ML4 B&I: Introduction to Machine Learning

Lecture 6- PCA, Matrix Completion, Autoencoders

Murat A. Erdogdu

Vector Institute, Fall 2022

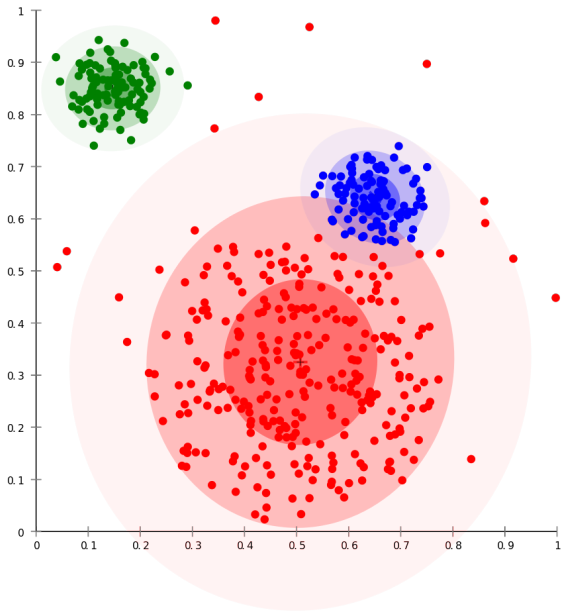
Today

- So far in this course: supervised learning
- Today we start unsupervised learning
  - No labels, so the purpose is to find patterns in data
  - Need to specify what kind of patterns to look for
- **This week:** dimensionality reduction
  - Linear dimensionality reduction (Principal Component Analysis)
    - Matrix completion is closely related to PCA.
  - Nonlinear dimensionality reduction (autoencoders).

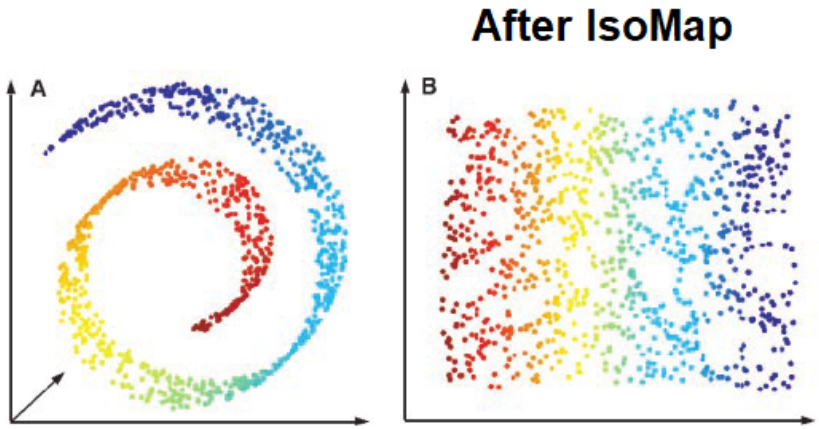
Motivating Examples

In this course, we'll focus on two simpler instances of unsupervised learning:

Clustering



Dimensionality Reduction

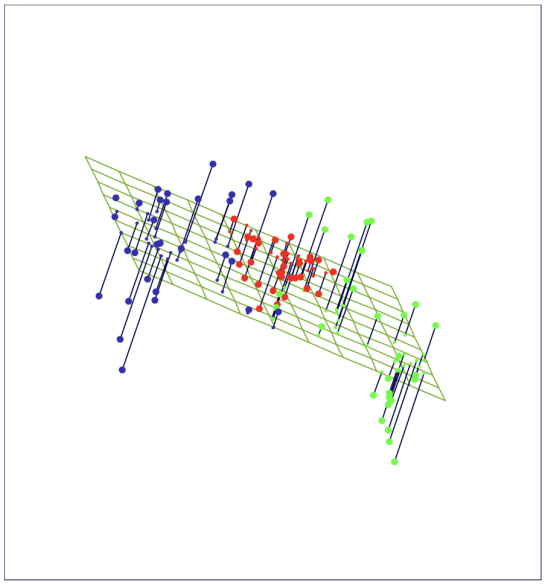


Linear Dimensionality Reduction

- We'll start with a simpler form of dimensionality reduction: linear dimensionality reduction
- **Example:** suppose you're a psychologist interested in modeling the variation in human personality
  - You've asked lots of participants to take a survey with lots of personality questions.
  - By figuring out which questions are highly correlated with each other, you can uncover the main factors describing human personality.
- A linear dimensionality reduction model found five key personality traits called the Big Five:
  - extraversion, agreeableness, openness to experience, conscientiousness, neuroticism
- In this lecture, we'll consider Principal Component Analysis (PCA)

PCA: Overview

- Principal Component Analysis (PCA) is our first unsupervised learning algorithm, and an example of linear dimensionality reduction.
- Dimensionality reduction: map data to a lower dimensional space
  - Save computation/memory
  - Reduce overfitting, achieve better generalization
  - Visualize in 2 dimensions
- Since PCA is a linear model, this mapping will be a projection.



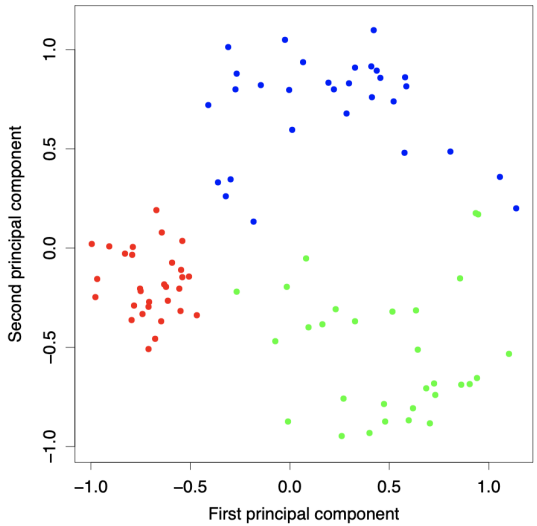
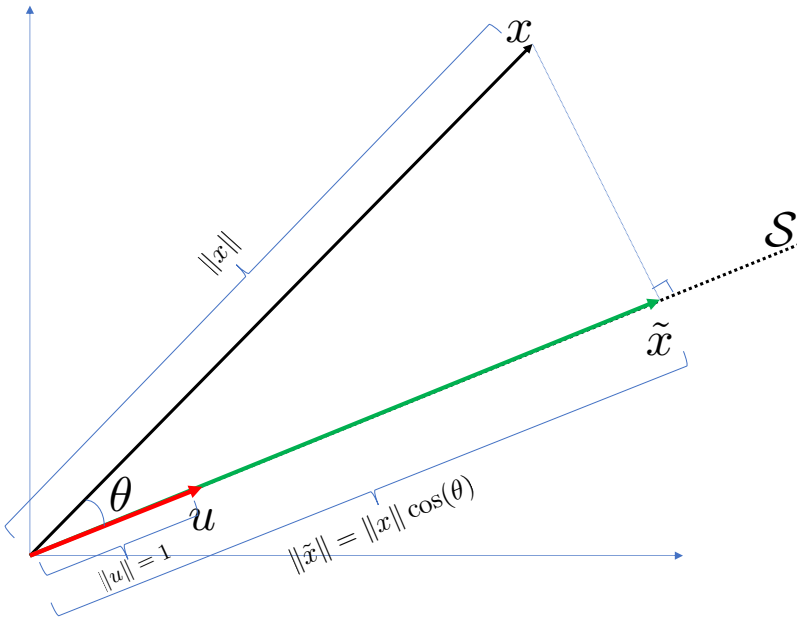


Image credit: Elements of Statistical Learning

Euclidean projection

Projection onto a 1-D subspace



- Subspace $\mathcal{S}$ is the line along the unit vector $\mathbf{u}$
  - $\mathbf{u}$ is a basis for $\mathcal{S}$: any point in $\mathcal{S}$ can be written as $z\mathbf{u}$ for some z.

