

CSC 411: Introduction to Machine Learning

CSC 411 Lecture 18: Matrix Factorizations

Mengye Ren and Matthew MacKay

University of Toronto

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

- 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

- 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

- 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

- 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 $\mathbf{U}\mathbf{z}$, 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

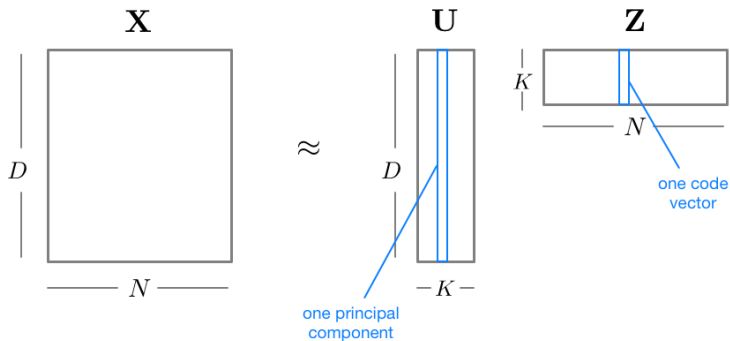
- Recall: each input vector $\mathbf{x}^{(i)}$ is approximated as $\mathbf{U}\mathbf{z}$, where \mathbf{U} is the orthogonal basis for the principal subspace, and \mathbf{z} is the code vector.
- Write this in matrix form: \mathbf{X} and \mathbf{Z} are matrices with one *column* per data point
 - We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

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

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

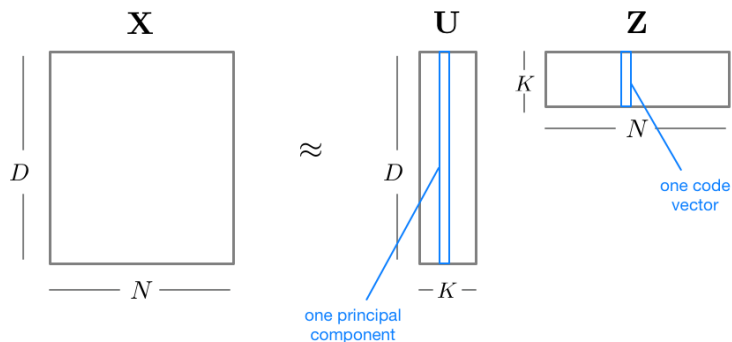
PCA as Matrix Factorization

- 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

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

Properties:

- \mathbf{U} , $\mathbf{\Sigma}$, and \mathbf{V}^T 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 $\mathbf{U}^T \mathbf{U} = \mathbf{I}_m$, where \mathbf{I}_m is the $m \times m$ identity matrix.
- \mathbf{V} is an orthonormal $n \times n$ matrix, $\mathbf{V}^T \mathbf{V} = \mathbf{I}_n$.
- $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix, with non-negative singular values, $\sigma_1, \sigma_2, \dots, \sigma_{\min\{m,n\}}$, 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)

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

$$\begin{matrix}
 \mathbf{M} & = & \mathbf{U} & \mathbf{\Sigma} & \mathbf{V}^* \\
 m \times n & & m \times m & m \times n & n \times n
 \end{matrix}$$

$$\mathbf{U} \quad \mathbf{U}^* = \mathbf{I}_m$$

$$\mathbf{V} \quad \mathbf{V}^* = \mathbf{I}_n$$

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 \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.



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?

-  YouTube^{CA} 400 hours of video are uploaded to YouTube every minute
-  amazon_{.ca} 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?

-  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.

