CSC 311: Introduction to Machine Learning

Lecture 10 - Matrix Factorizations & Recommender Systems

Richard Zemel & Murat A. Erdogdu

University of Toronto

Overview

- Recommender systems
- Movie recommendation example
- PCA as a matrix factorization
- Matrix completion task
- Alternating Least Square method (ALS)
- Gradient descent

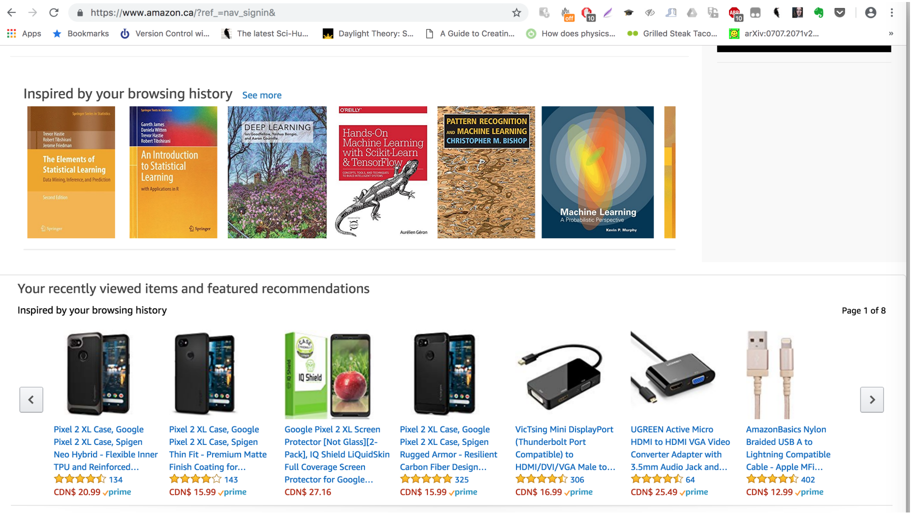
Recommender systems: Why?

- 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.

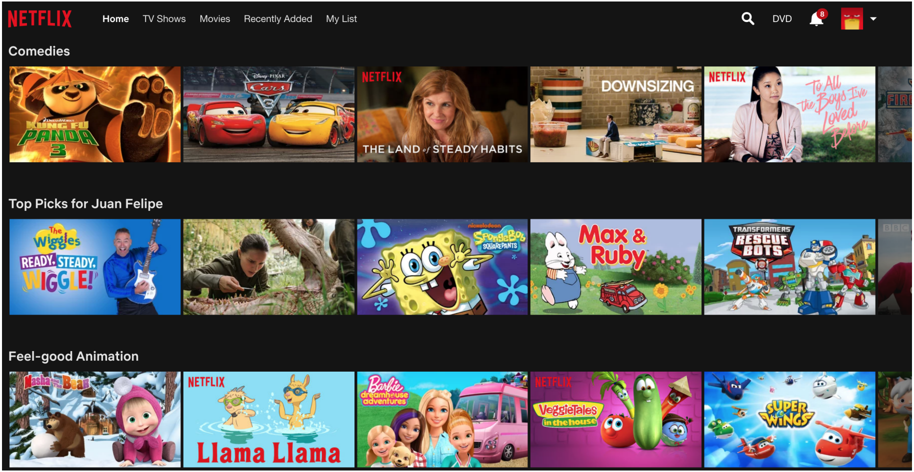
Recommender systems suggest items of interest and enjoyment to people based on their preferences

Some recommender systems in action



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

Some recommender systems in action



The Netflix problem

Movie recommendation: Users watch movies and rate them out of 5★.

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

Netflix Prize

PCA as a Matrix Factorization

- 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 lets 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 PCA: each input vector $\mathbf{x}^{(i)} \in \mathbb{R}^D$ is approximated as $\hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{z}^{(i)}$,

$$\mathbf{x}^{(i)} \approx \tilde{\mathbf{x}}^{(i)} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{z}^{(i)}$$

  where $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_i \mathbf{x}^{(i)}$ is the data mean, $\mathbf{U} \in \mathbb{R}^{D\times K}$ is the orthogonal basis for the principal subspace, and $\mathbf{z}^{(i)} \in \mathbb{R}^K$ is the code vector, and $\tilde{\mathbf{x}}^{(i)} \in \mathbb{R}^D$ is $\mathbf{x}^{(i)}$'s reconstruction or approximation.
- Assume that the data is centered: $\hat{\boldsymbol{\mu}} = 0$. Then, the approximation looks like

$$\mathbf{x}^{(i)} \approx \tilde{\mathbf{x}}^{(i)} = \mathbf{U}\mathbf{z}^{(i)}.$$

