Data Mining and Matrices

04 – Matrix Completion

Rainer Gemulla, Pauli Miettinen

May 02, 2013

Recommender systems

- Problem
  - Set of users
  - Set of items (movies, books, jokes, products, stories, ...)
  - Feedback (ratings, purchase, click-through, tags, ...)
  - Sometimes: metadata (user profiles, item properties, ...)
- Goal: Predict preferences of users for items
- Ultimate goal: Create item recommendations for each user
- Example

	Avatar	*The Matrix*	*Up*
Alice	?	4	2
Bob	3	2	?
Charlie	5	?	3

Outline

Collaborative filtering

- Key idea: Make use of past user behavior
- No domain knowledge required
- No expensive data collection needed
- Allows discovery of complex and unexpected patterns
- Widely adopted: Amazon, TiVo, Netflix, Microsoft
- Key techniques: neighborhood models, latent factor models

	Avatar	*The Matrix*	*Up*
Alice	?	4	2
Bob	3	2	?
Charlie	5	?	3

Leverage past behavior of other users and/or on other items.

A simple baseline

- m users, n items, $m \times n$ rating matrix $\mathbf{D}$
- Revealed entries $\Omega = \{\, (i,j) \mid \text{rating } \mathbf{D}_{ij} \text{ is revealed} \,\}$, $N = |\Omega|$
- **Baseline predictor**: $b_{ui} = \mu + b_i + b_j$
  - $\mu = \frac{1}{N} \sum_{(i,j)\in\Omega} \mathbf{D}_{ij}$ is the overall average rating
  - b_i is a *user bias* (user's tendency to rate low/high)
  - b_j is an *item bias* (item's tendency to be rated low/high)
- Least squares estimates: $\operatorname{argmin}_{b_*} \sum_{(i,j)\in\Omega} (\mathbf{D}_{ij} - \mu - b_i - b_j)^2$

D	*Avatar* (1.01)	*Matrix* (0.34)	*Up* (−1.32)
Alice (0.32)	? (4.5)	**4** (3.8)	**2** (2.1)
Bob (−1.34)	**3** (2.8)	**2** (2.2)	? (0.5)
Charlie (0.99)	**5** (5.2)	? (4.5)	**3** (2.8)

$$m = 3$$
$$n = 3$$
$$\Omega = \{\, (1,2), (1,3), (2,1), \ldots \,\}$$
$$N = 6$$
$$\mu = 3.17$$
$$b_{32} = 3.17 + 0.99 + 0.34 = 4.5$$

Baseline does not account for personal tastes.

When does a user like an item?

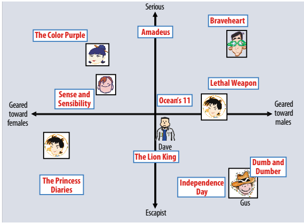
- **Neighborhood models** (kNN): When he likes similar items
  - Find the top-k most similar items the user has rated
  - Combine the ratings of these items (e.g., average)
  - Requires a similarity measure (e.g., Pearson correlation coefficient)

