Collaborative filtering

Recommender systems

Netflix problem: 480000 users, 18000 movies.

- Each user rates some subset of the movies with a score in $\{1, 2, \ldots, 5\}$.

  On average, each user rates around 200 movies, though the variance is high (e.g., some user has rated >17000 movies).

- Goal is to predict how users would rate movies they haven't seen.
- Common to reduce $\{1, 2, \ldots, 5\}$ to $\{-1, +1\}$ (e.g., $\{4, 5\} \mapsto +1$).

Common supervised learning approach:

- Get *features* for each user i and movie j (e.g., $\boldsymbol{x}_i \in \mathbb{R}^d$ and $\tilde{\boldsymbol{x}}_j \in \mathbb{R}^d$); goal is to predict rating as function of $(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_j)$.
- **Linear function**: $\boldsymbol{x}_i \tilde{\boldsymbol{x}}_j^\top \mapsto \boldsymbol{x}_i^\top \boldsymbol{W} \tilde{\boldsymbol{x}}_j$ for $\boldsymbol{W} \in \mathbb{R}^{d \times d}$.
  Can use SVM, logistic regression, etc.

What if you don't have any features?

Collaborative filtering

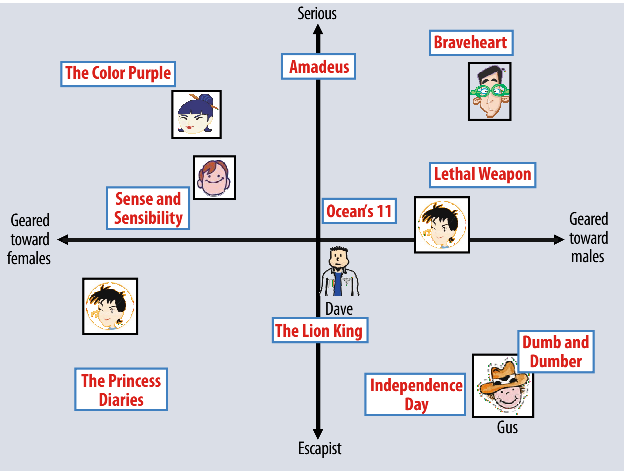
Collaborative filtering (CF):

- Users who rate the same movie similarly are likely to be similar.
- Movies that are rated similarly by the same user are likely to be similar.

Can we formulate a model that expresses this intuition?

- **Side goal for model**: learn *feature representations* of users and movies that are semantically meaningful.

User/movie parameters



(Graphic is from Koren, Bell, and Volinsky.)

- Each user i represented by two-dimensional parameter $\boldsymbol{u}_i = (u_{i,1}, u_{i,2})$.
- Each movie j represented by two-dimensional parameter $\boldsymbol{v}_j = (v_{j,1}, v_{j,2})$.

Simple statistical model for CF

- **Model parameters**: $\boldsymbol{\theta} := (\boldsymbol{u}_1, \dots, \boldsymbol{u}_m, \boldsymbol{v}_1, \dots, \boldsymbol{v}_n)$.
  - Parameter vector for user i: $\boldsymbol{u}_i \in \mathbb{R}^k$.
  - Parameter vector for movie j: $\boldsymbol{v}_j \in \mathbb{R}^k$.

  (Assume $k \leq \min\{m, n\}$.)
- **Distribution of ratings**: the $A_{i,j}$ are independent, and

$$A_{i,j} \sim \mathrm{N}\big(\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle, 1\big) .$$

$$\mathbb{E}\left\{\begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix}\right\} = \underbrace{\begin{bmatrix} \longleftarrow & \boldsymbol{u}_1^\top & \longrightarrow \\ \longleftarrow & \boldsymbol{u}_2^\top & \longrightarrow \\ & \vdots & \\ \longleftarrow & \boldsymbol{u}_m^\top & \longrightarrow \end{bmatrix}}_{m \times k \text{ matrix}} \underbrace{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{k \times n \text{ matrix}} .$$

Prediction and parameter estimation

Data: $\mathcal{A} := \{a_{i,j} \in \mathbb{R} : (i,j) \in \Omega\}$, for subset of user/movie pairs $\Omega \subseteq \{1, 2, \dots, m\} \times \{1, 2, \dots, n\}$.

