Recommendation Systems

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
July 9, 2020

Learning Goals Today

- Intro: Model for Recommendation Systems
- Collaborative Filtering
- Dimensionality Reduction: The UV Decomposition

Recommendation Systems
Introduction

RECOMMENDATION SYSTEMS

- *Recommendation systems* are
  - web applications that
  - predict user responses to options
- *Examples:*
  - Offering articles to online newspaper readers based on predicting reader interests
  - Offering online retailer suggestions to customers based on prior purchases / searches
- *Classification:*
  - *Content based systems:* characterize properties of items examined ☞ movie is "cowboy" movie if watched by many users liking cowboy movies
  - *Collaborative filtering systems:* recommend items based on similarity measures between users and/or items

Recommendation Systems: Model

- The *Utility Matrix:* Putting users and items into context
- *Long Tails:* Contain items that serve only small amounts of users
  - Long tail items not displayable in regular stores, while full range of products available online
- *Applications:*
  - Recommending products
  - Recommending movies
  - Recommending news articles

The Utility Matrix

Definition [Utility Matrix]:

- Let m be the number of users
- Let n be the number of items
- Let S be a set of ratings/values, including an element $\_$ representing "unknown"
- The utility matrix $M \in S^{m \times n}$ has m rows and n columns where

$$M_{ui} \in S \tag{1}$$

  reflects the *degree of preference* of user $u \in \{1, ..., m\}$ for item $i \in \{1, ..., n\}$.
- If $M_{ui} = \_$, the degree of preference of user u for item i is unknown.

The Utility Matrix: Example

- The utility matrix $M \in S^{m \times n}$ has m rows and n columns where

$$M_{ui} \in S$$

  reflects the *degree of preference* of user u for item i.
- If $M_{ui} = \_$, the degree of preference of user u for item i is unknown.

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

Utility matrix users $\times$ movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

THE UTILITY MATRIX: GOAL

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

Utility matrix reflecting users $\times$ movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- *Goal:* Predict values from S other than $\_$ for unknown entries $M_{ui} = \_$
- Note that in applications, not every value needs to be predicted
- Sufficiently many predictions for a user suffice

THE UTILITY MATRIX: EXAMPLE

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

Utility matrix reflecting users × movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- *HP* = Harry Potter, *TW* = Twilight, *SW* = Star Wars
- E.g. user *A* likes Twilight, user *B* likes Harry Potter
- *Possible question:* Will user *A* like movie *SW2*?
- Note similarity between *SW1* and *SW2*, note that *A* disliked *SW1*
- *Answer:* Possibly not!

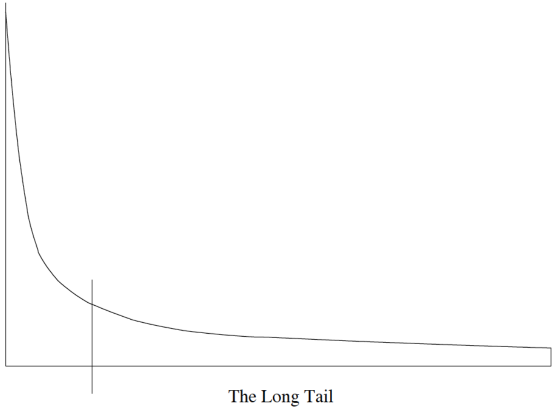
Populating the Utility Matrix

- Acquiring data from which to build utility matrix can be difficult
- *User Ratings:* Ask users to provide estimates; *however*
  - Users are unwilling to provide responses
  - Ratings are biased towards those willing
- *Infer from users' behaviour*
  - Once bought item / watched movie, rate as liked by user
  - Value system only has 0 and 1, where 0 reflects _

The Long Tail

- *Physical stores*
  - suffer from limited resources for items
  - e.g. can offer several thousands of books
  - *Recommendation:* Pick most purchased items and recommend to everyone
- *Online stores*
  - do not suffer from lack of resources
  - e.g. can offer several millions of books
  - *Recommendation:* Substantially more involved
- The Long Tail Phenomenon explains why recommendations systems are necessary

The Long Tail: Illustration



Items (x-axis) rated by popularity (y-axis); vertical bar: cutoff in physical stores

Adopted from `mmds.org`

Recommendation Systems: Applications

- *Product Recommendations*
  - *Amazon* offers products to returning users based on prior purchases
  - *Extreme example:* “Touching the Void” only increased in popularity after “Into Thin Air” appeared on the market
- *Movie Recommendations*
  - *Netflix* suggests movies to watch to users
  - Netflix offered one million dollars for algorithm beating their own recommendation system by 10%
  - Price was won in 2009 by team of researchers called “Bellkor’s Pragmatic Chaos”
- *News Articles*
  - Identify articles of interest to readers
  - Similarity based on similarity of important words and/or articles read by people with similar interests
  - *YouTube* is another example