PCA as Matrix Factorization

- PCA(on centered data): input vector $\mathbf{x}^{(i)}$ is approximated as $\mathbf{U}\mathbf{z}^{(i)}$

$$\mathbf{x}^{(i)} \approx \mathbf{U}\mathbf{z}^{(i)}$$

- Write this in matrix form, we have $\mathbf{X} \approx \mathbf{Z}\mathbf{U}^\top$ where $\mathbf{X}$ and $\mathbf{Z}$ are matrices with one *row* per data point

$$\mathbf{X} = \begin{bmatrix} [\mathbf{x}^{(1)}]^\top \\ [\mathbf{x}^{(2)}]^\top \\ \vdots \\ [\mathbf{x}^{(N)}]^\top \end{bmatrix} \in \mathbb{R}^{N\times D} \quad \text{and} \quad \mathbf{Z} = \begin{bmatrix} [\mathbf{z}^{(1)}]^\top \\ [\mathbf{z}^{(2)}]^\top \\ \vdots \\ [\mathbf{z}^{(N)}]^\top \end{bmatrix} \in \mathbb{R}^{N\times K}$$

- How to enforce $\mathbf{X} \approx \mathbf{Z}\mathbf{U}^\top$ or measure difference between them?
- Recall that the Frobenius norm of a matrix $\mathbf{Y}$ is defined as

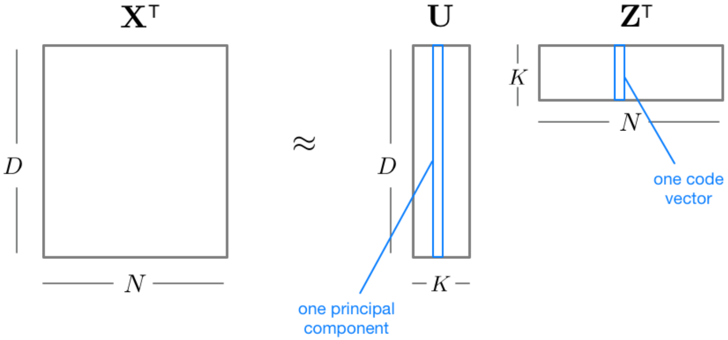
$$\|\mathbf{Y}\|_F^2 = \|\mathbf{Y}^\top\|_F^2 = \sum_{i,j} y_{ij}^2 = \sum_i \|\mathbf{y}^{(i)}\|^2.$$

- Writing the squared error in matrix form

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

PCA as Matrix Factorization

- So PCA is approximating $\mathbf{X} \approx \mathbf{Z}\mathbf{U}^\top$, or equivalently $\mathbf{X}^\top \approx \mathbf{U}\mathbf{Z}^\top$.



- 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 error $\|\mathbf{X}^\top - \mathbf{U}\mathbf{Z}^\top\|_F^2$.

Singular-Value Decomposition (SVD)

This has a close relationship to the Singular Value Decomposition (SVD) of $\mathbf{X}$ which is a matrix factorization technique. Consider an $N \times D$ matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ with SVD

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

Properties:

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

Note that standard SVD notation is $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$. We are using $\mathbf{X} = \mathbf{Q}\mathbf{S}\mathbf{U}^\top$ for notational convenience.