- Projection of $\mathbf{x}$ on $\mathcal{S}$ is denoted by $\text{Proj}_{\mathcal{S}}(\mathbf{x})$
- Recall: $\mathbf{x}^\top \mathbf{u} = \|\mathbf{x}\| \|\mathbf{u}\| \cos(\theta) = \|\mathbf{x}\| \cos(\theta)$
- $\text{Proj}_{\mathcal{S}}(\mathbf{x}) = \underbrace{\mathbf{x}^\top \mathbf{u}}_{\text{length of proj}} \cdot \underbrace{\mathbf{u}}_{\text{direction of proj}} = \|\tilde{\mathbf{x}}\| \mathbf{u}$

General subspaces

- How to project onto a K-dimensional subspace?
  - **Idea:** choose an orthonormal basis $\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_K\}$ for $\mathcal{S}$ (i.e. all unit vectors and orthogonal to each other)
  - Project onto each unit vector individually (as in previous slide), and sum together the projections.
- Mathematically, the projection is given as:

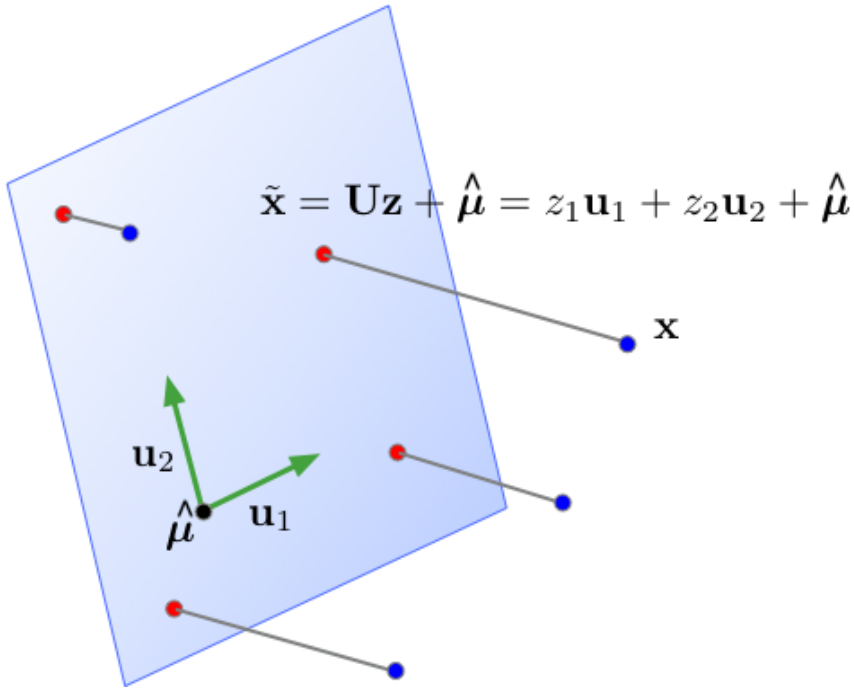
$$\text{Proj}_{\mathcal{S}}(\mathbf{x}) = \sum_{i=1}^{K} z_i \mathbf{u}_i \quad \text{where} \quad z_i = \mathbf{x}^\top \mathbf{u}_i.$$

- In vector form:

$$\text{Proj}_{\mathcal{S}}(\mathbf{x}) = \mathbf{U}\mathbf{z} \quad \text{where} \quad \mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

Projection onto a Subspace

- So far, we assumed the subspace passes through $\mathbf{0}$.
- In mathematical terminology, the "subspaces" we want to project onto are really affine spaces, and can have an arbitrary origin $\hat{\boldsymbol{\mu}}$.

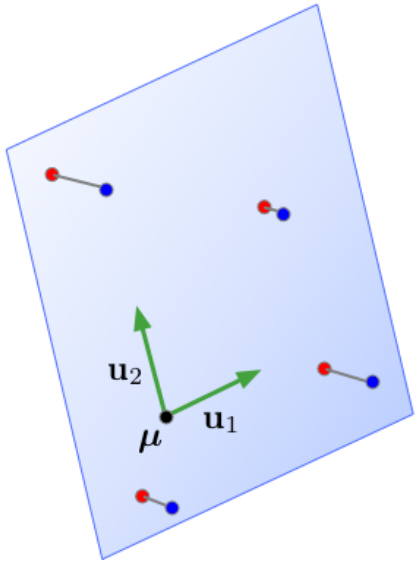


$$\mathbf{z} = \mathbf{U}^\top(\mathbf{x} - \hat{\boldsymbol{\mu}})$$

- In machine learning, $\tilde{\mathbf{x}}$ is also called the reconstruction of $\mathbf{x}$.
- $\mathbf{z}$ is its representation, or code.

Projection onto a Subspace

- If we have a K-dimensional subspace in a D-dimensional input space, then $\mathbf{x} \in \mathbb{R}^D$ and $\mathbf{z} \in \mathbb{R}^K$.
- If the data points $\mathbf{x}$ all lie close to their reconstructions, then we can approximate distances, etc. in terms of these same operations on the code vectors $\mathbf{z}$.
- If $K \ll D$, then it's much cheaper to work with $\mathbf{z}$ than $\mathbf{x}$.
- A mapping to a space that's easier to manipulate or visualize is called a representation, and learning such a mapping is representation learning.
- Mapping data to a low-dimensional space is called dimensionality reduction.



