Case Study 4: Collaborative Filtering

Collaborative Filtering
Matrix Completion
Alternating Least Squares

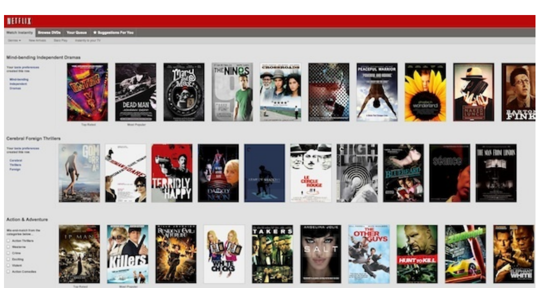
Machine Learning for Big Data
CSE547/STAT548, University of Washington

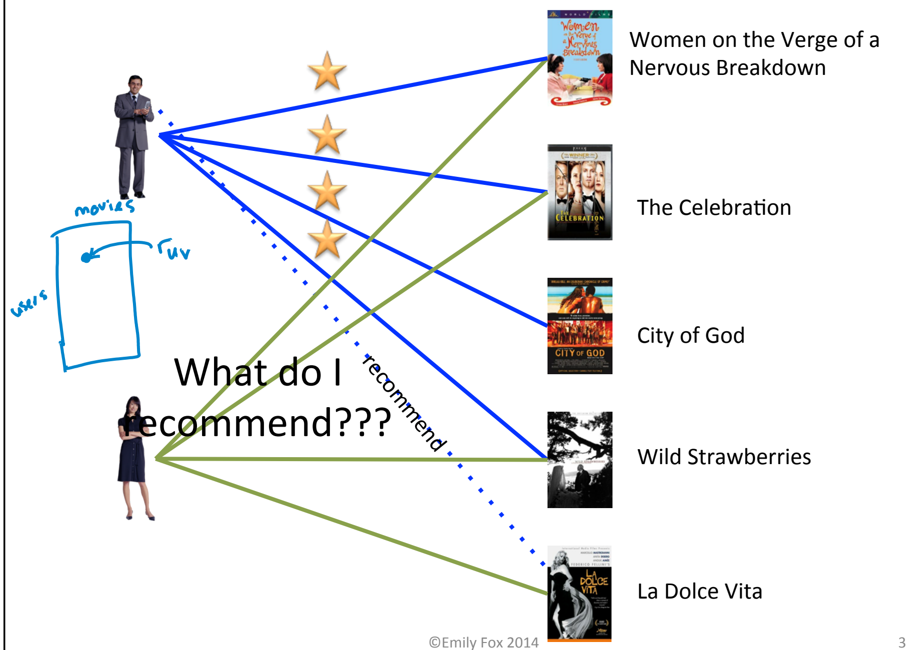
Emily Fox

February 11th, 2014

Collaborative Filtering

- **Goal:** Find movies of interest to a user based on movies watched by the user and others
- **Methods:** matrix factorization, GraphLab





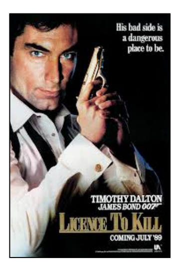
Cold-Start Problem

- **Challenge:** Cold-start problem (new movie or user)
- **Methods:** use features of movie/user

$$\phi(\text{Skyfall}) = \begin{array}{r} \text{action} \\ \text{romance} \\ \\ \\ \end{array} \begin{pmatrix} 7 \\ 2 \\ 0 \\ \vdots \end{pmatrix}$$

IN THEATERS

$$\phi(\text{FRWL}) = \begin{pmatrix} 8 \\ 1 \\ 6 \\ \vdots \end{pmatrix}$$



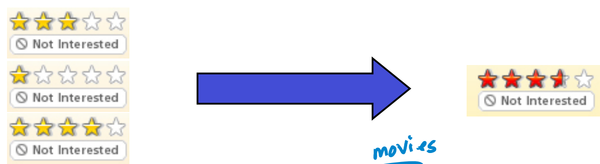
Netflix Prize

- Given 100 million ratings on a scale of 1 to 5, predict 3 million ratings to highest accuracy

skyfall

WVNB

FRWL



Pirates of the Caribbean

movies

users

- 17770 total movies
- 480189 total users
- Over 8 billion total ratings

8B params

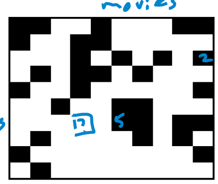
- How to fill in the blanks?

100M obs.

Figures from Ben Recht

Matrix Completion Problem

movies

X = 

users

2

17

5

X_{ij} known for black cells

X_{ij} unknown for white cells

Rows index users

Columns index movies

- Filling missing data?