is similar to

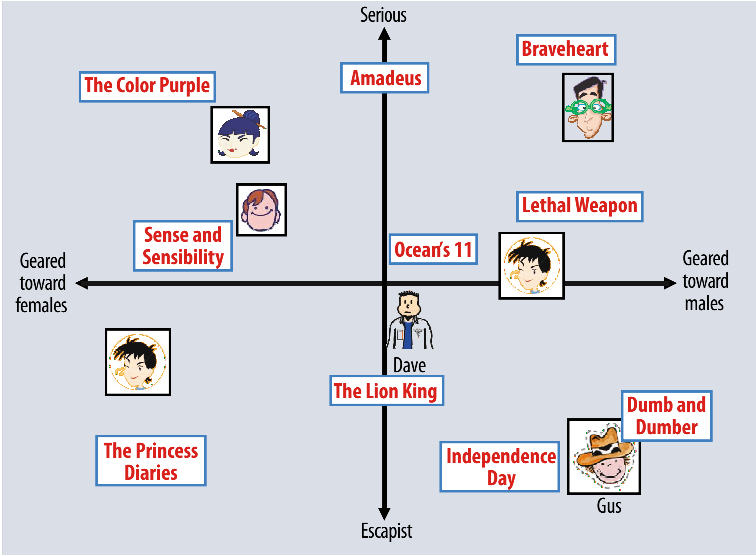
Unrated by Bob
$\rightarrow$ predict 4

Bob rated 4

- **Latent factor models** (LFM): When similar users like similar items
  - More holistic approach
  - Users and items are placed in the same "latent factor space"
  - Position of a user and an item related to preference (via dot products)



Intuition behind latent factor models (1)



Koren et al., 2009.

Intuition behind latent factor models (2)

- Does user $\mathbf{u}$ like item $\mathbf{v}$?
- Quality: measured via **direction** from origin ($\cos\angle(\mathbf{u},\mathbf{v})$)
  - Same direction $\rightarrow$ attraction: $\cos\angle(\mathbf{u},\mathbf{v}) \approx 1$
  - Opposite direction $\rightarrow$ repulsion: $\cos\angle(\mathbf{u},\mathbf{v}) \approx -1$
  - Orthogonal direction $\rightarrow$ oblivious: $\cos\angle(\mathbf{u},\mathbf{v}) \approx 0$
- Strength: measured via **distance** from origin ($\|\mathbf{u}\|\|\mathbf{v}\|$)
  - Far from origin $\rightarrow$ strong relationship: $\|\mathbf{u}\|\|\mathbf{v}\|$ large
  - Close to origin $\rightarrow$ weak relationship: $\|\mathbf{u}\|\|\mathbf{v}\|$ small
- Overall preference: measured via **dot product** ($\mathbf{u} \cdot \mathbf{v}$)

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\| \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \|\mathbf{u}\|\|\mathbf{v}\| \cos\angle(\mathbf{u},\mathbf{v})$$

  - Same direction, far out $\rightarrow$ strong attraction: $\mathbf{u} \cdot \mathbf{v}$ large positive
  - Opposite direction, far out $\rightarrow$ strong repulsion: $\mathbf{u} \cdot \mathbf{v}$ large negative
  - Orthogonal direction, any distance $\rightarrow$ oblivious: : $\mathbf{u} \cdot \mathbf{v} \approx 0$

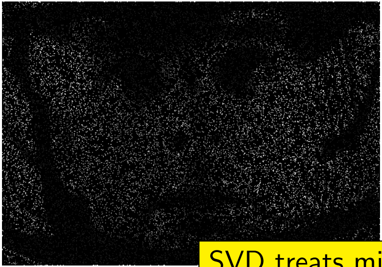
But how to select dimensions and where to place items and users?
Key idea: Pick dimensions that explain the known data well.

SVD and missing values

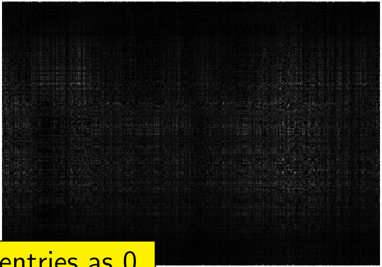
Input data

Rank-10 truncated SVD

10% of input data



Rank-10 truncated SVD



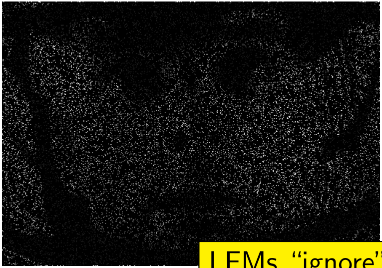
SVD treats missing entries as 0.

Latent factor models and missing values

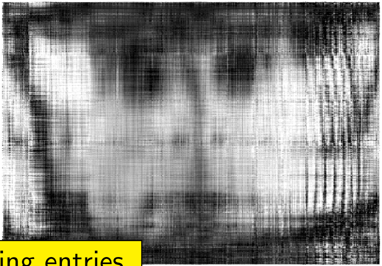
Input data

Rank-10 LFM

10% of input data



Rank-10 LFM



LFMs “ignore” missing entries.