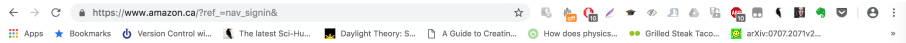
Recommender systems suggest items of interest and enjoyment to people based on their preferences

Some recommender systems in action

The screenshot displays the Netflix interface with the following sections:

- NETFLIX** navigation bar: Home, TV Shows, Movies, Recently Added, My List. Search, DVD, and notification icons are also present.
- Comedies** section: A row of movie posters including *Kung Fu Panda 3*, *Cars 3*, *The Land of Steady Habits*, *Downsizing*, and *To All the Boys I've Loved Before*.
- Top Picks for Juan Felipe** section: A row of personalized recommendations including *The Wiggles: Ready, Steady, Wiggle!*, a nature documentary, *SpongeBob SquarePants*, *Max & Ruby*, and *Transformers: Rescue Bots*.
- Feel-good Animation** section: A row of animated titles including *Masha and the Bear*, *Llama Llama*, *Barbie: The Princess Adventure*, *VeggieTales in the House*, and *Super Wings*.

Some recommender systems in action





Inspired by your browsing history [See more](#)



Your recently viewed items and featured recommendations

Inspired by your browsing history

Page 1 of 8

						
<p>Pixel 2 XL Case, Google Pixel 2 XL Case, Spigen Neo Hybrid - Flexible Inner TPU and Reinforced...</p> <p>★★★★☆ 134 CDN\$ 20.99 ✓prime</p>	<p>Pixel 2 XL Case, Google Pixel 2 XL Case, Spigen Thin Fit - Premium Matte Finish Coating for...</p> <p>★★★★☆ 143 CDN\$ 15.99 ✓prime</p>	<p>Google Pixel 2 XL Screen Protector [Not Glass][2- Pack], IQ Shield LiQuidSkin Full Coverage Screen Protector for Google...</p> <p>★★★★☆ 27.16</p>	<p>Pixel 2 XL Case, Google Pixel 2 XL Case, Spigen Rugged Armor - Resilient Carbon Fiber Design...</p> <p>★★★★☆ 325 CDN\$ 15.99 ✓prime</p>	<p>VicTsing Mini DisplayPort (Thunderbolt Port Compatible) to HDMI/DVI/VGA Male to...</p> <p>★★★★☆ 306 CDN\$ 16.99 ✓prime</p>	<p>UGREEN Active Micro HDMI to HDMI VGA Video Converter Adapter with 3.5mm Audio Jack and...</p> <p>★★★★☆ 64 CDN\$ 25.49 ✓prime</p>	<p>AmazonBasics Nylon Braided USB A to Lightning Compatible Cable - Apple MFL...</p> <p>★★★★☆ 402 CDN\$ 12.99 ✓prime</p>

Some recommender systems in action

The screenshot shows a web browser at https://www.amazon.ca/?ref_=nav_signin&. The top navigation bar includes 'Inspired by your browsing history' with a 'See more' link. Below this are six book covers: 'The Elements of Statistical Learning' (2nd Edition), 'An Introduction to Statistical Learning with Applications in R', 'DEEP LEARNING: An Introduction to Artificial Neural Networks, Deep Generative Models, and Advanced Computer Vision, Audio Processing, and Speech Recognition', 'Hands-On Machine Learning with Scikit-Learn & TensorFlow', 'PATTERN RECOGNITION AND MACHINE LEARNING' by Christopher M. Bishop, and 'Machine Learning: A Probabilistic Perspective' by Kevin P. Murphy.

The main section is titled 'Your recently viewed items and featured recommendations' and 'Inspired by your browsing history'. It contains a carousel of seven products with navigation arrows on either side. The products are:












