Machine Learning

Lecture 6

Unsupervised Learning

Vincent Adam & **Vicenç Gómez**

2023-2024

Content

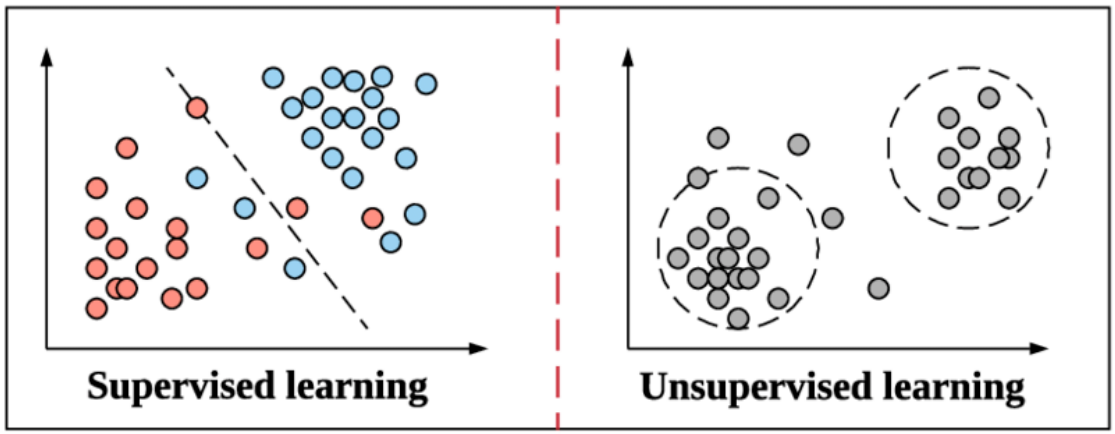
Content

Unsupervised learning

Material

- C. Bishop chap. 9
- D. Mackay pp. 284-292
- Shai[2] pp. 265-272 and pp. 301-305

Unsupervised learning



An unsupervised learning problem consists of:

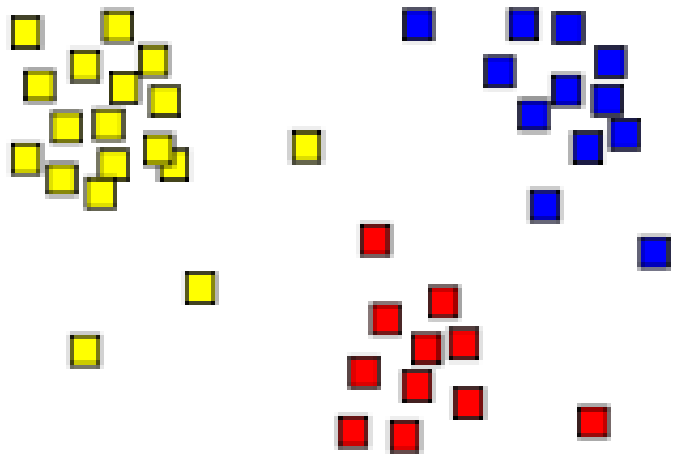
- A domain set $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^d$
- An unknown probability distribution $\mathcal{D}$ on $\mathcal{X}$
- An unlabelled training set $S = (x_1, \ldots, x_m)$ sampled from $\mathcal{D}$

The aim is to find structure in the data

Content

Clustering

Introduction



- Group sets of similar objects into clusters
- Requires a metric that measures distances between inputs
- Used for classification: new inputs are assigned to clusters

Clustering

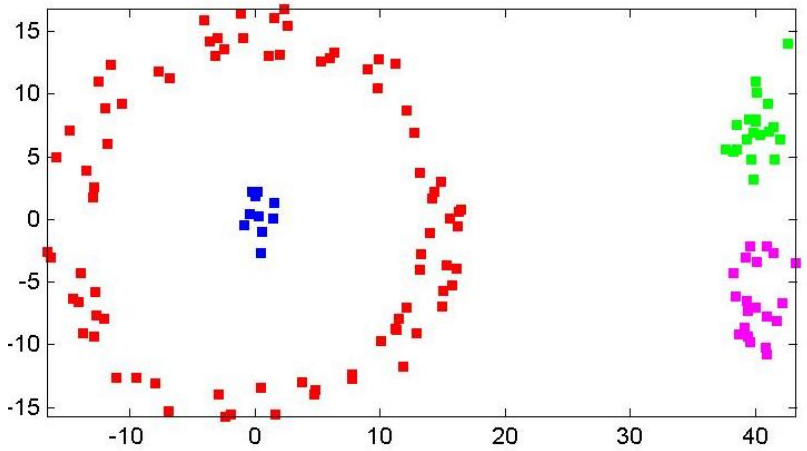
Advantages

- Most commonly used technique for unsupervised learning
- Can help find general patterns in complex data
- Every feature taken into account

Clustering

Disadvantages

- Difficult to define precisely what is meant by “cluster”
- Difficult to determine how many clusters are needed
- Algorithms have many parameters that have to be fine-tuned
- Data points might not be easily separable



- Common definitions of clusters:
  - Small distances between cluster members
  - Particular intervals or statistical distributions
  - Dense areas of the state space
- Clustering can be viewed as a multi-objective optimization problem

Clustering

The k-means algorithm

- Also known as Lloyd's algorithm
- Centroid-based algorithm
- Arguably the most popular clustering algorithm
- k: number of desired clusters
- Assumes distance metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
  - $d(x, x) = 0 \qquad \forall x \in \mathcal{X}$
  - $d(x, y) \geq 0 \qquad \forall (x, y) \in \mathcal{X} \times \mathcal{X}$
  - $d(x, y) = d(y, x) \quad \forall (x, y) \in \mathcal{X} \times \mathcal{X}$

- Given a partition $C_1, \ldots, C_k$ of S, the k-means cost is

$$G_{k-\text{means}}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu(C_i))^2,$$

where $\mu(C_i)$ is the centroid of subset C_i:

$$\mu(C_i) = \arg \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} d(x, \mu')^2$$

- Given a partition $C_1, \ldots, C_k$ of S, the k-means cost is

$$G_{k-\text{means}}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu(C_i))^2,$$

  where $\mu(C_i)$ is the centroid of subset C_i:

$$\mu(C_i) = \arg \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} d(x, \mu')^2$$

- An optimal clustering is the solution to the minimization problem

$$\arg \min_{C_1, \ldots, C_k} G_{k-\text{means}}(C_1, \ldots, C_k)$$

- Problem: NP-hard to minimize

k-means clustering

1. Input: training set $S = (x_1, \ldots, x_m)$, integer k
2. Randomly choose k points as centroids
3. Repeat until convergence:
   - Recompute clusters $C_1, \ldots, C_k$ given the centroids
   - Recompute centroids $\mu_1, \ldots, \mu_k$ given the clusters

Convergence occurs when clusters do not change between two consecutive iterations

- Given centroids $\mu_1, \ldots, \mu_k$, each cluster C_i is computed as

$$C_i = \left\{ x \in S : i = \arg\min_{j=1}^{k} \left\{ d(x, \mu_j)^2 \right\} \right\}$$

Clustering

Clusters and centroids

- Given centroids $\mu_1, \ldots, \mu_k$, each cluster C_i is computed as

$$C_i = \left\{ x \in S : i = \arg\min_{j=1}^{k} \left\{ d(x, \mu_j)^2 \right\} \right\}$$

- Given clusters $C_1, \ldots, C_k$, each centroid μ_i is computed as

$$\mu_i = \mu(C_i) = \arg\min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} d(x, \mu')^2$$

Clustering

Clusters and centroids

- Given centroids $\mu_1, \ldots, \mu_k$, each cluster C_i is computed as

$$C_i = \left\{ x \in S : i = \arg\min_{j=1}^{k} \left\{ d(x, \mu_j)^2 \right\} \right\}$$

- Given clusters $C_1, \ldots, C_k$, each centroid μ_i is computed as

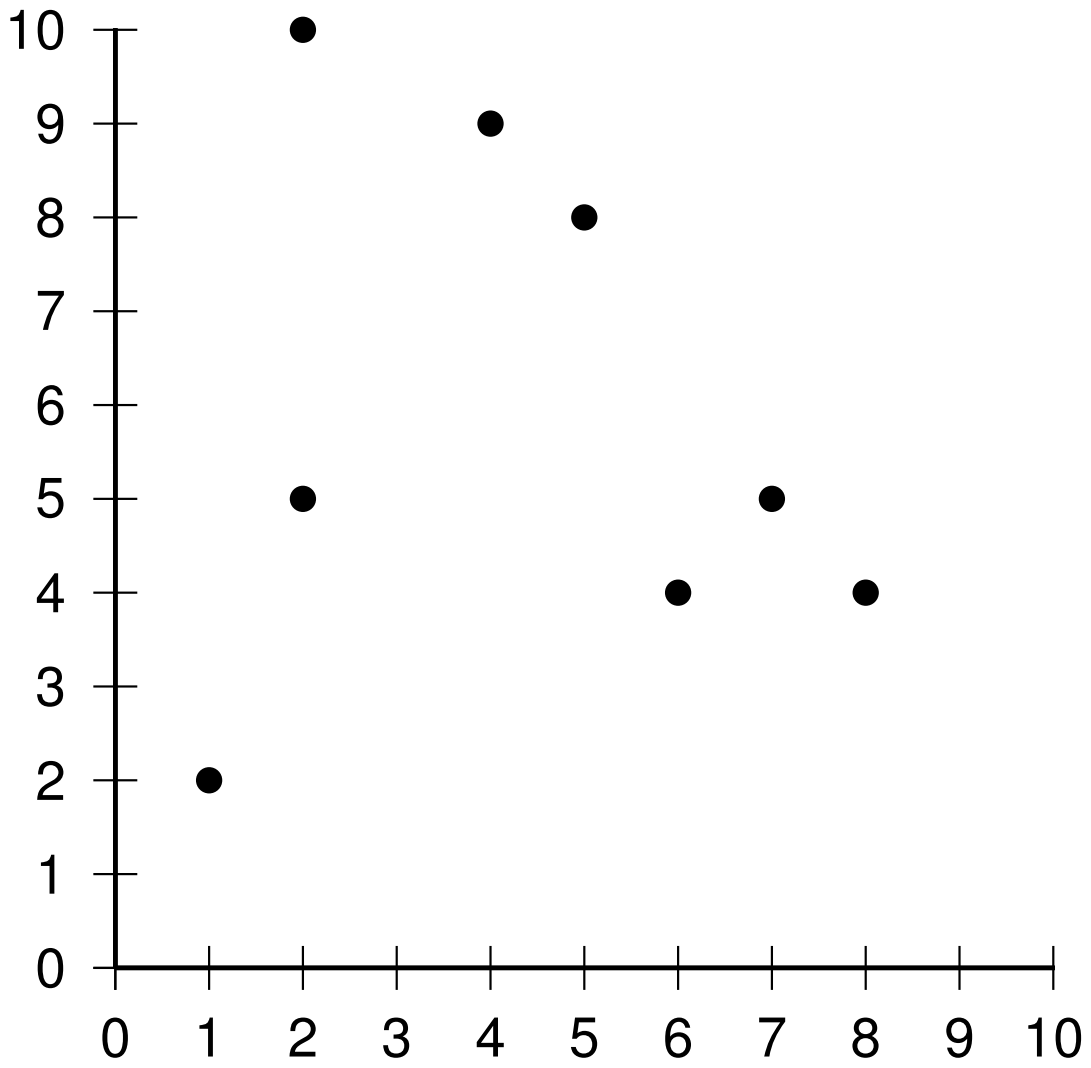
$$\mu_i = \mu(C_i) = \arg\min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} d(x, \mu')^2$$