Learning a Subspace

- How to choose a good subspace $\mathcal{S}$?
  - Origin $\hat{\boldsymbol{\mu}}$ is the empirical mean of the data
  - Need to choose a $D \times K$ matrix $\mathbf{U}$ with orthonormal columns.
- Two criteria:
  - Minimize the reconstruction error:

$$\min_{\mathbf{U}} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2$$

  - Maximize the variance of reconstructions: Find a subspace where data has the most variability.

$$\max_{\mathbf{U}} \frac{1}{N} \sum_{i} \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2$$

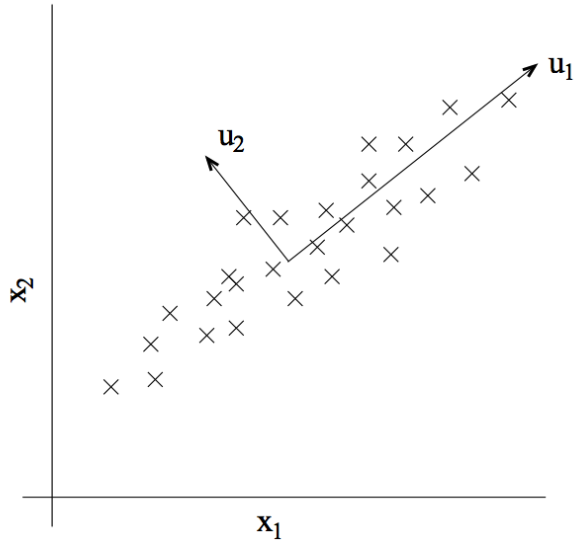
Learning a Subspace

- These two criteria are equivalent! I.e.,

$$\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)}-\tilde{\mathbf{x}}^{(i)}\|^2 = \text{const} - \frac{1}{N}\sum_{i}\|\tilde{\mathbf{x}}^{(i)}-\hat{\boldsymbol{\mu}}\|^2$$

- That is (See Appendix for a proof)

projected variance + reconstruction error = constant



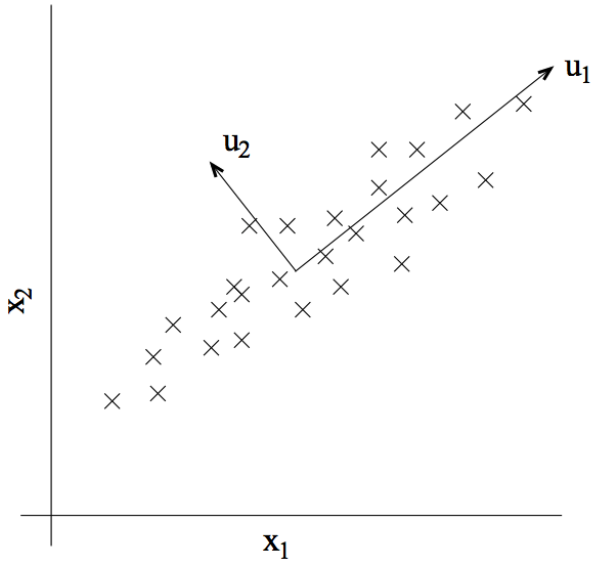
Principal Component Analysis

Choosing a subspace to maximize the projected variance, or minimize the reconstruction error, is called principal component analysis (PCA).

- Recall $\tilde{\mathbf{x}}^{(i)} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{U}^\top(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})$.
- Consider the empirical covariance matrix:

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

- The optimal PCA subspace is spanned by the top K eigenvectors of $\hat{\boldsymbol{\Sigma}}$.
  - Recall: $\hat{\boldsymbol{\Sigma}} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top$.
  - Choose $\mathbf{U}$ as the first K columns of $\mathbf{Q}$
- Eigenvectors (columns of $\mathbf{Q}$) are called principal components.



Recap

Recap:

- Dimensionality reduction aims to find a low-dimensional representation of the data.
- PCA projects the data onto a subspace which maximizes the projected variance, or equivalently, minimizes the reconstruction error.
- The optimal subspace is given by the top eigenvectors of the empirical covariance matrix.
- PCA gives a set of decorrelated features.
- See appendix for mathematics.

Applying PCA to faces

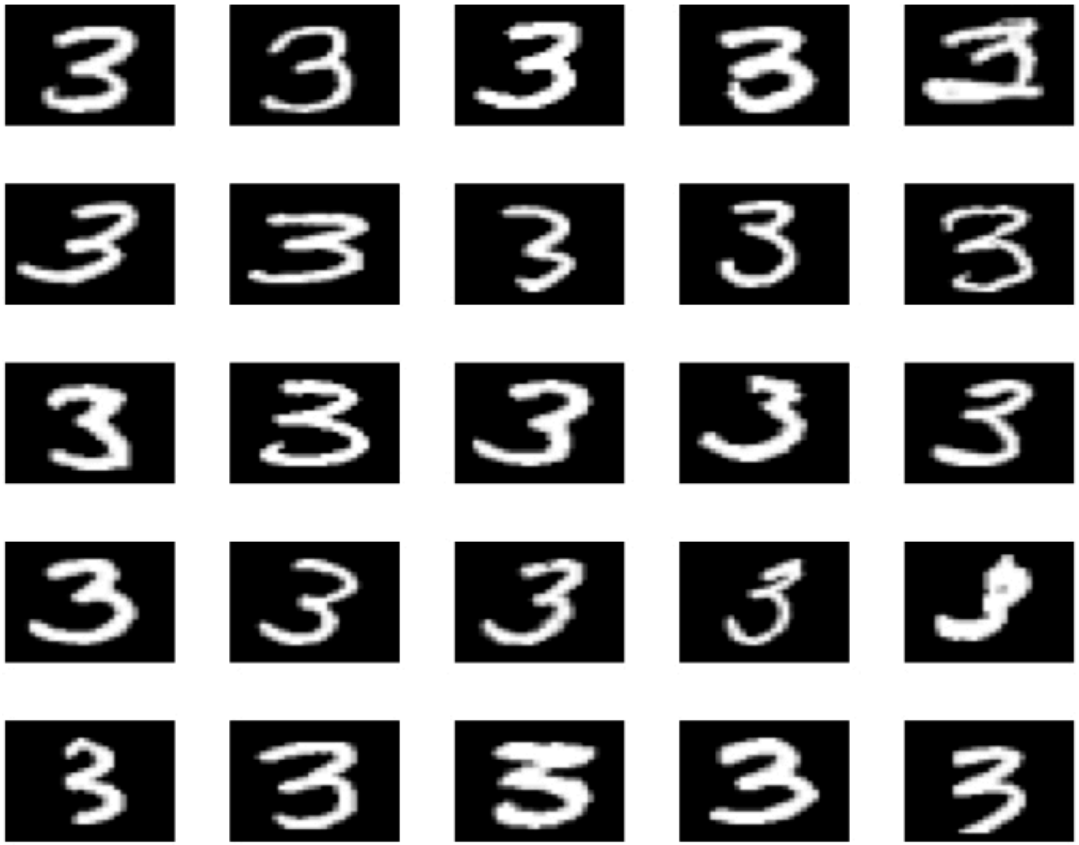
- Consider running PCA on 2429 19x19 grayscale images (CBCL data)
- Can get good reconstructions with only 3 components

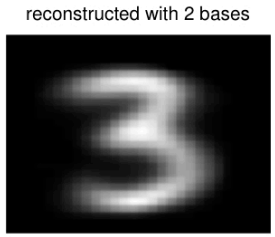
- PCA for pre-processing: can apply classifier to latent representation
  - Original data is 361 dimensional
  - For face recognition PCA with 3 components obtains 79% accuracy on face/non-face discrimination on test data vs. 76.8% for a Gaussian mixture model (GMM) with 84 states.
- Can also be good for visualization

