上海科技大学
ShanghaiTech University

CS182: Introduction to Machine Learning – Dimensionality Reduction

Yujiao Shi
SIST, ShanghaiTech
Spring, 2025

Recall: K-means Algorithm

Recall...

- Input: $\mathcal{D} = \{(\boldsymbol{x}^{(n)})\}_{n=1}^{N}, K$

1. Initialize cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
2. While NOT CONVERGED
   a. Assign each data point to the cluster with the nearest cluster center:
   $$z^{(n)} = \underset{k}{\operatorname{argmin}} \left\| \boldsymbol{x}^{(n)} - \boldsymbol{\mu}_k \right\|_2$$
   b. Recompute the cluster centers:
   $$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n\,:z^{(n)}\,=k} \boldsymbol{x}^{(n)}$$
   where N_k is the number of data points in cluster k

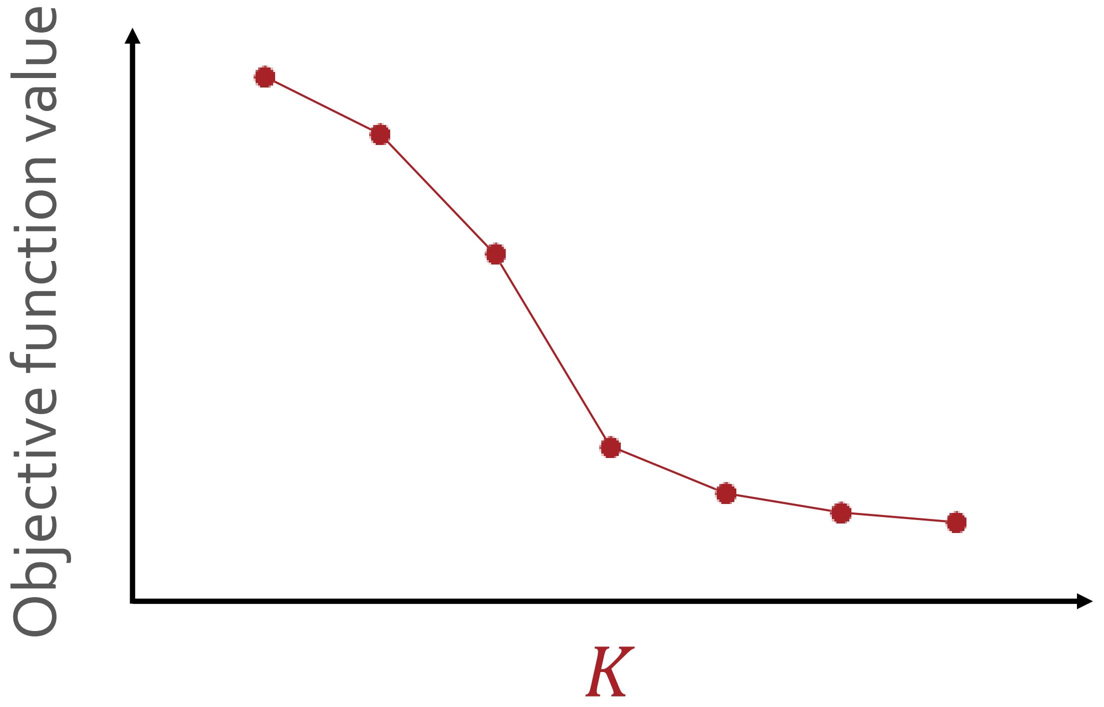
- Output: cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and cluster assignments $z^{(1)}, \ldots, z^{(N)}$

How do we set these hyperparameters?

Recall...

Setting K

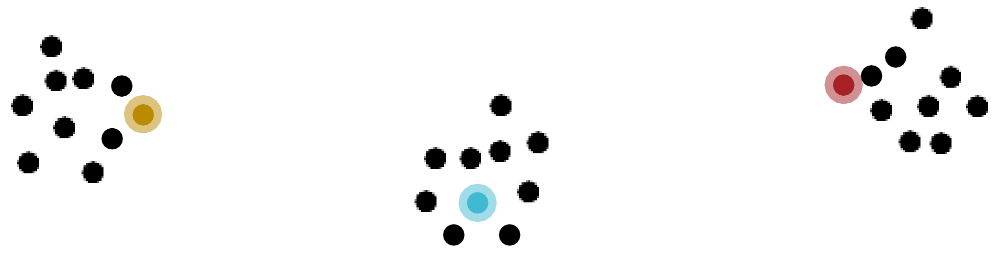
- Idea: choose the value of K that minimizes the objective function



- Better Idea: look for the characteristic "elbow" or largest decrease when going from $K-1$ to K

Initializing K-means: Lloyd's Method

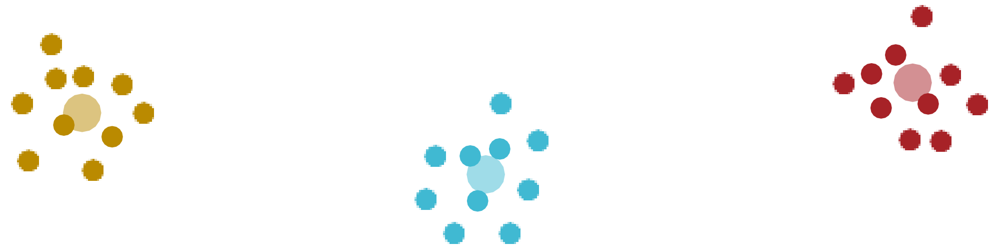
- Randomly choose K data points to be the initial cluster centers



Recall…

Initializing K-means: Lloyd's Method

- Randomly choose K data points to be the initial cluster centers

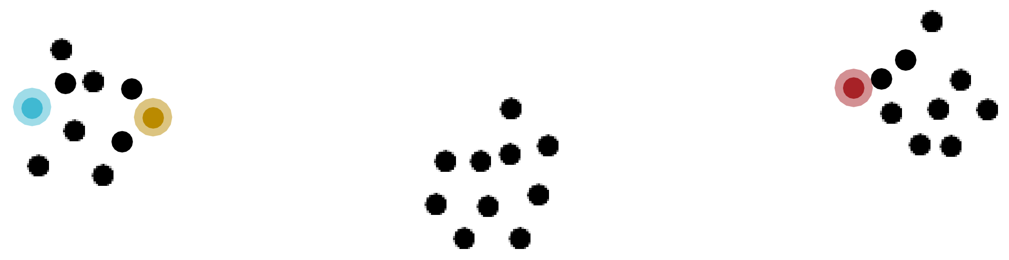


Recall…

上海科技大学 ShanghaiTech University

Initializing K-means: Lloyd's Method

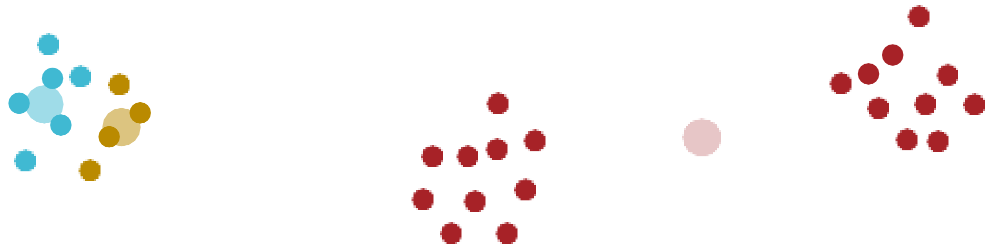
- Randomly choose K data points to be the initial cluster centers



Recall…

Initializing K-means: Lloyd's Method

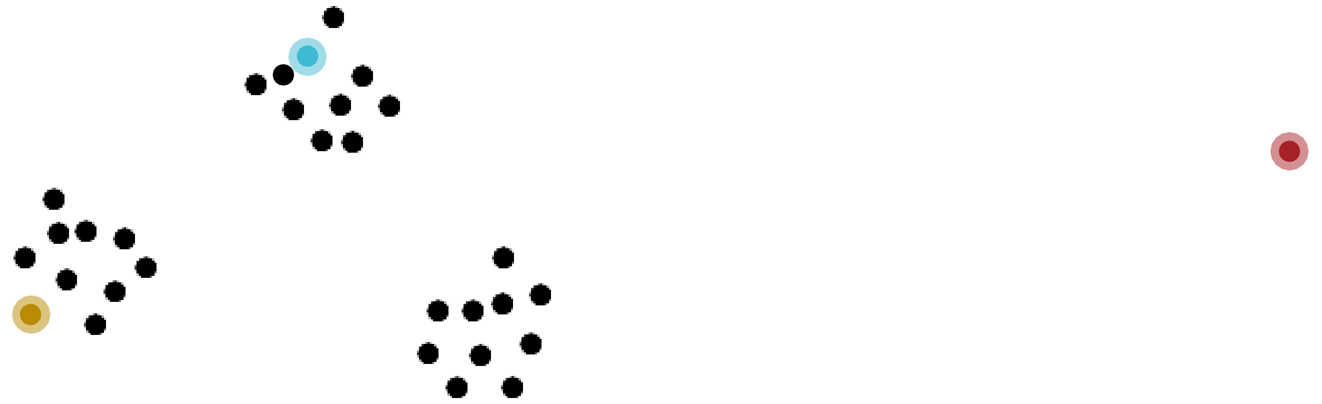
- Randomly choose K data points to be the initial cluster centers



Recall...

Initializing K-means: Furthest Point

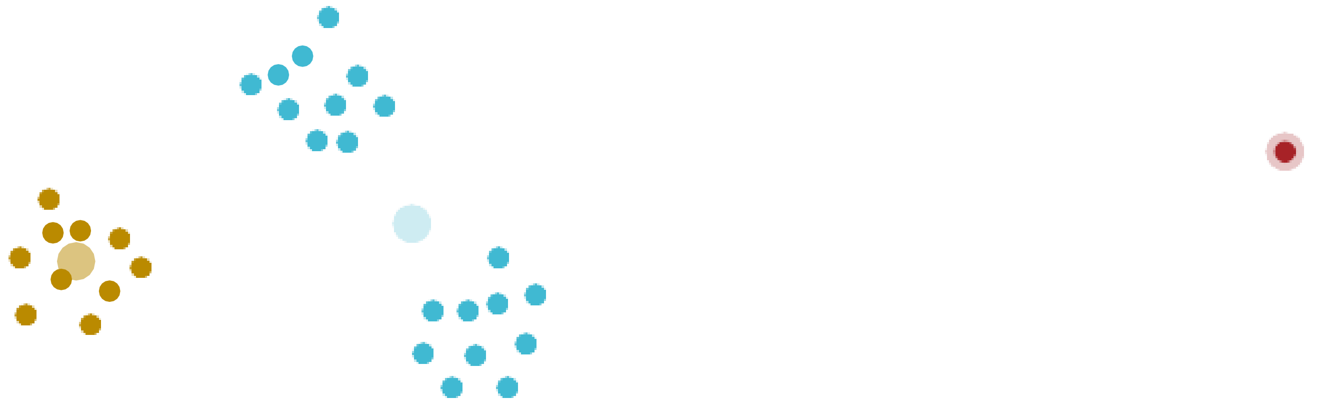
1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D\ (\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D\ (\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K-1$ times



Recall…

Initializing K-means: Furthest Point

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D(\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K-1$ times



Recall...

