CSC 411 Lecture 18: Matrix Factorizations

Roger Grosse, Amir-massoud Farahmand, and Juan Carrasquilla

University of Toronto

Overview

- Recall PCA: project data onto a low-dimensional subspace defined by the top eigenvalues of the data covariance
- We saw that PCA could be viewed as a linear autoencoder, which let us generalize to nonlinear autoencoders
- Today we consider another generalization, matrix factorizations
    - view PCA as a matrix factorization problem
    - extend to matrix completion, where the data matrix is only partially observed
    - extend to other matrix factorization models, which place different kinds of structure on the factors

PCA as Matrix Factorization

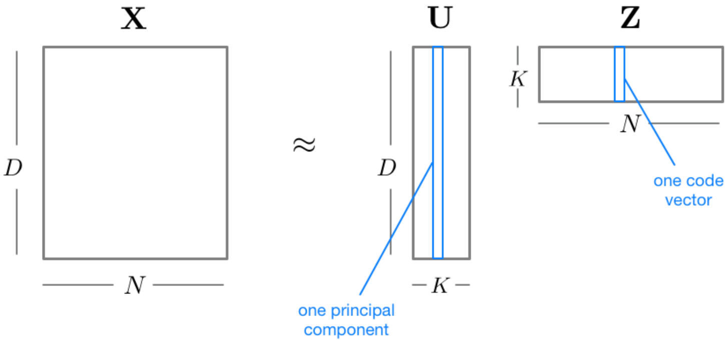
- Recall: each input vector $\mathbf{x}^{(i)}$ is approximated as $\mathbf{Uz}$, where $\mathbf{U}$ is the orthogonal basis for the principal subspace, and $\mathbf{z}$ is the code vector.
- Write this in matrix form: $\mathbf{X}$ and $\mathbf{Z}$ are matrices with one *column* per data point
  - I.e., for this lecture, we transpose our usual convention for data matrices.
- Writing the squared error in matrix form

$$\sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \mathbf{U}\mathbf{z}^{(i)}\|^2 = \|\mathbf{X} - \mathbf{UZ}\|_F^2$$

- Recall that the Frobenius norm is defined as $\|\mathbf{A}\|_F^2 = \sum_{i,j} a_{ij}^2$.

PCA as Matrix Factorization

- So PCA is approximating $\mathbf{X} \approx \mathbf{UZ}$.



- Based on the sizes of the matrices, this is a rank-K approximation.
- Since $\mathbf{U}$ was chosen to minimize reconstruction error, this is the *optimal* rank-K approximation, in terms of $||\mathbf{X} - \mathbf{UZ}||_F^2$.

PCA vs. SVD (optional)

This has a close relationship to the Singular Value Decomposition (SVD) of $\mathbf{X}$. This is a factorization

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$$

Properties:

- $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}^\top$ provide a real-valued matrix factorization of $\mathbf{X}$.
- $\mathbf{U}$ is a $n \times k$ matrix with orthonormal columns, $\mathbf{U}^\top\mathbf{U} = \mathbb{I}_k$, where $\mathbb{I}_k$ is the $k \times k$ identity matrix.
- $\mathbf{V}$ is an orthonormal $k \times k$ matrix, $\mathbf{V}^\top = \mathbf{V}^{-1}$.
- $\mathbf{S}$ is a $k \times k$ diagonal matrix, with non-negative singular values, $s_1, s_2, \ldots, s_k$, on the diagonal, where the singular values are conventionally ordered from largest to smallest.

It's possible to show that the first k singular vectors correspond to the first k principal components; more precisely, $\mathbf{Z} = \mathbf{S}\mathbf{V}^\top$

Matrix Completion

- We just saw that PCA gives the optimal low-rank matrix factorization.
- Two ways to generalize this:
  - Consider when $\mathbf{X}$ is only partially observed.
    - E.g., consider a sparse 1000×1000 matrix with 50,000 observations (only 5% observed).
    - A rank 5 approximation requires only 10,000 parameters, so it's reasonable to fit this.
    - Unfortunately, no closed form solution.
  - Impose structure on the factors. We can get lots of interesting models this way.