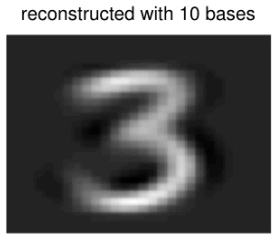
Applying PCA to faces: Learned basis

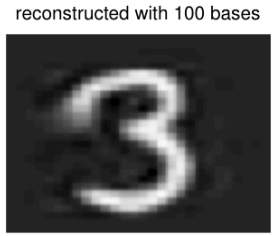
Principal components of face images ("eigenfaces")

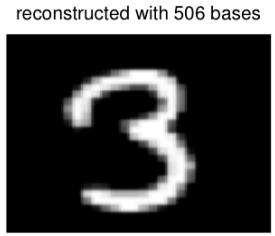
Applying PCA to digits

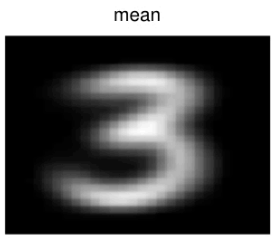


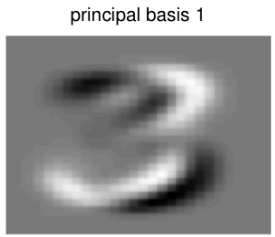


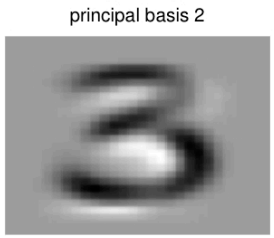


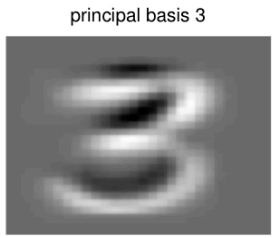










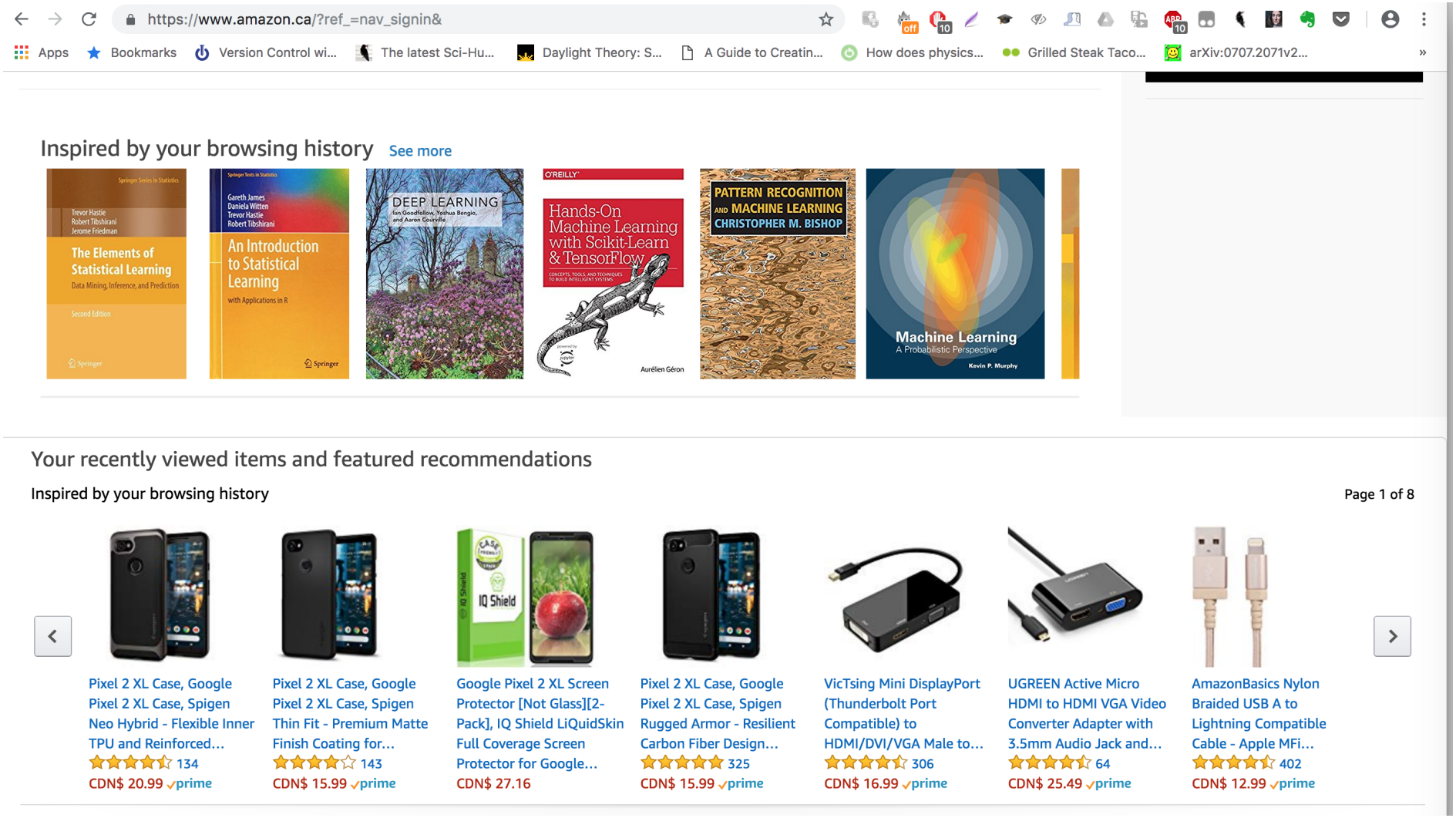


Next

Two more interpretations of PCA, which have interesting generalizations.