Latent factor models (simple form)

- Given rank r, find $m \times r$ matrix $\mathbf{L}$ and $r \times n$ matrix $\mathbf{R}$ such that

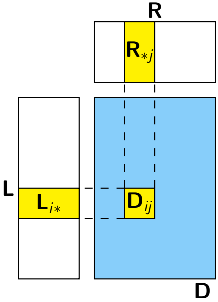
$$\mathbf{D}_{ij} \approx [\mathbf{LR}]_{ij} \qquad \text{for } (i,j) \in \Omega$$

- Least squares formulation

$$\min_{\mathbf{L},\mathbf{R}} \sum_{(i,j)\in\Omega} (\mathbf{D}_{ij} - [\mathbf{LR}]_{ij})^2$$

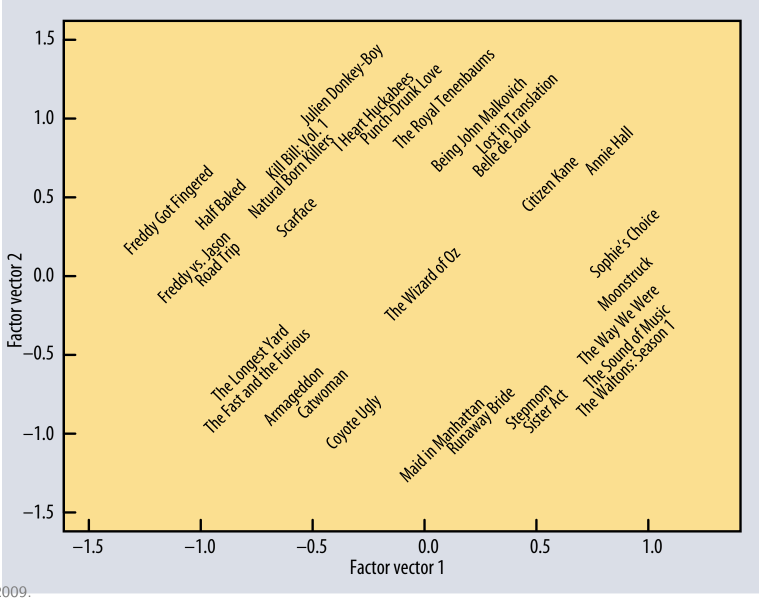
- Example ($r = 1$)

L \ **R**	*Avatar* (2.24)	*The Matrix* (1.92)	*Up* (1.18)
Alice (1.98)	? (4.4)	**4** (3.8)	**2** (2.3)
Bob (1.21)	**3** (2.7)	**2** (2.3)	? (1.4)
Charlie (2.30)	**5** (5.2)	? (4.4)	**3** (2.7)



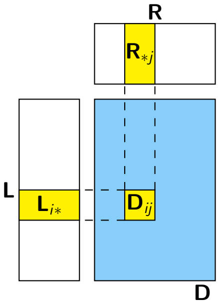
Example: Netflix prize data

($\approx$ 500k users, $\approx$ 17k movies, $\approx$ 100M ratings)



Koren et al., 2009.

Latent factor models (summation form)