Singular-Value Decomposition (SVD) continued

Properties of covariance matrices:

- Construct two positive semi-definite matrices $\mathbf{X}\mathbf{X}^\top$ ($N \times N$) and $\mathbf{X}^\top\mathbf{X}$ ($D \times D$).
- $\mathbf{X}\mathbf{X}^\top = \mathbf{Q}\mathbf{S}\mathbf{U}^\top(\mathbf{Q}\mathbf{S}\mathbf{U}^\top)^\top = \mathbf{Q}\mathbf{S}\mathbf{U}^\top\mathbf{U}\mathbf{S}\mathbf{Q}^\top = \mathbf{Q}\mathbf{S}^2\mathbf{Q}^\top$ is an eigendecomposition of $\mathbf{X}\mathbf{X}^\top$
- Similarly, $\mathbf{X}^\top\mathbf{X} = \mathbf{U}\mathbf{S}^2\mathbf{U}^\top$ is eigendecomposition of $\mathbf{X}^\top\mathbf{X}$.
- Assuming $N \geq D$, it can be shown that that $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}^\top\mathbf{X}$ will share D eigenvalues and the remaining $N - D$ eigenvalues of $\mathbf{X}\mathbf{X}^\top$ will be zero.

Motivation for matrix factorization: recall PCA

- Recall: The optimal PCA subspace is spanned by the top K eigenvectors of the covariance matrix

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^{\top}.$$

- When the data is centered ($\hat{\boldsymbol{\mu}} = 0$), this is equivalent to

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^{(i)} [\mathbf{x}^{(i)}]^{\top} = \frac{1}{N} \mathbf{X}^{\top} \mathbf{X}.$$

- You can center the data by subtracting the mean from each input vector:

$$\mathbf{x}^{(i)} \leftarrow \mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}} \quad \text{for} \quad i = 1, \ldots, N.$$

Motivation for matrix factorization: recall PCA

- Recall the spectral (or eigen-) decomposition of $\hat{\mathbf{\Sigma}}$

$$\hat{\mathbf{\Sigma}} = \frac{1}{N}\mathbf{X}^\top\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$$

- SVD of the data matrix $\mathbf{X} = \mathbf{Q}\mathbf{S}\mathbf{U}^\top = \mathbf{Z}\mathbf{U}^\top$, with $\mathbf{Z} = \mathbf{Q}\mathbf{S}$.
- The eigendecomposition of $\hat{\mathbf{\Sigma}}$ follows directly from the eigendecomposition of $\mathbf{X}^\top\mathbf{X}$:

$$\frac{1}{N}\mathbf{X}^\top\mathbf{X} = \frac{1}{N}\mathbf{U}\mathbf{S}\mathbf{Q}^\top\mathbf{Q}\mathbf{S}\mathbf{U}^\top = \mathbf{U}[\mathbf{S}^2/N]\mathbf{U}^\top.$$

- From $\mathbf{X}^\top\mathbf{X}/N$ it follows that the eigenvales λ_i's are related to the singular values $\lambda_i = \frac{1}{N}s_i^2$.
- The SVD gives $\mathbf{U}$ which is equivalent to the learned basis of PCA.
  - First K principal components corresponds first K columns of $\mathbf{U}$, i.e., $\mathbf{U}[:, :K]$ in python notation.
  - PCA reduces the dimension of data D to K. Low-dimensional representation is given by the first K columns of $\mathbf{Z}[:, :K]$. Rows of this matrix are the K-dimensional code vectors.

PCA as a matrix factorization

Ultimately, PCA with K principal components finds the *optimal* rank-K approximation of $\mathbf{X} \in \mathbb{R}^{N \times D}$, in terms of error $||\mathbf{X}^\top - \mathbf{U}\mathbf{Z}^\top||_F^2$.

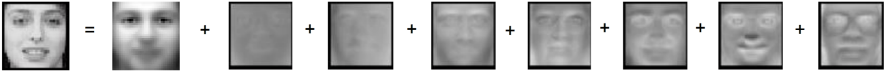
$$\min ||\mathbf{X}^\top - \mathbf{U}\mathbf{Z}^\top||_F^2 \quad \text{over} \quad \mathbf{Z} \in \mathbb{R}^{N \times K}, \mathbf{U} \in \mathbb{R}^{D \times K}.$$

Note that the case $K = D$ corresponds to the entire SVD of $\mathbf{X}$.

- Can we do something similar for recommender systems?

PCA as matrix factorization of $\mathbf{X}$

We have established that SVD provided a matrix factorization which we can interpret as a PCA. Recall



$$\bar{\mathbf{x}} = \mu + z_1\mathbf{u_1} + z_2\mathbf{u_2} + z_3\mathbf{u_3} + \dots$$

where the vectors $\mathbf{u_i}$ are the principal components of the data matrix $\mathbf{X}$ (the latent factors).
We can do the same for our ratings matrix $\mathbf{R}$. Rating of movie

 = **average user**+z_1**comedy user**+z_2**drama user**+z_3**action user**+. .