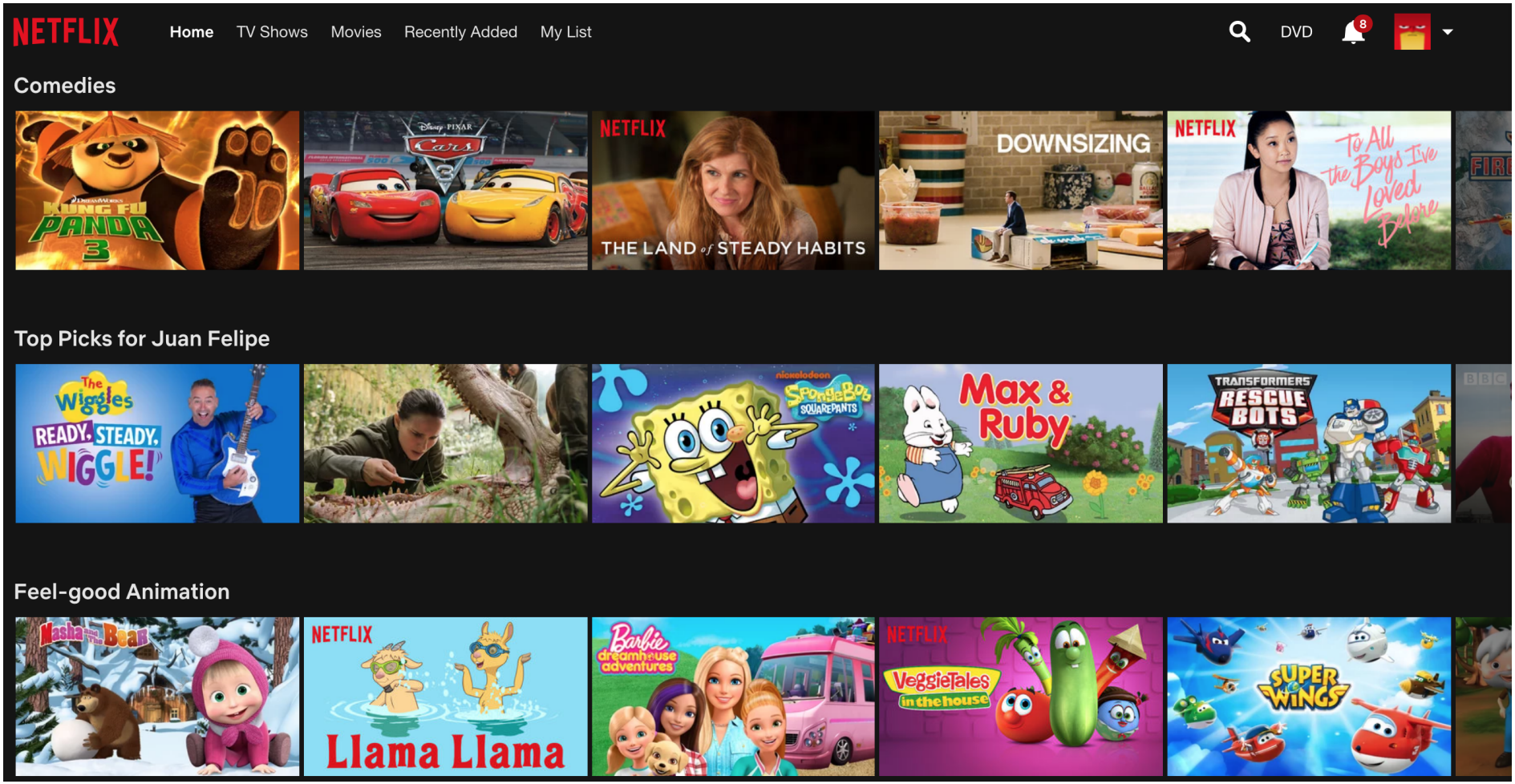
1. **Matrix factorization**
2. Autoencoder

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 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 for simplicity 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

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

- Can write the squared reconstruction error as

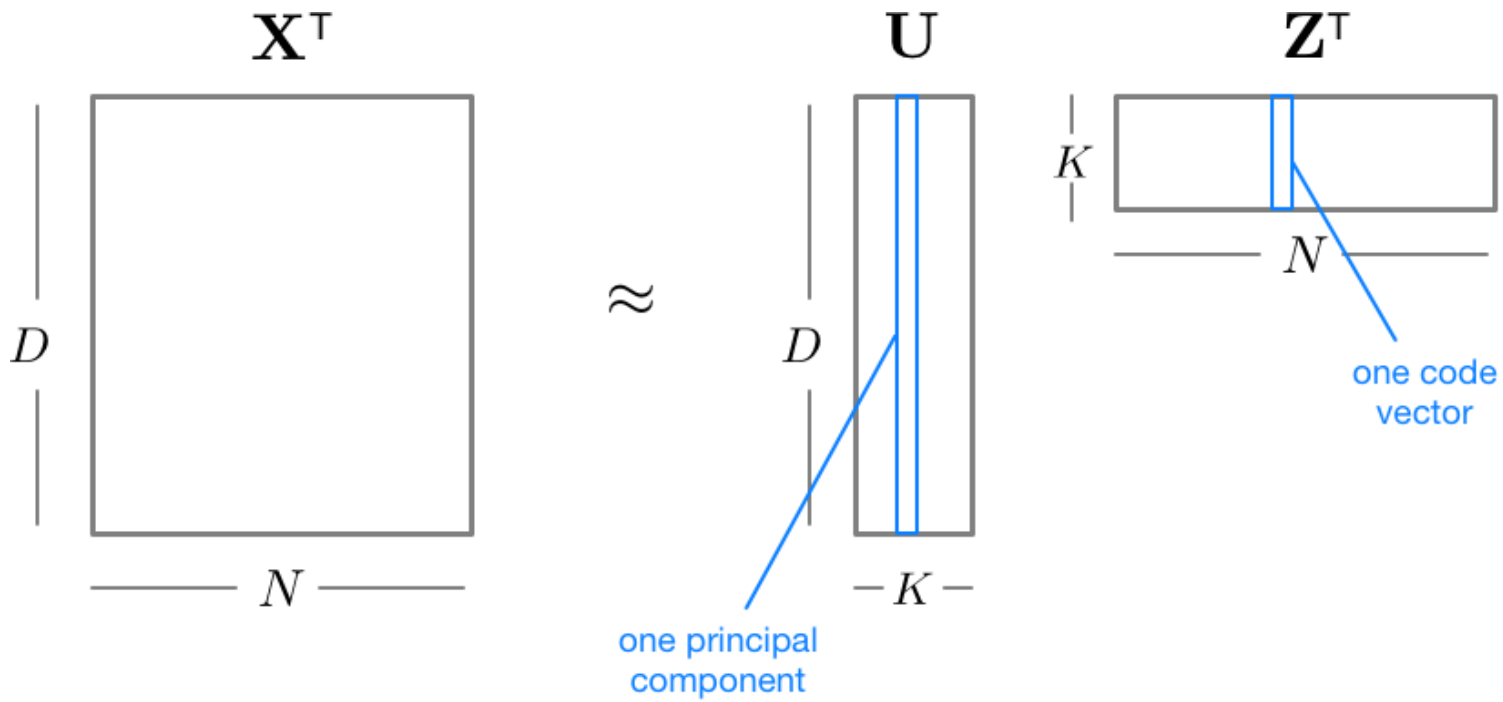
$$\sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \mathbf{U}\mathbf{z}^{(i)}\|^2 = \|\mathbf{X} - \mathbf{Z}\mathbf{U}^\top\|_F^2,$$

- $\|\cdot\|_F$ denotes the Frobenius norm:

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

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

Matrix Completion

- We just saw that PCA gives the optimal low-rank matrix factorization to a matrix $\mathbf{X}$.
- Can we generalize this to the case where $\mathbf{X}$ 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.