Content Based Recommendations

- Content based systems focus on properties of items
  - Determine features that describe the items
  - Represent items as vector in feature space
  - E.g. represent movies as bitvectors where entries relate to actors: 1 means actor plays in movie, 0 s/he doesn't
- For recommending items to users:
  - Develop user representations referring to the same feature space
  - E.g. represent movie watchers as vector where entries represent preferences for actors
  - *Recommendation:* Item bitvectors that are similar to user vector representations
  - Jaccard distance, Cosine distance etc.

Collaborative Filtering

Collaborative Filtering: Introduction

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

Utility matrix users $\times$ movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- Instead of item profiles, make direct use of utility matrix
- *Items* are represented by columns in utility matrix
- *Users* are represented by rows in utility matrix
- *Recommendations:*
  - Identify users that are similar to the particular user
  - Recommend items considered by the users identified as similar

How to compute user similarity?

Collaborative Filtering: Introduction

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

Utility matrix users $\times$ movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- A and B watched only one movie together, which they both liked
- A and C watched two movies together, but seem to have opposite opinions in both cases

Good similarity measure supposed to reflect this

Collaborative Filtering: Jaccard Distance

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

Utility matrix users $\times$ movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- Users = sets of movies, containing all movies they watched
- $$\mathrm{SIM}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{1}{5} < \frac{1}{2} = \frac{2}{4} = \frac{|A \cap C|}{|A \cup C|} = \mathrm{SIM}(A, C)$$
- *Conclusion:* Not a good idea when utility matrix contains ratings

Collaborative Filtering: Cosine Distance

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

Utility matrix users × movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- Users are vectors of integers
- Compute cosine of angle between user vectors
- Treat blanks as zeroes
  ☞ Questionable idea: missing rating = bad rating

Collaborative Filtering: Cosine Distance

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

Rounded utility matrix users $\times$ movies

Adopted from `mmds.org`

- Cosine(A,B):
$$\frac{4 \times 5}{\sqrt{4^2 + 5^2 + 1^2}\sqrt{5^2 + 5^2 + 4^2}} = 0.380$$
- Cosine(A,C):
$$\frac{5 \times 2 + 1 \times 4}{\sqrt{4^2 + 5^2 + 1^2}\sqrt{2^2 + 4^2 + 5^2}} = 0.322$$
- *Conclusion:* Points in the right direction

Collaborative Filtering: Rounding Data

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	1			1			
B	1	1	1				
C					1	1	
D		1					1

Utility matrix users $\times$ movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- Round at cutoff: 0, 1, 2 $\rightarrow$ 0; 3, 4, 5 $\rightarrow$ 1
- $$\text{SIM}(A, B) = \frac{1}{4} > 0 = \text{SIM}(A, C)$$
- *Conclusion:* Points in the right direction as well

COLLABORATIVE FILTERING: NORMALIZING DATA

	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

Utility matrix users $\times$ movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- Subtract average rating of respective user from each rating
  - Low ratings become negative numbers
  - High ratings become positive numbers
- Cosine distance:
  - Users with opposite views = vectors pointing in opposite directions
  - Users with similar views = small angle between vectors

COLLABORATIVE FILTERING: NORMALIZING DATA

	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

Utility matrix users × movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- User D essentially disappeared (because of too indifferent ratings)
- Cosine(A,B):

$$\frac{(2/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2}\sqrt{(1/3)^2 + (1/3)^2 + (-2/3)^2}} = 0.092$$

- Cosine(A,C):

$$\frac{(5/3) \times (-5/3) + (-7/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2}\sqrt{(-5/3)^2 + (1/3)^2 + (4/3)^2}} = -0.559$$

Collaborative Filtering: Normalizing Data

	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

Utility matrix users $\times$ movies, where $S = \{1, 2, 3, 4, 5, \_\}$

Adopted from `mmds.org`

- Cosine(A,B) = 0.092; Cosine(A,C) = -0.559
- *Conclusion:* Makes sense
  - A, B slight similarity, just one movie rated in common
  - A, C disagree to a stronger degree

Duality of Similarity

- Utility matrix tells about users, or items, or both
- While we focused on user similarity, techniques presented so far can be applied to identify similar items, too
- *However, difference* is that items are classifiable, while users are not
  - Movies can be classified according to genres
  - Users are rather heterogeneous in terms of genres
- *Consequence:* Similar items are easier to discover

