CSC 411: Introduction to Machine Learning

CSC 411 Lecture 18: Matrix Factorizations

Mengye Ren and Matthew MacKay

University of Toronto

Overview

- Recall PCA: project data onto a low-dimensional subspace defined by the top eigenvalues of the data covariance

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

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

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

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.

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
  - We transpose our usual convention for data matrices (for some parts of this lecture).

PCA as Matrix Factorization

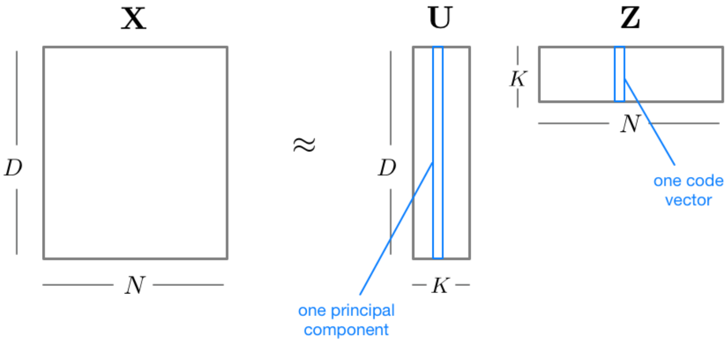
- Recall: each input vector $\mathbf{x}^{(i)}$ is approximated as $\mathbf{Uz}$, where $\mathbf{U}$ is the orthogonal basis for the principal subspace, and $\mathbf{z}$ is the code vector.
- Write this in matrix form: $\mathbf{X}$ and $\mathbf{Z}$ are matrices with one *column* per data point
  - We transpose our usual convention for data matrices (for some parts of this lecture).
- 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 $\|\mathbf{A}\|_F^2 = \sum_{i,j} a_{ij}^2$.

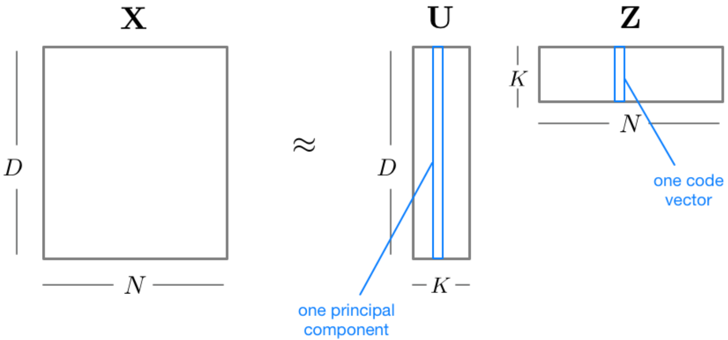
PCA as Matrix Factorization

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



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

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

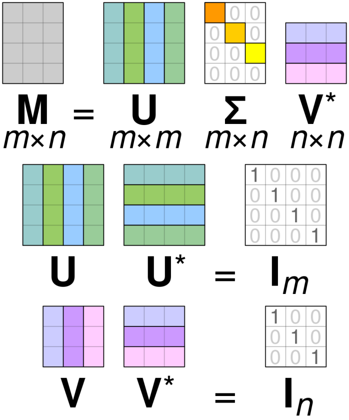
Properties:

- $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}^\top$ provide a real-valued matrix factorization of $\mathbf{X}$, an $m \times n$ matrix.
- $\mathbf{U}$ is a $m \times m$ matrix with orthonormal columns $\mathbf{U}^\top\mathbf{U} = I_m$, where I_m is the $m \times m$ identity matrix.
- $\mathbf{V}$ is an orthonormal $n \times n$ matrix, $\mathbf{V}^\top\mathbf{V} = I_n$.
- $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix, with non-negative singular values, $\sigma_1, \sigma_2, \ldots, \sigma_{\min\{m,n\}}$, on the diagonal, where the singular values are conventionally ordered from largest to smallest.

It's possible to show that the first n singular vectors correspond to the first n principal components; more precisely, $\mathbf{Z} = \mathbf{U}\mathbf{\Sigma}$

PCA vs. SVD (optional)

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



Matrix Completion

- We just saw that PCA gives the optimal low-rank matrix factorization.
- Two ways to generalize this:

Matrix Completion

- We just saw that PCA gives the optimal low-rank matrix factorization.
- Two ways to generalize this:
  - 1) Consider when **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.