Prediction

Given parameters $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{u}}_1, \dots, \hat{\boldsymbol{u}}_m, \hat{\boldsymbol{v}}_1, \dots, \hat{\boldsymbol{v}}_n)$, can *predict* $a_{i,j}$ for $(i,j) \notin \Omega$ (i.e., user/movie pairs you don't have ratings for):

$$\hat{a}_{i,j} := \langle \hat{\boldsymbol{u}}_i, \hat{\boldsymbol{v}}_j \rangle .$$

Maximum likelihood estimation

Log-likelihood of $\boldsymbol{\theta} = (\boldsymbol{u}_1, \dots, \boldsymbol{u}_m, \boldsymbol{v}_1, \dots, \boldsymbol{v}_n)$ given $\mathcal{A} := \{a_{i,j} : (i,j) \in \Omega\}$:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{A}) = -\frac{1}{2} \sum_{(i,j) \in \Omega} \big(a_{i,j} - \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle\big)^2 + (\text{terms not involving } \boldsymbol{\theta}) .$$

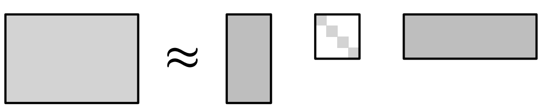
Unfortunately, this is generally hard to maximize.

Special case: complete set of ratings

Special case: have ratings for <u>all</u> user/movie pairs (i.e., $\Omega = \{1, \dots, m\} \times \{1, \dots, n\}$).

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{A}) = -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \big(a_{i,j} - \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle\big)^2 + (\text{terms not involving } \boldsymbol{\theta}) .$$

MLE is given by rank-k singular value decomposition (SVD):



$\boldsymbol{A}$ $(m \times n)$ $\approx$ $\boldsymbol{L}$ $(m \times k)$ $\boldsymbol{S}$ $(k \times k)$ $\boldsymbol{R}^\top$ $(k \times n)$

Use rows of $\boldsymbol{L}\boldsymbol{S}^{1/2}$ as the $\boldsymbol{u}_i$, and rows of $\boldsymbol{R}\boldsymbol{S}^{1/2}$ as the $\boldsymbol{v}_j$.

(Solution is not always unique!)

Matrix completion

General case: MLE is equivalent to (low rank) matrix completion problem:

$$\begin{aligned} \min_{\boldsymbol{X} \in \mathbb{R}^{m \times n}} \quad & \sum_{(i,j) \in \Omega} (a_{i,j} - X_{i,j})^2 \\ \text{s.t.} \quad & \operatorname{rank}(\boldsymbol{X}) \leq k . \end{aligned}$$

Objective is not convex due to rank constraint.

Known user parameters

$$\mathbb{E}\left\{\begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix}\right\} = \begin{bmatrix} \longleftarrow & \boldsymbol{u}_1^\top & \longrightarrow \\ \longleftarrow & \boldsymbol{u}_2^\top & \longrightarrow \\ & \vdots & \\ \longleftarrow & \boldsymbol{u}_m^\top & \longrightarrow \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}$$

Hypothetical situation: suppose all user parameters are already known.

MLE for movie parameters $\boldsymbol{v}_j$ given by ordinary least squares:

$$\boldsymbol{v}_j := \operatorname*{arg\,min}_{\boldsymbol{v}\in\mathbb{R}^k} \sum_{(i,j)\in\Omega} \left(\langle \boldsymbol{u}_i, \boldsymbol{v}\rangle - a_{i,j}\right)^2 .$$

Analogous if, instead, we suppose all movie parameters are already known: get MLE for user parameters $\boldsymbol{u}_j$ via ordinary least squares.

Idea: alternate between the two . . .