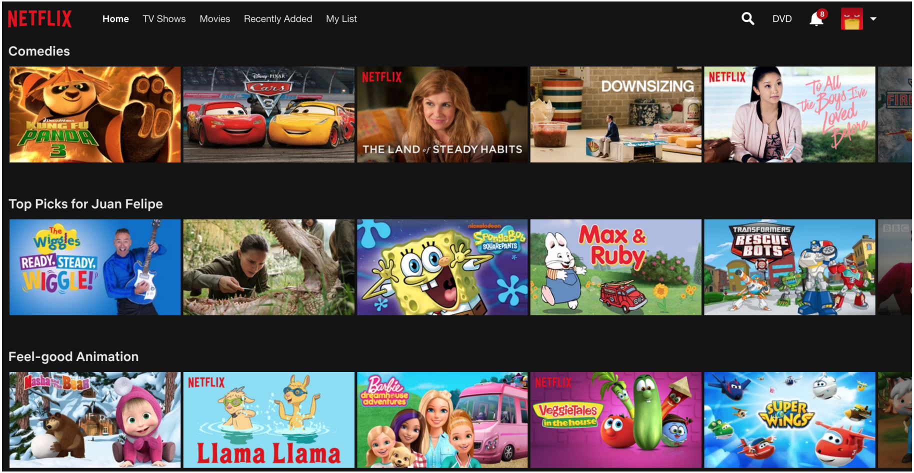
Recommender systems: Why?

- YouTube CA — 400 hours of video are uploaded to YouTube every minute
- amazon.ca — 353 million products and 310 million users
- Spotify — 83 million paying subscribers and streams about 35 million songs

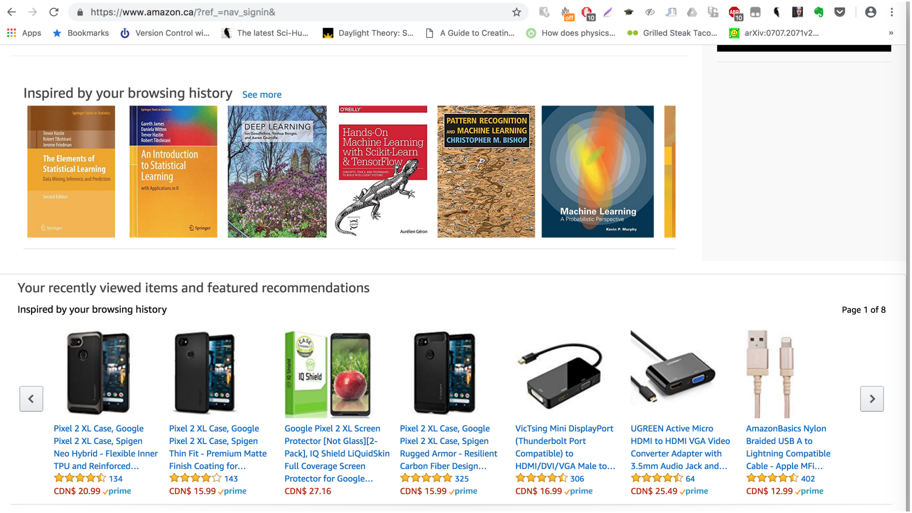
Who cares about all these videos, products and songs? People may care only about a few → Personalization: Connect users with content they may use/enjoy.

Recommender systems suggest items of interest and enjoyment to people based on their preferences

Some recommender systems in action



Some recommender systems in action



Ideally recommendations should combine global and session interests, look at your history if available, should adapt with time, be coherent and diverse, etc.

The Netflix problem

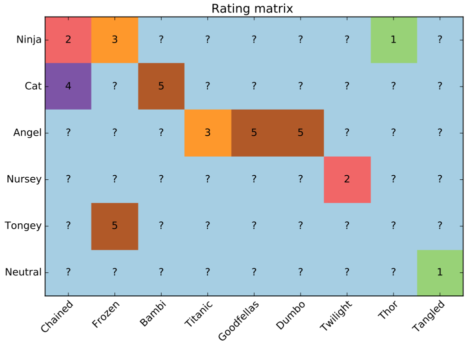
Movie recommendation: Users watch movies and rate them as good or bad.