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.

Recommender systems: Why?

- YouTube CA 400 hours of video are uploaded to YouTube every minute

Recommender systems: Why?

- 400 hours of video are uploaded to YouTube every minute
- 353 million products and 310 million users

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

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.

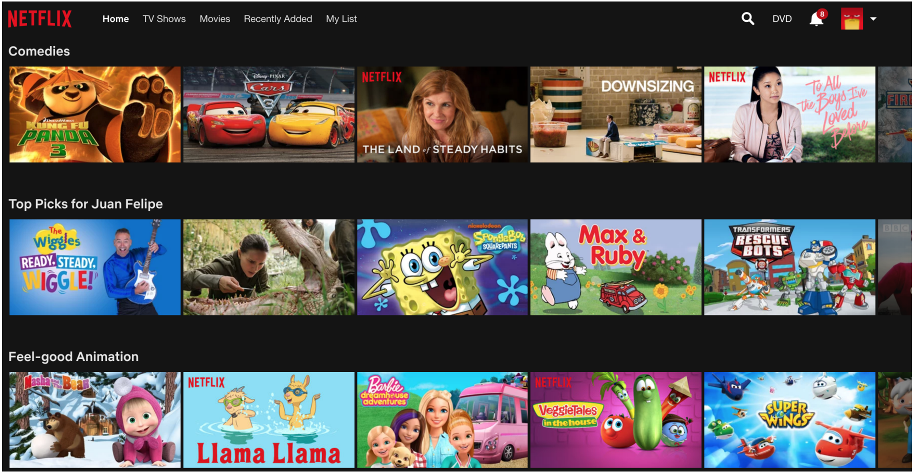
Recommender systems: Why?

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

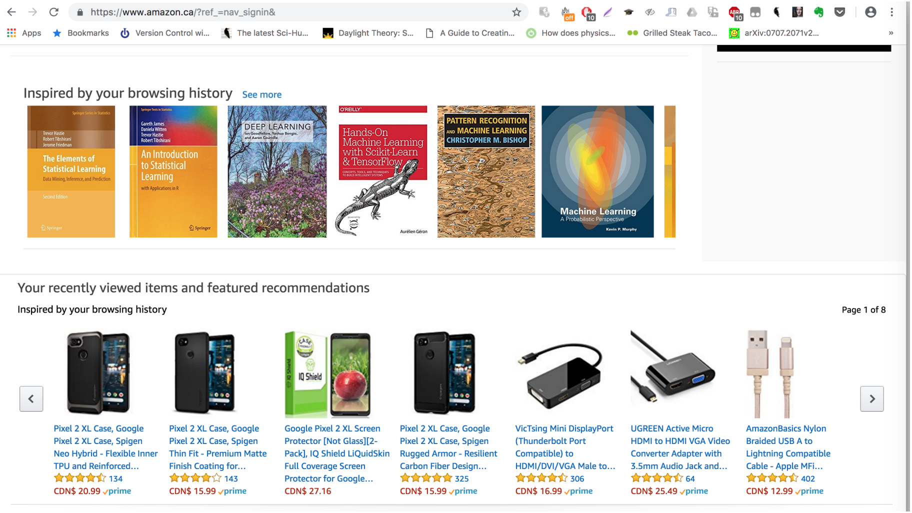
Who cares about all these videos, products and songs? People may care only about a few → **Personalization:** Connect users with content they may use/enjoy.