Duality of Similarity: Predictions

Predicting entries in utility matrix M

- First, normalize utility matrix (as described above)
- Let *sim* denote similarity measure of choice
- Let u be user, i be item; we would like to predict M_{ui}, where
  - only predicting M_{ui} is useless
  - we need to predict M_{ui} for many i, to put entries into mutual context

Duality of Similarity

Predicting entries in utility matrix M

- *First approach:* Select top m users $u_j, j = 1, ..., m$ similar to u and compute

$$M_{ui} = \frac{1}{m} \sum_{j=1}^{m} sim(u_j, u) M_{u_j i} \tag{2}$$

  - *Advantage:* One computation for several M_{ui} for one u
  - *Disadvantage:* Based on user similarity, which is less reliable

- *Second approach:* Select top m items $i_j, j = 1, ..., m$ similar to i and compute

$$M_{ui} = \frac{1}{m} \sum_{j=1}^{m} sim(i_j, i) M_{u i_j} \tag{3}$$

  - *Advantage:* Based on item similarity, which is more reliable
  - *Disadvantage:* Need to consider several items i for one u

Clustering Utility Matrix

- The utility matrix is sparse; many entries are missing
  - *Two items*, even if classified identically, miss users with entries for both of them
  - *Two users*, even if having identical interests, miss items that they both have entries for
- For increasing coherence, and decreasing sparsity: cluster items, or users, or both

Clustering Utility Matrix

- For clustering, apply iterative procedure (hierarchical clustering):
  - Cluster items, e.g. decreasing number of columns by factor of two
  - Entries for clustered columns are averages of single entries
  - Cluster users, e.g. decreasing number of rows by factor of two
  - Entries for clustered rows are averages of single entries

	HP	TW	SW
A	4	5	1
B	4.67		
C		2	4.5
D	3		3

Utility matrix after one iteration of clustering items

Adopted from `mmds.org`

Clustering Utility Matrix: Predictions

	HP	TW	SW
A	4	5	1
B	4.67		
C		2	4.5
D	3		3

Utility matrix after one iteration of clustering items

Adopted from `mmds.org`

- After clustering, predict items M_{ui} as follows:
  - Identify clusters of user u (cluster X) and item i (cluster Y)
  - Predict M_{ui} as M_{XY} in the clustered utility matrix
  - If M_{XY} is empty, use distance based methods to predict M_{XY}, and predict M_{ui} as M_{XY} when done

Dimensionality Reduction

THE UV-DECOMPOSITION

- Let M be utility matrix, for m users and n items
  Important: **In `https://mmds.org`, m and n are reversed**
- *Assumption:* There are $d \leq m, n$ hidden features such that
  - Users u can be represented as d-dimensional vectors across these features
  - Items i can be represented as d-dimensional vectors across these features
  - For example, for movies and watchers, hidden features may refer to genres
- How to reveal such hidden features?
- *Solution:* Apply *UV-decomposition* of M
- *Note:* Interpretation of meaning of hidden features may remain unclear

The UV-Decomposition

Definition [UV-Decomposition]

- Let $M \in \mathbb{R}^{m \times n}$ be a utility matrix; let $d \leq n, m$
- Let $U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{d \times n}$ such that

$$UV \in \mathbb{R}^{m \times n} \text{ approximates } M \in \mathbb{R}^{m \times n} \text{ closely}$$

- Then U, V is called a *UV-Decomposition (relative to d)* of M

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

UV-decomposition of matrix M

Adopted from `mmds.org`

The UV-Decomposition

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

UV-decomposition of matrix M

Adopted from `mmds.org`

- *Prediction:* Estimate missing entry M_{ui} as $(UV)_{ui} = \sum_{k=1}^{d} u_{uk} v_{ki}$
- *Example:* Predict missing M_{32} as $u_{31}v_{12} + u_{32}v_{22}$

UNIVERSITÄT
BIELEFELD

Measuring Closeness

Definition [Root-Mean-Square Error]

- Let $M \in \mathbb{R}^{m \times n}$ be decomposed into UV for $U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{d \times n}$
- Let l be the number of non-blank entries in M

The *root-mean-square error (RMSE)* of M and UV is defined to be

$$\sqrt{\frac{1}{l} \sum_{\substack{(u,i) \\ M_{ui} \neq -}} (M_{ui} - (UV)_{ui})^2} \tag{4}$$

that is the square root of the average over the squares of differences between M_{ui} and $(UV)_{ui}$ for all (u, i) where M_{ui} is not missing.