- In Euclidean space ($d(x, y) = \|x - y\|$), the centroid is given by

$$\mu_i = \mu(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

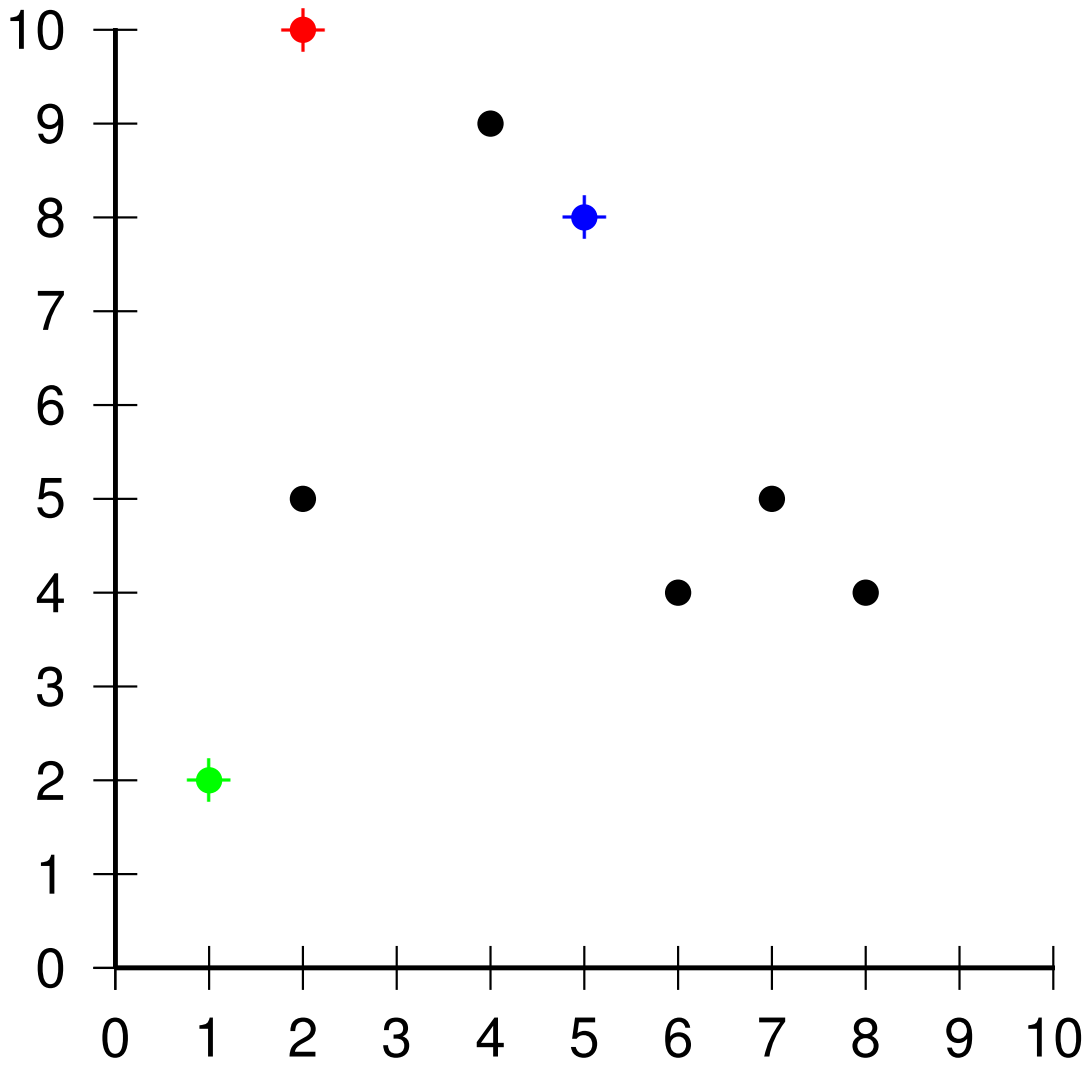
Clustering

K-means: Example



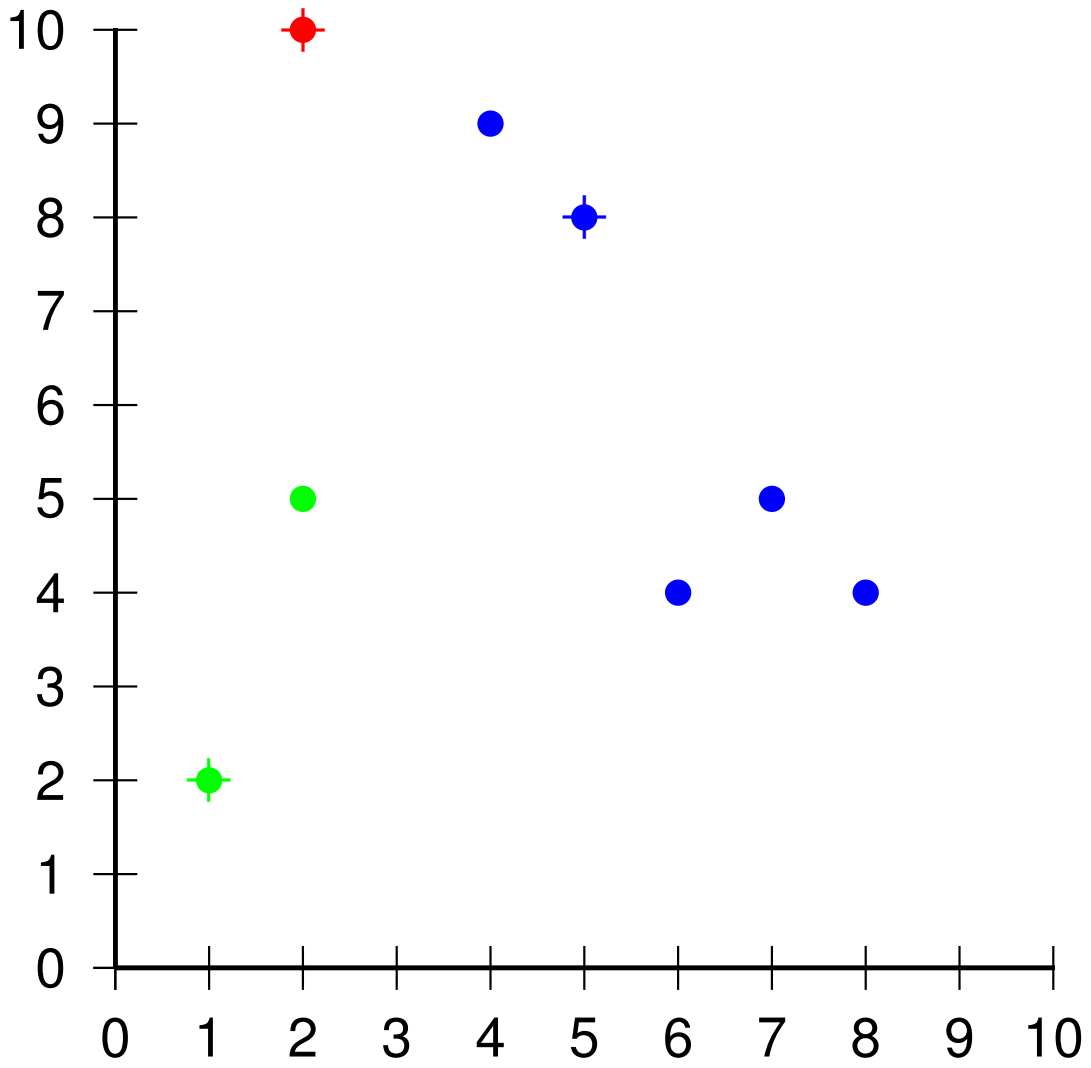
Clustering

K-means: Example



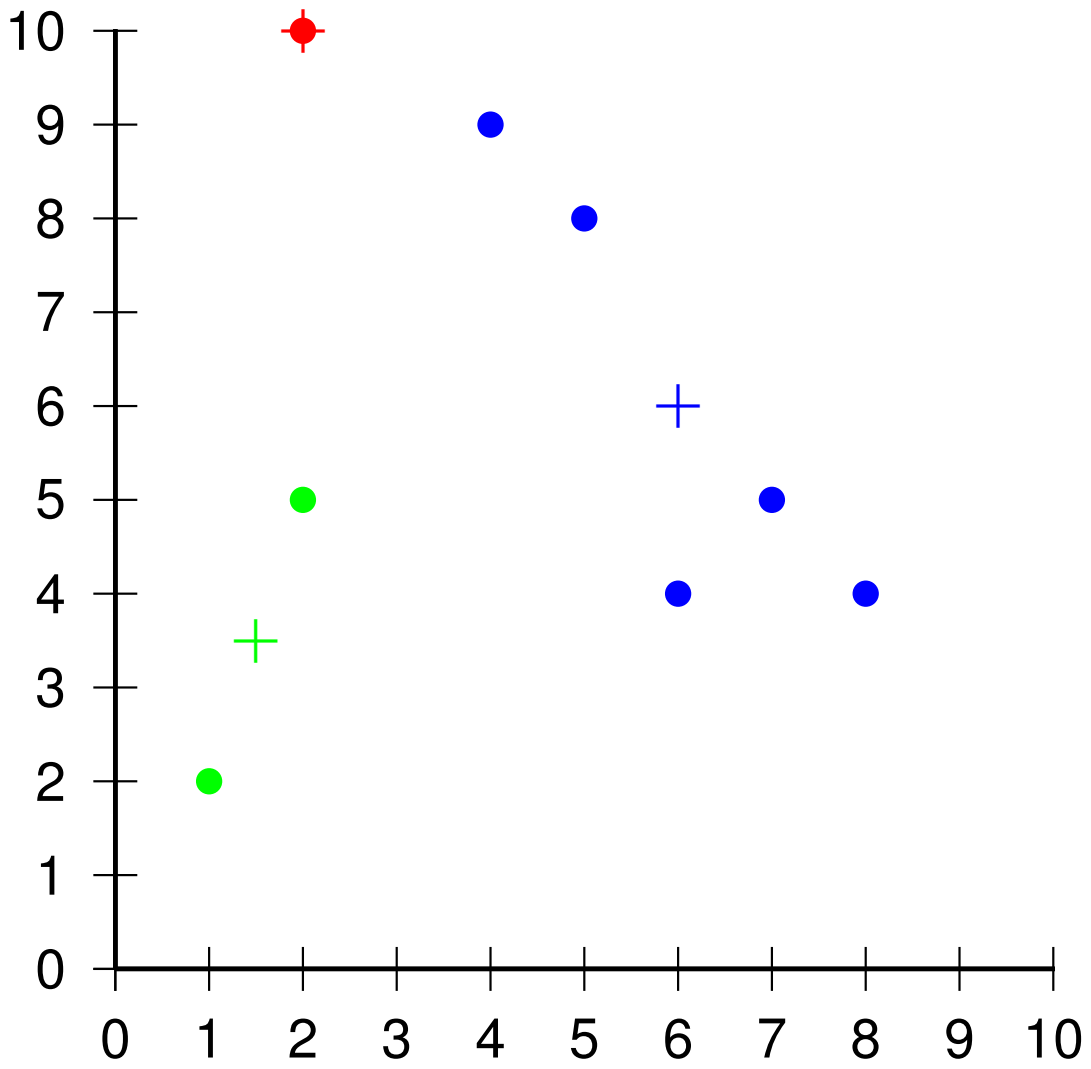
Clustering

K-means: Example



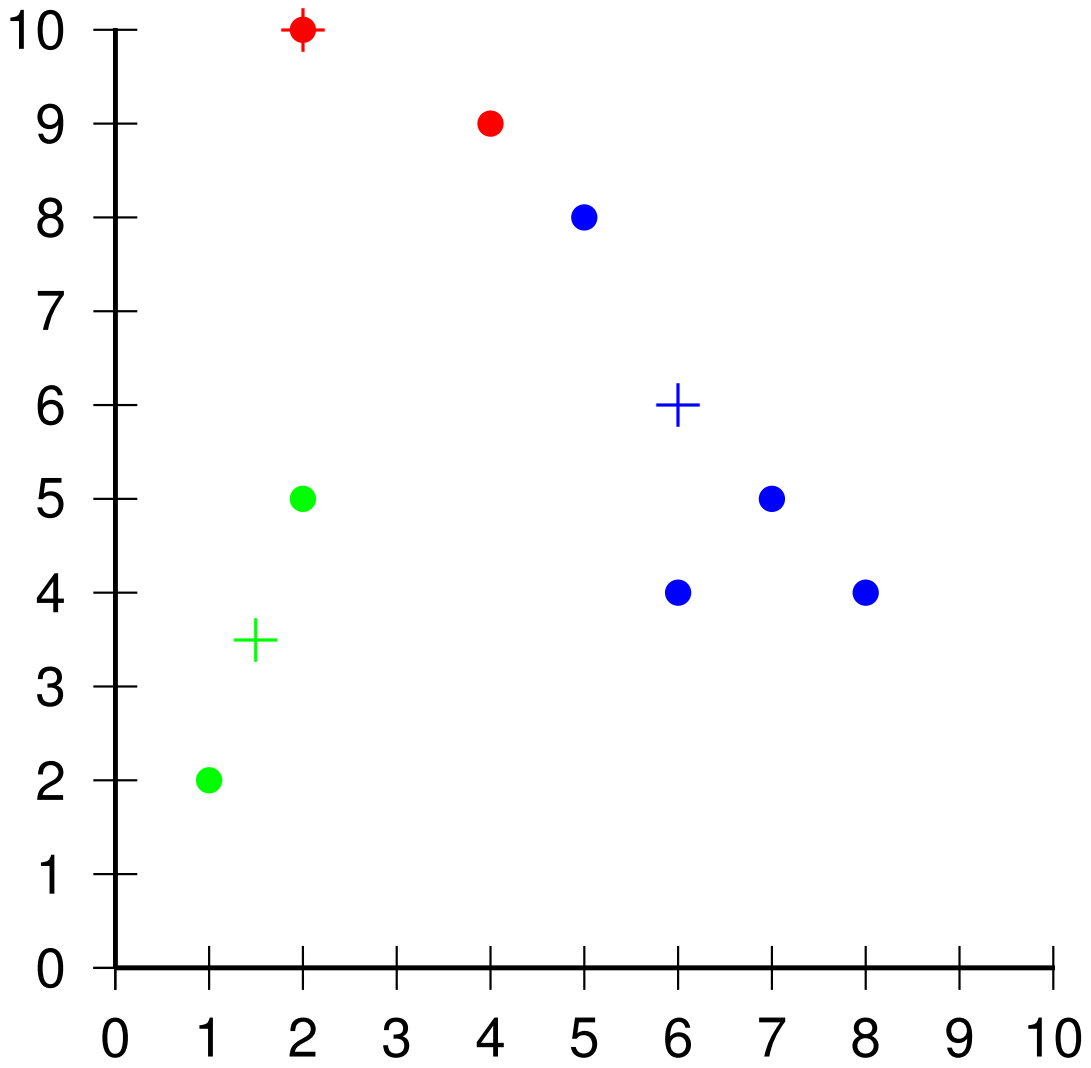
Clustering

K-means: Example



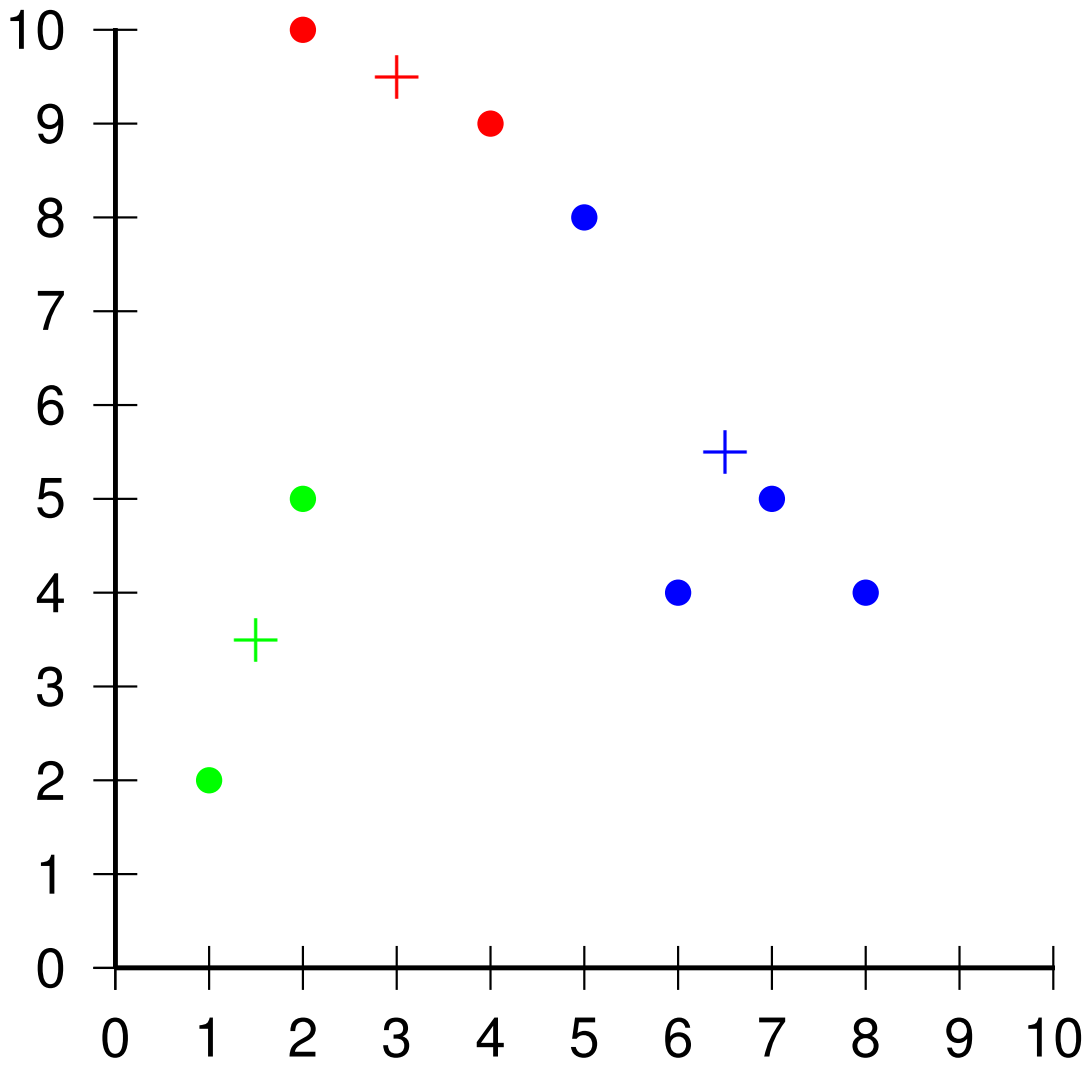
Clustering

K-means: Example



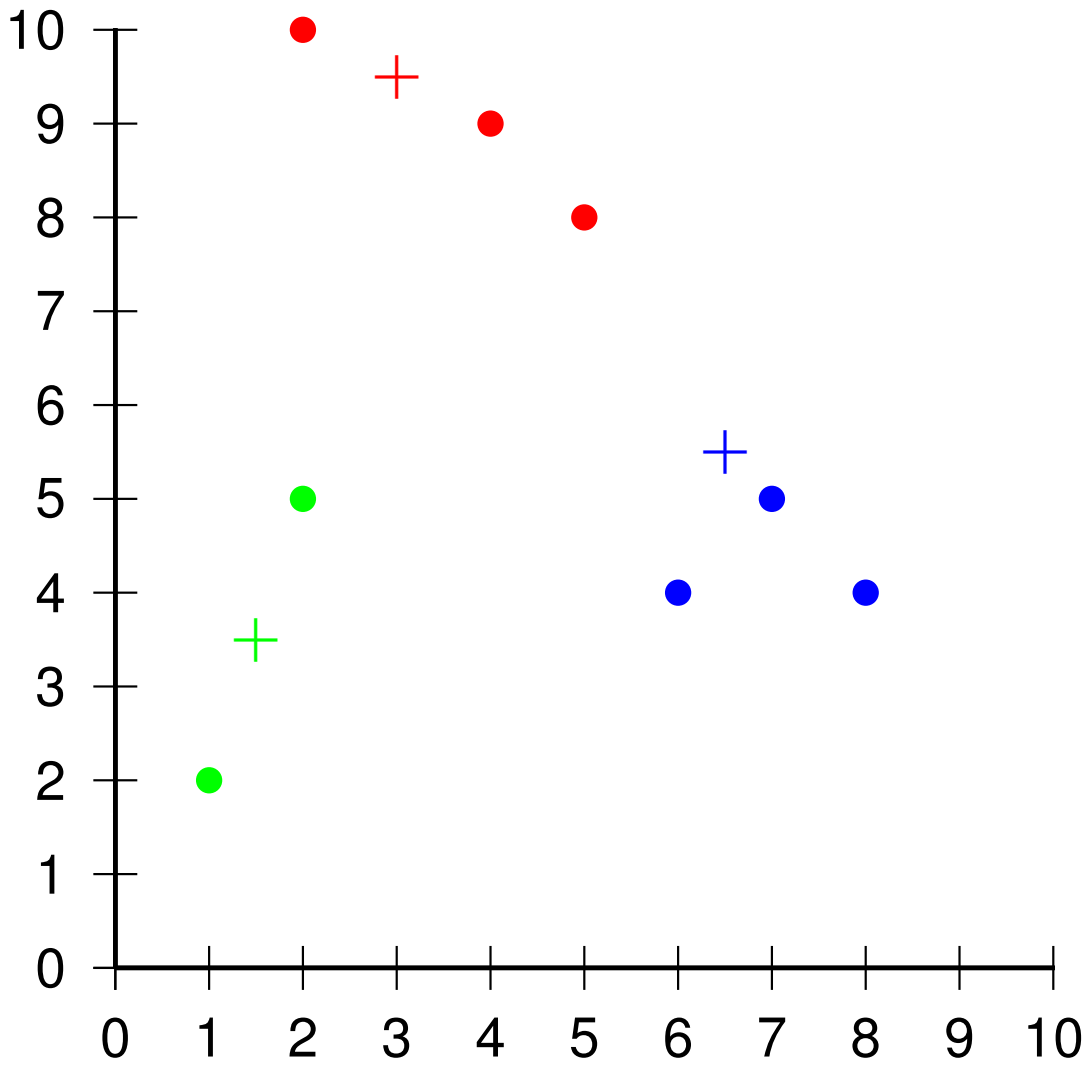
Clustering

K-means: Example



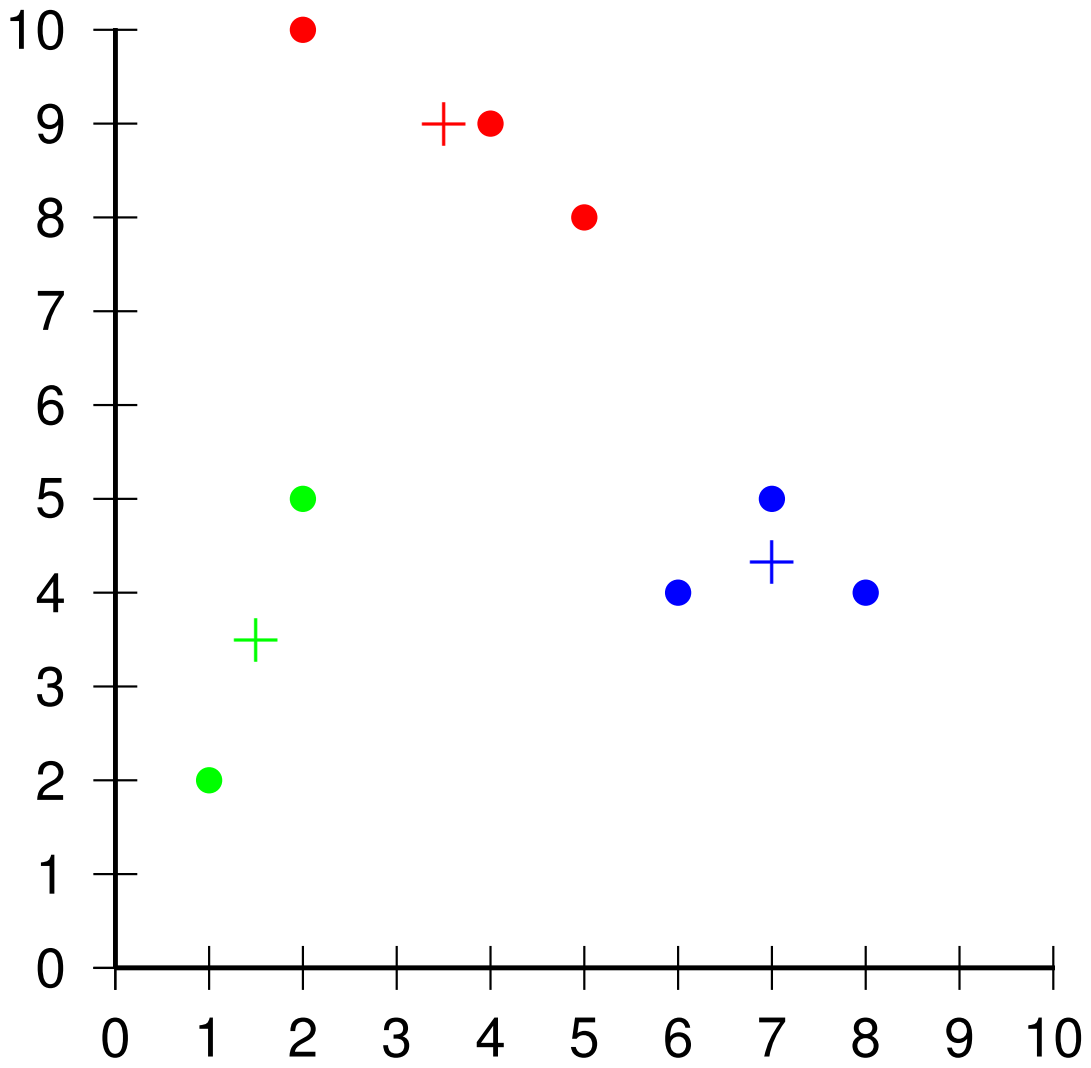
Clustering

K-means: Example



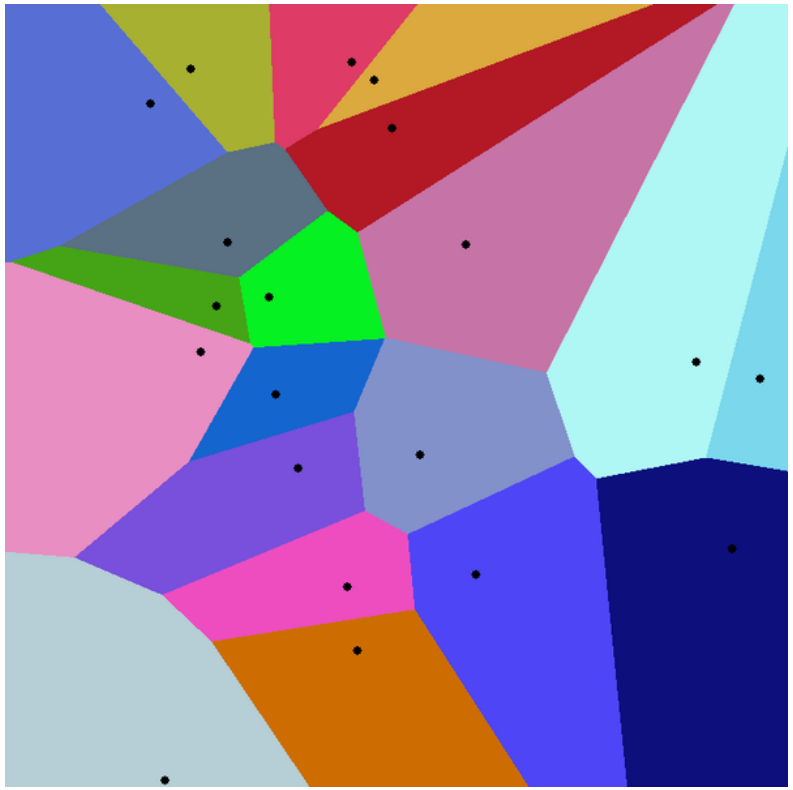
Clustering

K-means: Example



- Greedy algorithm
- Very fast compared to other clustering algorithms
- Tends to find clusters of comparable extent
- Resulting regions are Voronoi cells
- Expectation-maximization (interleaves assignment and update step)

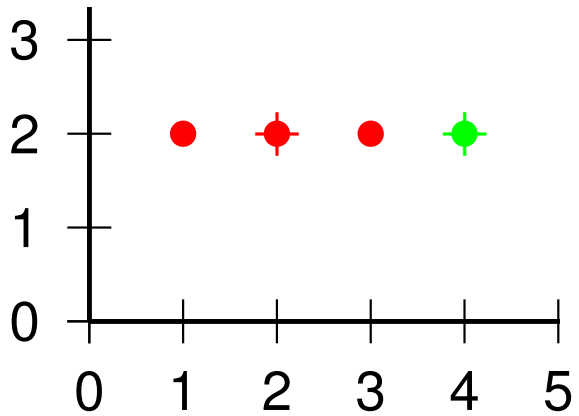
Learned representation of the output



- Highly dependent on the choice of k
- Highly dependent on the choice of initial centroids
- Cannot find clusters with more complicated shapes
- No guarantees on the magnitude of error (compared to optimal)
- Resulting cluster may not even be a local minimum

Clustering

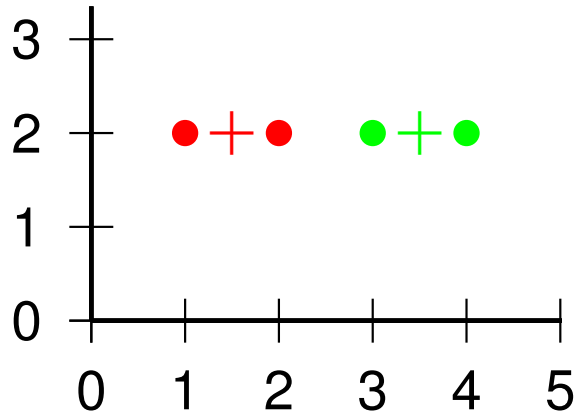
Example



Total cost is $1^2 + 0^2 + 1^2 + 0^2 = 2$

Clustering

Example



Total cost is $(0.5)^2 + (0.5)^2 + (0.5)^2 + (0.5)^2 = 1$

- Random restarts (different choice of initial centroids)
- Random partition (assign an initial cluster to each data point)
- k-medoids: arbitrary distance functions

How to choose k?

- G-means: enforces Gaussianity within clusters
- Elbow method: visual, heuristic
- Silhouette method: heuristic, considers consistency within clusters