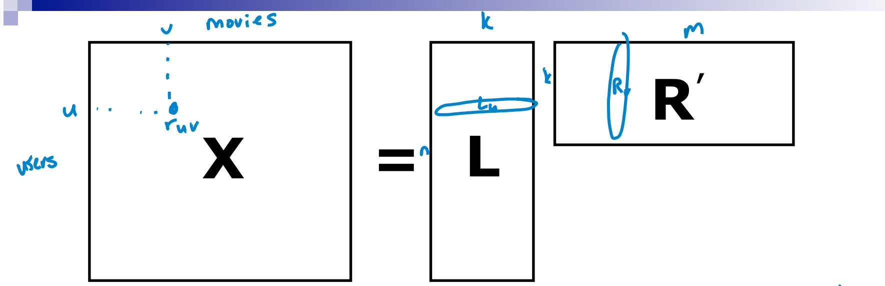
well-approx. by rank k matrix

$$\underset{n}{\overset{m}{X}} \approx \underset{n}{\overset{k}{L}} \; \overset{m}{\underset{k}{R^T}}$$

n·m params

$$\overbrace{k\cdot n + k\cdot m}^{\text{learn this}} \ll \overbrace{n\cdot m}^{\text{predict this}}$$

Interpreting Low-Rank Matrix Completion (aka Matrix Factorization)



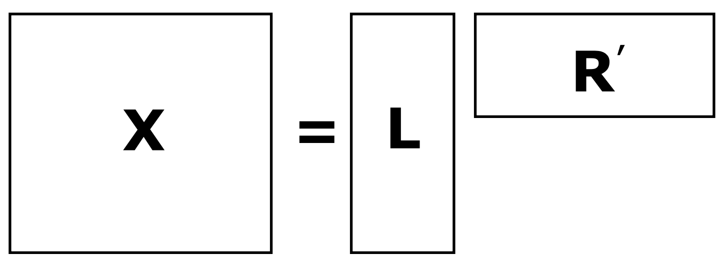
$r_{uv} \approx L_u \cdot R_v$

movie topic i "romance"

L_u $k \times 1$: how "much" user u likes topic i

R_v $k \times 1$: how "much" movie v is about topic i

Identifiability of Factors



- If r_{uv} is described by L_u , R_v what happens if we redefine the "topics" as

  $\tilde{L}_u = L_u \Lambda \qquad \tilde{R}_v = R_v \Lambda$ where $\Lambda\Lambda^T = I$ orthonormal matrix

- Then,

  $\tilde{L}_u \cdot \tilde{R}_v = L_u \Lambda \Lambda^T R_v = L_u \cdot R_v = r_{uv}$

  Invariant to orthonormal transformations

Matrix Completion via Rank Minimization

- Given observed values: $(u, v, r_{uv}) \in X$ some $r_{uv} = ?$ ← unobs. rating
- Find matrix Θ ($n \times m$)
- Such that: $\underbrace{\Theta_{uv} = r_{uv}}_{\text{fit } r_{uv} \neq ? \text{ perfectly}} \quad \forall r_{uv} \neq ?$ ← all obs. ratings
- But… want Θ to be low-rank
- Introduce bias: $\min_{\Theta} \text{rank}(\Theta)$ s.t. $\Theta_{uv} = r_{uv} \quad \forall r_{uv} \neq ?$ ← for $k < \min(n,m)$
- Two issues:
  - NP-hard
  - you can't hope to get exact matching

Approximate Matrix Completion

- Minimize squared error:
  - (Other loss functions are possible)

$$\min_{\Theta} \sum_{(u,v): r_{uv} \neq ?} (\Theta_{uv} - r_{uv})^2$$

- Choose rank *k:*

$$\underset{n \times m}{\Theta} = \underset{n \times k}{L} \cdot \underset{k \times m}{R^T}$$

- Optimization problem:

$$\min_{L,R} \sum_{r_{uv}} (L_u \cdot R_v - r_{uv})^2$$

non-convex opt. problem … local optima only

Coordinate Descent for Matrix Factorization

$$\min_{L,R} \sum_{(u,v):r_{uv}\neq ?} (L_u \cdot R_v - r_{uv})^2$$

- Fix movie factors R, optimize for user factors L
- First observation:

$$\min_{L_1,\dots,L_n} \sum_{(u,v):r_{uv}\neq ?} (L_u \cdot R_v - r_{uv})^2 =$$

V_u = set of movies user u rated

$$\min_{L_1,\dots,L_n} \sum_u \sum_{v\in V_u} (L_u \cdot R_v - r_{uv})^2 =$$

← ind. opt. problem for each user

$$\sum_u \underbrace{\min_{L_u} \sum_{v\in V_u} (L_u \cdot R_v - r_{uv})^2}_{\text{next slide}}$$

← data parallel problem

Minimizing Over User Factors

- For each user u: $\min_{L_u} \sum_{v\in V_u} (L_u \cdot R_v - r_{uv})^2$
- In matrix form:

$$\left\| \underbrace{\begin{bmatrix} \\ R_v \\ \\ \end{bmatrix}}_{|V_u| \times k} \underbrace{\begin{bmatrix} L_u \end{bmatrix}}_{k \times 1} - \underbrace{\begin{bmatrix} r_{uv} \end{bmatrix}}_{|V_u| \times 1} \right\|_2^2$$

Think of as $\|X\beta - y\|_2^2$ normal LS problem

- Second observation: Solve by
  - matrix inversion
  - gradient methods

  ⋮