User	Movie	Rating
	Thor	★ ☆ ☆ ☆ ☆
	Chained	★ ★ ☆ ☆ ☆
	Frozen	★ ★ ★ ☆ ☆
	Chained	★ ★ ★ ★ ☆
	Bambi	★ ★ ★ ★ ★
	Titanic	★ ★ ★ ☆ ☆
	Goodfellas	★ ★ ★ ★ ★
	Dumbo	★ ★ ★ ★ ★
	Twilight	★ ★ ☆ ☆ ☆
	Frozen	★ ★ ★ ★ ★
	Tangled	★ ☆ ☆ ☆ ☆

Because users only rate a few items, one would like to infer their preference for unrated items

Matrix completion problem

Matrix completion problem: Transform the table into a big users by movie matrix.



- Data: Users rate some movies. $\mathbf{R}_{\text{user,movie}}$. Very sparse
- Task: Finding missing data, e.g. for recommending new movies to users. Fill in the question marks
- Algorithms: Alternating Least Square method, Gradient Descent, Non-negative Matrix Factorization, low rank matrix Completion, etc.

Latent factor models

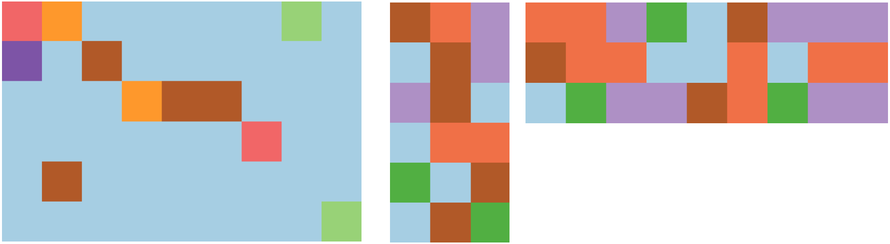
In our current setting, latent factor models attempt to explain the ratings by characterizing both items and users on a number of factors K inferred from the ratings patterns. For simplicity, we can associate these factors with idealized concepts like

- comedy
- drama
- action
- Children
- Quirkiness
- But also uninterpretable dimensions

Can we write down the ratings matrix $\mathbf{R}$ such that these (or similar) latent factors are automatically discovered?

Approach: Matrix factorization methods

$$\mathbf{R} \approx \mathbf{U} \quad \mathbf{Z}$$



Alternating least squares

Assume that the matrix $\mathbf{R}$ is low rank. One can attempt to factorize $\mathbf{R} \approx \mathbf{U}\mathbf{Z}$ in terms of **small** matrices

$$\mathbf{U} = \begin{bmatrix} - & \mathbf{u}_1^\top & - \\ & \vdots & \\ - & \mathbf{u}_D^\top & - \end{bmatrix} \text{ and } \mathbf{Z} = \begin{bmatrix} | & & | \\ \mathbf{z}_1 & \dots & \mathbf{z}_N \\ | & & | \end{bmatrix}$$

Using the squared error loss, a matrix factorization corresponds to solving $\min_{\mathbf{U},\mathbf{Z}} f(\mathbf{U}, \mathbf{Z})$, with $f(\mathbf{U}, \mathbf{Z}) = \frac{1}{2} \sum_{r_{ij} \text{ Observed}} \left(r_{ij} - \mathbf{u}_i^\top \mathbf{z}_j \right)^2$.

The objective is non-convex and in fact it's NP-hard to optimize. (See Low-Rank Matrix Approximation with Weights or Missing Data is NP-hard by Gillis and Glineur, 2011)

As a function of either $\mathbf{U}$ or $\mathbf{Z}$ individually, the problem is convex. But have a chicken-and-egg problem, just like with K-means and mixture models!

Alternating Least Squares (ALS): fix $\mathbf{U}$ and optimize $\mathbf{Z}$, followed by fix $\mathbf{U}$ and optimize $\mathbf{Z}$, and so on until convergence.