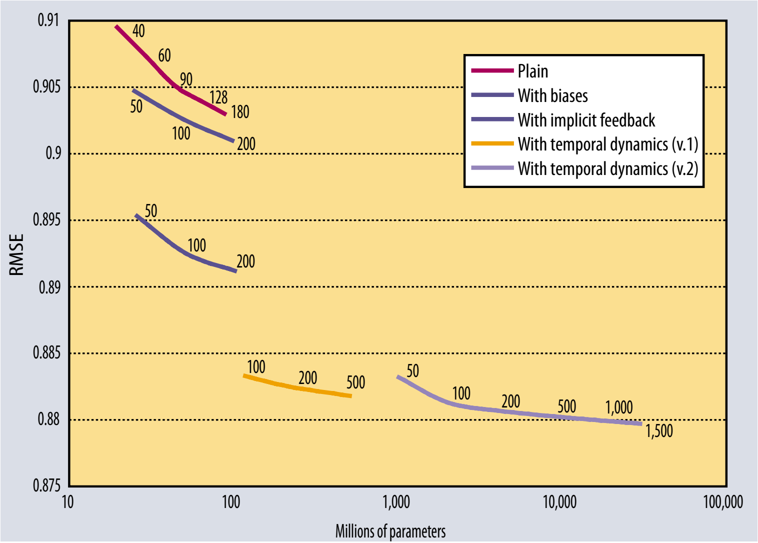
- Least squares formulation prone to overfitting
- More general **summation form**:

$$L = \sum_{(i,j)\in\Omega} l_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j}) + R(\mathbf{L}, \mathbf{R}),$$

  - L is **global loss**
  - $\mathbf{L}_{i*}$ and $\mathbf{R}_{*j}$ are user and item **parameters**, resp.
  - l_{ij} is **local loss**, e.g., $l_{ij} = (\mathbf{D}_{ij} - [\mathbf{LR}]_{ij})^2$
  - R is **regularization term**, e.g., $R = \lambda(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2)$
- Loss function can be more sophisticated
  - Improved predictors (e.g., include user and item bias)
  - Additional feedback data (e.g., time, implicit feedback)
  - Regularization terms (e.g., weighted depending on amount of feedback)
  - Available metadata (e.g., demographics, genre of a movie)

Example: Netflix prize data

Root mean square error of predictions



Koren et al., 2009.

Outline

The matrix completion problem

Complete these matrices!

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & ? & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & ? & ? & ? & ? \\ 1 & ? & ? & ? & ? \\ 1 & ? & ? & ? & ? \\ 1 & ? & ? & ? & ? \end{pmatrix}$$

Matrix completion is impossible without additional assumptions!

Let's assume that underlying full matrix is "simple" (here: rank 1).

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

When/how can we recover a low-rank matrix from a sample of its entries?

Rank minimization

Definition (rank minimization problem)

Given an $n \times n$ data matrix $\mathbf{D}$ and an index set Ω of revealed entries. The *rank minimization problem* is

$$\begin{array}{lll} \text{minimize} & \text{rank}(\mathbf{X}) & \\ \text{subject to} & \mathbf{D}_{ij} = \mathbf{X}_{ij} & (i,j) \in \Omega \\ & \mathbf{X} \in \mathbb{R}^{n \times n}. & \end{array}$$

- Seeks for "simplest explanation" fitting the data
- If unique and sufficient samples, recovers $\mathbf{D}$ (i.e., $\mathbf{X} = \mathbf{D}$)
- NP-hard

Time complexity of existing rank minimization algorithms double exponential in n (and also slow in practice).

Nuclear norm minimization

- Rank: $\text{rank}(\mathbf{D}) = |\{\sigma_k(\mathbf{D}) > 0 : 1 \leq k \leq n\}| = \sum_{k=1}^{n} I_{\sigma_k(\mathbf{D})>0}$
- **Nuclear norm**: $\|\mathbf{D}\|_* = \sum_{k=1}^{n} \sigma_k(\mathbf{D})$

Definition (nuclear norm minimization)

Given an $n \times n$ data matrix $\mathbf{D}$ and an index set Ω of revealed entries. The *nuclear minimization problem* is

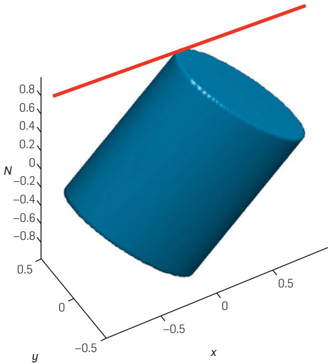
$$\begin{array}{lll} \text{minimize} & \|\mathbf{X}\|_* & \\ \text{subject to} & \mathbf{D}_{ij} = \mathbf{X}_{ij} & (i,j) \in \Omega \\ & \mathbf{X} \in \mathbb{R}^{n \times n}. & \end{array}$$

- A heuristic for rank minimization
- Nuclear norm is convex function (thus local optimum is global opt.)