`https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html`

- Distribution-based algorithm (generative model)

- Distribution-based algorithm (generative model)
- Each cluster i is a Gaussian distribution with parameters μ_i, Σ_i

$$P(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right)$$

- Distribution-based algorithm (generative model)
- Each cluster i is a Gaussian distribution with parameters μ_i, Σ_i

$$P(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right)$$

- Assumes data points is generated according to:

- Distribution-based algorithm (generative model)
- Each cluster i is a Gaussian distribution with parameters μ_i, Σ_i

$$P(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right)$$

- Assumes data points is generated according to:
  1. The component i is sampled from $P(i) = c_i$

- Distribution-based algorithm (generative model)
- Each cluster i is a Gaussian distribution with parameters μ_i, Σ_i

$$P(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right)$$

- Assumes data points is generated according to:
  1. The component i is sampled from $P(i) = c_i$
  2. The datapoint x is sampled from a Gaussian $P(x|i)$

Clustering

Gaussian mixture models

- Distribution-based algorithm (generative model)
- Each cluster i is a Gaussian distribution with parameters μ_i, Σ_i

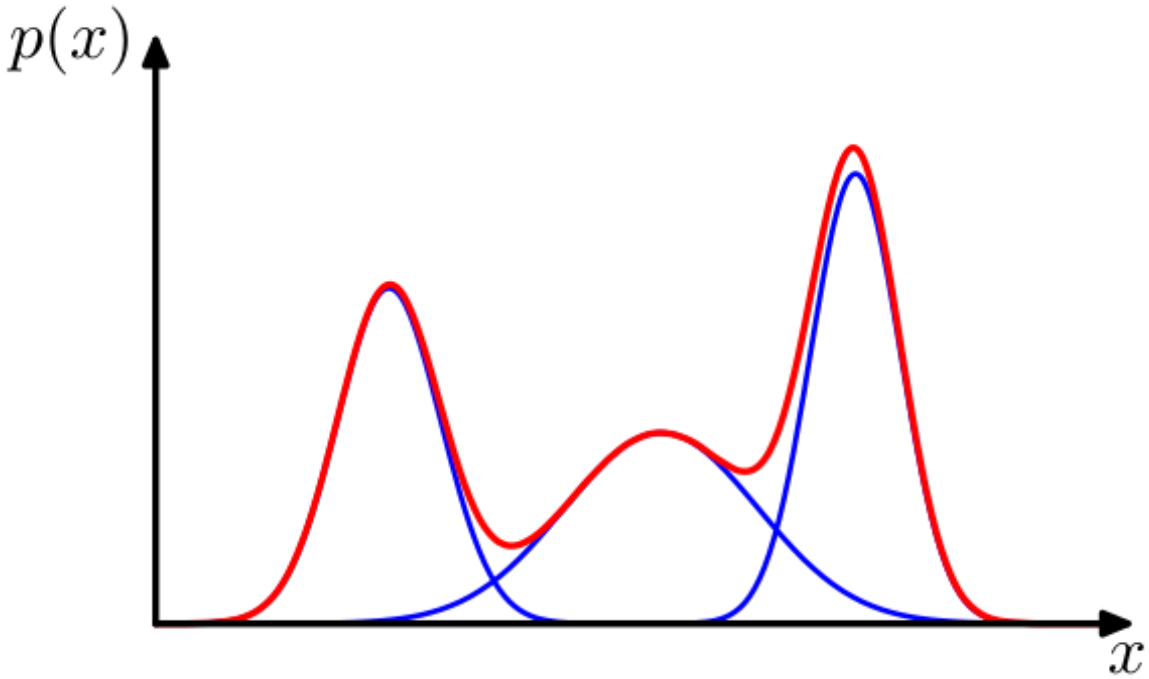
$$P(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right)$$

- Assumes data points is generated according to:
  1. The component i is sampled from $P(i) = c_i$
  2. The datapoint x is sampled from a Gaussian $P(x|i)$
- The model is a linear combination of Gaussians

$$P_\theta(x) = \sum_{i=1}^{k} c_i \mathcal{N}(x; \mu_i, \Sigma_i) \qquad \text{with } \theta = \{c_i, \mu_i, \Sigma_i\}_{i=1}^k$$

