UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Recommender Systems

Matrix Factorization

Rodrygo Santos
rodrygo@dcc.ufmg.br

Quick recap

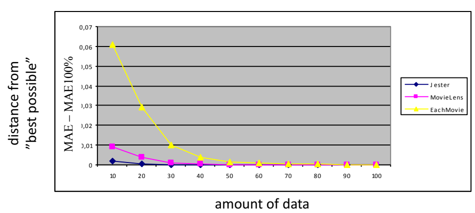
UFMG COMPUTER SCIENCE

- How will I like item *x*?
  - User-based
    - *What have similar users found of item x?*
  - Item-based
    - *What have I found of items similar to x?*
- Key difference: neighborhoods
  - User-based: unstable, hard to precompute
  - Item-based: stable, easy to precompute

How robust is this?

UFMG COMPUTER SCIENCE

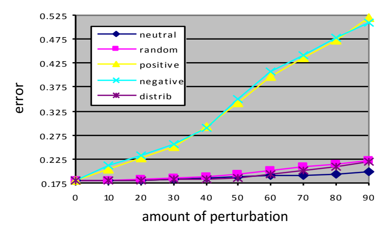
- What if we have sparse data?
  - E.g., new user *a*, new item *p*?
    - No available rating r_{ap}



How robust is this?

UFMG COMPUTER SCIENCE

- What if we have perturbations?
  - Positively rate your own items
  - Negatively rate your competitors'



Memorize or model?

UFMG COMPUTER SCIENCE

- Memory-based
  - Online "learning"
  - Online prediction
  - *Lazy, instance-based learning (k-NN like)*

Problems?

- Costly recommendation
  - $O(mn)$ worst case
- Poor robustness
  - Sparsity
  - Perturbations
- Overfit representation
  - I like Star Wars, you like Star Trek
  - ***Are we neighbors?***

Memorize or model?

UFMG COMPUTER SCIENCE

- Memory-based
  - Online "learning"
  - Online prediction
  - *Lazy, instance-based learning (k-NN like)*
- Model-based
  - Offline learning
  - Online prediction

Can we compactly represent users' tastes and item descriptions?

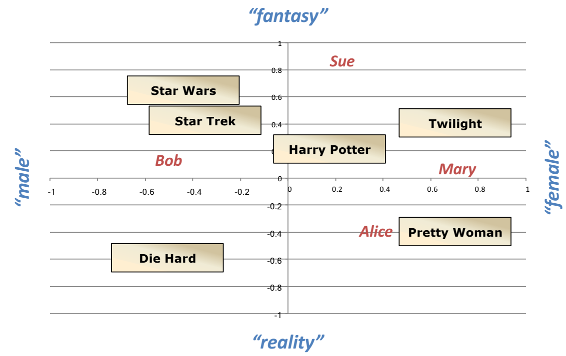
Dimensionality reduction

UFMG COMPUTER SCIENCE

- Given an *m* x *n* utility matrix
  - User-based CF is *n*-dimensional
  - Item-based CF is *m*-dimensional
    - *m* and *n* could be in the hundreds of millions
- *Can we **reduce the dimensions** of the utility matrix while effectively **retaining information** about each user's preferences?*

Lower dimensional projection

UFMG COMPUTER SCIENCE



Where did this come from?

UFMG COMPUTER SCIENCE

- Vocabulary mismatch problem
  - Queries are vectors over keywords
  - Documents are vectors over keywords
  - *We want to match **concepts**!*
- Latent semantic indexing (LSI) (Deerwester et al., ASIS 1988)
  - Represent queries and documents in a compact and robust space of latent concepts
- Actually a bit older... *(like a century!)*

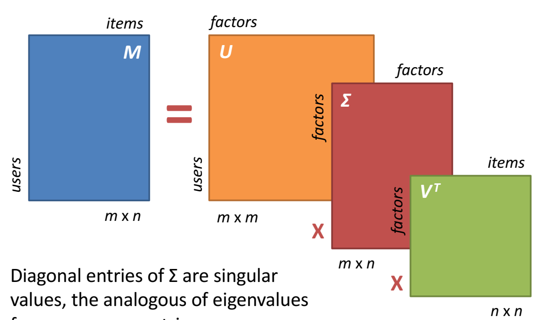
Singular value decomposition

UFMG COMPUTER SCIENCE

- Reduce dimensionality of the problem
  - Results in small, fast model
  - Richer, denser neighbor network
- One of various matrix factorization techniques
  - Techniques based on linear systems (e.g., LU)
  - Techniques based on eigenvalues (e.g., SVD)