上海科技大学 ShanghaiTech University

Initializing K-means: Furthest Point

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D\,(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D\,(\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K-1$ times

- Works great in the case of well-clustered data!
- Can struggle with outliers...

Recall...

Initializing K-means: K-means++

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D\left(\boldsymbol{x}\right)$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. *Sample* the next cluster center with probability proportional to $D\left(\boldsymbol{x}\right)^2$
4. Repeat 2 and 3 $K-1$ times

i	$D\left(\boldsymbol{x}^{(i)}\right)$	$D\left(\boldsymbol{x}^{(i)}\right)^2$	Probability of Being Selected
1	4	16	$16/123$
2	7	49	$49/123$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	1	1	$1/123$
Total		123	$123/123 = 1$

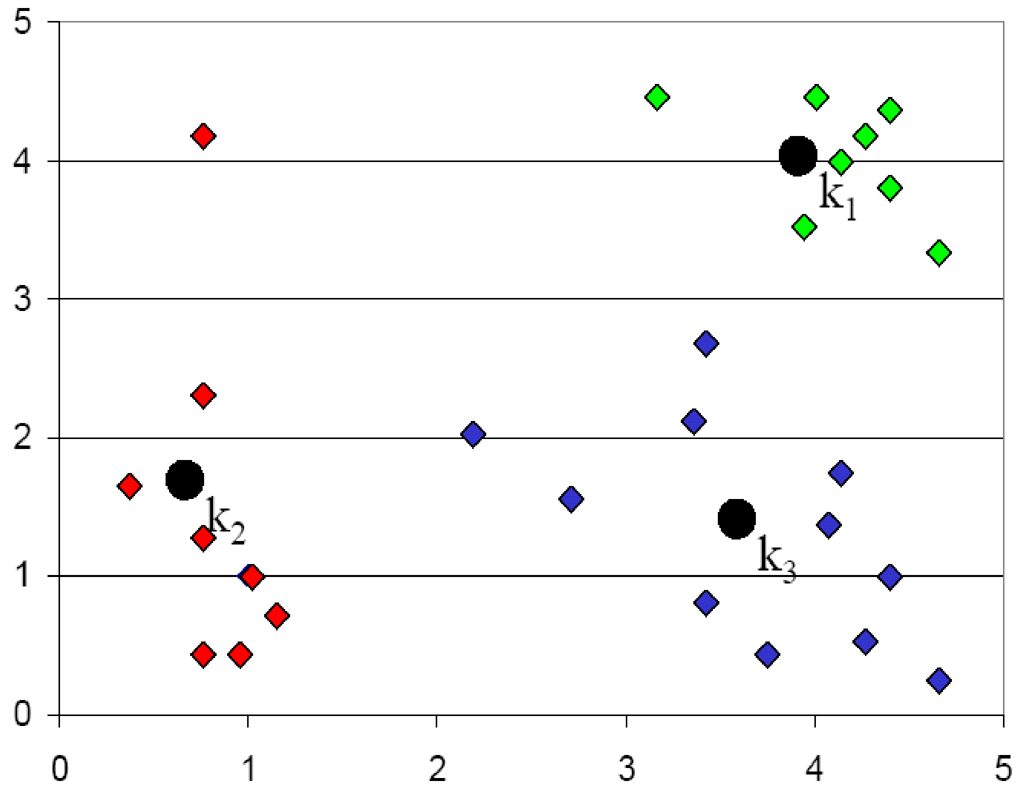
Recall...

Initializing K-means: K-means++

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. *Sample* the next cluster center with probability proportional to $D(\boldsymbol{x})^2$
4. Repeat 2 and 3 $K-1$ times

- K-means++ achieves a $O(\log K)$ approximation to the optimal clustering in expectation!
- All initialization methods can benefit from multiple random restarts

Problem with K-means



	Cluster 1	Cluster 2	Cluster 3
Individual 1	1	0	0
Individual 2	0	1	0
Individual 3	0	1	0
Individual 4	1	0	0
Individual 5	…	…	…
Individual 6	…	…	…
Individual 7	…	…	…
Individual 8	…	…	…
Individual 9	…	…	…
Individual 10	…	…	…

Hard assignment of samples into three clusters

Probabilistic Sof-Clustering of Samples into Three Clusters

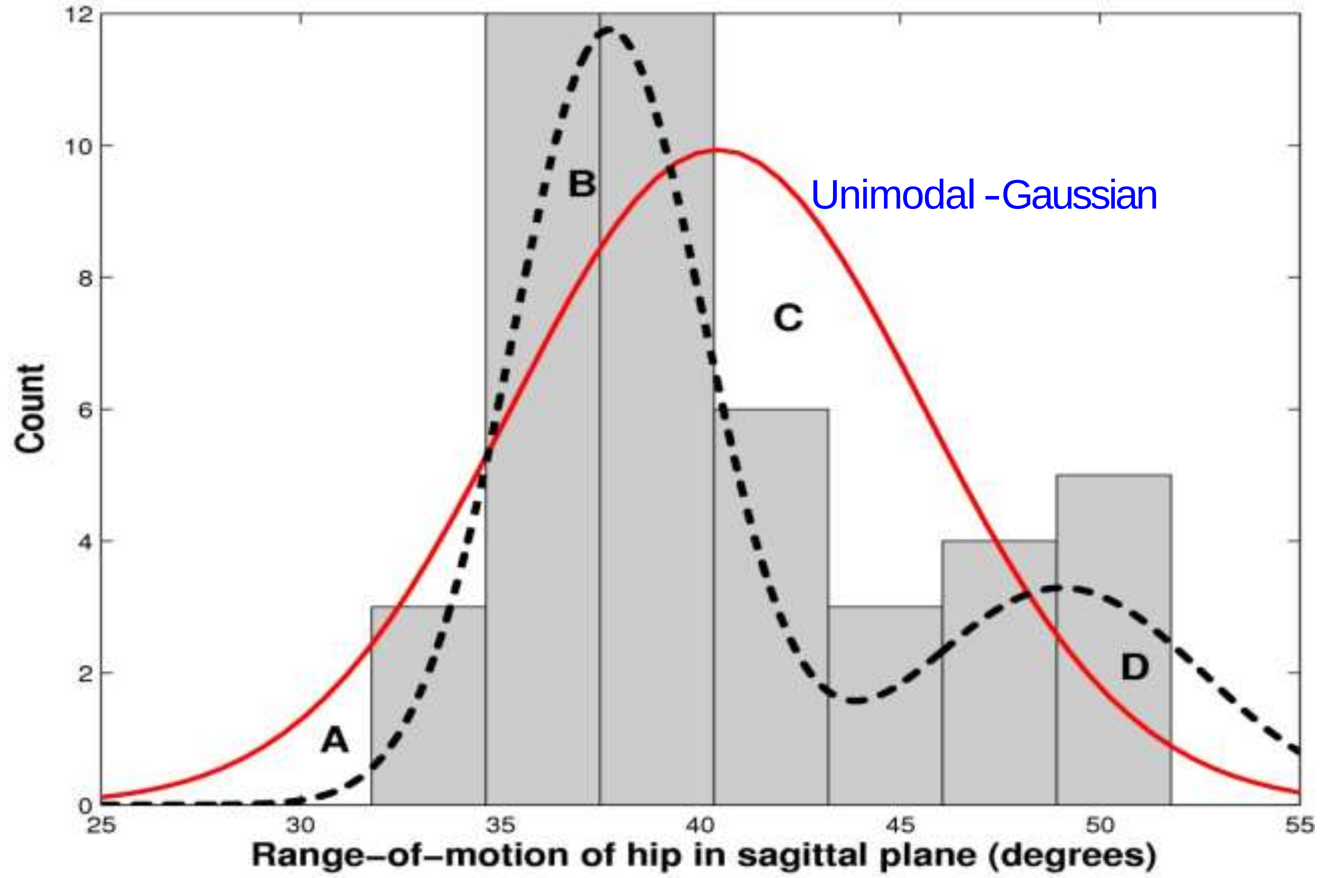
Probability of	Cluster 1	Cluster 2	Cluster 3	Sum
Individual 1	0.1	0.4	0.5	1
Individual 2	0.8	0.1	0.1	1
Individual 3	0.7	0.2	0.1	1
Individual 4	0.10	0.05	0.85	1
Individual 5	…	…	…	1
Individual 6	…	…	…	1
Individual 7	…	…	…	1
Individual 8	…	…	…	1
Individual 9	…	…	…	1
Individual 10	…	…	…	1

- Each sample can be assigned to more than one clusters with a certain probability.
- For each sample, the probabilities for all clusters should sum to 1. (i.e., each row should sum to 1.)
- Each cluster is explained by a cluster center variable (i.e., cluster mean)