The Netflix problem

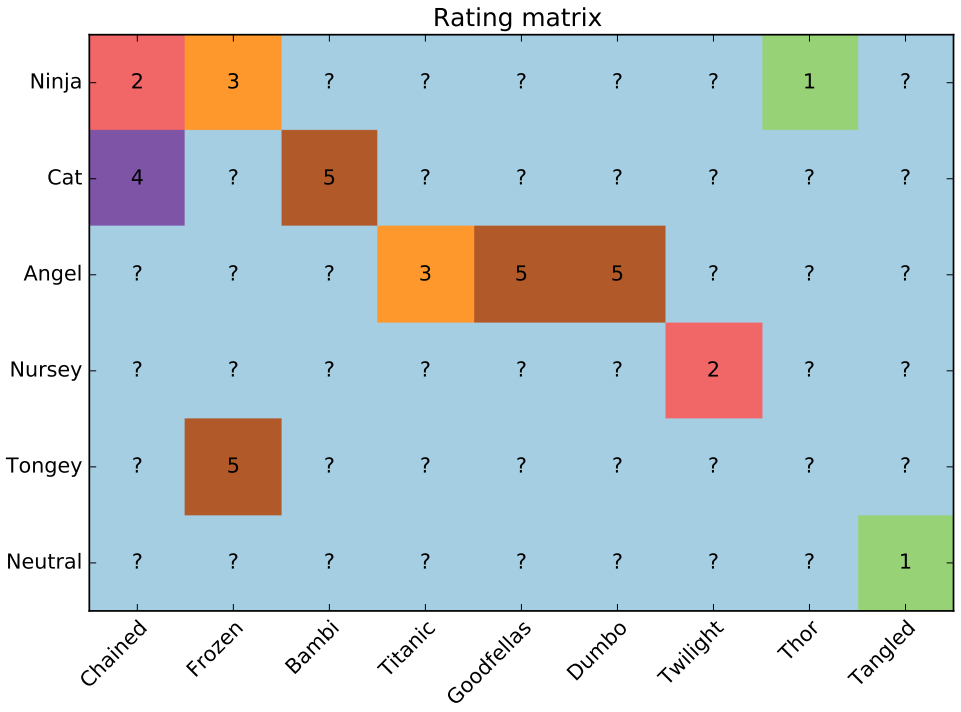
Movie recommendation: Users watch movies and rate them as good or bad.

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

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**: Predict missing entries, i.e. how a user would rate a movie they haven't previously rated
- **Evaluation Metric**: Squared error (used by Netflix Competition). Is this a reasonable metric?

Matrix Completion

- 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?

Matrix Completion

- Let the representation of user i in the K-dimensional space be $\mathbf{u}_i$ and the representation of movie j be $\mathbf{z}_j$
  - Intuition: maybe the first entry of $\mathbf{u}_i$ says how much the user likes horror films, and the first entry of $\mathbf{z}_j$ says how much movie j is a horror film.
- Assume the rating user i gives to movie j is given by a dot product: $R_{ij} \approx \mathbf{u}_i^\top \mathbf{z}_j$
- 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!

Matrix Completion

- 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.

- What's different about the Netflix problem?
  - Most entries are missing!
  - We only want to count the error for the observed entries.

Matrix Completion

- Let $O = \{(n, m) : \text{ entry } (n, m) \text{ of matrix } \mathbf{R} \text{ is observed}\}$
- Using the squared error loss, matrix completion requires solving

$$\min_{\mathbf{U},\mathbf{Z}} \frac{1}{2} \sum_{(i,j)\in O} \left(R_{ij} - \mathbf{u}_i^\top \mathbf{z}_j\right)^2$$

- The objective is non-convex in $\mathbf{U}$ and $\mathbf{Z}$ jointly.
- 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 (next lecture)!

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

- Want to minimize the squared error cost with respect to the factor $\mathbf{U}$. (The case of $\mathbf{Z}$ is exactly symmetric.)
- We can decompose the cost into a sum of independent terms:

$$\sum_{(i,j)\in O} \left(R_{ij} - \mathbf{u}_i^\top \mathbf{z}_j\right)^2 = \sum_i \underbrace{\sum_{j:(i,j)\in O} \left(R_{ij} - \mathbf{u}_i^\top \mathbf{z}_j\right)^2}_{\text{only depends on } \mathbf{u}_i}$$

  This can be minimized independently for each $\mathbf{u}_i$.
- This is a linear regression problem in disguise. Its optimal solution is:

$$\mathbf{u}_i = \left(\sum_{j:(i,j)\in O} \mathbf{z}_j \mathbf{z}_j^\top\right)^{-1} \sum_{j:(i,j)\in O} R_{ij}\mathbf{z}_j$$

Alternating Least Squares

ALS for Matrix Completion problem

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

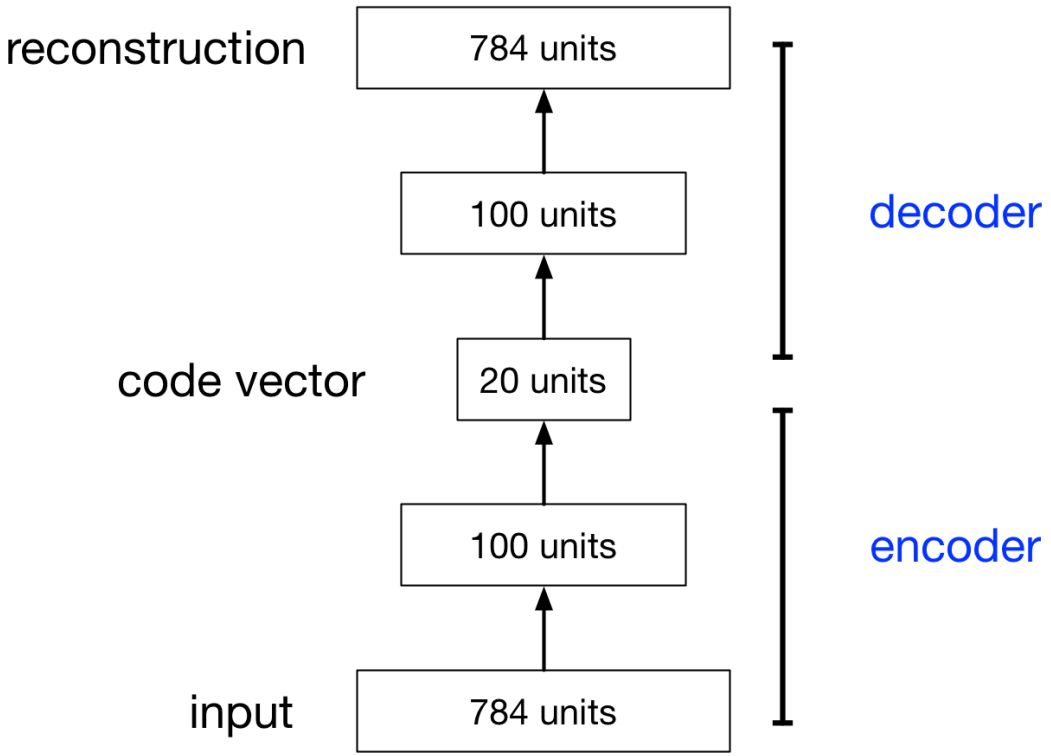
Next

Two more interpretations of PCA, which have interesting generalizations.

1. Matrix factorization
2. **Autoencoder**

Autoencoders

- An autoencoder is a feed-forward neural net whose job is to take an input $\mathbf{x}$ and predict $\mathbf{x}$.
- To make this non-trivial, we need to add a bottleneck layer whose dimension is much smaller than the input.