Alternating (regularized) least squares

- Initialize $\hat{\boldsymbol{u}}_i \in \mathbb{R}^k$ for each user i and $\hat{\boldsymbol{v}}_j \in \mathbb{R}^k$ for each movie j.
- For $t = 1, 2, \dots$:
  - For each user $i = 1, 2, \dots, m$,
    $$\hat{\boldsymbol{u}}_i := \operatorname*{arg\,min}_{\boldsymbol{u}\in\mathbb{R}^k} \sum_{(i,j)\in\Omega} \left(\langle \boldsymbol{u}, \hat{\boldsymbol{v}}_j\rangle - a_{i,j}\right)^2 + \lambda\|\boldsymbol{u}\|_2^2 .$$
  - For each movie $j = 1, 2, \dots, n$,
    $$\hat{\boldsymbol{v}}_j := \operatorname*{arg\,min}_{\boldsymbol{v}\in\mathbb{R}^k} \sum_{(i,j)\in\Omega} \left(\langle \hat{\boldsymbol{u}}_i, \boldsymbol{v}\rangle - a_{i,j}\right)^2 + \lambda\|\boldsymbol{v}\|_2^2 .$$

(Could also switch or randomize order of updates.)

Alternating stochastic gradient method

When $|\Omega|$ is very large, each iteration can be expensive.

Alternating stochastic gradient method (for $\lambda = 0$)

- Initialize $\hat{\boldsymbol{u}}_i \in \mathbb{R}^k$ for each user i and $\hat{\boldsymbol{v}}_j \in \mathbb{R}^k$ for each movie j.
- For $t = 1, 2, \dots$:
  - Pick $(i_t, j_t) \in \Omega$ uniformly at random.
  - Update:
    $$\begin{aligned} \hat{\boldsymbol{u}}_{i_t} &:= \hat{\boldsymbol{u}}_{i_t} - 2\eta\left(\langle \hat{\boldsymbol{u}}_{i_t}, \hat{\boldsymbol{v}}_{j_t}\rangle - a_{i_t,j_t}\right)\hat{\boldsymbol{v}}_{j_t} , \\ \hat{\boldsymbol{v}}_{j_t} &:= \hat{\boldsymbol{v}}_{j_t} - 2\eta\left(\langle \hat{\boldsymbol{u}}_{i_t}, \hat{\boldsymbol{v}}_{j_t}\rangle - a_{i_t,j_t}\right)\hat{\boldsymbol{u}}_{i_t} . \end{aligned}$$

(Could also switch or randomize order of updates.)

Example

Ran alternating (regularized) least squares on 800000 ratings of movies from $m = 6040$ users and $n = 3952$ movies.

Movie parameters $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n \in \mathbb{R}^k$ give *feature representations* of movies.
Are they semantically meaningful?

Some nearest-neighbor pairs $(\boldsymbol{v}_i, \mathrm{NN}(\boldsymbol{v}_i))$:

- Toy Story (1995), Toy Story 2 (1999)
- Sense and Sensibility (1995), Emma (1996)
- Heat (1995), Carlito's Way (1993)
- The Crow (1994), Blade (1998)
- Forrest Gump (1994), Dances with Wolves (1990)
- Mrs. Doubtfire (1993), The Bodyguard (1992) ???
- . . .

Final remarks

- **Initialization**: can't initialize with all $\boldsymbol{u}_i = \boldsymbol{v}_j = \boldsymbol{0}$.
  (Try random instead.)
- How to pick k or λ? Cross validation.
- **Model with global/per-user/per-movie biases**:
  Parameters: $\boldsymbol{\theta} = (\boldsymbol{u}_1, \dots, \boldsymbol{u}_m, \boldsymbol{v}_1, \dots, \boldsymbol{v}_n, b_1, \dots, b_m, c_1, \dots, c_n, \mu)$

$$A_{i,j} \sim \mathrm{N}\big(\mu + b_i + c_j + \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle, 1\big) .$$

- **Combination with user/movie features** $\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_j \in \mathbb{R}^d$:
  Parameters: $\boldsymbol{\theta} = (\boldsymbol{u}_1, \dots, \boldsymbol{u}_m, \boldsymbol{v}_1, \dots, \boldsymbol{v}_n, b_1, \dots, b_m, c_1, \dots, c_n, \mu, \boldsymbol{W})$

$$A_{i,j} \mid \boldsymbol{X} = \boldsymbol{x}_i \wedge \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}_j \sim \mathrm{N}\Big(\mu + b_i + c_j + \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle + \boldsymbol{x}_i^\top \boldsymbol{W} \tilde{\boldsymbol{x}}_j, 1\Big) .$$

- Many other variations!

Key takeaways

1. Simple statistical model for CF; two methods to (attempt to) compute the MLE.
2. Some possibile generalizations of the CF model.