Mixture Model

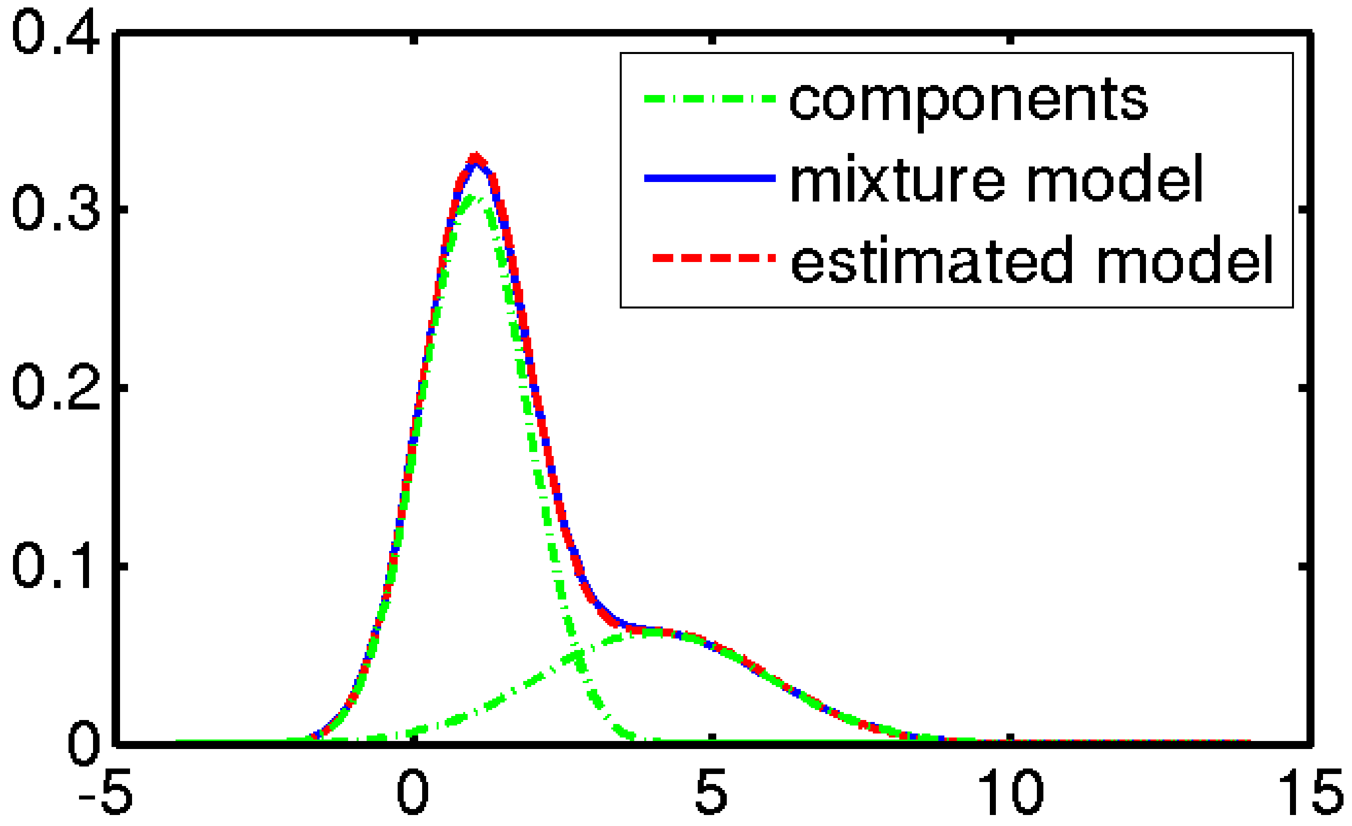
- A density model **p**(**x**) may be multi-modal.

Multi--model: how do we model this?



Mixture Model

- We may be able to model it as a mixture of uni-modal distributions (e.g., Gaussians).
- Each mode may correspond to a different sub-population (e.g., male and female).



GMM

Recall...

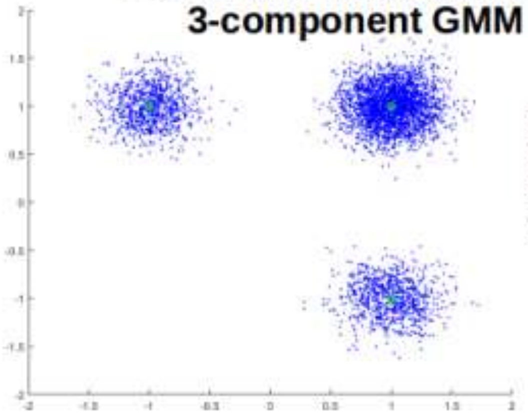
- The generative story for each $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$
  - First choose one of the K mixture components as

$$\boldsymbol{z}_n \sim \text{Multinomial}(\boldsymbol{z}_n|\pi) \qquad \text{(from the prior } p(\boldsymbol{z}) \text{ over } \boldsymbol{z})$$

  - Suppose $\boldsymbol{z}_n = k$. Now generate $\boldsymbol{x}_n$ from the k-th Gaussian as

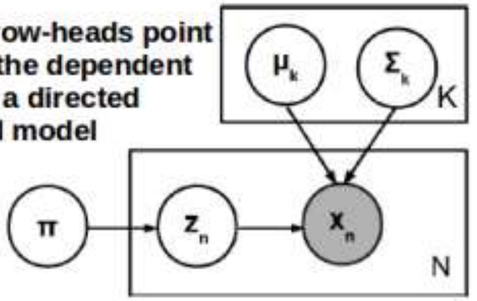
$$\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(from the data distr. } p(\boldsymbol{x}|\boldsymbol{z}))$$

Some simulated data from a 3-component GMM



Directed Graphical Model for a K-component GMM

Note: Arrow-heads point towards the dependent nodes in a directed graphical model



Shaded nodes: Observed

White nodes: Unknowns

Parameter Estimation with Latent Variables

Recall...

- Consider the 'incomplete" data log likelihood

$$\log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}|\Theta) = \log \sum_{\mathbf{Z}} q(\mathbf{Z}) \frac{p(\mathbf{X},\mathbf{Z}|\Theta)}{q(\mathbf{Z})} \quad \text{(where } q(\mathbf{Z}) \text{ is some dist.)}$$

$$\geq \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\Theta)}{q(\mathbf{Z})} \quad \text{(concave } f\text{, Jensen's Ineq.: } f(\textstyle\sum \lambda_i x_i) \geq \sum \lambda_i f(x_i))$$

$$\log p(\mathbf{X}|\Theta) \geq \sum_{\mathbf{Z}} q(\mathbf{Z}) \log p(\mathbf{X},\mathbf{Z}|\Theta) - \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \log q(\mathbf{Z})}_{\text{doesn't depend on } \Theta} = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log p(\mathbf{X},\mathbf{Z}|\Theta) + \text{const.}$$

- If we set $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X},\Theta)$, the above inequality becomes equality

$$\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\Theta)}{q(\mathbf{Z})} = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\Theta) \log \frac{\cancel{p(\mathbf{Z}|\mathbf{X},\Theta)}\, p(\mathbf{X}|\Theta)}{\cancel{p(\mathbf{Z}|\mathbf{X},\Theta)}} = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\Theta) \log p(\mathbf{X}|\Theta)$$

$$= \log p(\mathbf{X}|\Theta) \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\Theta) = \log p(\mathbf{X}|\Theta)$$

- Thus for $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X},\Theta)$, we have

$$\log p(\mathbf{X}|\Theta) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\Theta) \log p(\mathbf{X},\mathbf{Z}|\Theta) + \text{const.} = \mathbb{E}[\log p(\mathbf{X},\mathbf{Z}|\Theta)] + \text{const.}$$

- Thus $\log p(\mathbf{X}|\Theta)$ is tightly lower-bounded by $\mathbb{E}[\log p(\mathbf{X},\mathbf{Z}|\Theta)]$ which EM maximizes

The Expectation Maximization (EM) Algorithm

Recall...

Initialize the parameters: Θ^{old}. Then alternate between these steps:

- **E (Expectation) step:**
  - Compute the posterior $p(\mathbf{Z}|\mathbf{X}, \Theta^{old})$ over latent variables $\mathbf{Z}$ using Θ^{old}
  - Compute the expected complete data log-likelihood w.r.t. *this* posterior

$$\mathcal{Q}(\Theta, \Theta^{old}) = \mathbb{E}_{p(\mathbf{Z}|\mathbf{X},\Theta^{old})}[\log p(\mathbf{X}, \mathbf{Z}|\Theta)] = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \Theta^{old}) \log p(\mathbf{X}, \mathbf{Z}|\Theta)$$

- **M (Maximization) step:**
  - Maximize the expected complete data log-likelihood w.r.t. Θ

$$\begin{aligned}\Theta^{new} &= \arg\max_{\Theta} \mathcal{Q}(\Theta, \Theta^{old}) \quad \text{(if doing MLE)} \\ \Theta^{new} &= \arg\max_{\Theta} \{\mathcal{Q}(\Theta, \Theta^{old}) + \log p(\Theta)\} \quad \text{(if doing MAP)}\end{aligned}$$

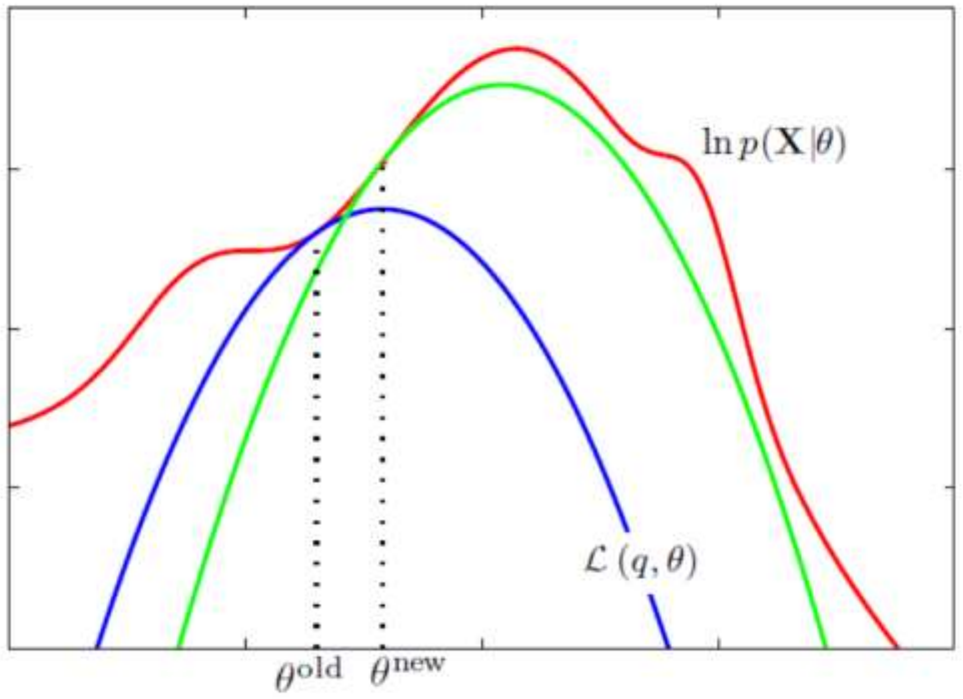
- If the log-likelihood or the parameter values not converged then set $\Theta^{old} = \Theta^{new}$ and go to the E step.

EM: A View in the Parameter Space

上海科技大学 ShanghaiTech University

Recall...