Can be optimized (more) efficiently via semidefinite programming.

Why nuclear norm minimization?

Figure 1. Unit ball of the nuclear norm for symmetric 2 × 2 matrices. The red line depicts a random one-dimensional affine space. Such a subspace will generically intersect a sufficiently large nuclear norm ball at a rank one matrix.



$$\begin{pmatrix} x & y \\ y & z \end{pmatrix}$$

- Consider SVD of $\mathbf{D} = \mathbf{U}\Sigma\mathbf{V}^T$
- Unit nuclear norm ball $=$ convex combination (σ_k) of rank-1 matrices of unit Frobenius $(\mathbf{U}_{*k}\mathbf{V}_{*k}^T)$
- Extreme points have low rank (in figure: rank-1 matrices of unit Frobenius norm)
- Nuclear norm minimization: inflate unit ball as little as possible to reach $\mathbf{D}_{ij} = \mathbf{X}_{ij}$
- Solution lies at extreme point of inflated ball $\rightarrow$ (hopefully) low rank

Relationship to LFMs

- Recall regularized LFM ($\mathbf{L}$ is $m \times r$, $\mathbf{R}$ is $r \times n$):

$$\min_{\mathbf{L},\mathbf{R}} \sum_{(i,j)\in\Omega} (\mathbf{D}_{ij} - [\mathbf{LR}]_{ij})^2 + \lambda \left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right)$$

- View as matrix completion problem by enforcing $\mathbf{D}_{ij} = [\mathbf{LR}]_{ij}$:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right) \\ \text{subject to} & \mathbf{D}_{ij} = \mathbf{X}_{ij} \qquad (i,j) \in \Omega \\ & \mathbf{LR} = \mathbf{X}. \end{array}$$

- One can show: for r chosen larger than rank of nuclear norm optimum, equivalent to nuclear norm minimization
- For some intuition, suppose $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ at optimum $\mathbf{L}$ and $\mathbf{R}$:

$$\begin{aligned} \tfrac{1}{2}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right) &\leq \tfrac{1}{2}\left(\|\mathbf{U}\Sigma^{1/2}\|_F^2 + \|\Sigma^{1/2}\mathbf{V}^T\|_F^2\right) \\ &= \tfrac{1}{2}\textstyle\sum_{i=1}^n \sum_{k=1}^r (\mathbf{U}_{ik}^2\sigma_k + \mathbf{V}_{ik}^2\sigma_k) \\ &= \textstyle\sum_{k=1}^r \sigma_k = \|\mathbf{X}\|_* \end{aligned}$$

When can we hope to recover **D**? (1)

Assume **D** is the 5×5 all-ones matrix (rank 1).

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & ? & ? & ? & ? \\ \mathbf{1} & ? & ? & ? & ? \\ \mathbf{1} & ? & ? & ? & ? \\ \mathbf{1} & ? & ? & ? & ? \end{pmatrix}$$

Ok

$$\begin{pmatrix} \mathbf{1} & ? & ? & \mathbf{1} & ? \\ ? & ? & \mathbf{1} & ? & ? \\ ? & \mathbf{1} & ? & ? & \mathbf{1} \\ \mathbf{1} & ? & \mathbf{1} & ? & ? \\ ? & \mathbf{1} & \mathbf{1} & ? & ? \end{pmatrix}$$

Ok

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & ? \\ \mathbf{1} & \mathbf{1} & ? & ? & ? \\ \mathbf{1} & ? & ? & ? & ? \\ \mathbf{1} & ? & ? & \mathbf{1} & ? \\ \mathbf{1} & ? & ? & ? & ? \end{pmatrix}$$

Not unique
(column missed)

$$\begin{pmatrix} \mathbf{1} & ? & ? & ? & ? \\ ? & \mathbf{1} & ? & ? & ? \\ ? & ? & \mathbf{1} & ? & ? \\ ? & ? & ? & \mathbf{1} & ? \\ ? & ? & ? & ? & \mathbf{1} \end{pmatrix}$$

Not unique
(insufficient samples)

Sampling strategy and sample size matter.