Example

- In the example from above

$$\text{RMSE}(M, UV) = \sqrt{\frac{1}{23}(5 - (u_{11}v_{11} + u_{12}v_{21}))^2 + \ldots + (4 - (u_{51}v_{14} + u_{52}v_{24})^2}$$

UV Decomposition: Incremental Computation

Computing U, V: Idea

- Start with arbitrary (while still reasonably chosen) U, V
- Iterating through elements U_{uk}, V_{ki}, decrease RMSE(M, UV) by adjusting single entries U_{uk} or V_{ki} in U or V
- Do this until convergence; eventually, U, V may reflect local minima
- Repeat this by varying inital choices for U, V to get global minimum or suitable local minimum

UV Decomposition: Incremental Computation

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

Matrix M to be decomposed into UV

Adopted from `mmds.org`

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Initial choice for U, V

Adopted from `mmds.org`

Initial RMSE: $\sqrt{\frac{75}{23}} = 1.806$

UV Decomposition: Incremental Computation

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

Matrix M to be decomposed into UV

Adopted from `mmds.org`

$$\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Varying $x = U_{11}$

Adopted from `mmds.org`

Minimize contribution from $x = U_{11}$ to sum of squares:

$$(5-(x+1))^2 + (2-(x+1))^2 + (4-(x+1))^2 + (4-(x+1))^2 + (3-(x+1))^2$$

UV Decomposition: Incremental Computation

Minimize contribution from $x = U_{11}$ to sum of squares:

$(5-(x+1))^2 + (2-(x+1))^2 + (4-(x+1))^2 + (4-(x+1))^2 + (3-(x+1))^2$

which simplifies to

$$(4-x)^2 + (1-x)^2 + (3-x)^2 + (3-x)^2 + (2-x)^2$$

Take derivative and set to zero:

$-2 \times ((4-x)+(1-x)+(3-x)+(3-x)+(2-x)) = 0 \quad \text{or} \quad -2 \times (13-5x) = 0$

from which we obtain $x = 2.6$.

UV Decomposition: Incremental Computation

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix}$$

Matrix M to be decomposed into UV

Adopted from `mmds.org`

$$\begin{bmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} y & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2.6y+1 & 3.6 & 3.6 & 3.6 & 3.6 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Varying $y = V_{11}$

Adopted from `mmds.org`

Minimize contribution from $y = V_{11}$ to sum of squares:

$$(5-(2.6y+1))^2 + (3-(y+1))^2 + (2-(y+1))^2 + (2-(y+1))^2 + (4-(y+1))^2$$

UV Decomposition: Incremental Computation

Minimize contribution from $y = V_{11}$ to sum of squares:

$(5-(2.6y+1))^2+(3-(y+1))^2+(2-(y+1))^2+(2-(y+1))^2+(4-(y+1))^2$

which simplifies to

$$(4-2.6y)^2+(2-y)^2+(1-y)^2+(1-y)^2+(3-y)^2$$

Take derivative and set to zero:

$$-2 \times (2.6(4-2.6y)+(2-y)+(1-y)+(1-y)+(3-y)) = 0$$

from which we obtain $y = 1.617$.

UV Decomposition: Incremental Computation

- $\sum_i$ be shorthand for sum over all i such that m_{ui} is not missing
- $\sum_u$ be shorthand for sum over all u such that m_{ui} is not missing
- $\sum_{j \neq k}$ shorthand for sum over all $j = 1, ..., d$ except for $j = k$
- General formula for determining optimal $x = U_{uk}$:

$$x = \frac{\sum_i V_{ki}(M_{ui} - \sum_{j \neq k} U_{uj} V_{ji})}{\sum_i V_{ki}^2} \tag{5}$$

- General formula for determining optimal $y = V_{ki}$:

$$y = \frac{\sum_u U_{uk}(M_{ui} - \sum_{j \neq k} U_{uj} V_{ji})}{\sum_u U_{uk}^2} \tag{6}$$

Complete UV-Decomposition Algorithm

There are four *issues* to deal with:

1. Preprocessing M
   - Normalize M; undo normalization when making predictions
2. Initializing U and V
   - Let a be average across non-blank elements of M
   - Choose $\sqrt{a/d}$ for each entry of U and V
   - Perturb value $\sqrt{a/d}$ randomly and independently for varying initialization
3. Determine order in which to optimize elements of U, V
   - Do row-by-row or column-by-column
   - Choose entries randomly
4. Convergence? Stop the iteration.
   - Stop when improvements in RMSE become negligible

Materials / Outlook

- See *Mining of Massive Datasets*, chapter 9.1, 9.3, 9.4
- As usual, see `http://www.mmds.org/` in general for further resources
- Next lecture: "Various Topics"