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

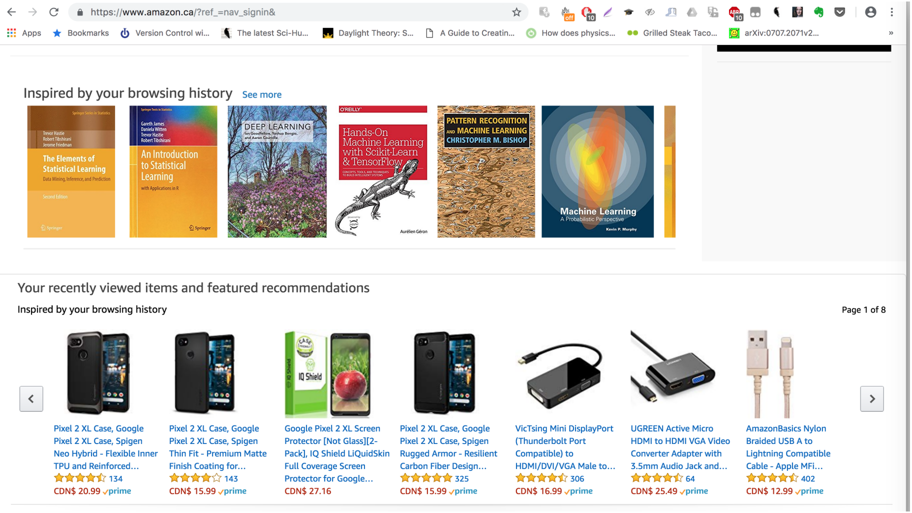
Some recommender systems in action



Some recommender systems in action



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	★ ☆ ☆ ☆ ☆

The Netflix problem

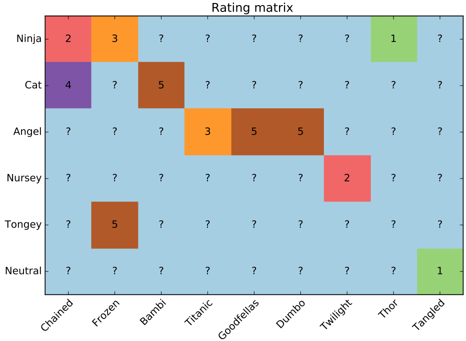
Movie recommendation: Users watch movies and rate them as good or bad.

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 a representation movies and users as vectors in $\mathbb{R}^K$
- 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

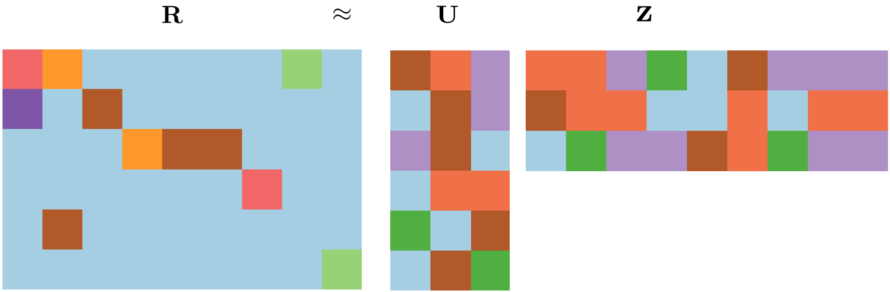
- Let the representation of user n in the K-dimensional space be $\mathbf{u}_n$ and the representation of movie m be $\mathbf{z}_m$
- Assume the rating user n gives to movie m is given by a dot product: $R_{nm} \approx \mathbf{u}_n^T \mathbf{z}_m$
- In matrix form, if:

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

  then: $\mathbf{R} \approx \mathbf{UZ}$

- This is a matrix factorization problem!

Approach: Matrix factorization methods



Alternating least squares

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

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

Alternating least squares

- Let $O = \{(n, m) : \text{ entry } (n, m) \text{ of matrix } R \text{ is observed}\}$
- 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 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)

Alternating least squares

- Let $O = \{(n,m):\ \text{entry } (n,m) \text{ of matrix } R \text{ is observed}\}$
- 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 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 **U** or **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

- Let $O = \{(n, m) : \text{ entry } (n, m) \text{ of matrix } R \text{ is observed}\}$
- 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 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 **U** or **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 **Z** and optimize **U**, followed by fix **U** and optimize **Z**, and so on until convergence.

Alternating least squares

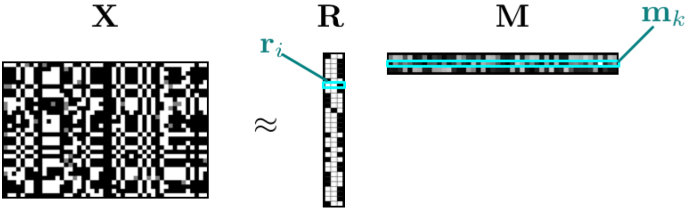
ALS for Matrix Completion algorithm

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

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

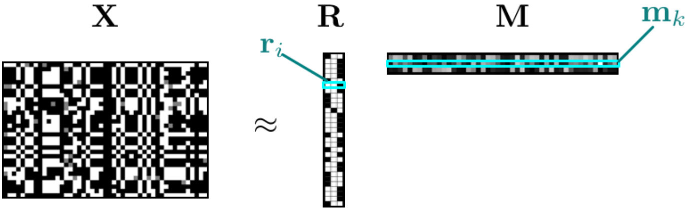
K-Means

- It's possible to view K-means as a matrix factorization.
- Stack the indicator 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

- It's possible to view K-means as a matrix factorization.
- Stack the indicator 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 $\mathbf{RM}$.