ALS for Matrix Completion algorithm

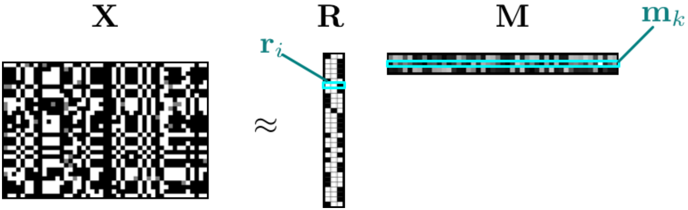
1. Initialize $\mathbf{U}$ and $\mathbf{Z}$ randomly
2. repeat
3. **for** $i = 1, .., D$ **do**
4. $\mathbf{u}_i = \left(\sum_{j:r_{ij}\neq 0} \mathbf{z}_j \mathbf{z}_j^\top\right)^{-1} \sum_{j:r_{ij}\neq 0} r_{ij}\mathbf{z}_j$
5. **for** $j = 1, .., N$ **do**
6. $\mathbf{z}_j = \left(\sum_{i:r_{ij}\neq 0} \mathbf{u}_i \mathbf{u}_i^\top\right)^{-1} \sum_{i:r_{ij}\neq 0} r_{ij}\mathbf{u}_i$
7. until convergence

See also the paper "Probabilistic Matrix Factorization" in the course readings.

More matrix factorizations

K-Means

- It's even possible to view K-means as a matrix factorization!
- Stack the indicator vectors $\mathbf{r}_i$ for assignments into a matrix $\mathbf{R}$, and stack the cluster centers $\boldsymbol{\mu}_k$ into a matrix $\mathbf{M}$.
- "Reconstruction" of the data is given by $\mathbf{RM}$.

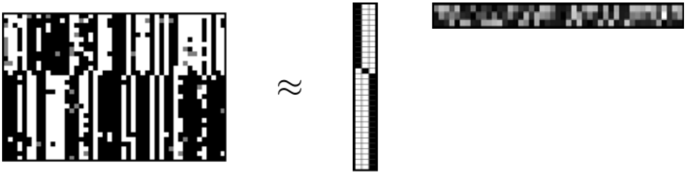


- K-means distortion function in matrix form:

$$\sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2 = ||\mathbf{X} - \mathbf{RM}||_F^2$$

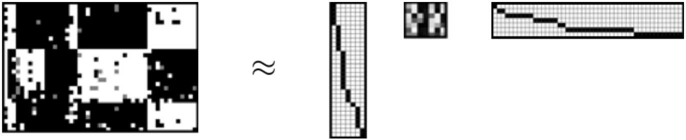
K-Means

- Can sort by cluster for visualization:



Co-clustering (optional)

- We can take this a step further. Co-clustering clusters both the rows and columns of a data matrix, giving a block structure.
- We can represent this as the indicator matrix for rows, times the matrix of means for each block, times the indicator matrix for columns

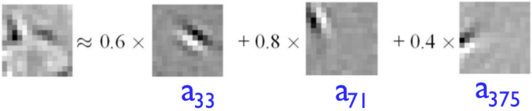


Sparse Coding

- Efficient coding hypothesis: the structure of our visual system is adapted to represent the visual world in an efficient way
  - E.g., be able to represent sensory signals with only a small fraction of neurons having to fire (e.g. to save energy)
- Olshausen and Field fit a sparse coding model to natural images to try to determine what's the most efficient representation.
- They didn't encode anything specific about the brain into their model, but the learned representations bore a striking resemblance to the representations in the primary visual cortex

Sparse Coding

- This algorithm works on small (e.g. 20×20) image patches, which we reshape into vectors (i.e. ignore the spatial structure)
- Suppose we have a dictionary of basis functions $\{\mathbf{a}_k\}_{k=1}^K$ which can be combined to model each patch
- Each patch is approximated as a linear combination of a small number of basis functions
- This is an overcomplete representation, in that typically $K > D$ (e.g. more basis functions than pixels)



$$\mathbf{x} \approx \sum_{k=1}^{K} s_k \mathbf{a}_k = \mathbf{A}\mathbf{s}$$

- Since we use only a few basis functions, $\mathbf{s}$ is a sparse vector.

