even if they are given

Matrix Factorization vs Factorization Machine

solve

Matrix Factorization (MF) vs Factorization Machine (FM)

- MF: decompose a large rating matrix (user-by-item) into the product of two small matrices
  - A user-by-latent factor matrix
  - A latent-factor-by-item matrix
- FM: $y = \sum_{i=0}^{d} \theta_i x_i + \sum_{(j,k) \in C_2} \langle \boldsymbol{v}_j, \boldsymbol{v}_k \rangle x_j x_k$
- It turns out that MF is a special case of FM
  - When using only user's ratings on items as the clues, FM=MF
  - When user features and item features are given, FM can integrate these features into model

Factorization machines (FM)

- Formula

$$y = \sum_{i=0}^{d} \theta_i x_i + \sum_{(j,k) \in C_2} \langle \boldsymbol{v}_j, \boldsymbol{v}_k \rangle x_j x_k$$

  - y: target
  - $x_1, \ldots, x_d$: features
  - $\theta_0,\ \theta_1 \ldots,\ \theta_d, \boldsymbol{v}_1,\ \boldsymbol{v}_d$: parameters to learn, each $\boldsymbol{v}_j$ is a vector of length ℓ
  - C_2: 2-combination of elements in $[x_1, \ldots, x_d]$

FM vs MF

- Given (i, j, r_{ij}): user i's rating on item j is r_{ij}
  - Target: r_{ij}
  - Features: $(\underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{|U|}, \underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{|I|})$
  - Prediction model:

$$\hat{r}_{ij} = \theta_0 + \theta_i 1 + \theta_j 1 + \langle \boldsymbol{v}_i \boldsymbol{v}_j \rangle 1 \cdot 1$$

  - Ref: Prediction model of MF:

$$\hat{r}_{ij} = w_0 + b_i + c_j + \boldsymbol{p}_i \boldsymbol{q}_j$$

FM can integrate other features

- User features: gender, age, annual income, …
- Item features: category, brand, price, …
- Contextual features: weather, holiday, …
- FM combines MF and these features into one unified model
  - $\hat{r}_{ij} = \theta_0 + \sum_k \theta_k x_k + \langle \boldsymbol{v}_m \boldsymbol{v}_n \rangle x_m x_n$
  - $(\underbrace{0,\ldots,0,1,0,\ldots,0}_{|U|},\underbrace{0,\ldots,0,1,0,\ldots,0}_{|I|},\underbrace{20,1,100,5,\ldots}_{\text{Other features (e.g., age, gender, price, …)}})$

Summary

- We derived FM from the perspective of improving the poly-2 model
- We derived MF from the perspective of decomposing a matrix
- It turns out that MF is a special case of FM

Quiz

- Matrix factorization describes user-item relationship in high-dimensional space (true or false)
- In matrix factorization, what would happen if we set the number of latent factors to be larger than *m* and *n*?