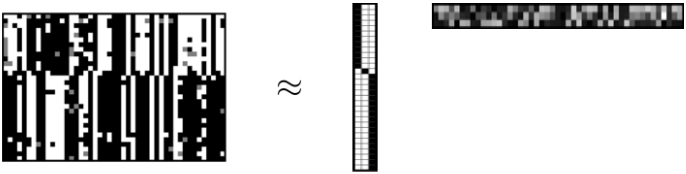


- K-means distortion function in matrix form:

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

K-Means

- Can sort by cluster for visualization:

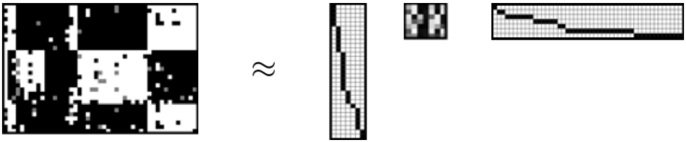


Co-clustering (optional)

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

Co-clustering (optional)

- 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)

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

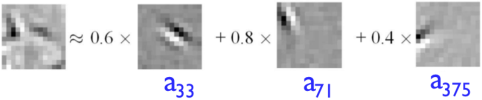
Sparse Coding

- 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:

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

- This is an **overcomplete** representation, in that typically $K > D$ (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 **s** to accurately reconstruct the image, but encourage sparsity in **s**.
- What cost function should we use?

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

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

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

- Why is the normalization constraint on $\mathbf{a}_k$ needed?

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

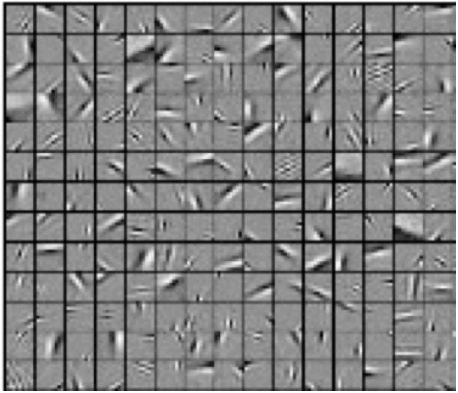
$$\min_{\{\mathbf{s}_i\},\mathbf{A}} \quad \sum_{i=1}^{N} \left\|\mathbf{x} - \mathbf{A}\mathbf{s}_i\right\|^2 + \beta\|\mathbf{s}_i\|_1$$

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

- Why is the normalization constraint on $\mathbf{a}_k$ needed?
- Reconstruction term can be written in matrix form as $\|\mathbf{X} - \mathbf{AS}\|_F^2$, 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

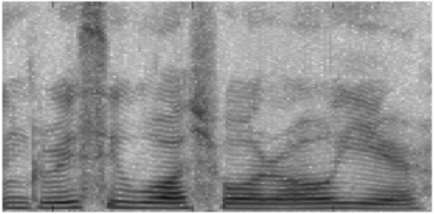
- Basis functions learned from natural images:



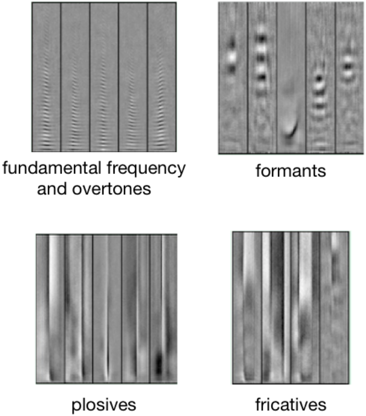
Sparse Coding: Learning the Dictionary

- The sparse components are oriented edges, similar to what a 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)

Applying sparse coding to speech signals:



example speech spectrogram (log amplitude)



(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.