- E-step: Update of q makes the $\mathcal{L}(q,\Theta)$ curve touch the $\log p(\mathbf{X}|\Theta)$ curve
- M-step gives the maxima Θ^{new} of $\mathcal{L}(q,\Theta)$
- Next E-step readjusts $\mathcal{L}(q,\Theta)$ curve (green) to meet $\log p(\mathbf{X}|\Theta)$ curve again
- This continues until a local maxima of $\log p(\mathbf{X}|\Theta)$ is reached



Learning GMM by EM

Recall…

1. Initialize the Parameters $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ randomly, or using K-means

2. Iterate until convergence (e.g., when $\log p(X|\Theta)$ ceases to increase

   a) E-step:

$$\gamma_{ik} = p\left(z_i = k \middle| x_i, \Theta^{old}\right) = \frac{\pi_k^{old} \mathcal{N}\left(x_i \middle| \mu_k^{old}, \Sigma_k^{old}\right)}{\sum_{j=1}^{K} \pi_j^{old} \mathcal{N}\left(x_i \middle| \mu_j^{old}, \Sigma_j^{old}\right)}$$

   b) M-step:

$$\pi_k^{new} = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}$$

$$\mu_k^{new} = \frac{\sum_{i=1}^{N} \gamma_{ik} x_i}{\sum_{i=1}^{N} \gamma_{ik}}$$

$$\Sigma_k^{new} = \frac{\sum_{i=1}^{N} \gamma_{ik} (x_i - \mu_k^{new})(x_i - \mu_k^{new})^\top}{\sum_{i=1}^{N} \gamma_{ik}}$$

DIMENSIONALITY REDUCTION

High Dimension Data

Examples of high dimensional data:

- High resolution images (millions of pixels)

High Dimension Data

Examples of high dimensional data:

– Multilingual News Stories
  (vocabulary of hundreds of thousands of words)

The Tribune Prints the News
Hilo Tribune.
We Lead Others Follow
KILAUEA CRATER CONTINUES IN ERUPTION
FOUR DAYS OF ACTIVITY
ANOTHER OUTBREAK OCCURS
HONOLULU GETS APPROPRIATIONS FOR HARBOR AND LIGHTHOUSE

報 功

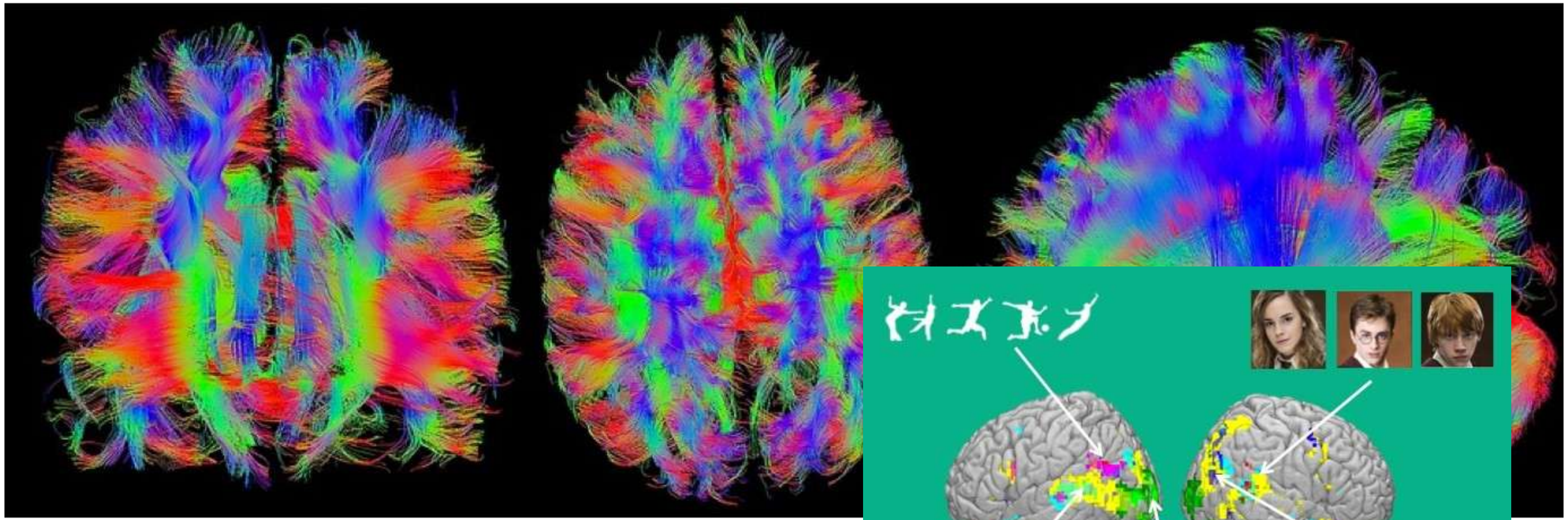
Македония
PRAVA BRAĆA
Истински братя
DEKLARACIJA
Декларация

குடி அரசு
"KUDI-ARASU"

High Dimension Data

Examples of high dimensional data:

– Brain Imaging Data (100s of MBs per scan)



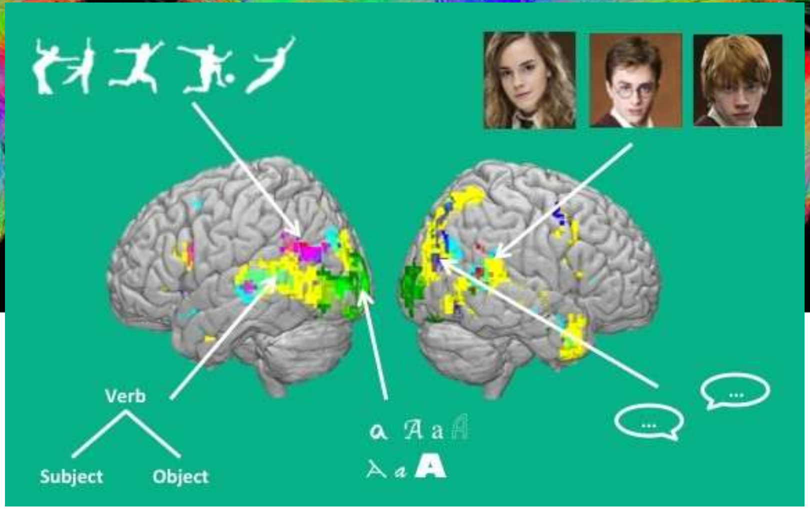


Image from (Wehbe et al., 2014)

Image from https://pixabay.com/en/brain-mrt-magnetic-resonance-imaging-1728449/

Learning Representations

Dimensionality Reduction Algorithms:

Powerful (often unsupervised) learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.

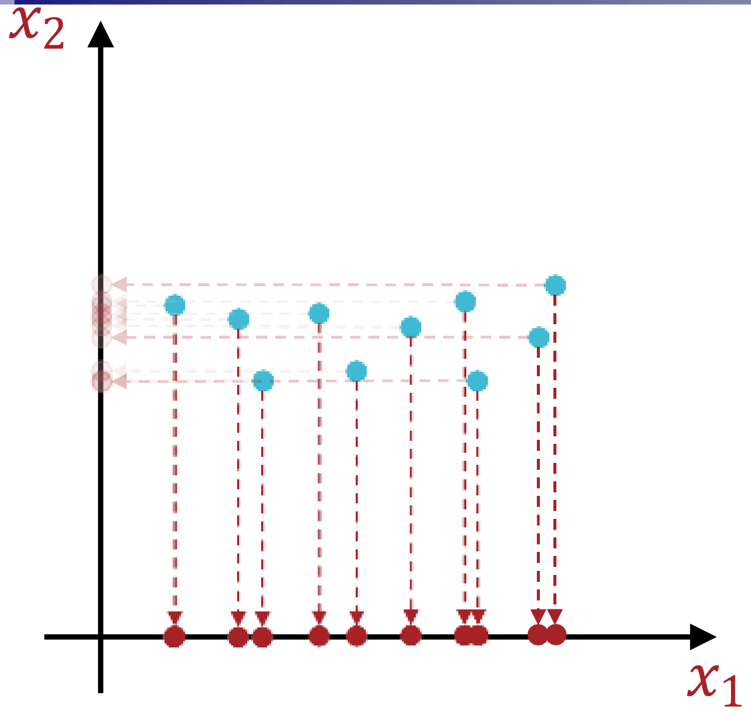
Examples:

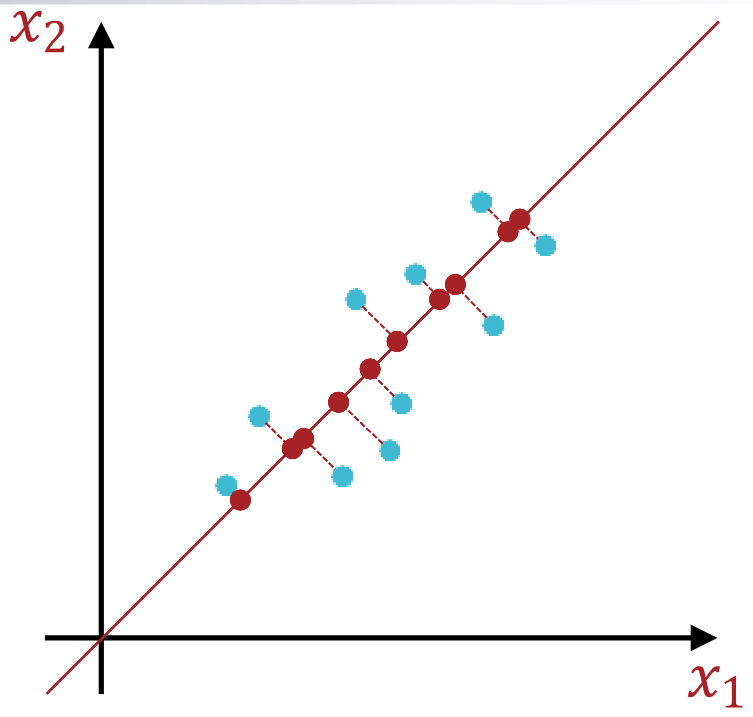
PCA, Kernel PCA, ICA, CCA, t-SNE, Autoencoders, VAEs

Useful for:

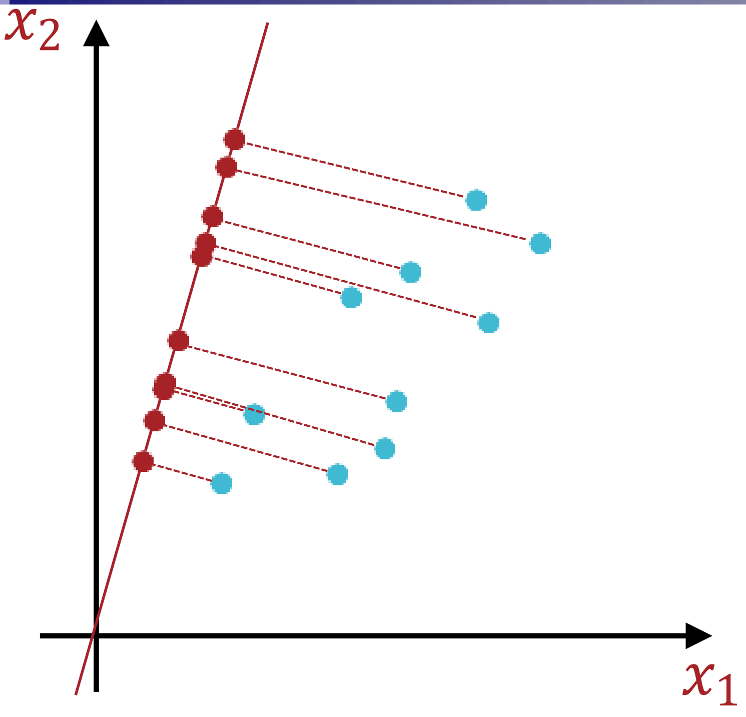
- Visualization
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions $\rightarrow$ better generalization
- Noise removal (improving data quality)