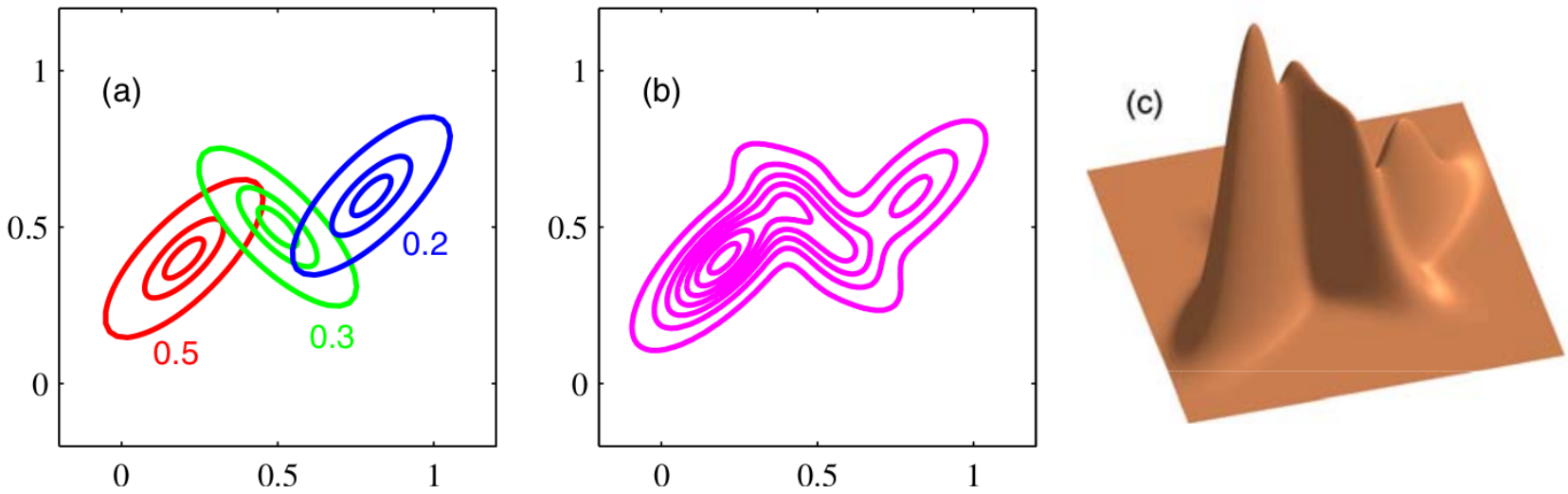
Clustering

GMMs: Example in one dimension



Clustering

GMMs: Example in two dimensions



- Objective: maximize the (log-)likelihood for the entire dataset

$$L(\theta) = \sum_{j=1}^{m} \log \left(\sum_{i=1}^{k} c_i \mathcal{N}(x_j; \mu_i, \Sigma_i) \right)$$

- Objective: maximize the (log-)likelihood for the entire dataset

$$L(\theta) = \sum_{j=1}^{m} \log \left(\sum_{i=1}^{k} c_i \mathcal{N}(x_j; \mu_i, \Sigma_i) \right)$$

- Direct maximization is hard (no closed-form solution)

- Objective: maximize the (log-)likelihood for the entire dataset

$$L(\theta) = \sum_{j=1}^{m} \log \left(\sum_{i=1}^{k} c_i \mathcal{N}(x_j; \mu_i, \Sigma_i) \right)$$

- Direct maximization is hard (no closed-form solution)
- We don't know which component generated x (latent variable)

- Objective: maximize the (log-)likelihood for the entire dataset

$$L(\theta) = \sum_{j=1}^{m} \log \left(\sum_{i=1}^{k} c_i \mathcal{N}(x_j; \mu_i, \Sigma_i) \right)$$

- Direct maximization is hard (no closed-form solution)
- We don't know which component generated x (latent variable)
- Instead, optimize expected (log-) likelihood

$$L'(\theta) = \sum_{j=1}^{m} \sum_{i=1}^{k} P(i|x_j) \log \left(c_i \mathcal{N}(x_j; \mu_i, \Sigma_i) \right)$$

- Objective: maximize the (log-)likelihood for the entire dataset

$$L(\theta) = \sum_{j=1}^{m} \log \left(\sum_{i=1}^{k} c_i \mathcal{N}(x_j; \mu_i, \Sigma_i) \right)$$

- Direct maximization is hard (no closed-form solution)
- We don't know which component generated x (latent variable)
- Instead, optimize expected (log-) likelihood

$$L'(\theta) = \sum_{j=1}^{m} \sum_{i=1}^{k} P(i|x_j) \log \left(c_i \mathcal{N}(x_j; \mu_i, \Sigma_i) \right)$$

- Options: Gradient-based or (more used) Expectation-Maximization

EM for Mixture of Gaussians

1. Input: training set $S = (x_1, \ldots, x_m)$, integer k
2. Randomly form mixture of k Gaussians with parameters

$$\theta = \{c_i, \mu_i, \Sigma_i\}_{i=1}^{k}$$

3. Iterate until convergence:
   - E-step: compute probability of each x belonging to each cluster i

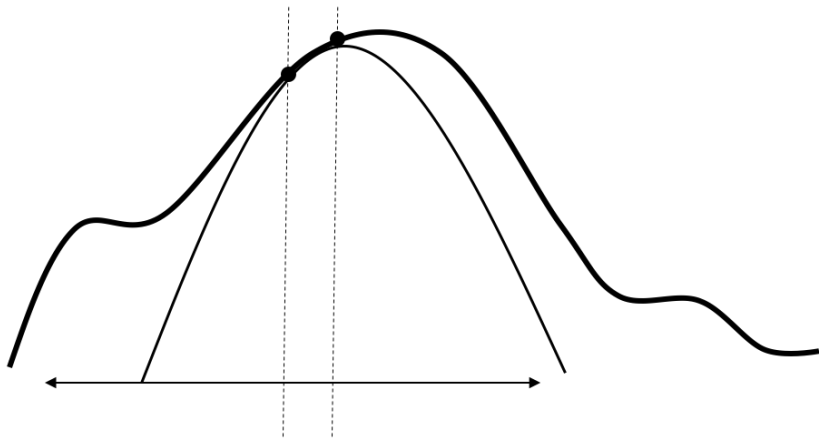
$$P_\theta(i|x)$$

   - M-step: find new parameters θ' for given $P_\theta(i|x)$ (in closed-form)

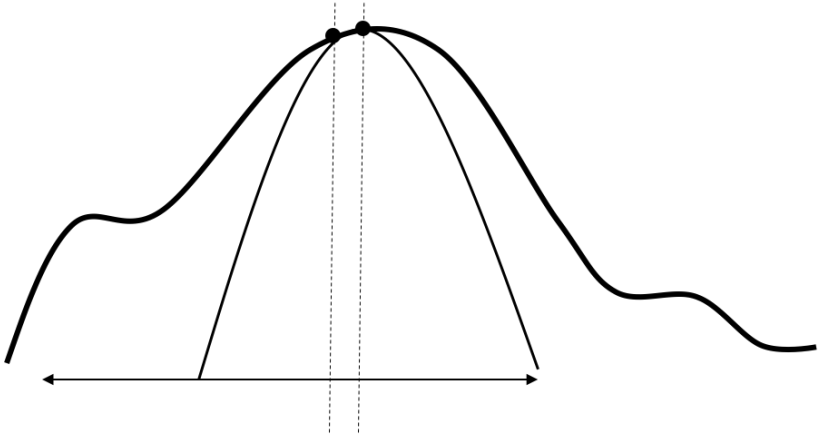
Intuition

Iteration *t*

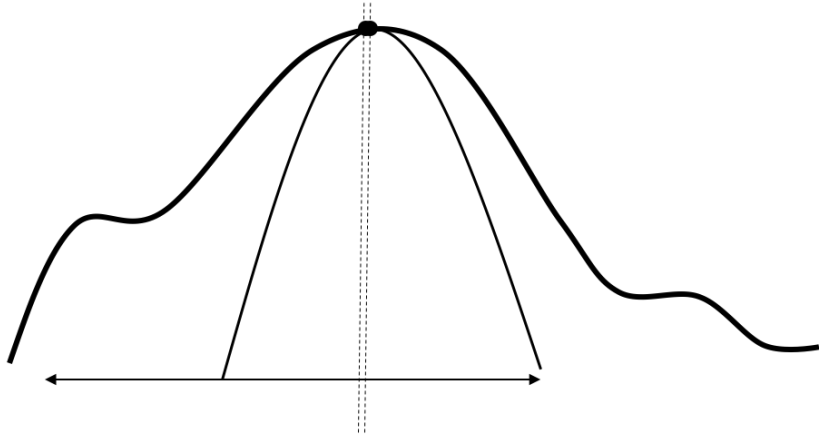
Iteration *t+1*



Iteration *t+2*



Iteration *t+3*



- For $\Sigma_1 = \cdots = \Sigma_k = I$, the expectation step for $x \in S$ is given by

$$P_\theta(i|x) = \frac{1}{Z} P(i) P(x|i) = \frac{1}{Z} c_i \exp\left(-\frac{1}{2}\|x - \mu_i\|^2\right)$$

Z: normalization factor which ensures that $\sum_i P_\theta(i|x) = 1$

- For $\Sigma_1 = \cdots = \Sigma_k = I$, the maximization step is given by

$$\arg \max_{c,\mu_1,\ldots,\mu_k} \sum_{j=1}^{m} \sum_{i=1}^{k} P_\theta(i|x_j) \log \left(c_i \exp \left(-\frac{1}{2} \|x_j - \mu_i\|^2 \right) \right)$$

$$= \arg \max_{c,\mu_1,\ldots,\mu_k} \sum_{j=1}^{m} \sum_{i=1}^{k} P_\theta(i|x_j) \left(\log(c_i) - \frac{1}{2} \|x_j - \mu_i\|^2 \right)$$

- For $\Sigma_1 = \cdots = \Sigma_k = I$, the maximization step is given by

$$\arg \max_{c,\mu_1,\ldots,\mu_k} \sum_{j=1}^{m} \sum_{i=1}^{k} P_\theta(i|x_j) \log \left(c_i \exp \left(-\frac{1}{2} \|x_j - \mu_i\|^2 \right) \right)$$

$$= \arg \max_{c,\mu_1,\ldots,\mu_k} \sum_{j=1}^{m} \sum_{i=1}^{k} P_\theta(i|x_j) \left(\log(c_i) - \frac{1}{2} \|x_j - \mu_i\|^2 \right)$$

- Analytical solution:

$$\mu_i = \frac{\sum_{j=1}^{m} P_\theta(i|x_j) x_j}{\sum_{j=1}^{m} P_\theta(i|x_j)},$$

$$c_i = \frac{\sum_{j=1}^{m} P_\theta(i|x_j)}{\sum_{i'=1}^{k} \sum_{j=1}^{m} P_\theta(i'|x_j)}.$$

Other settings, depending on covariance model:

- **Isotropic**: diagonal covariance with constant value along diagonal

$$\Sigma = \sigma^2 I$$

- **Diagonal**: diagonal covariance with different values along diagonal

$$\Sigma = \text{diag}(\sigma_d^2)$$

- **Full**: no assumption

Clustering

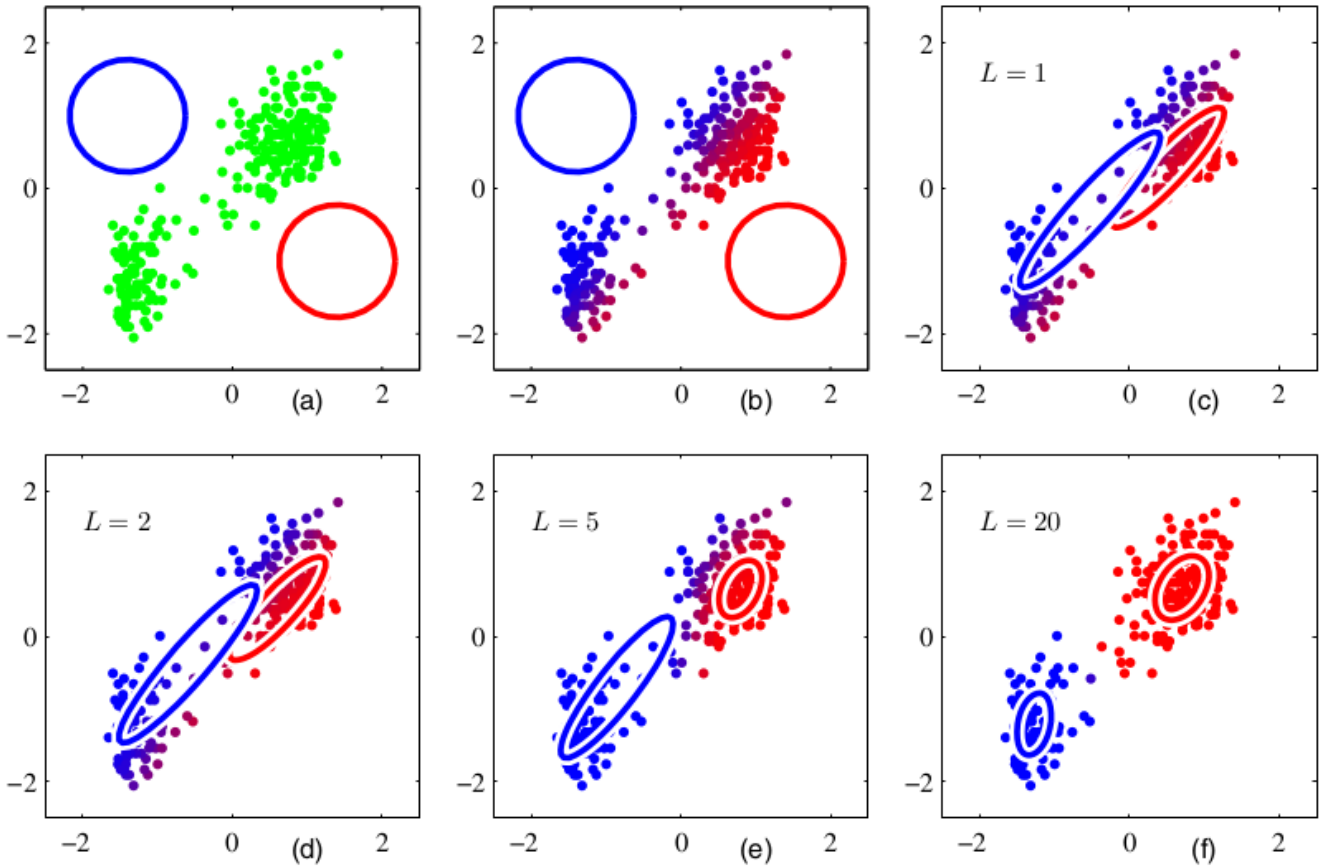
GMMs compared to k-means

- Soft assignment: probability $P(i|x)$ of belonging to cluster i
- Mean ($\approx$ centroid): weighted average of all data points
- Mixture of Gaussians is often called soft k-means
- k-means is a special case where
  - The responsibilities are binary
  - Only means are updated
  - Covariances are assumed diagonal and fixed