These latent factors are idealized, the real latent factors do not necessarily reveal these semantic concepts so clearly.

Matrix Completion

- We just saw that PCA gives the optimal low-rank matrix factorization.
- Two ways to generalize this:
  - 1) Consider when $\mathbf{X}$ is only partially observed.
    - 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.
  - 2) Impose structure on the factors. We can get lots of interesting models this way.

The Netflix problem

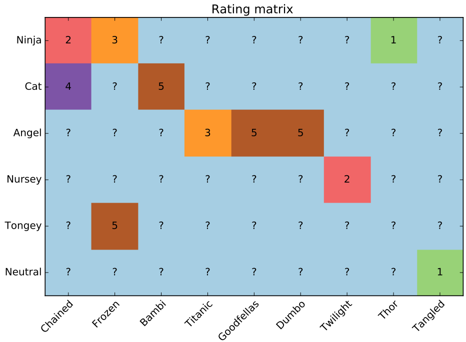
Movie recommendation: Users watch movies and rate them as good or bad.

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 N users by M movies matrix $\mathbf{R}$



- 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 movies and users on a number of factors K inferred from the ratings patterns.
- That is, we seek representations for movies and users as vectors in $\mathbb{R}^K$ that can ultimately be translated to ratings.
- For simplicity, we can associate these factors (i.e. the dimensions of the vectors) with idealized concepts like
  - comedy
  - drama
  - action
  - But also uninterpretable dimensions

Can we use the sparse ratings matrix $\mathbf{R}$ to find these latent factors automatically?

Alternating least squares

- Let the representation of user n in the K-dimensional space be $\mathbf{u}_n$ and the representation of movie m be $\mathbf{z}_m$
- Assume the rating user n gives to movie m is given by a dot product: $R_{nm} \approx \mathbf{u}_n^T \mathbf{z}_m$
- In matrix form, if:

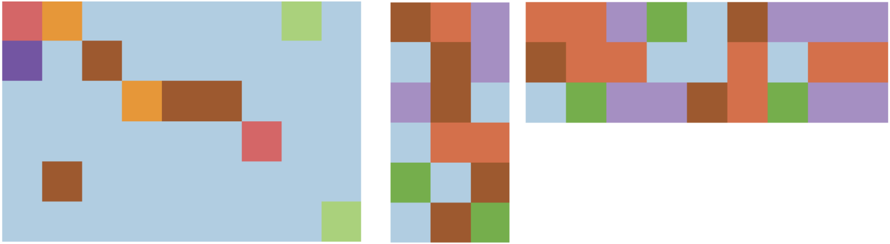
$$\mathbf{U} = \begin{bmatrix} — & \mathbf{u}_1^\top & — \\ & \vdots & \\ — & \mathbf{u}_N^\top & — \end{bmatrix} \text{ and } \mathbf{Z}^\top = \begin{bmatrix} | & & | \\ \mathbf{z}_1 & \dots & \mathbf{z}_M \\ | & & | \end{bmatrix}$$

  then: $\mathbf{R} \approx \mathbf{U}\mathbf{Z}^\top$

- This is a matrix factorization problem!

Approach: Matrix factorization methods

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



Cost for Matrix Factorization for Recommender Systems

- Recall PCA: To enforce $\mathbf{X}^\top \approx \mathbf{U}\mathbf{Z}^\top$, we minimized

$$\min_{\mathbf{U},\mathbf{Z}} \|\mathbf{X}^\top - \mathbf{U}\mathbf{Z}^\top\|_{\mathrm{F}}^2 = \sum_{i,j} (x_{ji} - \mathbf{u}_i^\top \mathbf{z}_j)^2$$

  where $\mathbf{u}_i$ and $\mathbf{z}_i$ are the i-th rows of matrices $\mathbf{U}$ and $\mathbf{Z}$, respectively.

- How do we enforce $\mathbf{R} \approx \mathbf{U}\mathbf{Z}^\top$
  - Try

$$\min_{\mathbf{U},\mathbf{Z}} \sum_{i,j} (R_{ij} - \mathbf{u}_i^\top \mathbf{z}_j)^2$$

  - Most entries of $\mathbf{R}$ are missing!