Slide adapted from Nina Balcan



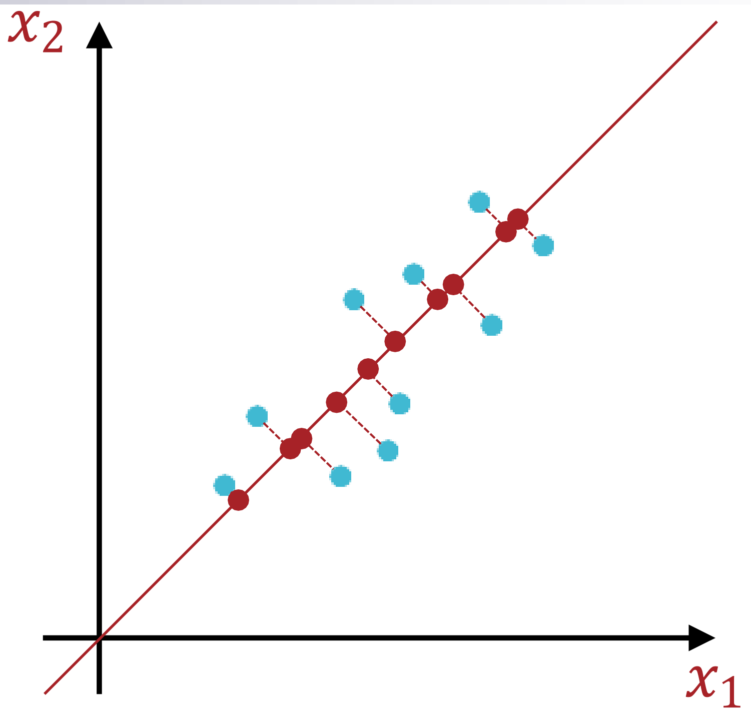


Feature Elimination ∈ Dimensionality Reduction



Option A

Option B (TOXIC)



Option C

Which projection do you prefer (Q1) and why (Q2)?

Background: Sample Variance and Covariance

- Given a collection of N 1-dimensional samples $[x^{(1)}, x^{(2)}, \ldots, x^{(N)}]$ from some random variable, the **sample variance** is

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \hat{\mu}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \frac{1}{N}\sum_{n=1}^{N} x^{(n)}\right)^2$$

- Given a collection of N D-dimensional samples $[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}]$ from some random variable, the **sample covariance** between dimension j and k is

$$\Sigma_{jk} = \frac{1}{N}\sum_{i=1}^{N}\left(x_j^{(i)} - \hat{\mu}_j\right)\left(x_k^{(i)} - \hat{\mu}_k\right) \text{ where } \hat{\mu}_d = \frac{1}{N}\sum_{n=1}^{N} x_d^{(n)}$$

Background: Sample Variance and Covariance

- Given a collection of N 1-dimensional samples $\left[x^{(1)}, x^{(2)}, \dots, x^{(N)}\right]$ from some random variable, the **sample variance** is

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \hat{\mu}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \frac{1}{N}\sum_{n=1}^{N} x^{(n)}\right)^2$$

- Given a collection of N D-dimensional samples $\left[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\right]$ from some random variable, the **sample covariance matrix** is

$$\Sigma = \frac{1}{N} X^T X \quad \text{where} \quad X = \begin{bmatrix} (\boldsymbol{x}^{(1)} - \boldsymbol{\mu})^T \\ (\boldsymbol{x}^{(2)} - \boldsymbol{\mu})^T \\ \vdots \\ (\boldsymbol{x}^{(N)} - \boldsymbol{\mu})^T \end{bmatrix}$$

Centering the Data

- To be consistent, we will constrain principal components to be *orthogonal unit vectors* that begin at the origin
- Preprocess data to be centered around the origin:

1. $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}$
2. $\tilde{\boldsymbol{x}}^{(n)} = \boldsymbol{x}^{(n)} - \boldsymbol{\mu} \; \forall \, n$
3. $X = \begin{bmatrix} \tilde{\boldsymbol{x}}^{(1)^T} \\ \tilde{\boldsymbol{x}}^{(2)^T} \\ \vdots \\ \tilde{\boldsymbol{x}}^{(N)^T} \end{bmatrix}$

Reconstruction Error

- The projection of $\tilde{\boldsymbol{x}}^{(n)}$ onto a vector $\boldsymbol{v}$ is

$$\boldsymbol{z}^{(n)} = \underbrace{\left(\frac{\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}}{\|\boldsymbol{v}\|_2}\right)}_{\text{Length of projection}} \underbrace{\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}}_{\text{Direction of projection}}$$

Reconstruction Error

- The projection of $\tilde{\boldsymbol{x}}^{(n)}$ onto a unit vector $\boldsymbol{v}$ is

$$\mathbf{z}^{(n)} = \left(\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}$$

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\|\tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}\right\|_2^2$$

$$\begin{aligned}
&\left\|\tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}\right\|_2^2 \\
&\quad = {\tilde{\boldsymbol{x}}^{(n)}}^T\tilde{\boldsymbol{x}}^{(n)} - 2\left(\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)} + \left(\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)}\right)\left(\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T\boldsymbol{v} \\
&\quad = {\tilde{\boldsymbol{x}}^{(n)}}^T\tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)} \\
&\quad = \left\|\tilde{\boldsymbol{x}}^{(n)}\right\|_2^2 - \left(\boldsymbol{v}^T\tilde{\boldsymbol{x}}^{(n)}\right)^2
\end{aligned}$$

Minimizing the Reconstruction Error $\Updownarrow$ Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v} \right\|_2^2$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} \right\|_2^2 - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \sum_{n=1}^{N} \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2 \longleftarrow$$

Variance of projections ($\tilde{\boldsymbol{x}}^{(n)}$ are centered)

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T \left(\sum_{n=1}^{N} \tilde{\boldsymbol{x}}^{(n)} \tilde{\boldsymbol{x}}^{(n)^T} \right) \boldsymbol{v}$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X) \boldsymbol{v}$$

Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X)\boldsymbol{v}$$

$$\begin{aligned}\mathcal{L}(\boldsymbol{v},\lambda) &= \boldsymbol{v}^T (X^T X)\boldsymbol{v} - \lambda(\|\boldsymbol{v}\|_2^2 - 1) \\ &= \boldsymbol{v}^T (X^T X)\boldsymbol{v} - \lambda(\boldsymbol{v}^T \boldsymbol{v} - 1)\end{aligned}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = 2(X^T X)\boldsymbol{v} - 2\lambda\boldsymbol{v}$$

$$\rightarrow 2(X^T X)\hat{\boldsymbol{v}} - 2\lambda\hat{\boldsymbol{v}} = 0 \rightarrow (X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}$$

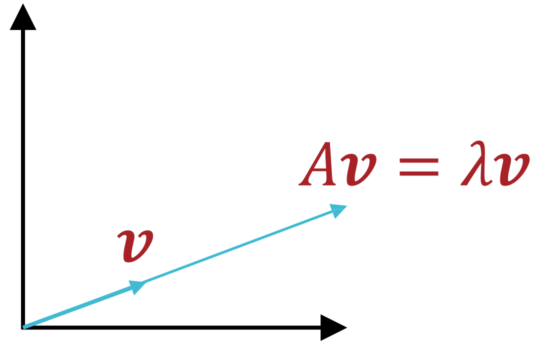
- $\hat{\boldsymbol{v}}$ is an eigenvector of $X^T X$ and λ is the corresponding eigenvalue!

Background: Eigenvectors & Eigenvalues

- Given a square matrix $A \in \mathbb{R}^{N \times N}$, a vector $\boldsymbol{v} \in \mathbb{R}^{N \times 1}$ is an

  eigenvector of A iff there exists some scalar λ such that

$$A\boldsymbol{v} = \lambda\boldsymbol{v}$$



Intuition: A scales or stretches $\boldsymbol{v}$ but does not rotate it

- Key property: the eigenvectors of symmetric matrices (e.g., the covariance matrix of a data set) are orthogonal!

Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X)\boldsymbol{v}$$

$$\begin{aligned}\mathcal{L}(\boldsymbol{v},\lambda) &= \boldsymbol{v}^T (X^T X)\boldsymbol{v} - \lambda(\|\boldsymbol{v}\|_2^2 - 1) \\ &= \boldsymbol{v}^T (X^T X)\boldsymbol{v} - \lambda(\boldsymbol{v}^T\boldsymbol{v} - 1)\end{aligned}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = 2(X^T X)\boldsymbol{v} - 2\lambda\boldsymbol{v}$$

$$\rightarrow 2(X^T X)\hat{\boldsymbol{v}} - 2\lambda\hat{\boldsymbol{v}} = 0 \rightarrow (X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}$$

- $\hat{\boldsymbol{v}}$ is an eigenvector of $X^T X$ and λ is the corresponding eigenvalue!
- But which one?