When can we hope to recover **D**? (2)

Consider the following rank-1 matrices and assume few revealed entries.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Ok ("incoherent")

$$\begin{pmatrix} 20 & 20 & 22 & 20 & 20 \\ 20 & 20 & 22 & 20 & 20 \\ 22 & 22 & 24 & 22 & 22 \\ 20 & 20 & 22 & 20 & 20 \\ 20 & 20 & 22 & 20 & 20 \end{pmatrix}$$

Ok ("incoherent")

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Bad ("coherent")
$\rightarrow$ first row required

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Bad ("coherent")
$\rightarrow$ $(1,1)$-entry required

Properties of **D** matter.

When can we hope to recover $\mathbf{D}$? (3)

Exact conditions under which matrix completion "works" is active research area:

- Which sampling schemes? (e.g., random, WR/WOR, active)
- Which sample size?
- Which matrices? (e.g., "incoherent" matrices)
- Noise (e.g., independent, normally distributed noise)

Theorem (Candès and Recht, 2009)

Let $\mathbf{D} = \mathbf{U}\Sigma\mathbf{V}^T$. If $\mathbf{D}$ is incoherent in that

$$\max_{ij} \mathbf{U}_{ij}^2 \leq \frac{\mu_B}{n} \qquad \text{and} \qquad \max_{ij} \mathbf{V}_{ij}^2 \leq \frac{\mu_B}{n}$$

for some $\mu_B = O(1)$, and if $\operatorname{rank}(\mathbf{D}) \leq \mu_B^{-1} n^{1/5}$, then $O(n^{6/5} r \log n)$ random samples without replacement suffice to recover $\mathbf{D}$ exactly with high probability.

Outline

Overview

Latent factor models in practice

- Millions of users and items
- Billions of ratings
- Sometimes quite complex models

Many algorithms have been applied to large-scale problems

- Gradient descent and quasi-Newton methods
- Coordinate-wise gradient descent
- Stochastic gradient descent
- Alternating least squares

Continuous gradient descent

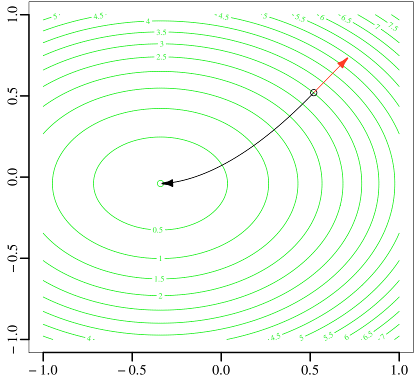
- Find minimum θ^* of function L
- Pick a starting point θ_0
- Compute gradient $L'(\theta_0)$
- Walk downhill
- Differential equation

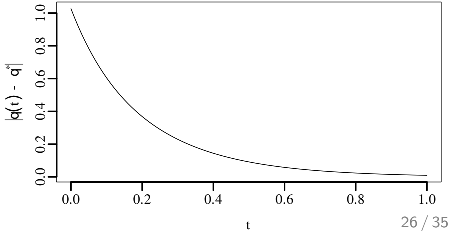
$$\frac{\partial \theta(t)}{\partial t} = -L'(\theta(t))$$

with boundary cond. $\theta(0) = \theta_0$

- Under certain conditions

$$\theta(t) \to \theta^*$$





Discrete gradient descent

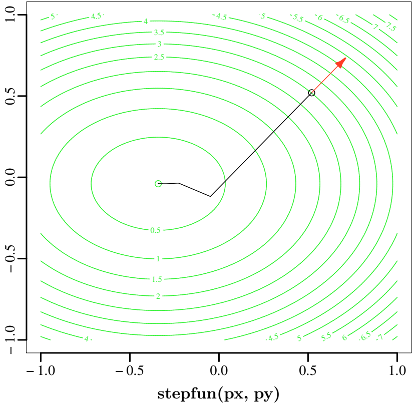
- Find minimum θ^* of function L
- Pick a starting point θ_0
- Compute gradient $L'(\theta_0)$
- Jump downhill
- Difference equation