Alternating least squares

- Let $O = \{(n, m) :$ entry (n, m) of matrix $\mathbf{R}$ is observed$\}$
- Using the squared error loss, a matrix factorization corresponds to solving

$$\min_{\mathbf{U},\mathbf{Z}} \frac{1}{2} \sum_{(n,m)\in O} \left(R_{nm} - \mathbf{u}_n^\top \mathbf{z}_m \right)^2$$

- The objective is non-convex in $\mathbf{U}$ and $\mathbf{Z}$ and in fact it's generally NP-hard to minimize the above cost function.
- As a function of either $\mathbf{U}$ or $\mathbf{Z}$ individually, the problem is convex and easy to optimize. We can use coordinate descent, just like with K-means and mixture models!

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

Alternating least squares

ALS for Matrix Completion algorithm

1. Initialize $\mathbf{U}$ and $\mathbf{Z}$ randomly
2. repeat until convergence
3. $\quad$ **for** $n = 1, .., N$ **do**
4. $\qquad \mathbf{u}_n = \left(\sum_{m:(n,m)\in O} \mathbf{z}_m \mathbf{z}_m^\top\right)^{-1} \sum_{m:(n,m)\in O} R_{nm}\mathbf{z}_m$
5. $\quad$ **for** $m = 1, .., M$ **do**
6. $\qquad \mathbf{z}_m = \left(\sum_{n:(n,m)\in O} \mathbf{u}_n \mathbf{u}_n^\top\right)^{-1} \sum_{n:(n,m)\in O} R_{nm}\mathbf{u}_n$

Gradient descent method

- We can also do full gradient descent for matrix completion.
- Minimize $f(\mathbf{U}, \mathbf{Z})$ with GD. Both $\mathbf{U}, \mathbf{Z}$ are variables. Gradient descent step:

$$\begin{bmatrix} \mathbf{U} \\ \mathbf{Z} \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{U} \\ \mathbf{Z} \end{bmatrix} - \alpha \nabla f(\mathbf{U}, \mathbf{Z}) \tag{1}$$

- Computation of the gradient term per iteration is expensive if all the index pairs in the ratings matrix are considered and $\mathbf{R}$ is large (e.g. Netflix).

Stochastic gradient descent method

Stochastic gradient descent for matrix completion (recall SGD from lecture 8). Attempt to minimize $f(\mathbf{U}, \mathbf{Z}) = \frac{1}{2} \sum_{(n,m)\in O} \left(R_{nm} - \mathbf{u}_n^\top \mathbf{z}_m \right)^2$. For a randomly chosen observed pair (n, m) in $\mathbf{R}$, the SGD update:

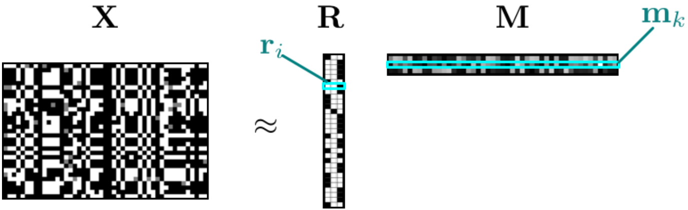
$$\begin{bmatrix} \mathbf{u}_n \\ \mathbf{z}_m \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{u}_n \\ \mathbf{z}_m \end{bmatrix} - \alpha \begin{bmatrix} \left(R_{nm} - \mathbf{u}_n^\top \mathbf{z}_m \right) \mathbf{z}_m \\ \left(R_{nm} - \mathbf{u}_n^\top \mathbf{z}_m \right) \mathbf{u}_n \end{bmatrix} \tag{2}$$

Algorithm:

1. Initialize $\mathbf{U}$ and $\mathbf{Z}$
2. repeat until "convergence"
3. Randomly select a pair $(n, m) \in O$ among observed elements of $\mathbf{R}$
4. $\mathbf{u}_n \leftarrow \mathbf{u}_n - \alpha \left(R_{nm} - \mathbf{u}_n^\top \mathbf{z}_m \right) \mathbf{z}_m$
5. $\mathbf{z}_m \leftarrow \mathbf{z}_m - \alpha \left(R_{nm} - \mathbf{u}_n^\top \mathbf{z}_m \right) \mathbf{u}_n$

K-Means

- It's possible to view K-means as a matrix factorization.
- Stack 1-of-K vectors $\mathbf{r}_i$ for assignments into a $N \times K$ matrix $\mathbf{R}$, and stack the cluster centers $\mathbf{m}_k$ into a matrix $K \times D$ matrix $\mathbf{M}$.
- "Reconstruction" of the data (replace each point with its cluster center) is given by $\mathbf{RM}$.