Factorization with SVD

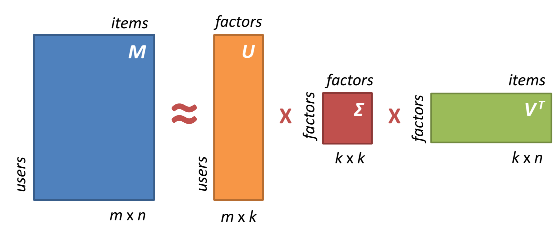
UFMG COMPUTER SCIENCE



Diagonal entries of Σ are singular values, the analogous of eigenvalues for non-square matrices

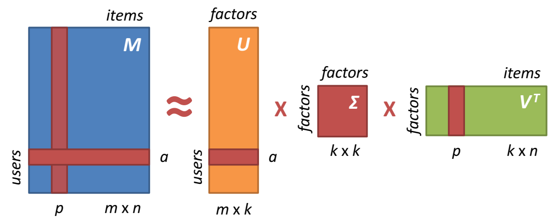
Factorization with SVD

UFMG COMPUTER SCIENCE



- "Truncated" SVD: keep *k* "most important" factors
  - Best rank-*k* approximation (by Frobenius norm)

Predictions with SVD



$$\hat{r}_{ap} = \bar{r}_a + \frac{\sum_{b \in N} sim(a,b) \times (r_{bp} - \bar{r}_b)}{\sum_{b \in N} sim(a,b)} \approx \hat{r}_{ap} = \bar{r}_a + U_a \times \Sigma \times V_p^T$$

SVD pros

- Prediction quality generally increases…
  - Noisy ratings filtered out
  - Nontrivial correlations detected
- … although it may also decrease
  - Original ratings are not taken into account
- Depends on the amount of reduction
  - Normally 20 to 100 factors (Koren, KDD 2009)
  - But ***it really depends*** on the target domain

SVD cons

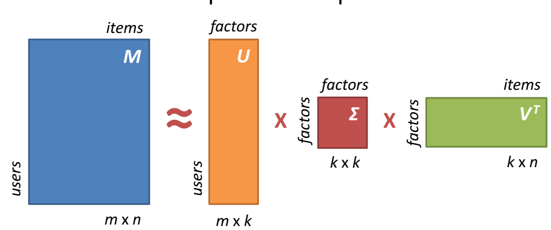
- Lack of transparency
  - Optimal dimensions do not correspond to user-comprehensible concepts
- Missing values
  - SVD is undefined for incomplete matrices
  - Utility matrix has lots of missing values
- Computation complexity
  - SVD computation is $O(m^2n + n^3)$

Missing values

- SVD assumes a complete matrix
  - If it's complete, we don't need a recommender!
- What to do with missing values?
  - Impute (assume they are a mean)
  - Normalize (assume they are 0)
  - Ignore!

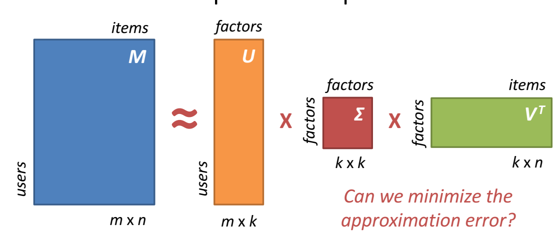
Computational complexity

- Standard SVD is (very) slow
  - All we need is a good approximation
- Model it as an optimization problem!



Computational complexity

- Standard SVD is (very) slow
  - All we need is a good approximation
- Model it as an optimization problem!



Can we minimize the approximation error?