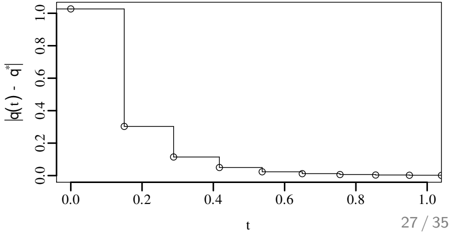
$$\theta_{n+1} = \theta_n - \epsilon_n L'(\theta_n)$$

- Under certain conditions, approximates CGD in that

$$\theta^n(t) = \theta_n + \text{“steps of size } t\text{”}$$

satisfies the ODE as $n \to \infty$



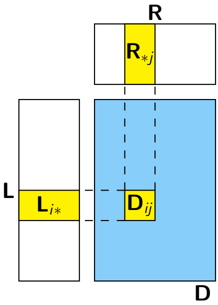


Gradient descent for LFMs

- Set $\theta = (\mathbf{L}, \mathbf{R})$ and write

$$L(\theta) = \sum_{(i,j)\in\Omega} L_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j})$$

$$\nabla_{\mathbf{L}_{i*}} L(\theta) = \sum_{j\in\{\, j' \mid (i,j')\in\Omega\,\}} \nabla_{\mathbf{L}_{i*}} L_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j})$$



- GD epoch
  1. Compute gradient
     - ★ Initialize zero matrices $\mathbf{L}^{\nabla}$ and $\mathbf{R}^{\nabla}$
     - ★ For each entry $(i, j) \in \Omega$, update gradients

$$\mathbf{L}^{\nabla}_{i*} \leftarrow \mathbf{L}^{\nabla}_{i*} + \nabla_{\mathbf{L}_{i*}} L_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j})$$

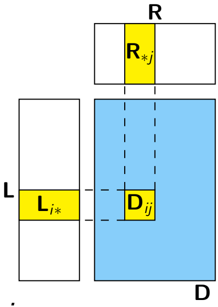
$$\mathbf{R}^{\nabla}_{*j} \leftarrow \mathbf{R}^{\nabla}_{*j} + \nabla_{\mathbf{R}_{*j}} L_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j})$$

  2. Update parameters

$$\mathbf{L} \leftarrow \mathbf{L} - \epsilon_n \mathbf{L}^{\nabla}$$

$$\mathbf{R} \leftarrow \mathbf{R} - \epsilon_n \mathbf{R}^{\nabla}$$

Computing the gradient (example)



Simplest form (unregularized)

$$L_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j}) = (\mathbf{D}_{ij} - \mathbf{L}_{i*}\mathbf{R}_{*j})^2$$

Gradient computation

$$\nabla_{\mathbf{L}_{i'k}} L_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j}) = \begin{cases} 0 & \text{if } i' \neq i \\ -2\mathbf{R}_{kj}(\mathbf{D}_{ij} - \mathbf{L}_{i*}\mathbf{R}_{*j}) & \text{if } i' = i \end{cases}$$

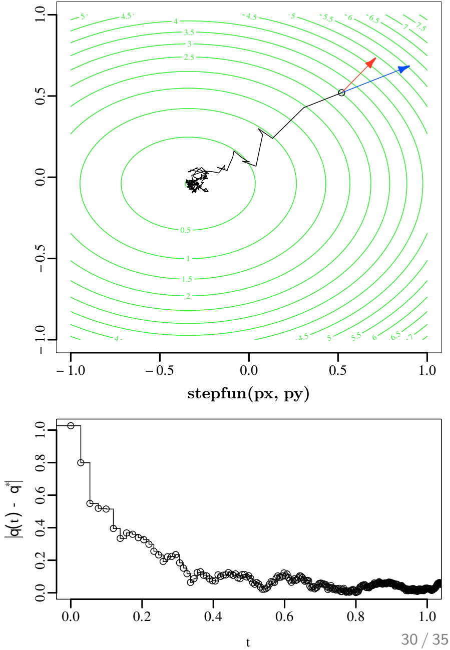
Local gradient of entry $(i, j) \in \Omega$ nonzero only on row $\mathbf{L}_{i*}$ and column $\mathbf{R}_{*j}$.