Linear Autoencoders

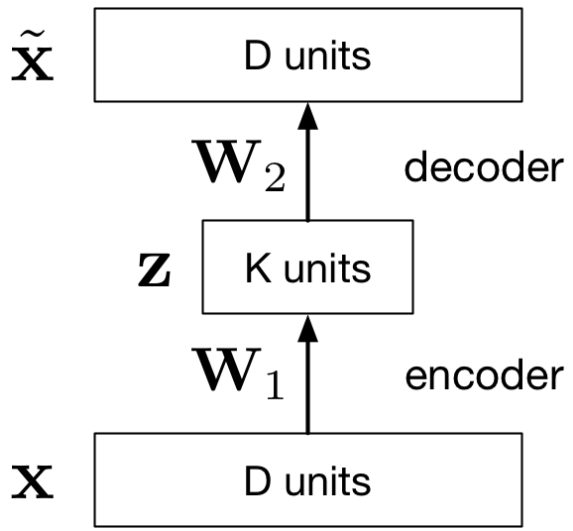
Why autoencoders?

- Map high-dimensional data to two dimensions for visualization
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
  - Unlabled data can be much more plentiful than labeled data

Linear Autoencoders

- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss.

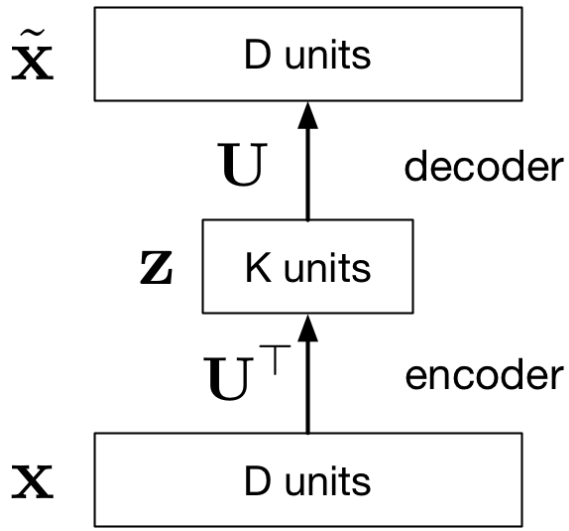
$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$



- This network computes $\tilde{\mathbf{x}} = \mathbf{W}_2\mathbf{W}_1\mathbf{x}$, which is a linear function.
- If $K \geq D$, we can choose $\mathbf{W}_2$ and $\mathbf{W}_1$ such that $\mathbf{W}_2\mathbf{W}_1$ is the identity matrix. This isn't very interesting.
- But suppose $K < D$:
  - $\mathbf{W}_1$ maps $\mathbf{x}$ to a K-dimensional space, so it's doing dimensionality reduction.

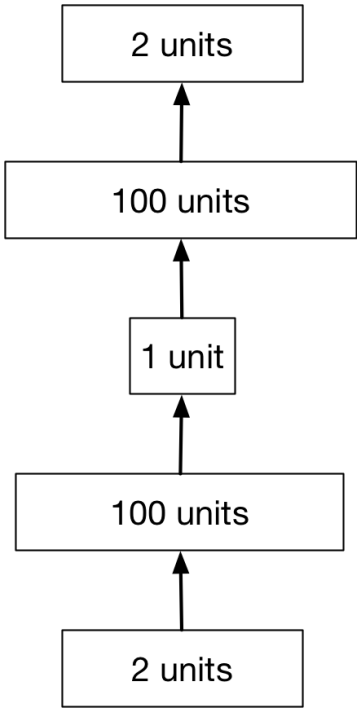
Linear Autoencoders

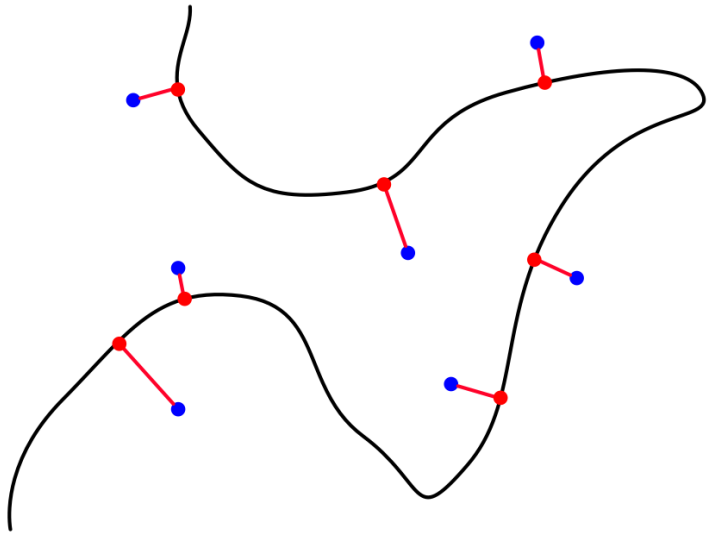
- Observe that the output of the autoencoder must lie in a K-dimensional subspace spanned by the columns of $\mathbf{W}_2$. This is because $\tilde{\mathbf{x}} = \mathbf{W}_2\mathbf{z}$
- We saw that the best possible (min error) K-dimensional linear subspace in terms of reconstruction error is the PCA subspace.
- The autoencoder can achieve this by setting $\mathbf{W}_1 = \mathbf{U}^\top$ and $\mathbf{W}_2 = \mathbf{U}$.
- Therefore, the optimal weights for a linear autoencoder are just the principal components!



Nonlinear Autoencoders

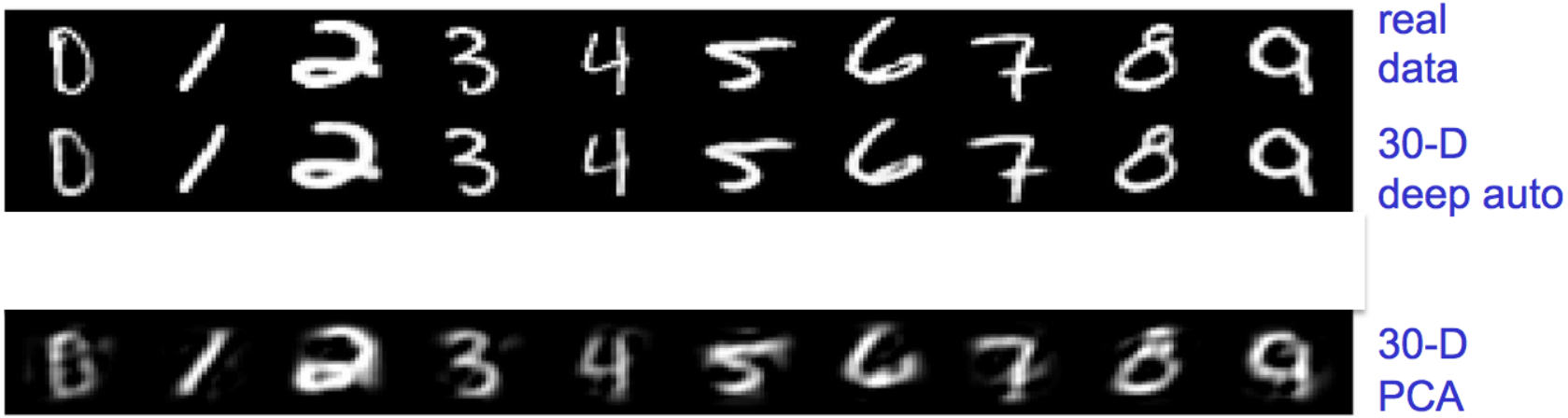
- Deep nonlinear autoencoders learn to project the data, not onto a subspace, but onto a nonlinear manifold
- This manifold is the image of the decoder.
- This is a kind of nonlinear dimensionality reduction.