- Pixel 2 XL Case, Google Pixel 2 XL Case, Spigen Neo Hybrid - Flexible Inner TPU and Reinforced...**
★★★★★ 134
CDN\$ 20.99 ✓prime
- Pixel 2 XL Case, Google Pixel 2 XL Case, Spigen Thin Fit - Premium Matte Finish Coating for...**
★★★★☆ 143
CDN\$ 15.99 ✓prime
- Google Pixel 2 XL Screen Protector [Not Glass][2-Pack], IQ Shield LiQuidSkin Full Coverage Screen Protector for Google...**
CDN\$ 27.16
- Pixel 2 XL Case, Google Pixel 2 XL Case, Spigen Rugged Armor - Resilient Carbon Fiber Design...**
★★★★★ 325
CDN\$ 15.99 ✓prime
- VicTsing Mini DisplayPort (Thunderbolt Port Compatible) to HDMI/DVI/VGA Male to...**
★★★★★ 306
CDN\$ 16.99 ✓prime
- UGREEN Active Micro HDMI to HDMI VGA Video Converter Adapter with 3.5mm Audio Jack and...**
★★★★★ 64
CDN\$ 25.49 ✓prime
- AmazonBasics Nylon Braided USB A to Lightning Compatible Cable - Apple MFL...**
★★★★★ 402
CDN\$ 12.99 ✓prime

Page 1 of 8

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}

Rating matrix

Ninja	2	3	?	?	?	?	?	1	?
Cat	4	?	5	?	?	?	?	?	?
Angel	?	?	?	3	5	5	?	?	?
Nursey	?	?	?	?	?	?	2	?	?
Tongey	?	5	?	?	?	?	?	?	?
Neutral	?	?	?	?	?	?	?	?	1
Chained	Frozen	Bambi	Titanic	Goodfellas	Dumbo	Twilight	Thor	Tangled	

- **Data:** Users rate some movies. $\mathbf{R}_{\text{user},\text{movie}}$ 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

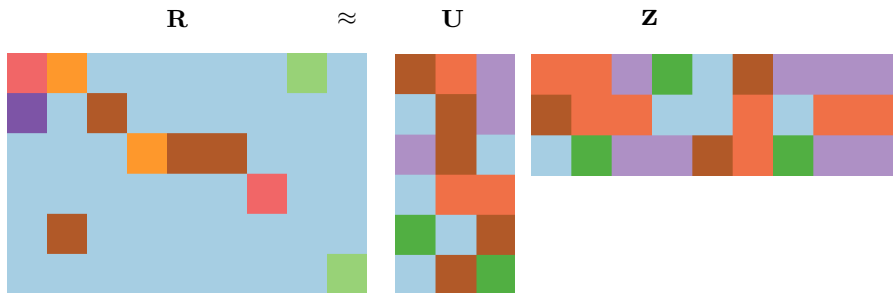
- Let the representation of user n in the K -dimensional space be \mathbf{u}_n and the representation of movie m be \mathbf{z}_m
- Assume the rating user n gives to movie m is given by a dot product:
 $R_{nm} \approx \mathbf{u}_n^T \mathbf{z}_m$
- In matrix form, if:

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

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

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

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

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

- 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

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

- 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 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

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

- 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 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

3 **for** $n = 1, \dots, N$ **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** $m = 1, \dots, M$ **do**

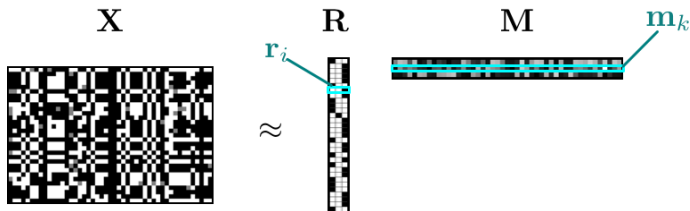
6
$$\mathbf{z}_m = \left(\sum_{n:(n,m) \in O} \mathbf{u}_n \mathbf{u}_n^\top \right)^{-1} \sum_{n:(n,m) \in O} R_{nm} \mathbf{u}_n$$

7 until convergence

See also the paper “Probabilistic Matrix Factorization” in the course readings.