Stochastic gradient descent

- Find minimum θ^* of function L
- Pick a starting point θ_0
- Approximate gradient $\hat{L}'(\theta_0)$
- Jump "approximately" downhill
- Stochastic difference equation

$$\theta_{n+1} = \theta_n - \epsilon_n \hat{L}'(\theta_n)$$

- Under certain conditions, asymptotically approximates (continuous) gradient descent



Stochastic gradient descent for LFMs

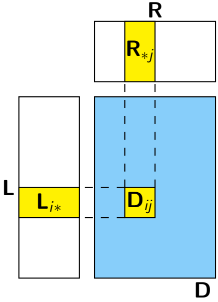
- Set $\theta = (\mathbf{L}, \mathbf{R})$ and use

$$L(\theta) = \sum_{(i,j)\in\Omega} L_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j})$$

$$L'(\theta) = \sum_{(i,j)\in\Omega} L'_{ij}(\mathbf{L}_{i*}, \mathbf{R}_{*j})$$

$$\hat{L}'(\theta, z) = N L'_{i_z j_z}(\mathbf{L}_{i_z *}, \mathbf{R}_{* j_z}),$$

where $N = |\Omega|$ and $z = (i_z, j_z) \in \Omega$



- SGD epoch
  1. Pick a random entry $z \in \Omega$
  2. Compute approximate gradient $\hat{L}'(\theta, z)$
  3. Update parameters

$$\theta_{n+1} = \theta_n - \epsilon_n \hat{L}'(\theta_n, z)$$

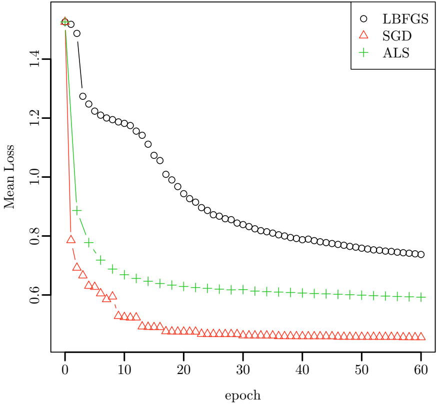
  4. Repeat N times

SGD step affects only current row and column.

SGD in practice

Step size sequence $\{\epsilon_n\}$ needs to be chosen carefully

- Pick initial step size based on sample (of some rows and columns)
- Reduce step size gradually
- **Bold driver heuristic**: After every epoch
  - Increase step size slightly when loss decreased (by, say, 5%)
  - Decrease step size sharply when loss increased (by, say, 50%)



Netflix data (unregularized)

Outline

Lessons learned

- Collaborative filtering methods learn from past user behavior
- Latent factor models are best-performing single approach for collaborative filtering
  - But often combined with other methods
- Users and items are represented in common latent factor space
  - Holistic matrix-factorization approach
  - Similar users/item placed at similar positions
  - Low-rank assumption = few "factors" influence user preferences
- Close relationship to matrix completion problem
  - Reconstruct a partially observed low-rank matrix
- SGD is simple and practical algorithm to solve LFMs in summation form

Suggested reading

- Y. Koren, R. Bell, C. Volinsky
  Matrix factorization techniques for recommender systems
  IEEE Computer, 42(8), p. 30–37, 2009
  `http://research.yahoo.com/pub/2859`

- E. Candès, B. Recht
  Exact matrix completion via convex optimization
  Communications of the ACM, 55(6), p. 111–119, 2012
  `http://doi.acm.org/10.1145/2184319.2184343`

- And references in the above articles