Nonlinear Autoencoders

- Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA)



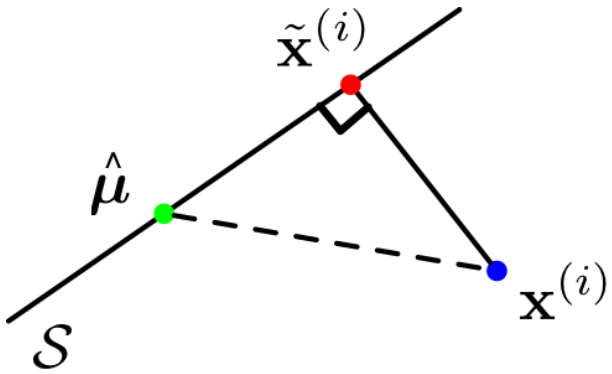
Conclusion

- We covered Principal Component Analysis, which is a fundamental dimension reduction technique in machine learning.
- We discussed matrix completion and matrix factorization, and showed their applications on recommender systems.
- We also covered non-linear dimension reduction techniques called auto-encoders.
- Lester Mackey is giving a talk tomorrow – don't miss!

Pythagorean Theorem

- Observation 1: $\|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 = \|\mathbf{z}^{(i)}\|^2$
  - Variance of reconstructions is equal to variance of code vectors: $\frac{1}{N}\sum_i \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 = \frac{1}{N}\sum_i \|\mathbf{z}^{(i)}\|^2$ (exercise $\frac{1}{N}\sum_i \mathbf{z}^{(i)} = 0$)

- **Observation 2**: orthogonality of $\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}$ and $\tilde{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}$
  (Two vectors $\mathbf{a}, \mathbf{b}$ are orthogonal $\iff \mathbf{a}^\top \mathbf{b} = 0$)
- Recall $\tilde{\mathbf{x}}^{(i)} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{U}^\top(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})$.



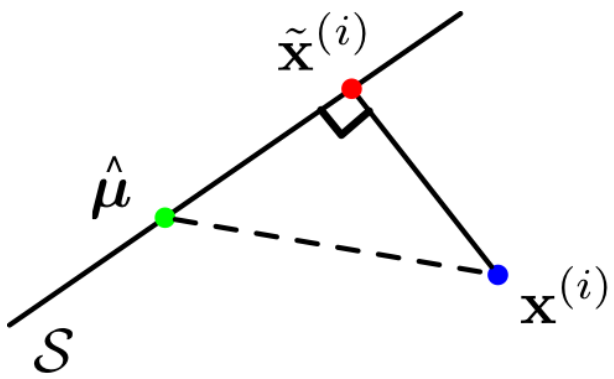
$$
\begin{aligned}
&(\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}})^\top(\tilde{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}) \\
&= (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top \mathbf{U}\mathbf{U}^\top(\hat{\boldsymbol{\mu}} - \mathbf{x}^{(i)} + \mathbf{U}\mathbf{U}^\top(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})) \\
&= (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top \mathbf{U}\mathbf{U}^\top(\hat{\boldsymbol{\mu}} - \mathbf{x}^{(i)}) + (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top \mathbf{U}\mathbf{U}^\top(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}) \\
&= 0
\end{aligned}
$$

Pythagorean Theorem

The Pythagorean Theorem tells us:

$$\|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 + \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2 = \|\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 \qquad \text{for each } i$$

By averaging over data and from observation 2, we obtain



$$\underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2}_{\text{projected variance}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2}_{\text{reconstruction error}}$$

$$= \underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}\|^2}_{\text{constant}}$$

Therefore,

$$\text{projected variance} = \text{constant} - \text{reconstruction error}$$

Maximizing the variance is equivalent to minimizing the reconstruction error!

Supplement: Deriving PCA

- For $K = 1$, we are fitting a unit vector $\mathbf{u}$, and the code is a scalar $z^{(i)} = \mathbf{u}^\top(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})$. Let's maximize the projected variance. From observation 1, we have

$$\frac{1}{N}\sum_i \|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\|^2 = \frac{1}{N}\sum_i [z^{(i)}]^2 = \frac{1}{N}\sum_i (\mathbf{u}^\top(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}))^2$$

$$= \frac{1}{N}\sum_{i=1}^N \mathbf{u}^\top(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top\mathbf{u} \qquad (\mathbf{a}^\top\mathbf{b})^2 = \mathbf{a}^\top\mathbf{b}\mathbf{b}^\top\mathbf{a}$$

$$= \mathbf{u}^\top\left[\frac{1}{N}\sum_{i=1}^N (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top\right]\mathbf{u}$$

$$= \mathbf{u}^\top\hat{\boldsymbol{\Sigma}}\mathbf{u}$$

$$= \mathbf{u}^\top\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top\mathbf{u} \qquad \text{Spectral Decomposition } \hat{\boldsymbol{\Sigma}} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top$$

$$= \mathbf{a}^\top\boldsymbol{\Lambda}\mathbf{a} \qquad \text{for } \mathbf{a} = \mathbf{Q}^\top\mathbf{u}$$

$$= \sum_{j=1}^D \lambda_j a_j^2$$

Supplement: Deriving PCA

- Maximize $\mathbf{a}^\top \mathbf{\Lambda a} = \sum_{j=1}^{D} \lambda_j a_j^2$ for $\mathbf{a} = \mathbf{Q}^\top \mathbf{u}$.
  - This is a change-of-basis to the eigenbasis of $\mathbf{\Sigma}$.
- Assume the λ_i are in sorted order, $\lambda_1 \geq \lambda_2, \geq \ldots$
- Observation: since $\mathbf{u}$ is a unit vector, then by unitarity, $\mathbf{a}$ is also a unit vector: $\mathbf{a}^\top \mathbf{a} = \mathbf{u}^\top \mathbf{Q}\mathbf{Q}^\top \mathbf{u} = \mathbf{u}^\top \mathbf{u}$, i.e., $\sum_j a_j^2 = 1$.
- By inspection, set $a_1 = \pm 1$ and $a_j = 0$ for $j \neq 1$.
- Hence, $\mathbf{u} = \mathbf{Q}\mathbf{a} = \mathbf{q}_1$ (the top eigenvector).

- A similar argument shows that the kth principal component is the kth eigenvector of $\mathbf{\Sigma}$.