Clustering

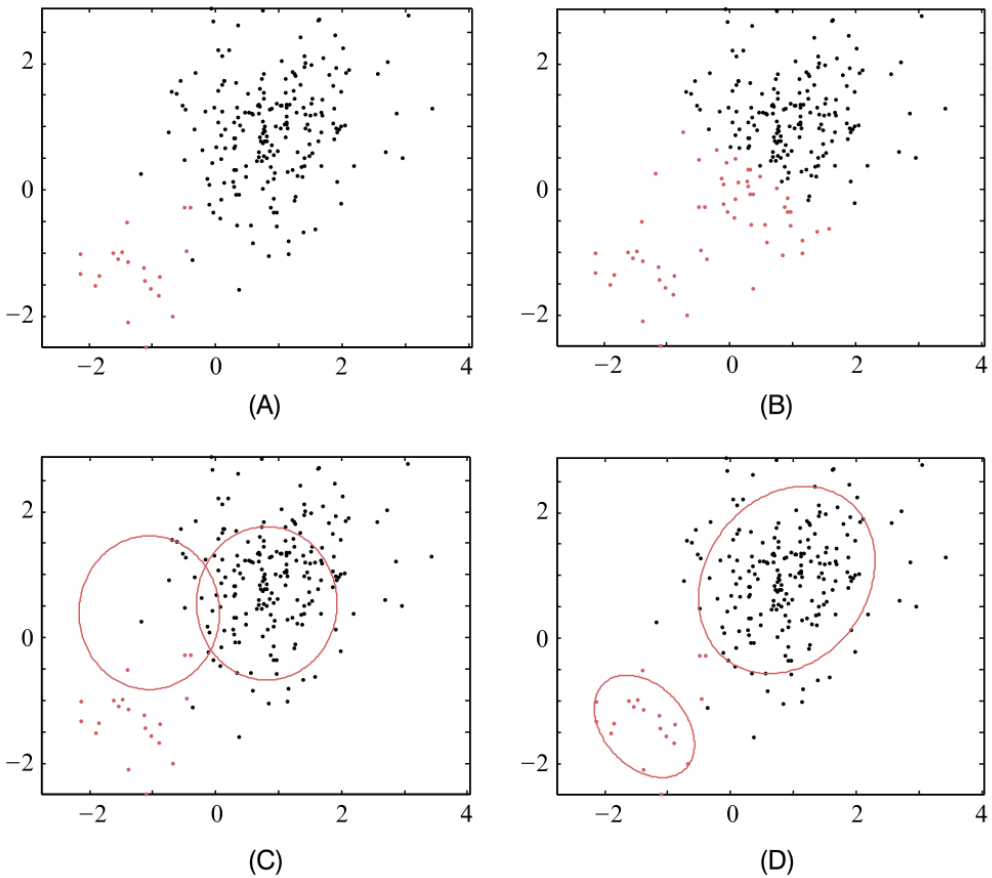
Example of Gaussian Mixture Model

- Using two components and arbitrary covariances



Clustering

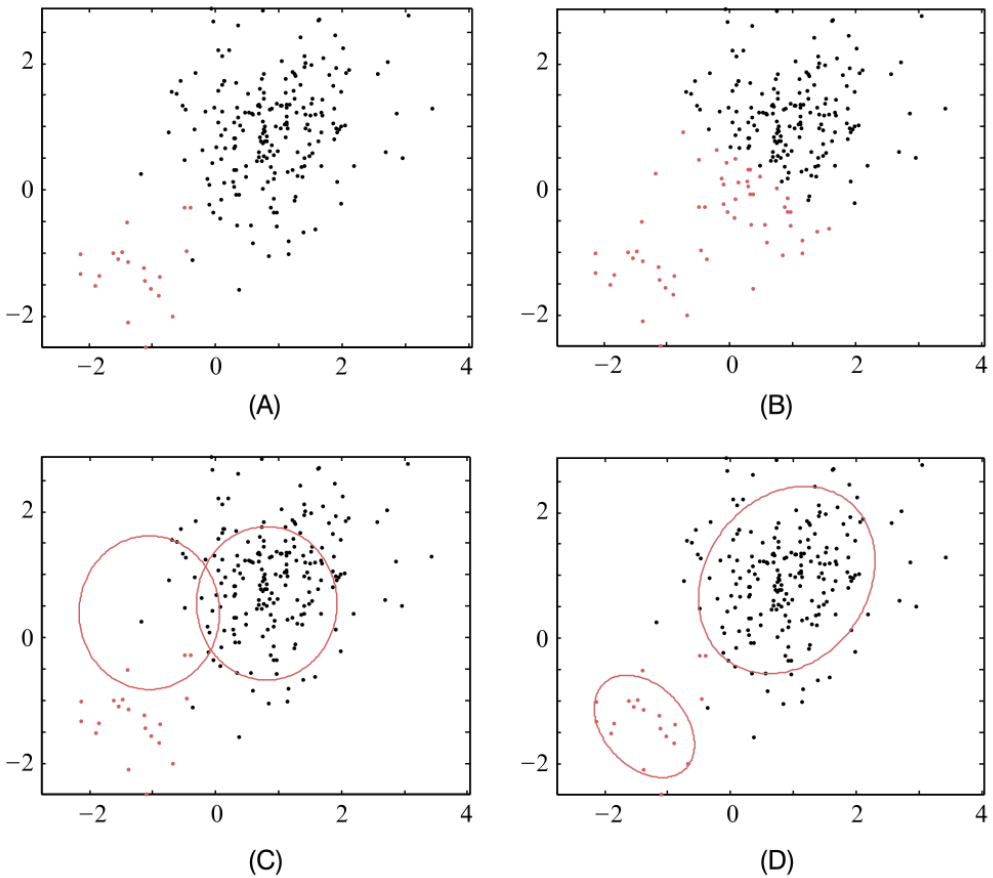
Example of GMMs compared to K-means



- (A) Data: two *unbalanced* clusters

Clustering

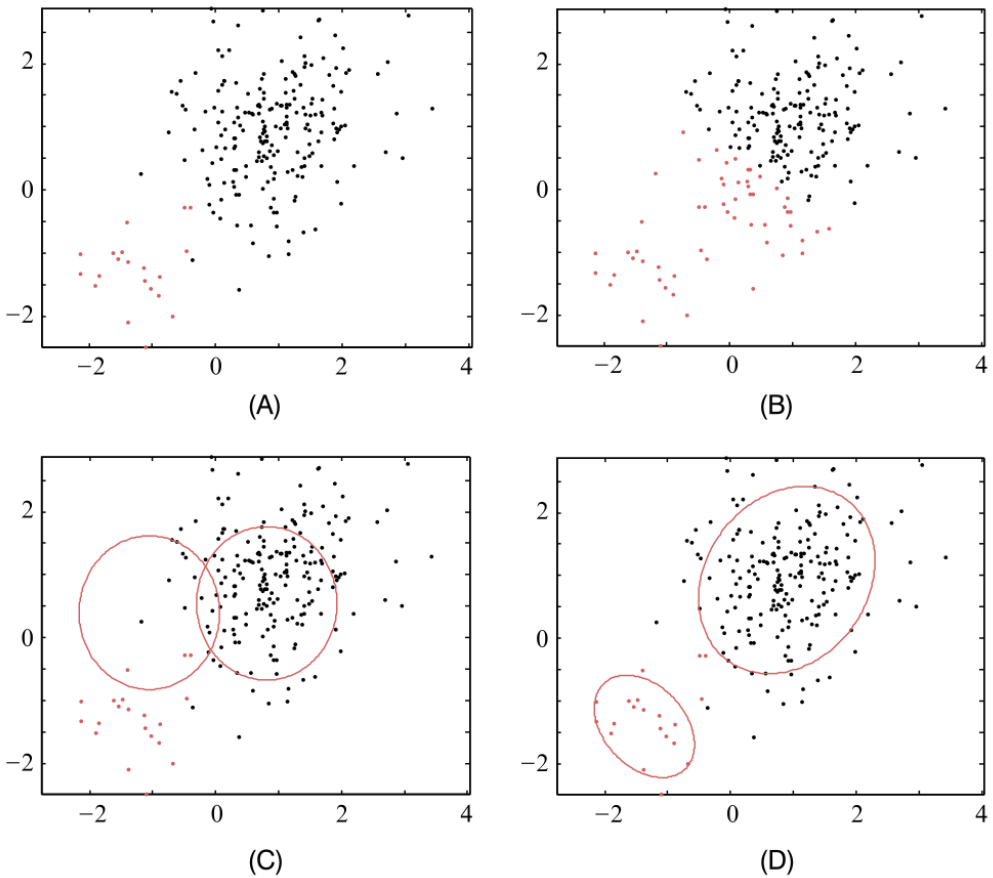
Example of GMMs compared to K-means



- (B) K-means assigns wrongly points to the majority component

Clustering

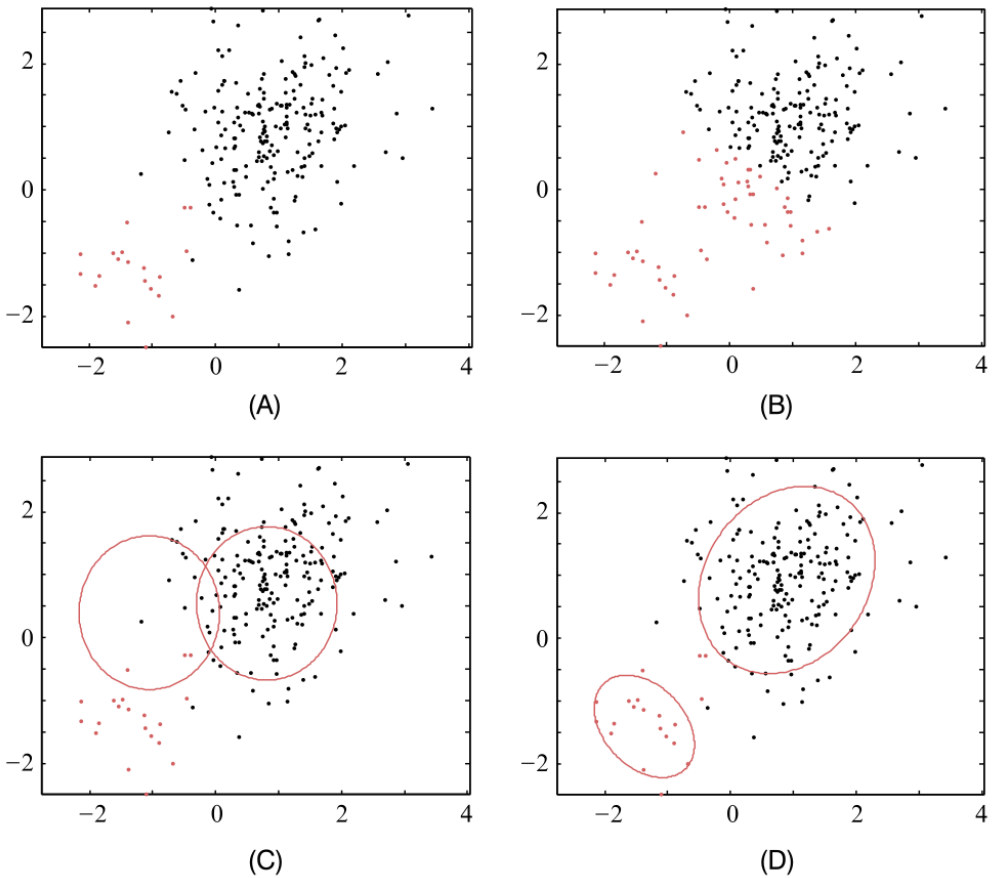
Example of GMMs compared to K-means



- (C) A random initialization for GMM

Clustering

Example of GMMs compared to K-means

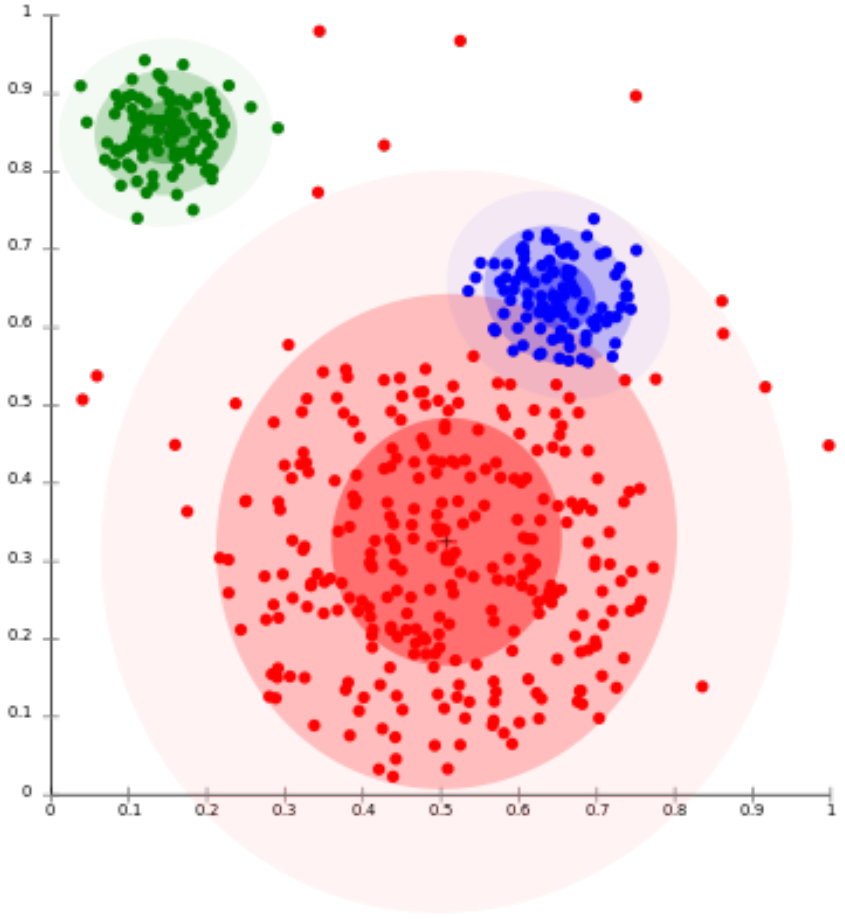


- (D) GMM result correctly captures the clustering

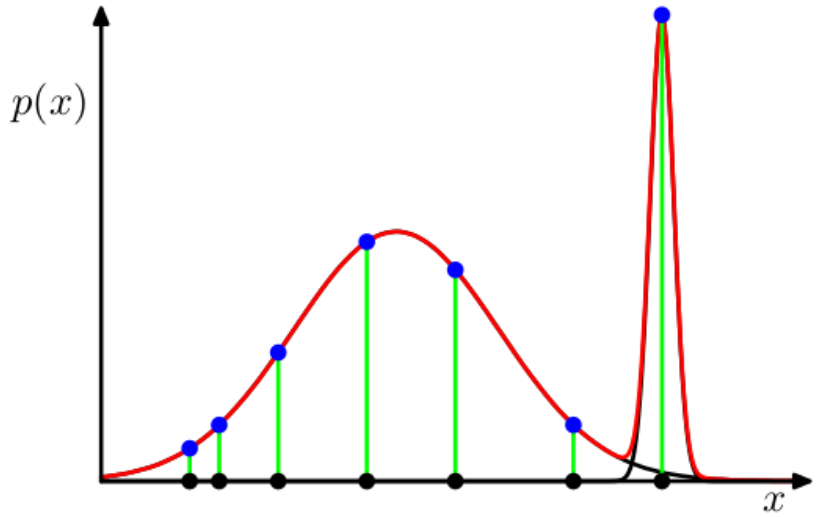
Clustering

Example of Gaussian Mixture Model

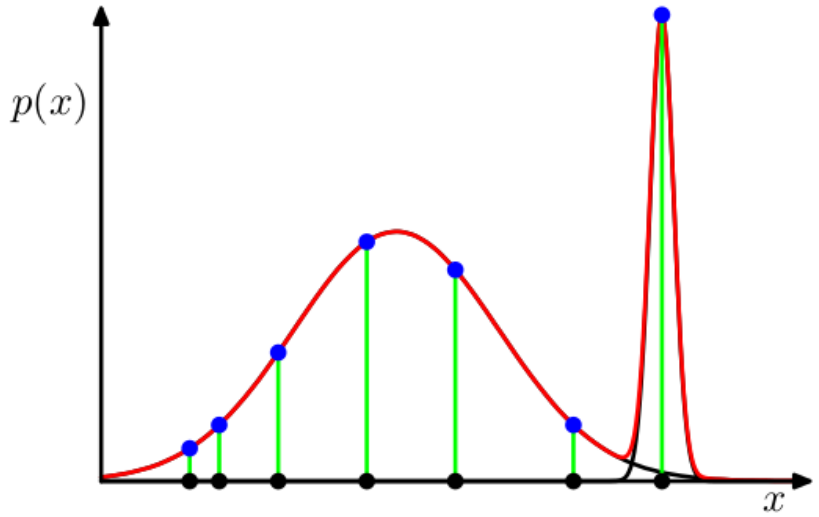
- Can capture more complex structure using arbitrary covariances