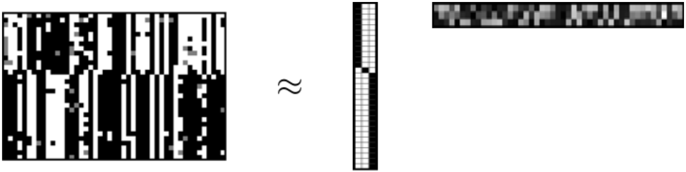


- K-means distortion function in matrix form:

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

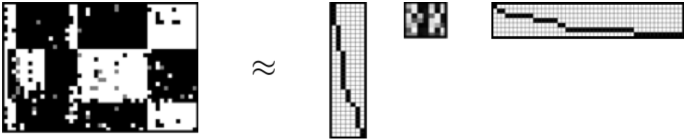
K-Means

- Can sort by cluster for visualization:



Co-clustering

- We can take this a step further.
- Idea: feature dimensions can be redundant, and some feature dimensions cluster together.
- 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

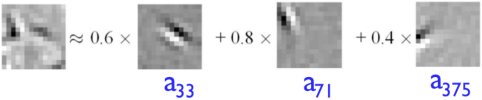
Sparse Coding

- 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:

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

- This is an overcomplete representation, in that typically $K > D$ for sparse coding problems (e.g. more basis functions than pixels)
- The requirement that $\mathbf{s}$ is sparse makes things interesting

Sparse Coding



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

Since we use only a few basis functions, **s** is a sparse vector.

Sparse Coding

- We'd like choose $\mathbf{s}$ to accurately reconstruct the image, $\mathbf{x} \approx \mathbf{A}\mathbf{s}$ 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

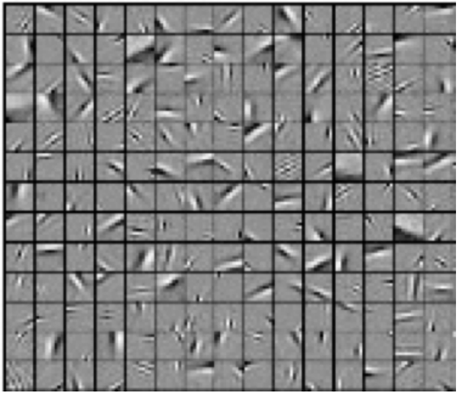
$$\min_{\{\mathbf{s}_i\},\mathbf{A}} \quad \sum_{i=1}^{N} \left\|\mathbf{x}^{(i)} - \mathbf{A}\mathbf{s}_i\right\|^2 + \beta\|\mathbf{s}_i\|_1$$

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

- 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 $\mathbf{s}_i$ as columns
- 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:

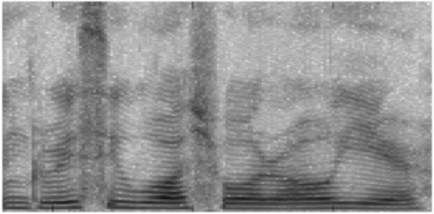


Sparse Coding: Learning the Dictionary

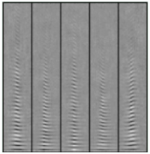
- The sparse components are oriented edges, similar to what a neural networks learn
- But the learned dictionary is much more diverse than the first-layer neural net representations: tiles the space of location, frequency, and orientation in an efficient way

Sparse Coding

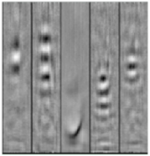
Applying sparse coding to speech signals:



example speech spectrogram (log amplitude)

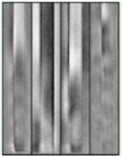


fundamental frequency and overtones



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 ALS, 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.