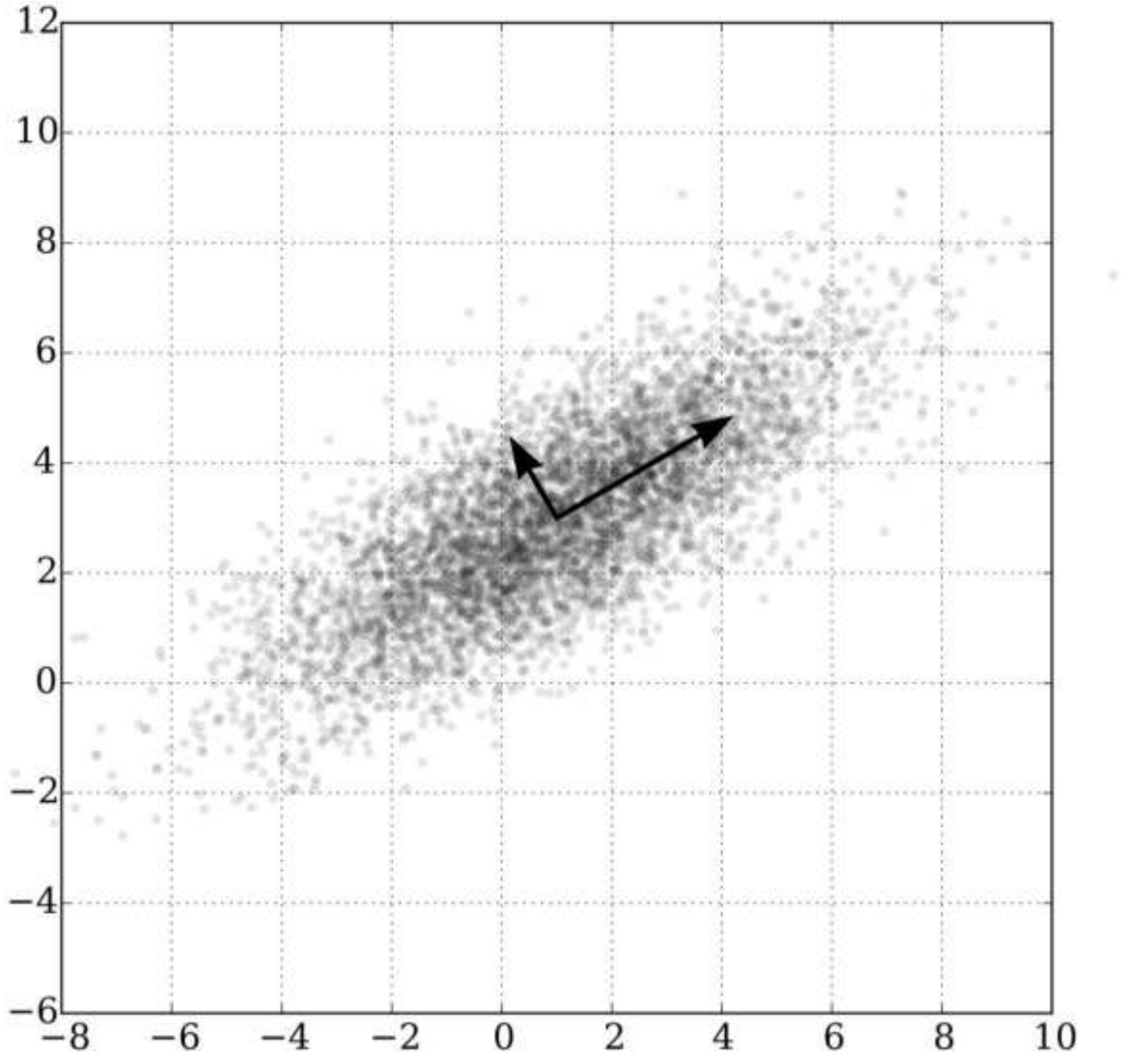
Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}}\, \boldsymbol{v}^T(X^TX)\boldsymbol{v}$$

$$(X^TX)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}} \rightarrow \hat{\boldsymbol{v}}^T(X^TX)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}} = \lambda$$

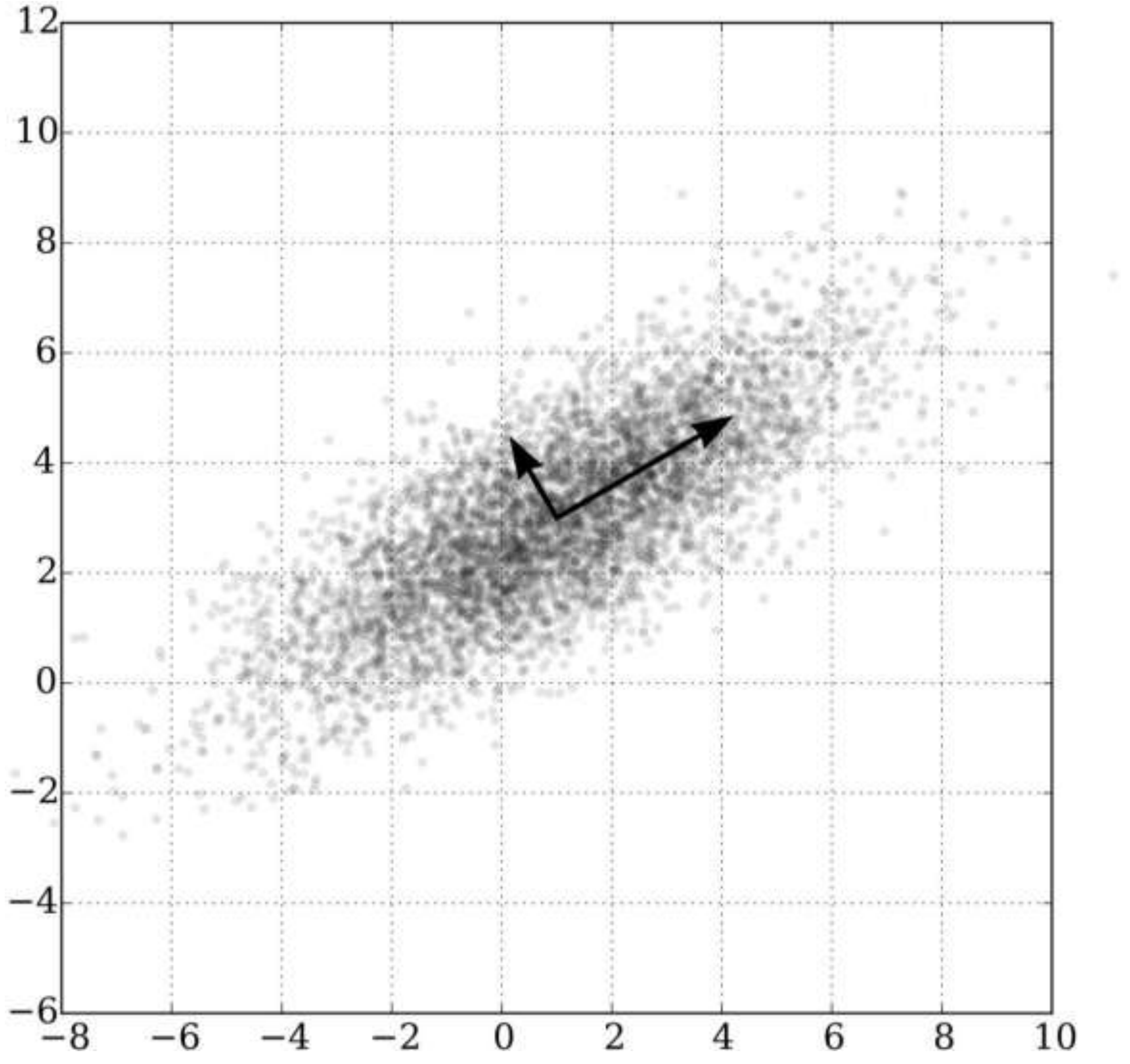
- The first principal component is the eigenvector $\hat{\boldsymbol{v}}_1$ that corresponds to the largest eigenvalue λ_1
- The second principal component is the eigenvector $\hat{\boldsymbol{v}}_2$ that corresponds to the second largest eigenvalue λ_2
  - $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$ are orthogonal
- Etc …
- λ_i is a measure of how much variance falls along $\hat{\boldsymbol{v}}_i$

Principal Components: Example



Source: https://en.wikipedia.org/wiki/Principal_component_analysis#/media/File:GaussianScatterPCA.svg

How can we efficiently find principal components (eigenvectors)?



Source: https://en.wikipedia.org/wiki/Principal_component_analysis#/media/File:GaussianScatterPCA.svg

Singular Value Decomposition (SVD) for PCA

- Every real-valued matrix $X \in \mathbb{R}^{N \times D}$ can be expressed as

$$X = USV^T$$

where:

1. $U \in \mathbb{R}^{N \times N}$ - columns of U are eigenvectors of XX^T
2. $V \in \mathbb{R}^{D \times D}$ - columns of V are eigenvectors of $X^T X$
3. $S \in \mathbb{R}^{N \times D}$ - diagonal matrix whose entries are the eigenvalues of X $\rightarrow$ squared entries are the eigenvalues of XX^T and $X^T X$

PCA Algorithm

- Input: $\mathcal{D} = \{(\boldsymbol{x}^{(n)})\}_{n=1}^{N}, \rho$

1. Center the data
2. Use SVD to compute the eigenvalues and eigenvectors of $X^T X$
3. Collect the top ρ eigenvectors (corresponding to the ρ largest eigenvalues), $V_\rho \in \mathbb{R}^{D \times \rho}$
4. Project the data into the space defined by V_ρ, $Z = X V_\rho$

- Output: Z, the transformed (potentially lower-dimensional) data

How many PCs should we use?

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}\right)\right\}_{n=1}^{N}, \rho$

1. Center the data
2. Use SVD to compute the eigenvalues and eigenvectors of $X^T X$
3. Collect the top ρ eigenvectors (corresponding to the ρ largest eigenvalues), $V_\rho \in \mathbb{R}^{D \times \rho}$
4. Project the data into the space defined by V_ρ, $Z = X V_\rho$

- Output: Z, the transformed (potentially lower-dimensional) data

Choosing the number of PCs

- Define a percentage of explained variance for the i^{th} PC:

$$\lambda_i / \sum \lambda_j$$

- Select all PCs above some threshold of explained variance, e.g., 5%

- Keep selecting PCs until the total explained variance exceeds some threshold, e.g., 90%