- Singularities in the likelihood function

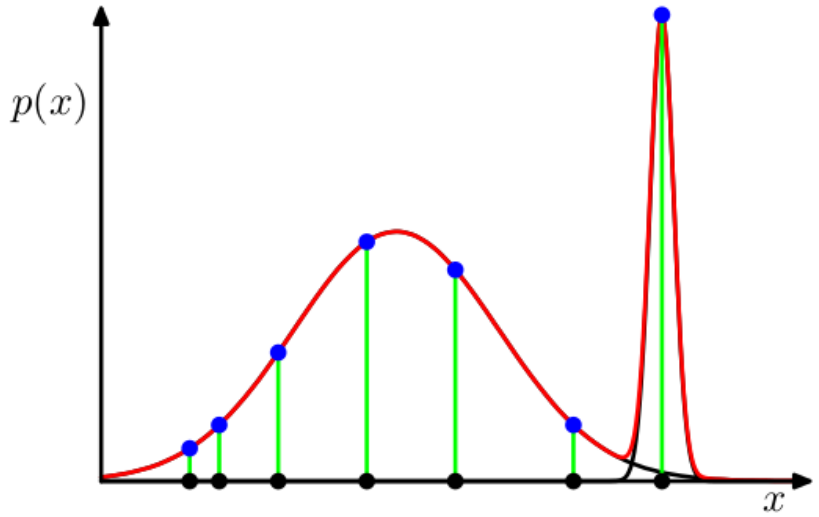


- Singularities in the likelihood function



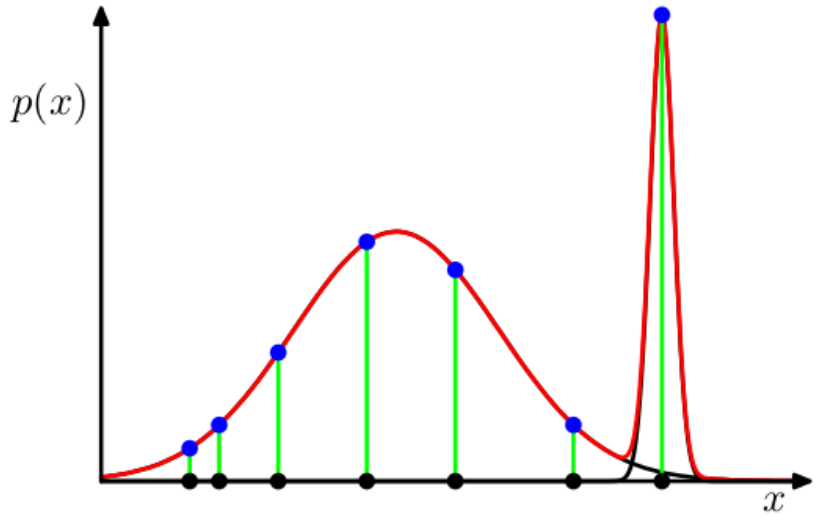
- Identifiability: $k!$ possible ways of assigning parameters

- Singularities in the likelihood function



- Identifiability: $k!$ possible ways of assigning parameters
  - Multiple minima, but all equally good

- Singularities in the likelihood function



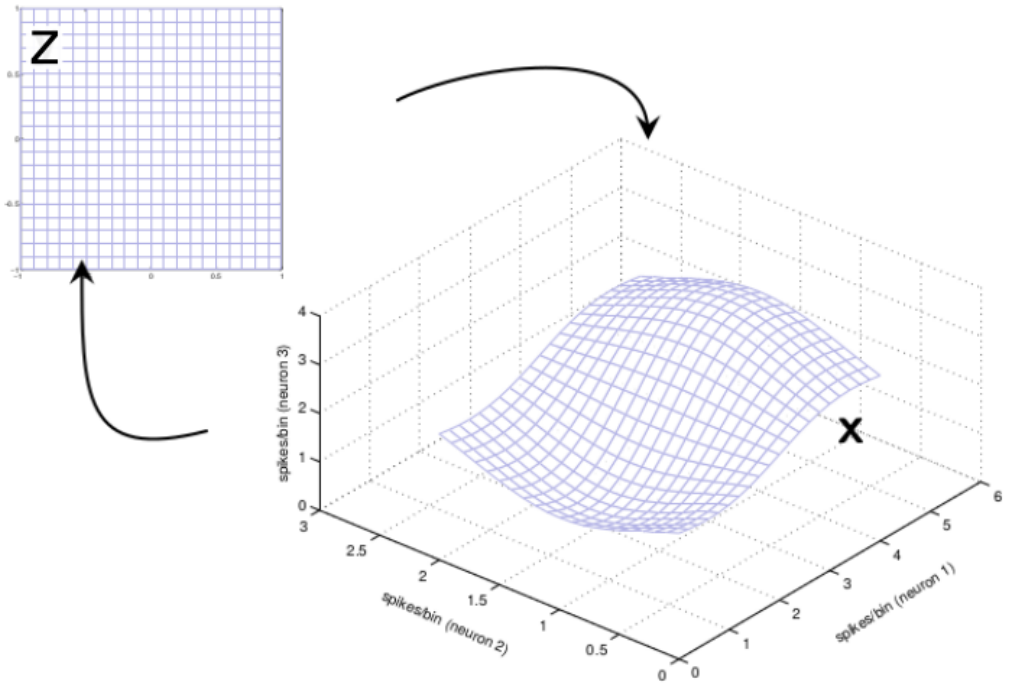
- Identifiability: $k!$ possible ways of assigning parameters
  - Multiple minima, but all equally good
  - In practice, only problematic for interpretability

Content

Dimensionality Reduction

Preliminaries

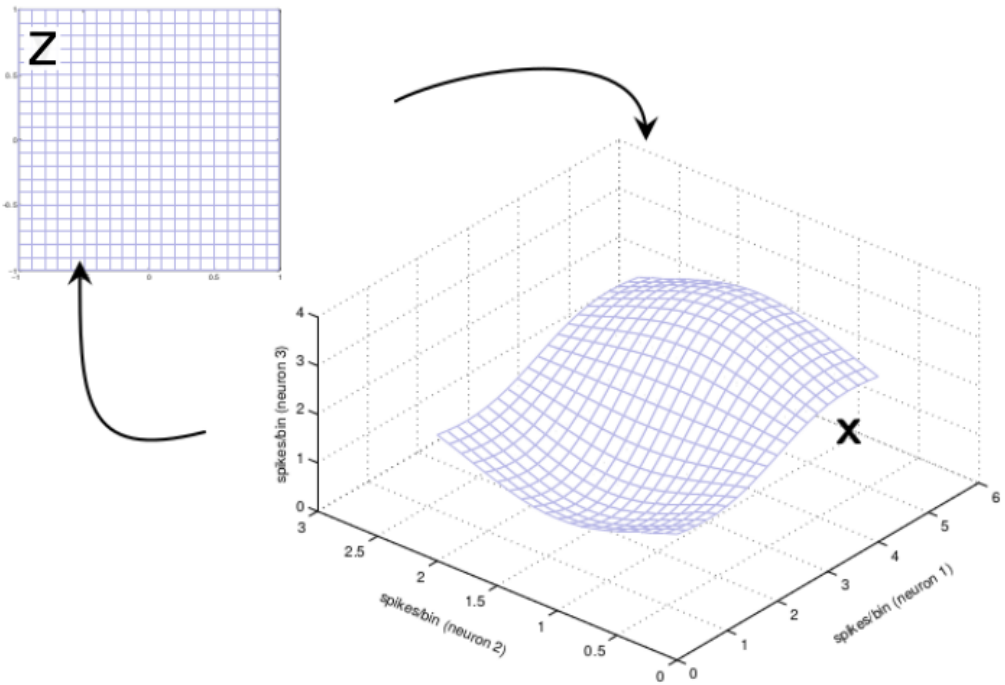
- High-dimensional data $\mathbf{x}_i \in \mathbb{R}^D, i = 1, \ldots, N$
- But not all possible vectors appear in the data set



Dimensionality Reduction

Preliminaries

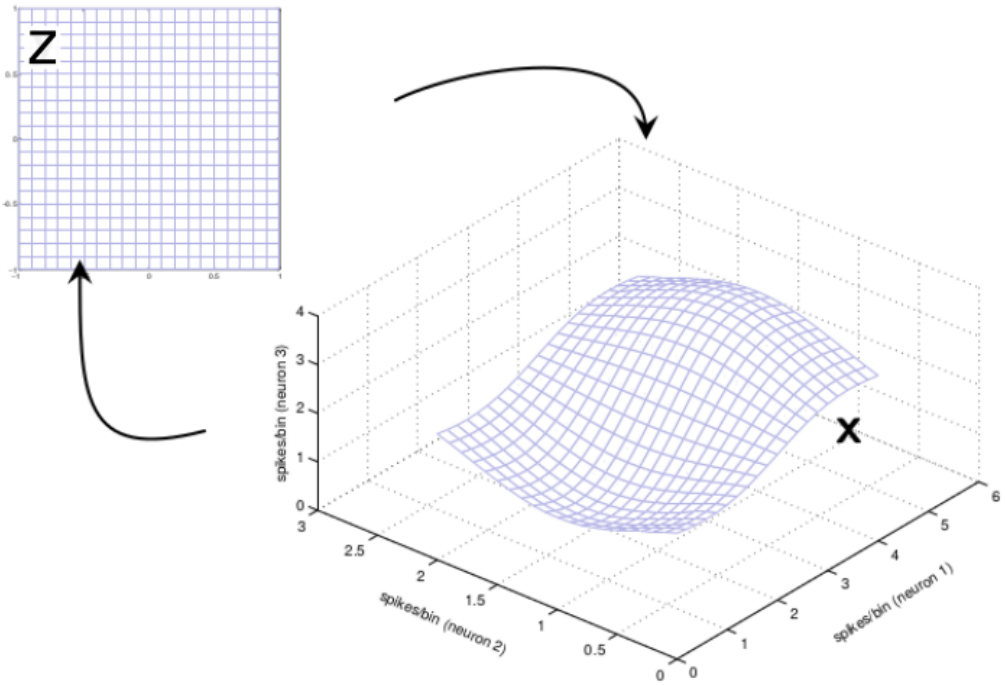
- Data live on a **low-dimensional** manifold
  - subset of possible values
  - smoothly varying and dense
  - may be parametrized by “latent variables”



Dimensionality Reduction

Preliminaries

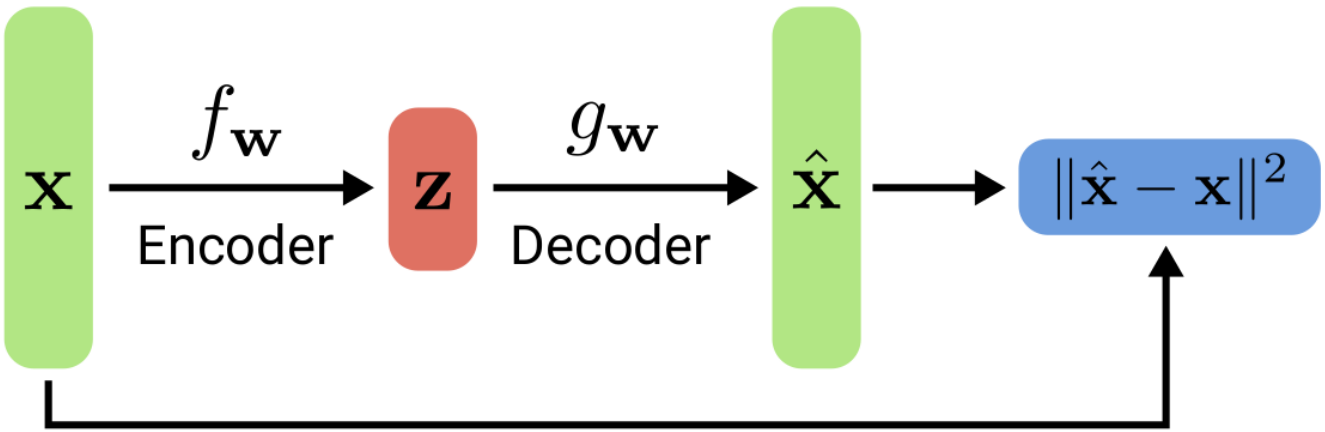
- **Goal:** Find the manifold
  - More precisely, find $\mathbf{z}_i \in \mathbb{R}^Q, (Q < D)$ so that $\mathbf{z}_i$ parametrizes the location of $\mathbf{x}_i$ on the manifold



Autoencoder perspective:

- Encoder produces a compressed latent representation $\mathbf{z} \in \mathbb{R}^Q$
- Decoder produces a reconstructed input $\hat{\mathbf{x}} \in \mathbb{R}^D$

$$f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{z} \qquad g_{\mathbf{w}} : \mathbf{z} \mapsto \hat{\mathbf{x}}$$



Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**
  - Multidimensional Scaling (MDS)

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**
  - Multidimensional Scaling (MDS)
- Nonlinear - pre-processing, then factorization

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**
  - Multidimensional Scaling (MDS)
- Nonlinear - pre-processing, then factorization
  - **Isomap**

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**
  - Multidimensional Scaling (MDS)
- Nonlinear - pre-processing, then factorization
  - **Isomap**
  - Locally Linear Embeddings (LLE)

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**
  - Multidimensional Scaling (MDS)
- Nonlinear - pre-processing, then factorization
  - **Isomap**
  - Locally Linear Embeddings (LLE)
  - Maximum Variance Unfolding (MVU)

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**
  - Multidimensional Scaling (MDS)
- Nonlinear - pre-processing, then factorization
  - **Isomap**
  - Locally Linear Embeddings (LLE)
  - Maximum Variance Unfolding (MVU)
  - Uniform Manifold Approximation and Projection (UMAP)

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**
  - Multidimensional Scaling (MDS)
- Nonlinear - pre-processing, then factorization
  - **Isomap**
  - Locally Linear Embeddings (LLE)
  - Maximum Variance Unfolding (MVU)
  - Uniform Manifold Approximation and Projection (UMAP)
  - Stochastic Neighbour Embedding (SNE)

Dimensionality Reduction

Preliminaries

- Uses of dimensionality reduction
  - Structure discovery
  - Visualization
  - Pre-processing
- Classical Methods (linear) - just factorization
  - **Principal Component Analysis (PCA)**
  - Multidimensional Scaling (MDS)
- Nonlinear - pre-processing, then factorization
  - **Isomap**
  - Locally Linear Embeddings (LLE)
  - Maximum Variance Unfolding (MVU)
  - Uniform Manifold Approximation and Projection (UMAP)
  - Stochastic Neighbour Embedding (SNE)
  - ...

