7-2: Diving Deeper with SVD

Introduction to Recommender Systems

Introduction

- Last time: introduced matrix factorization
  - Singular Value Decomposition
- This video: details on how SVDs work
  - Algebraic understanding
  - Dealing with missing data
  - Updating with new data

The Algebra of an SVD

- Rating matrix M decomposes to $U\Sigma V^T$
- U, V orthogonal
  - translate vectors into & out of low-dim space
- Σ diagonal matrix of singular values
- Truncate: keep *k* ‘most important’ dimensions (highest singular values)
  - Best rank-k approximation (by RMSE/Frobenius norm)
  - de-noises the data

Computing an SVD

- Well-known algorithms in many linear algebra packages
  - Matlab
  - LINPACK
  - ARPACK
- Very slow

Missing Data

- SVD formulation (and many solvers) assume matrix is complete
  - But if it's complete, don't need recommender
- What to do with missing values?
  - ‘Impute’ — assume they are a mean
  - Normalize data first — assume they are 0
  - Next lecture: ignore them

Adding New Data (folding in)

- New user joins the system and rates some items
  - They weren't in last night's model build
- Vector spaces to the rescue!
  - Multiply user rating vector by item-feature matrix
  - This gives you user-feature vector

General SVD Practice

- Build models regularly
- Fold-in user's current ratings for live recommendation
- Impute means or pre-normalize data to handle missing data
  - Pre-normalizing lets you use standard sparse matrix and vector arithmetic

7-2: Diving Deeper with SVD

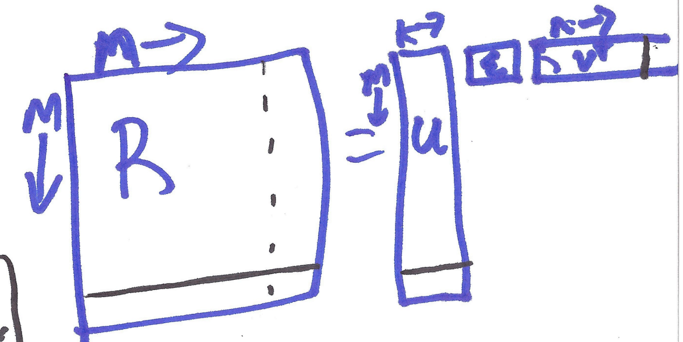
m = # users
n = # items

$m \times n$

$$R = U \Sigma V^T$$

$m \times k$ U - user-feature matrix
$n \times k$ V - item-feature matrix
orthoganal

$k \times k$ Σ - weights
diagonal
only keep k largest
- best rank-k approx.
by global RMSE



$$p_{a,i} = \sum_{f=1}^{k} u_{af} \sigma_f v_{if}$$

$$R = U \Sigma V^T$$

$$\vec{u}_a = \Sigma V^T \vec{r}_a^T$$