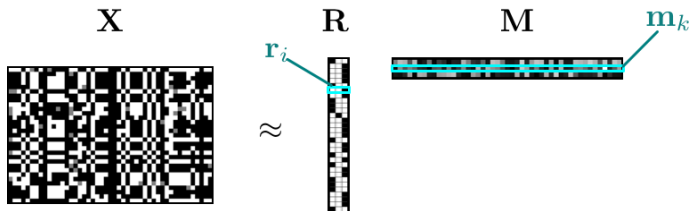
K-Means

- 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

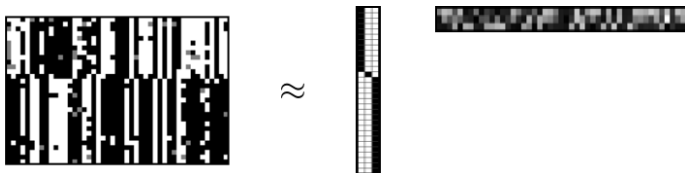
- It's possible to view K-means as a matrix factorization.
- Stack the indicator vectors \mathbf{r}_i for assignments into a $N \times K$ matrix \mathbf{R} , and stack the cluster centers \mathbf{m}_k into a $K \times D$ matrix \mathbf{M} .
- “Reconstruction” of the data (replace each point with its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



- K-means distortion function in matrix form:

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

- Can sort by cluster for visualization:

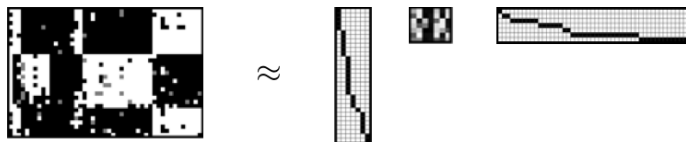


Co-clustering (optional)

- 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



- **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)

- **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

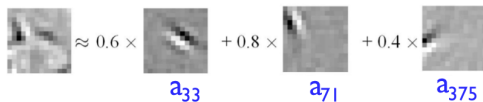
Sparse Coding

- 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

- 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 $K > D$ (e.g. more basis functions than pixels)
- The requirement that \mathbf{s} is sparse makes things interesting



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

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?

- 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

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

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

- 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

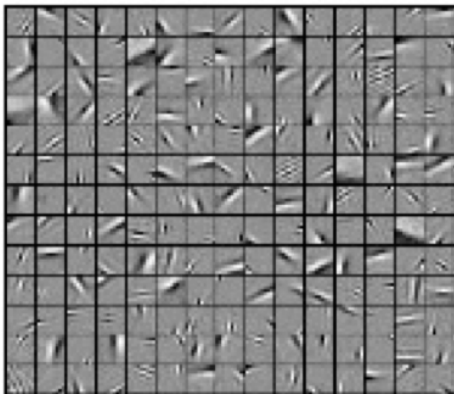
$$\min_{\{\mathbf{s}_i\}, \mathbf{A}} \sum_{i=1}^N \|\mathbf{x} - \mathbf{A}\mathbf{s}_i\|^2 + \beta \|\mathbf{s}_i\|_1$$

subject to $\|\mathbf{a}_k\|^2 \leq 1$ 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

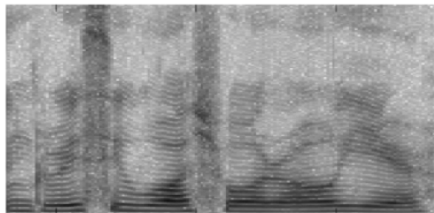
- Basis functions learned from natural images:



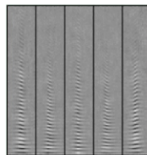
Sparse Coding: Learning the Dictionary

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

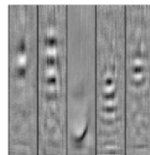
Applying sparse coding to speech signals:



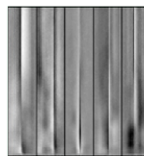
example speech spectrogram (log amplitude)



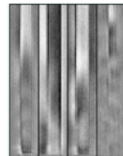
fundamental frequency and overtones



formants



plosives



fricatives

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

- 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.