Dimensionality Reduction

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$

Dimensionality Reduction

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Mean vector

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

Dimensionality Reduction

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Mean vector

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

- Sample covariance matrix

$$\mathbf{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$$

Dimensionality Reduction

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Mean vector

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

- Sample covariance matrix

$$\mathbf{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$$

  - $\Sigma \in \mathbb{R}^{D \times D}$

Dimensionality Reduction

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Mean vector

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

- Sample covariance matrix

$$\mathbf{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$$

  - $\Sigma \in \mathbb{R}^{D \times D}$
  - A real, symmetric, and positive semi-definite matrix

Dimensionality Reduction

Preliminaries

- **v** is an **eigenvector**, with scalar **eigenvalue** λ, of a matrix **A** if

$$\mathbf{Av} = \lambda\mathbf{v}$$

Dimensionality Reduction

Preliminaries

- **v** is an **eigenvector**, with scalar **eigenvalue** λ, of a matrix **A** if

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

- **v** can have any norm, but we will define it to be unity, $\mathbf{v}^\top\mathbf{v} = 1$

Dimensionality Reduction

Preliminaries

- **v** is an **eigenvector**, with scalar **eigenvalue** λ, of a matrix **A** if

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

- **v** can have any norm, but we will define it to be unity, $\mathbf{v}^\top\mathbf{v} = 1$
- For $\mathbf{S} \in \mathbb{R}^{D\times D}$ real, symmetric, postive semi-definite:

Dimensionality Reduction

Preliminaries

- $\mathbf{v}$ is an **eigenvector**, with scalar **eigenvalue** λ, of a matrix $\mathbf{A}$ if

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

- $\mathbf{v}$ can have any norm, but we will define it to be unity, $\mathbf{v}^\top\mathbf{v} = 1$
- For $\mathbf{S} \in \mathbb{R}^{D\times D}$ real, symmetric, postive semi-definite:
  - There are D eigenvector-eigenvalue pairs $(\mathbf{v}_j, \lambda_j)$

Dimensionality Reduction

Preliminaries

- $\mathbf{v}$ is an **eigenvector**, with scalar **eigenvalue** λ, of a matrix $\mathbf{A}$ if

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

- $\mathbf{v}$ can have any norm, but we will define it to be unity, $\mathbf{v}^\top\mathbf{v} = 1$
- For $\mathbf{S} \in \mathbb{R}^{D\times D}$ real, symmetric, postive semi-definite:
  - There are D eigenvector-eigenvalue pairs $(\mathbf{v}_j, \lambda_j)$
  - The D eigenvectors are orthogonal, forming an **orthonormal basis**

$$\sum_j \mathbf{v}_j\mathbf{v}_j^\top = \mathbf{I}$$

Dimensionality Reduction

Preliminaries

- **v** is an **eigenvector**, with scalar **eigenvalue** λ, of a matrix **A** if

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

- **v** can have any norm, but we will define it to be unity, $\mathbf{v}^\top\mathbf{v} = 1$
- For $\mathbf{S} \in \mathbb{R}^{D\times D}$ real, symmetric, postive semi-definite:
  - There are D eigenvector-eigenvalue pairs $(\mathbf{v}_j, \lambda_j)$
  - The D eigenvectors are orthogonal, forming an **orthonormal basis**

$$\sum_j \mathbf{v}_j\mathbf{v}_j^\top = \mathbf{I}$$

  - Any vector **u** can be written as

$$\mathbf{u} = \left(\sum_j \mathbf{v}_j\mathbf{v}_j^\top\right)\mathbf{u} = \sum_j (\mathbf{v}_j^\top\mathbf{u})\mathbf{v}_j = \sum_j u_j\mathbf{v}_j$$

Eigendecomposition

- Define matrices

$$\mathbf{V} = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_D \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \cdots & \\ & & & \lambda_D \end{bmatrix}$$

Eigendecomposition

- Define matrices

$$\mathbf{V} = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_D \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \cdots & \\ & & & \lambda_D \end{bmatrix}$$

- Then we can write the eigenvector definition as:

$$\mathbf{SV} = \mathbf{V}\Lambda$$

Eigendecomposition

- Define matrices

$$\mathbf{V} = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_D \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \cdots & \\ & & & \lambda_D \end{bmatrix}$$

- Then we can write the eigenvector definition as:

$$\mathbf{SV} = \mathbf{V}\Lambda$$

- For symmetric $\mathbf{S}$ (i.e., orthonormal $\mathbf{V}$):

$$\mathbf{S} = \mathbf{SVV}^\top = \mathbf{V}\Lambda\mathbf{V}^\top = \sum_j \lambda_j \mathbf{v}_j \mathbf{v}_j^\top$$

Eigendecomposition

- Define matrices

$$\mathbf{V} = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_D \\ | & | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_D \end{bmatrix}$$

- Then we can write the eigenvector definition as:

$$\mathbf{SV} = \mathbf{V}\Lambda$$

- For symmetric $\mathbf{S}$ (i.e., orthonormal $\mathbf{V}$):

$$\mathbf{S} = \mathbf{SVV}^\top = \mathbf{V}\Lambda\mathbf{V}^\top = \sum_j \lambda_j \mathbf{v}_j \mathbf{v}_j^\top$$

- This is called eigendecomposition of the matrix $\mathbf{S}$

Dimensionality Reduction

Finding eigenvectors and eigenvalues

In theory

- Just algebra

$$\mathbf{Sv} = \lambda\mathbf{v} \implies (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$$

In practice

Dimensionality Reduction

Finding eigenvectors and eigenvalues

In theory

- Just algebra

$$\mathbf{Sv} = \lambda\mathbf{v} \implies (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$$

- Solve:

$$|\mathbf{S} - \lambda\mathbf{I}| = 0$$

(polynomial in λ) to find eigvals

In practice

Dimensionality Reduction

Finding eigenvectors and eigenvalues

In theory

- Just algebra

$$\mathbf{Sv} = \lambda\mathbf{v} \implies (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$$

- Solve:

$$|\mathbf{S} - \lambda\mathbf{I}| = 0$$

(polynomial in λ) to find eigvals

- Solve:

$$(\mathbf{S} - \lambda_i\mathbf{I})\mathbf{v} = 0$$

(linear system) to find eigvecs

In practice

Dimensionality Reduction

Finding eigenvectors and eigenvalues

In theory

- Just algebra

$$\mathbf{Sv} = \lambda\mathbf{v} \implies (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$$

- Solve:

$$|\mathbf{S} - \lambda\mathbf{I}| = 0$$

(polynomial in λ) to find eigvals

- Solve:

$$(\mathbf{S} - \lambda_i\mathbf{I})\mathbf{v} = 0$$

(linear system) to find eigvecs

In practice

- Use a software package.
  In MATLAB `[V,L] = eig(S)`

Dimensionality Reduction

Finding eigenvectors and eigenvalues

In theory

- Just algebra

$$\mathbf{Sv} = \lambda\mathbf{v} \implies (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$$

- Solve:

$$|\mathbf{S} - \lambda\mathbf{I}| = 0$$

(polynomial in λ) to find eigvals

- Solve:

$$(\mathbf{S} - \lambda_i\mathbf{I})\mathbf{v} = 0$$

(linear system) to find eigvecs

In practice

- Use a software package.
  In MATLAB `[V,L] = eig(S)`
- returns the matrices $\mathbf{V}$ (`V`) and $\mathbf{\Lambda}$ (`L`) defined before

Dimensionality Reduction

Finding eigenvectors and eigenvalues

In theory

- Just algebra

$$\mathbf{Sv} = \lambda\mathbf{v} \implies (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$$

- Solve:

$$|\mathbf{S} - \lambda\mathbf{I}| = 0$$

(polynomial in λ) to find eigvals

- Solve:

$$(\mathbf{S} - \lambda_i\mathbf{I})\mathbf{v} = 0$$

(linear system) to find eigvecs

In practice

- Use a software package.
  In MATLAB `[V,L] = eig(S)`
- returns the matrices $\mathbf{V}$ (`V`) and $\mathbf{\Lambda}$ (`L`) defined before
- `eig` usually returns eigvals in increasing order, but don't count on it

Dimensionality Reduction

Finding eigenvectors and eigenvalues

In theory

- Just algebra

$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v} \implies (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$$

- Solve:

$$|\mathbf{S} - \lambda\mathbf{I}| = 0$$

(polynomial in λ) to find eigvals

- Solve:

$$(\mathbf{S} - \lambda_i\mathbf{I})\mathbf{v} = 0$$

(linear system) to find eigvecs

In practice

- Use a software package.
  In MATLAB `[V,L] = eig(S)`
- returns the matrices $\mathbf{V}$ (`V`) and $\mathbf{\Lambda}$ (`L`) defined before
- `eig` usually returns eigvals in increasing order, but don't count on it
- `eigs` can find largest or smallest k eigvals (and corresponding eigvecs)

Principal Component Analysis (PCA)

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$

Principal Component Analysis (PCA)

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Corresponding latent variables $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)^\top \in \mathbb{R}^{N \times Q}$

Principal Component Analysis (PCA)

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Corresponding latent variables $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)^\top \in \mathbb{R}^{N \times Q}$
- While $\mathbf{X}$ is observed, $\mathbf{Z}$ is not (needs to be inferred)

Principal Component Analysis (PCA)

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Corresponding latent variables $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)^\top \in \mathbb{R}^{N \times Q}$
- While $\mathbf{X}$ is observed, $\mathbf{Z}$ is not (needs to be inferred)
- Typically, $Q < D$ assumed (compressed representation)

Principal Component Analysis (PCA)

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Corresponding latent variables $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)^\top \in \mathbb{R}^{N \times Q}$
- While $\mathbf{X}$ is observed, $\mathbf{Z}$ is not (needs to be inferred)
- Typically, $Q < D$ assumed (compressed representation)
- Goal: learn a **linear bidirectional mapping** $\mathcal{X} \leftrightarrow \mathcal{Z}$ such that as much information of $\mathcal{X}$ as possible is retained in $\mathcal{Z}$

Principal Component Analysis (PCA)

Preliminaries

- $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Corresponding latent variables $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)^\top \in \mathbb{R}^{N \times Q}$
- While $\mathbf{X}$ is observed, $\mathbf{Z}$ is not (needs to be inferred)
- Typically, $Q < D$ assumed (compressed representation)
- Goal: learn a **linear bidirectional mapping** $\mathcal{X} \leftrightarrow \mathcal{Z}$ such that as much information of $\mathcal{X}$ as possible is retained in $\mathcal{Z}$
- We want to encode $\mathbf{x} \mapsto \mathbf{z}$ such that if we decode $\mathbf{z} \mapsto \hat{\mathbf{x}}$, then $\hat{\mathbf{x}}$ is a good approximation of the original $\mathbf{x}$ (in most cases $\hat{\mathbf{x}} \neq \mathbf{x}$)

Principal Component Analysis (PCA)

Derivation as loss minimization

- Assume the following **linear mapping** from latent space to observation space

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij}\mathbf{v}_j$$

where $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$ is the **data mean** and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$ an **orthonormal basis**.

Principal Component Analysis (PCA)

Derivation as loss minimization

- Assume the following **linear mapping** from latent space to observation space

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij} \mathbf{v}_j$$

where $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$ is the **data mean** and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$ an **orthonormal basis**.

- Goal: minimize the L_2 **reconstruction loss** wrt. $\mathbf{Z}$ and $\mathbf{V}$:

$$\mathcal{L}(\mathbf{Z}, \mathbf{V}) = \sum_{i=1}^{N} \| \hat{\mathbf{x}}_i - \mathbf{x}_i \|^2 = \sum_{i=1}^{N} \| \underbrace{\bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij} \mathbf{v}_j}_{=\hat{\mathbf{x}}_i} - \mathbf{x}_i \|^2$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- Consider $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$ an **orthonormal basis**. Expanding $\mathcal{L}$

$$
\begin{aligned}
\mathcal{L}(\mathbf{Z}, \mathbf{V}) &= \sum_{i=1}^{N} \parallel \bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij}\mathbf{v}_j - \mathbf{x}_i \parallel^2 \\
&= \sum_{i=1}^{N} \parallel \sum_{j=1}^{Q} z_{ij}\mathbf{v}_j + \bar{\mathbf{x}} - \mathbf{x}_i \parallel^2 \\
&= \sum_{i=1}^{N} \left[\sum_{j=1}^{Q} z_{ij}^2 + 2 \sum_{j=1}^{Q} z_{ij}\mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i) + \parallel \bar{\mathbf{x}} - \mathbf{x}_i \parallel^2 \right]
\end{aligned}
$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- The **reconstruction loss** can be minimized in closed for wrt $\mathbf{Z}$:

$$\mathcal{L}(\mathbf{Z}, \mathbf{V}) = \sum_{i=1}^{N} \left[\sum_{j=1}^{Q} \left[z_{ij}^2 + 2 z_{ij} \mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i) \right] + \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2 \right]$$

$$\frac{\partial \mathcal{L}(\mathbf{Z}, \mathbf{V})}{\partial z_{ij}} = 2 z_{ij} + 2 \mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i) = 0$$

$$\implies z_{ij}^* = -\mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i)$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- The **reconstruction loss** can be minimized in closed for wrt $\mathbf{Z}$:

$$\mathcal{L}(\mathbf{Z}, \mathbf{V}) = \sum_{i=1}^{N} \left[\sum_{j=1}^{Q} \left[z_{ij}^2 + 2 z_{ij} \mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i) \right] + \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2 \right]$$

$$\frac{\partial \mathcal{L}(\mathbf{Z}, \mathbf{V})}{\partial z_{ij}} = 2 z_{ij} + 2 \mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i) = 0$$

$$\implies z_{ij}^* = -\mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i)$$

- For $\mathbf{Z} = \mathbf{Z}^*$, the reconstruction loss simplifies to:

$$\mathcal{L}(\mathbf{Z}^*, \mathbf{V}) = \sum_{i=1}^{N} \left[- \sum_{j=1}^{Q} \left[{z_{ij}^*}^2 + \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2 \right] \right]$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- The **reconstruction loss** at $z_{ij}^* = -\mathbf{v}_j^\top (\bar{\mathbf{x}} - \mathbf{x}_i)$ can be rewritten

$$\mathcal{L}(\mathbf{Z}^*, \mathbf{V}) = \sum_{i=1}^{N} \left[-\sum_{j=1}^{Q} {z_{ij}^*}^2 + \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2 \right]$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- The **reconstruction loss** at $z_{ij}^* = -\mathbf{v}_j^\top(\bar{\mathbf{x}} - \mathbf{x}_i)$ can be rewritten

$$\begin{aligned}\mathcal{L}(\mathbf{Z}^*, \mathbf{V}) &= \sum_{i=1}^{N}\left[-\sum_{j=1}^{Q} {z_{ij}^*}^2 + \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2\right] \\ &= -\sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S}\mathbf{v}_j + \sum_{i=1}^{N} \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2\end{aligned}$$

- with $\mathbf{S}$ the **scatter matrix** (unnormalized sample covariance) of $\mathbf{x}$

$$\mathbf{S} = \sum_{i=1}^{N} (\bar{\mathbf{x}} - \mathbf{x}_i)(\bar{\mathbf{x}} - \mathbf{x}_i)^\top$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- To enforce $\| \mathbf{v}_j \| = 1$, we introduce **Lagrange multipliers** λ_j:

$$\mathcal{L}(\mathbf{Z}^*, \mathbf{V}, \boldsymbol{\lambda}) = -\sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{i=1}^{N} \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2 + \sum_{j=1}^{Q} \lambda_j (\mathbf{v}_j^\top \mathbf{v}_j - 1)$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- To enforce $\| \mathbf{v}_j \| = 1$, we introduce **Lagrange multipliers** λ_j:

$$\mathcal{L}(\mathbf{Z}^*, \mathbf{V}, \boldsymbol{\lambda}) = -\sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{i=1}^{N} \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2 + \sum_{j=1}^{Q} \lambda_j (\mathbf{v}_j^\top \mathbf{v}_j - 1)$$

$$\frac{\partial \mathcal{L}(\mathbf{Z}, \mathbf{V}, \boldsymbol{\lambda})}{\partial z_{ij}} = -2\mathbf{S}\mathbf{v}_j + 2\lambda_j \mathbf{v}_j = 0$$

$$\implies \mathbf{S}\mathbf{v}_j = \lambda_j \mathbf{v}_j$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- To enforce $\| \mathbf{v}_j \| = 1$, we introduce **Lagrange multipliers** λ_j:

$$\mathcal{L}(\mathbf{Z}^*, \mathbf{V}, \boldsymbol{\lambda}) = -\sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{i=1}^{N} \| \bar{\mathbf{x}} - \mathbf{x}_i \|^2 + \sum_{j=1}^{Q} \lambda_j (\mathbf{v}_j^\top \mathbf{v}_j - 1)$$

$$\frac{\partial \mathcal{L}(\mathbf{Z}, \mathbf{V}, \boldsymbol{\lambda})}{\partial z_{ij}} = -2\mathbf{S}\mathbf{v}_j + 2\lambda_j \mathbf{v}_j = 0$$

$$\implies \mathbf{S}\mathbf{v}_j = \lambda_j \mathbf{v}_j$$

- We see that $\{\boldsymbol{\lambda}, \mathbf{V}\}$ is the solution of an **eigenvalue problem**
- Also, $\mathcal{L}$ is minimized by choosing (for $\mathbf{V}$) the eigenvectors $\mathbf{v}_j$ of $\mathbf{S}$ corresponding to the top Q eigenvalues