Quantifying the error

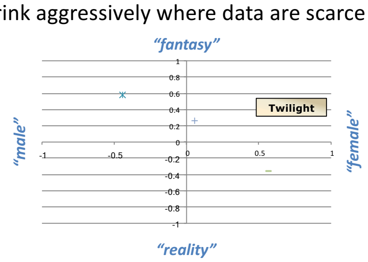
- Approximated SVD
  - $\hat{r}_{ap} = U_a V_p^T$ (*Σ* embedded in *U* and *V*)
- Approximation error
  - $e_{ap} = r_{ap} - \hat{r}_{ap}$

Defining the problem

- We want to minimize the error over all ***known ratings***, hence ***ignoring missing*** values
  - $\min_{U,V} \sum_{(a,p)} (r_{ap} - U_a V_p^T)^2$ (r_{ap} *is known*)
- Problem: overfitting
  - We are learning from few examples after all
- Solution: regularization
  - $\min_{U,V} \sum_{(a,p)} (r_{ap} - U_a V_p^T)^2 + \lambda(|U_a|^2 + |V_p|^2)$

The effect of regularization

- Occam's razor (aka KISS)
  - Allow rich model where there are sufficient data
  - Shrink aggressively where data are scarce



A funky solution

Simon Funk's SVD

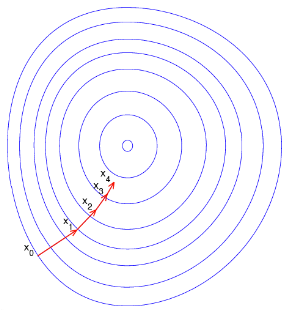
- One of the most interesting findings during the ***Netflix Prize*** came out of a blog post
  - *"Ok, so here's where I tell all about how I got to be tied for third place on the Netflix prize."* Simon Funk, 2006
- Incremental, iterative, and approximate way to compute the SVD using ***gradient descent***

Gradient descent

- Problem
  - $\min F(\mathbf{x})$
- Solution
  - Gradient points in the direction of the greatest decrease (or increase) of a function
  - Why not step iteratively in that direction?
    - $\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \nabla F(\mathbf{x}_i)$
  - γ_i is the learning rate
    - Higher = faster training, lower accuracy

Gradient descent

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \nabla F(\mathbf{x}_i)$$



A simplified example

- Goal is to decompose the left-side matrix

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

A simplified example

- An initial (random) guess

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

A simplified example

- Not that impressive…

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

- RMSE = 1.806

A simplified example

- An improved guess

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

A simplified example

- An improved guess

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

- Contribution of the first row $(5-(x+1))^2 + (2-(x+1))^2 + (4-(x+1))^2 + (4-(x+1))^2 + (3-(x+1))^2$

A simplified example

- An improved guess

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

- How to choose x to minimize the error?
  - Take the gradient!

A simplified example

- The error contribution

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

- Simplifies to

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

- Taking the derivative and equating to 0

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

A simplified example

- Replacing $x = 2.6$

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

A simplified example

- Replacing $x = 2.6$

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} 3.6 & 3.6 & 3.6 & 3.6 & 3.6 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

- RMSE ***reduced*** from 1.806 to 1.642

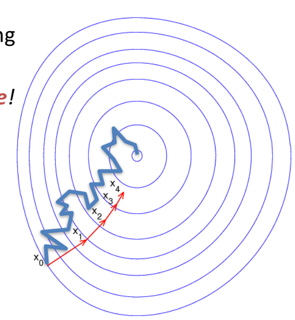
Funk's SVD

initialize matrices U, V
for each factor f:
    until f has converged:
        for each rating r_{ap}:
            predict $\hat{r}_{ap}$
            update U_{af}
            update V_{pf}

Do we need to use every rating in every iteration?

Stochastic gradient descent

- We can sample training ratings randomly
- *Improves **convergence**!*
  - *Aka **faster** learning*



Summary

UFMG COMPUTER SCIENCE

- Matrix factorization is dominant
  - Superior to classical nearest neighbor techniques
  - Uses a more compact memory-efficient model
- Matrix factorization is flexible
  - Can integrate multiple forms of feedback, user and item biases, temporal dynamics, and confidence levels (Koren et al., Computer 2009)
- SVD is not the only factorization approach

Writing Assignment #3

UFMG COMPUTER SCIENCE

- Write a one-page summary about:
  - Matrix factorization techniques for recommender systems (Computer 2009) by Yehuda Koren, Robert Bell, Chris Volinsky
  - Due Mon, Apr 17 @ 23:55 via Moodle