Sparse Coding

- We'd like choose $\mathbf{s}$ to accurately reconstruct the image, but encourage sparsity in $\mathbf{s}$.
- What cost function should we use?
- Inference in the sparse coding model:

$$\min_{\mathbf{s}} ||\mathbf{x} - \mathbf{A}\mathbf{s}||^2 + \beta||\mathbf{s}||_1$$

- Here, β is a hyperparameter that trades off reconstruction error vs. sparsity.
- There are efficient algorithms for minimizing this cost function (beyond the scope of this class)

Sparse Coding: Learning the Dictionary

- We can learn a dictionary by optimizing both $\mathbf{A}$ and $\{\mathbf{s}_i\}_{i=1}^N$ to trade off reconstruction error and sparsity

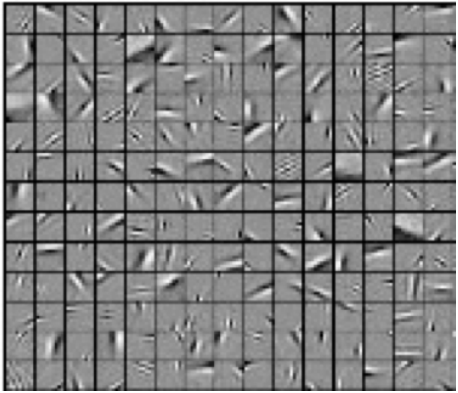
$$\min_{\{\mathbf{s}_i\},\mathbf{A}} \quad \sum_{i=1}^{N} \|\mathbf{x} - \mathbf{A}\mathbf{s}_i\|^2 + \beta\|\mathbf{s}_i\|_1$$

$$\text{subject to} \quad \|\mathbf{a}_k\|^2 \leq 1 \text{ for all } k$$

- Why is the normalization constraint on $\mathbf{a}_k$ needed?
- Reconstruction term can be written in matrix form as $\|\mathbf{X} - \mathbf{A}\mathbf{S}\|_F^2$, where $\mathbf{S}$ combines the $\mathbf{s}_i$.
- Can fit using an alternating minimization scheme over $\mathbf{A}$ and $\mathbf{S}$, just like K-means, EM, low-rank matrix completion, etc.

Sparse Coding: Learning the Dictionary

- Basis functions learned from natural images:

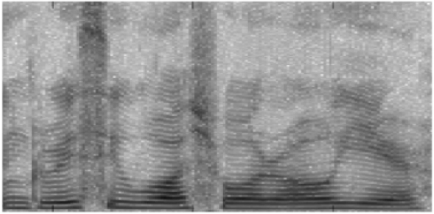


Sparse Coding: Learning the Dictionary

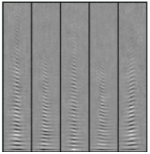
- The sparse components are oriented edges, similar to what a conv net learns
- But the learned dictionary is much more diverse than the first-layer conv net representations: tiles the space of location, frequency, and orientation in an efficient way
- Each basis function has similar response properties to cells in the primary visual cortex (the first stage of visual processing in the brain)

Sparse Coding

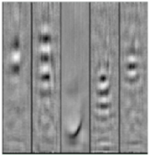
Applying sparse coding to speech signals:



example speech spectrogram (log amplitude)

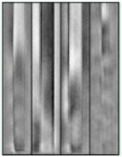


fundamental frequency and overtones



formants

plosives



fricatives

(Grosse et al., 2007, "Shift-invariant sparse coding for audio classification")

Summary

- PCA can be viewed as fitting the optimal low-rank approximation to a data matrix.
- Matrix completion is the setting where the data matrix is only partially observed
  - Solve using ILS, an alternating procedure analogous to EM
- PCA, K-means, co-clustering, sparse coding, and lots of other interesting models can be viewed as matrix factorizations, with different kinds of structure imposed on the factors.