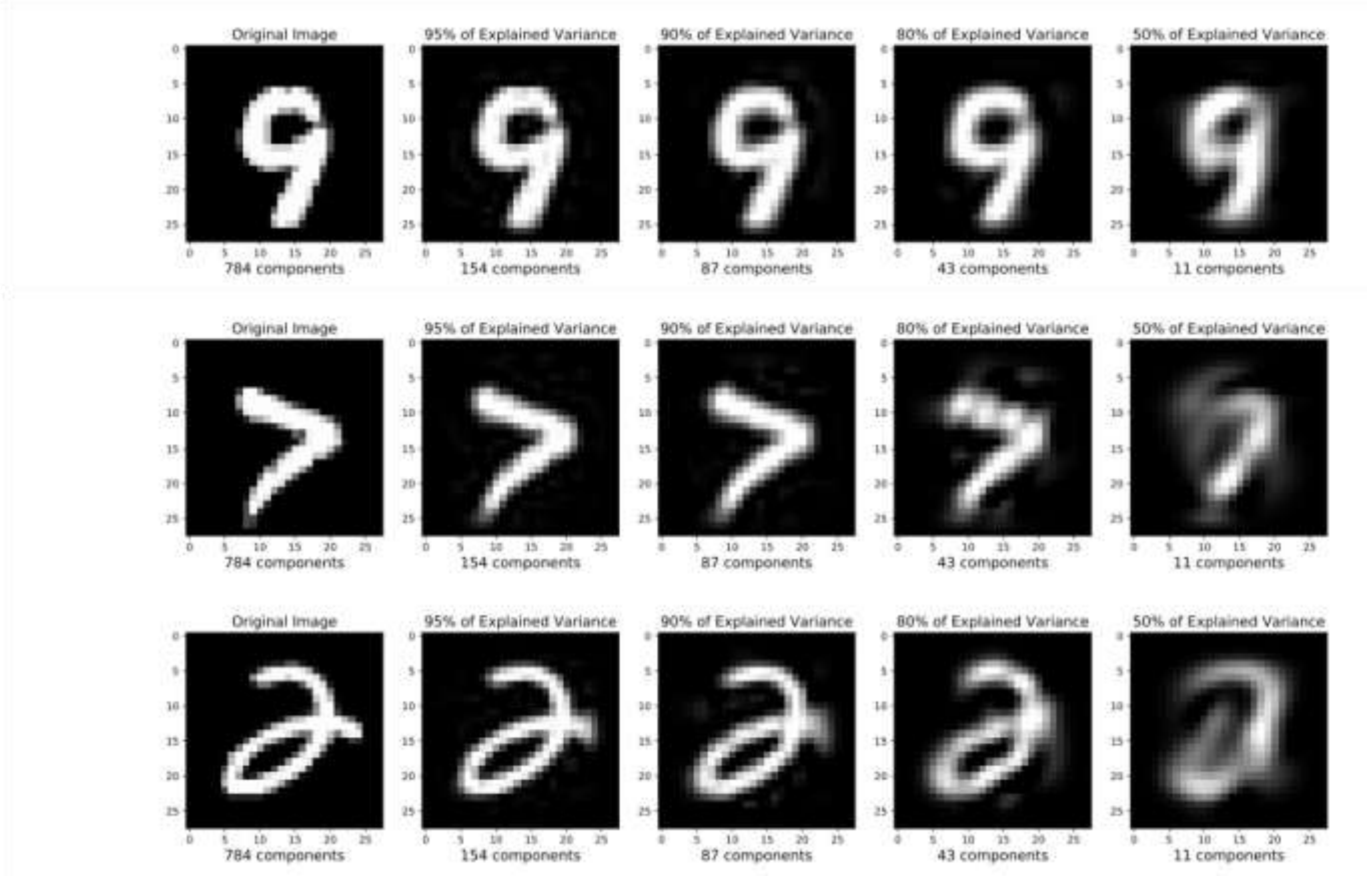
- Evaluate on some downstream metric

PCA EXAMPLES

Projecting MNIST digits

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Report percent of variance explained for K components
3. Then project back up to 28x28 image (i.e. a vector $\tilde{\mathbf{x}}^{(i)}$ of length 784) to visualize how much information was preserved



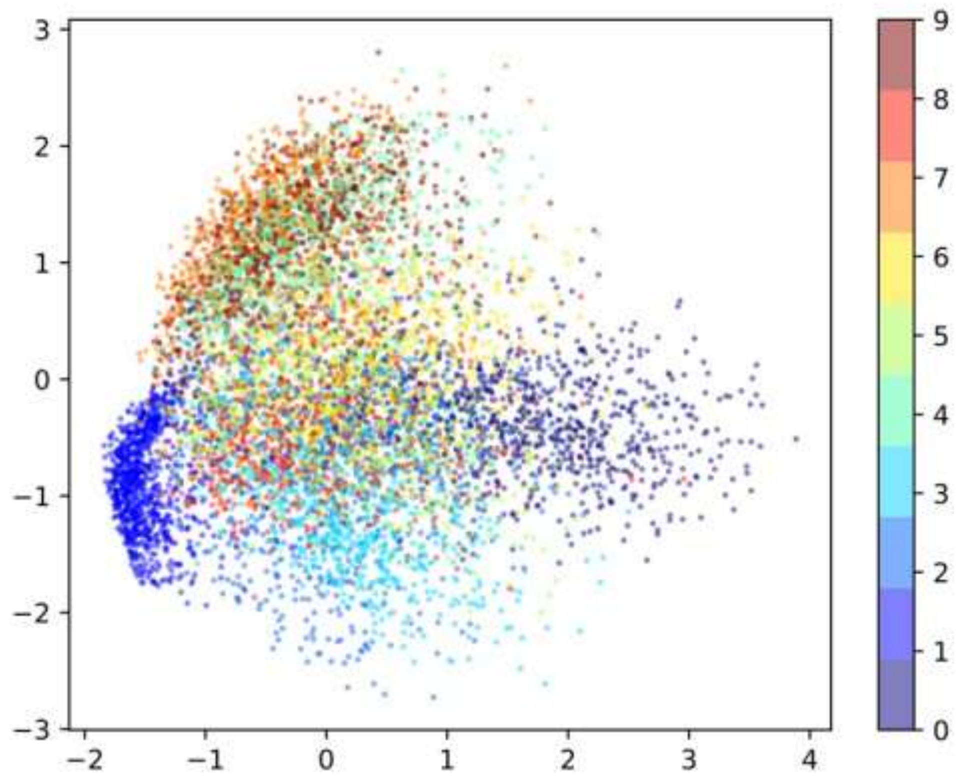
Takeaway: Using fewer principal components K leads to higher reconstruction error.

But even a small number (say 43) still preserves a lot of information about the original image.

Projecting MNIST digits

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K=2 components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Plot the 2 dimensional points $\mathbf{u}^{(i)}$ and label with the (unknown to PCA) label $y^{(i)}$ as the color
3. Here we look at all ten digits 0 - 9



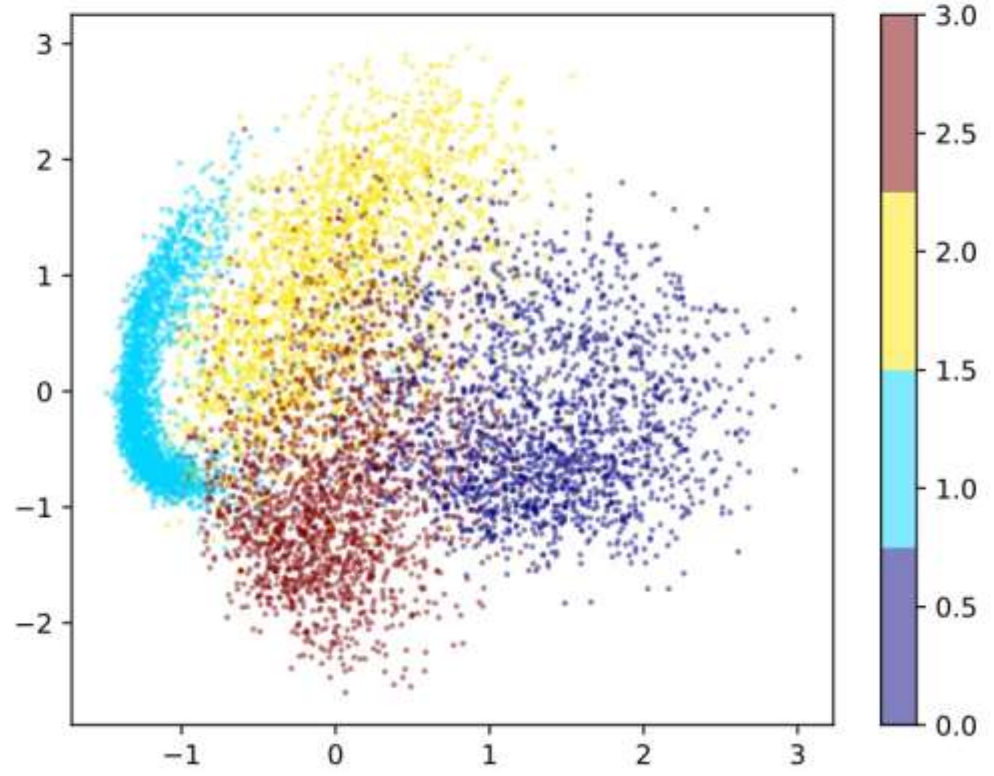
Takeaway: Even with a tiny number of principal components K=2, PCA learns a representation that captures the *latent* information about the type of digit

Projecting MNIST digits

上海科技大学
ShanghaiTech University

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K=2 components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Plot the 2 dimensional points $\mathbf{u}^{(i)}$ and label with the (unknown to PCA) label $y^{(i)}$ as the color
3. Here we look at just four digits 0, 1, 2, 3



Takeaway: Even with a tiny number of principal components K=2, PCA learns a representation that captures the *latent* information about the type of digit

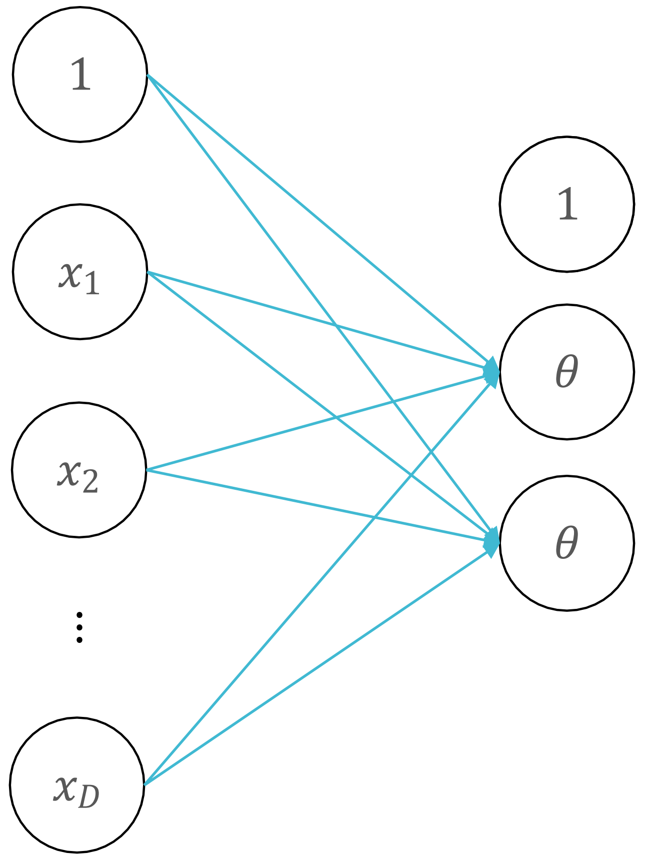
Learning Objectives

Dimensionality Reduction / PCA

You should be able to…

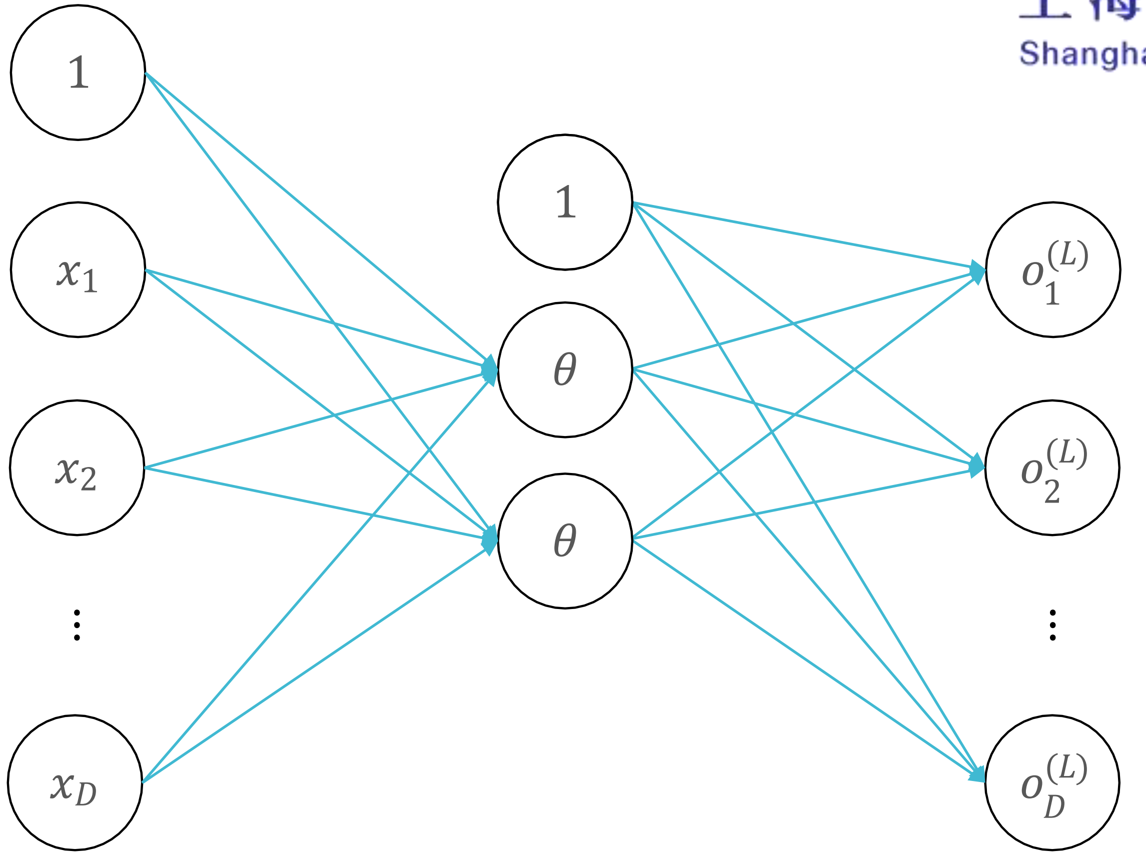
1. Define the sample mean, sample variance, and sample covariance of a vector-valued dataset
2. Identify examples of high dimensional data and common use cases for dimensionality reduction
3. Draw the principal components of a given toy dataset
4. Establish the equivalence of minimization of reconstruction error with maximization of variance
5. Given a set of principal components, project from high to low dimensional space and do the reverse to produce a reconstruction
6. Explain the connection between PCA, eigenvectors, eigenvalues, and covariance matrix
7. Use common methods in linear algebra to obtain the principal components

Autoencoders



Insight: neural networks implicitly learn low-dimensional representations of inputs in hidden layers

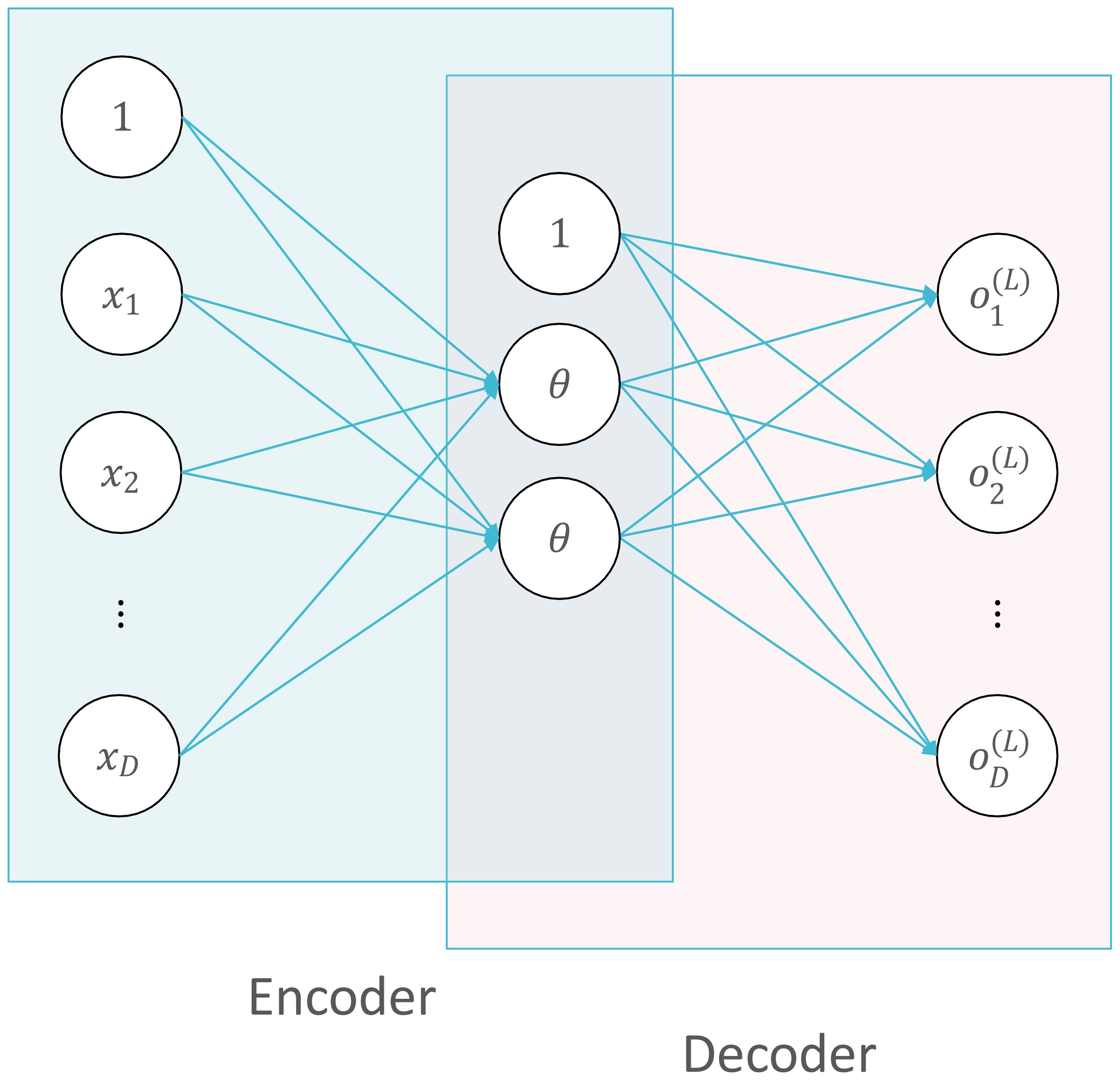
Autoencoders



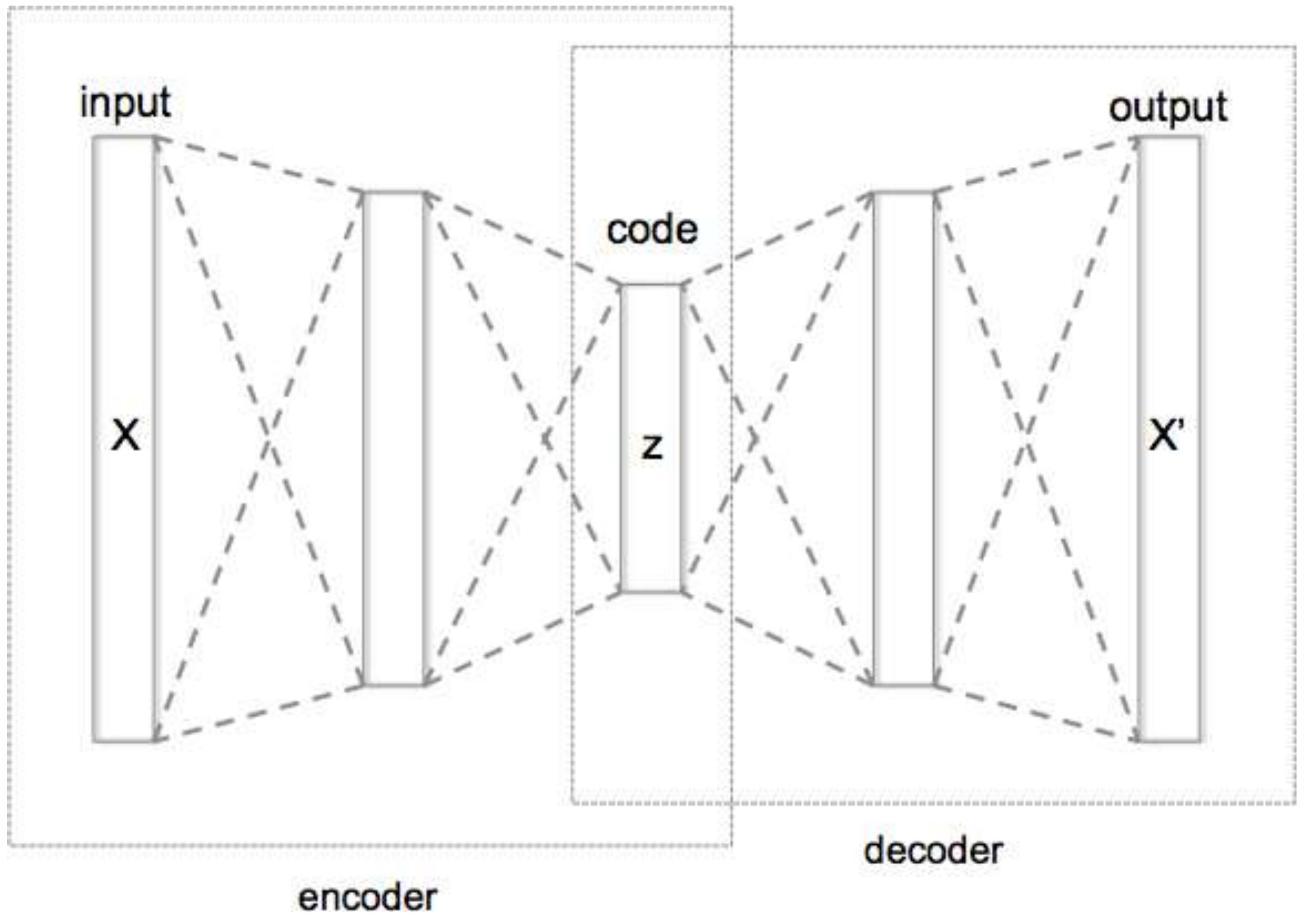
- Learn the weights by minimizing the reconstruction loss:

$$e(\boldsymbol{x}) = \left\|\boldsymbol{x} - \boldsymbol{o}^{(L)}\right\|_2^2$$

Autoencoders



Deep Autoencoders



Source: https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_structure.png