Principal Component Analysis (PCA)

Derivation as loss minimization

- Consider again the **linear model** that we started with:

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij} \mathbf{v}_j$$

Principal Component Analysis (PCA)

Derivation as loss minimization

- Consider again the **linear model** that we started with:

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + \sum_{j=1}^{Q} z_{ij} \mathbf{v}_j$$

- Both the PCA decoder and encoder are simple **linear mappings**

Decoder:

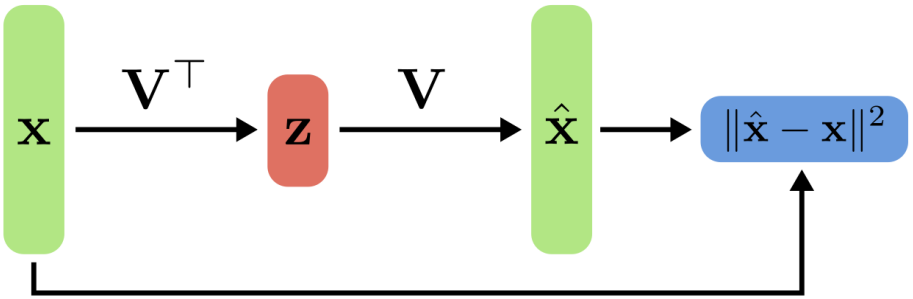
$$\hat{\mathbf{x}} = \mathbf{V}\mathbf{z} + \bar{\mathbf{x}}$$

Encoder:

$$\mathbf{z} = \mathbf{V}^\top (\mathbf{x} - \bar{\mathbf{x}})$$

Principal Component Analysis (PCA)

Algorithm

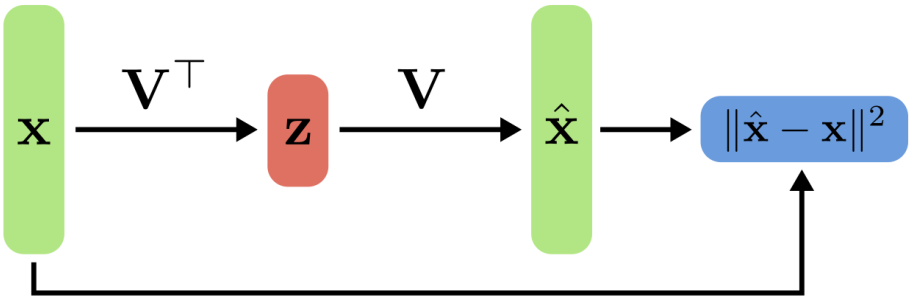


PCA recipe

- $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$

Principal Component Analysis (PCA)

Algorithm

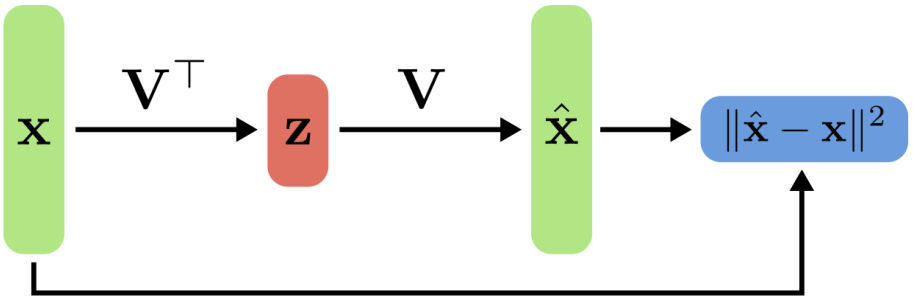


PCA recipe

- $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Compute the **data mean $\bar{\mathbf{x}}$** and **scatter matrix** $\mathbf{S} = \sum_{i=1}^{N} (\bar{\mathbf{x}} - \mathbf{x}_i)(\bar{\mathbf{x}} - \mathbf{x}_i)^\top$

Principal Component Analysis (PCA)

Algorithm

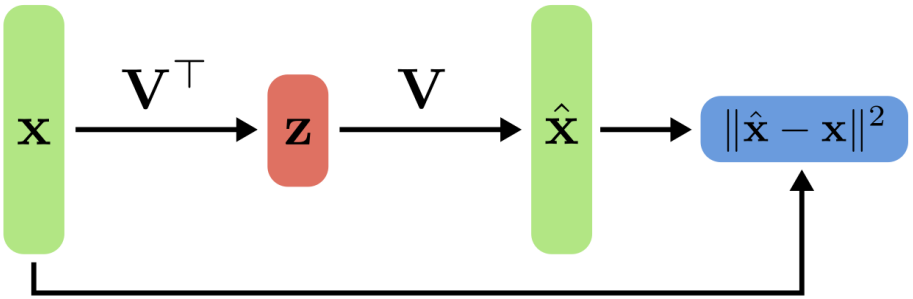


PCA recipe

- $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Compute the **data mean $\bar{\mathbf{x}}$** and **scatter matrix** $\mathbf{S} = \sum_{i=1}^{N} (\bar{\mathbf{x}} - \mathbf{x}_i)(\bar{\mathbf{x}} - \mathbf{x}_i)^\top$
- Compute the **eigen decomposition** of **S**

Principal Component Analysis (PCA)

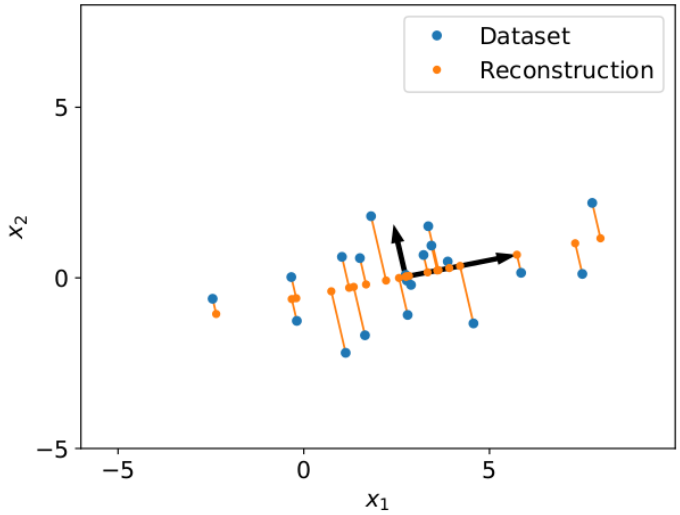
Algorithm

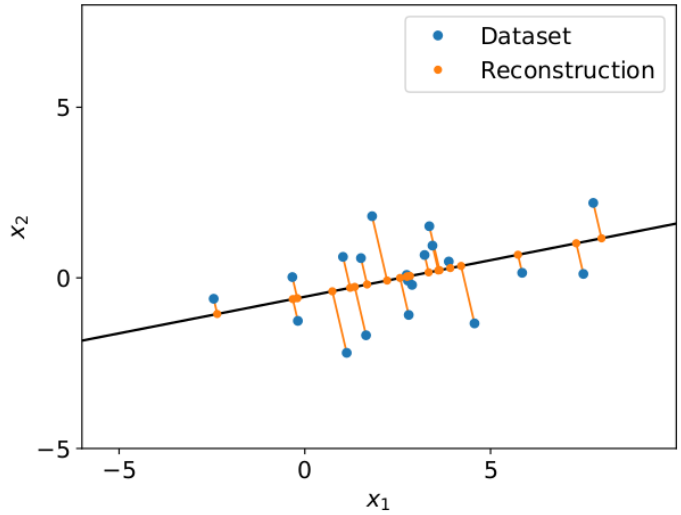


PCA recipe

- $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$ a dataset of observations $\mathbf{x}_i \in \mathbb{R}^D$
- Compute the **data mean $\bar{\mathbf{x}}$** and **scatter matrix** $\mathbf{S} = \sum_{i=1}^{N} (\bar{\mathbf{x}} - \mathbf{x}_i)(\bar{\mathbf{x}} - \mathbf{x}_i)^\top$
- Compute the **eigen decomposition** of **S**
- Select Q eigenvectors corresponding to the Q largest eigenvalues for **V**

Principal Component Analysis (PCA)

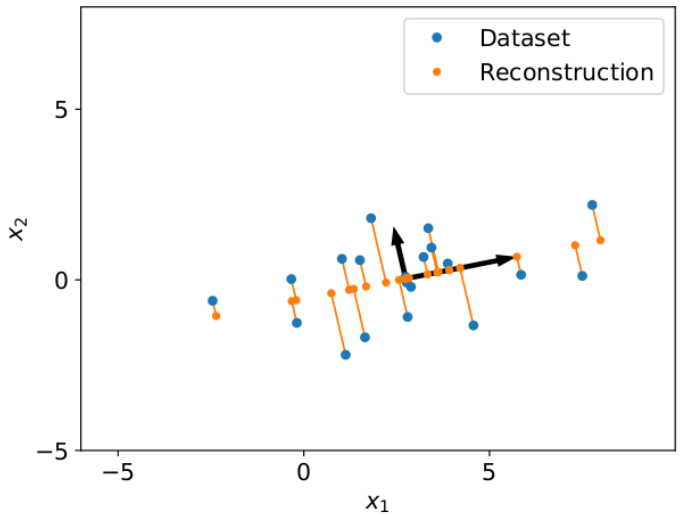


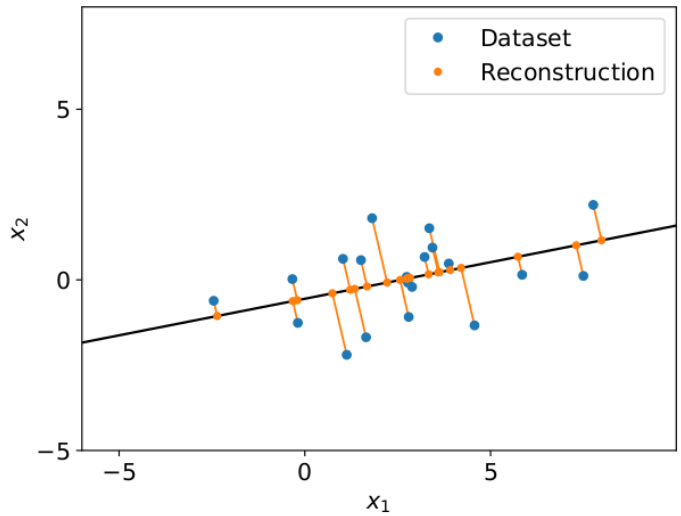


There are two perspectives on PCA:

- Can be formulated as a **minimization** of the reconstruction error

Principal Component Analysis (PCA)

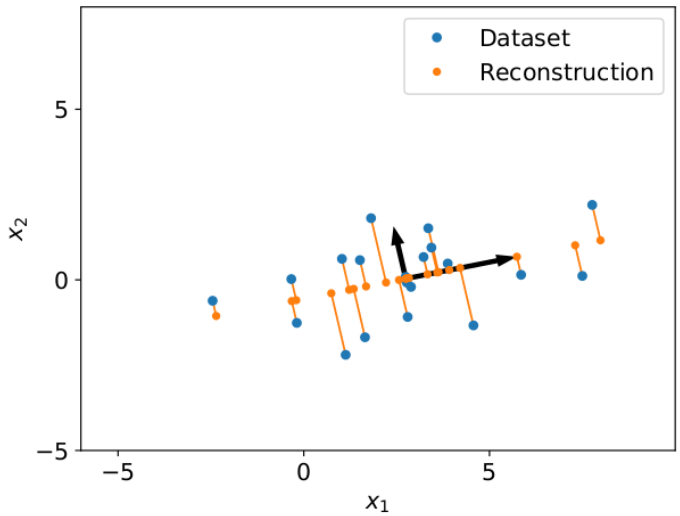


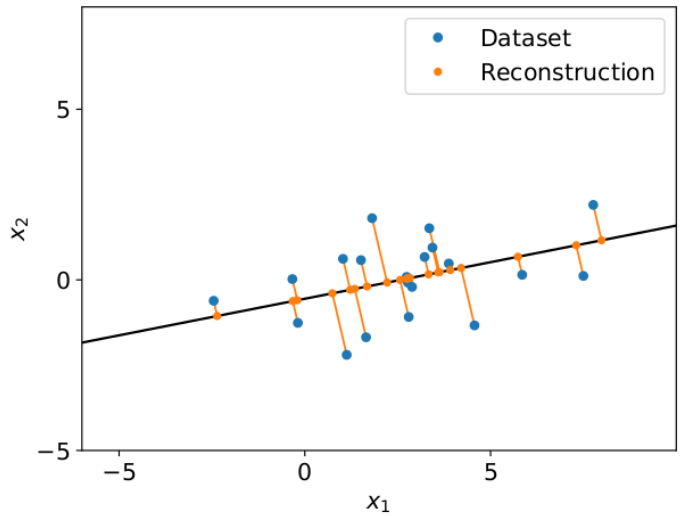


There are two perspectives on PCA:

- Can be formulated as a **minimization** of the reconstruction error
- But also as a **variance maximization** of latent points

Principal Component Analysis (PCA)





There are two perspectives on PCA:

- Can be formulated as a **minimization** of the reconstruction error
- But also as a **variance maximization** of latent points
- Find an Q-dimensional embedding that captures most of the variation in the original dataset with $Q \ll D$

Principal Component Analysis (PCA)

Variance Maximization Perspective

Consider the following (one-dimensional) encoding of vector $\mathbf{x}$:

$$\mathbf{z} = \mathbf{v}^\top (\mathbf{x} - \bar{\mathbf{x}})$$

Goal: **maximize the variance** in latent space:

$$\begin{aligned}
\text{Var}(\mathbf{z}) &= \mathbb{E}\left[\left(\mathbf{v}^\top(\mathbf{x} - \bar{\mathbf{x}}) - \mathbb{E}\left[\mathbf{v}^\top(\mathbf{x} - \bar{\mathbf{x}})\right]\right)^2\right] \\
&= \mathbb{E}\left[\left(\mathbf{v}^\top(\mathbf{x} - \bar{\mathbf{x}})\right)^2\right] && \text{as } \mathbb{E}[\mathbf{x}] = \bar{\mathbf{x}} \\
&= \mathbb{E}\left[\mathbf{v}^\top(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^\top \mathbf{v}\right] \\
&\propto \mathbf{v}^\top \mathbf{S} \mathbf{v} && \text{as } \mathbf{S} \text{ is not normalized}
\end{aligned}$$

Principal Component Analysis (PCA)

Variance Maximization Perspective

- Again assume a Q-dimensional orthonormal basis $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$

Principal Component Analysis (PCA)

Variance Maximization Perspective

- Again assume a Q-dimensional orthonormal basis $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$
- Constrained optimization problem

$$\boldsymbol{\lambda}^*, \mathbf{V}^* = \text{argmax}_{\boldsymbol{\lambda}, \mathbf{V}} \sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{j=1}^{Q} \lambda_j (\mathbf{v}_j^\top \mathbf{v}_j - 1)$$

Principal Component Analysis (PCA)

Variance Maximization Perspective

- Again assume a Q-dimensional orthonormal basis $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$
- Constrained optimization problem

$$\boldsymbol{\lambda}^*, \mathbf{V}^* = \text{argmax}_{\boldsymbol{\lambda}, \mathbf{V}} \sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{j=1}^{Q} \lambda_j (\mathbf{v}_j^\top \mathbf{v}_j - 1)$$

- Solution: the Q largest eigenvalues and corresponding eigenvectors of S

Principal Component Analysis (PCA)

Variance Maximization Perspective

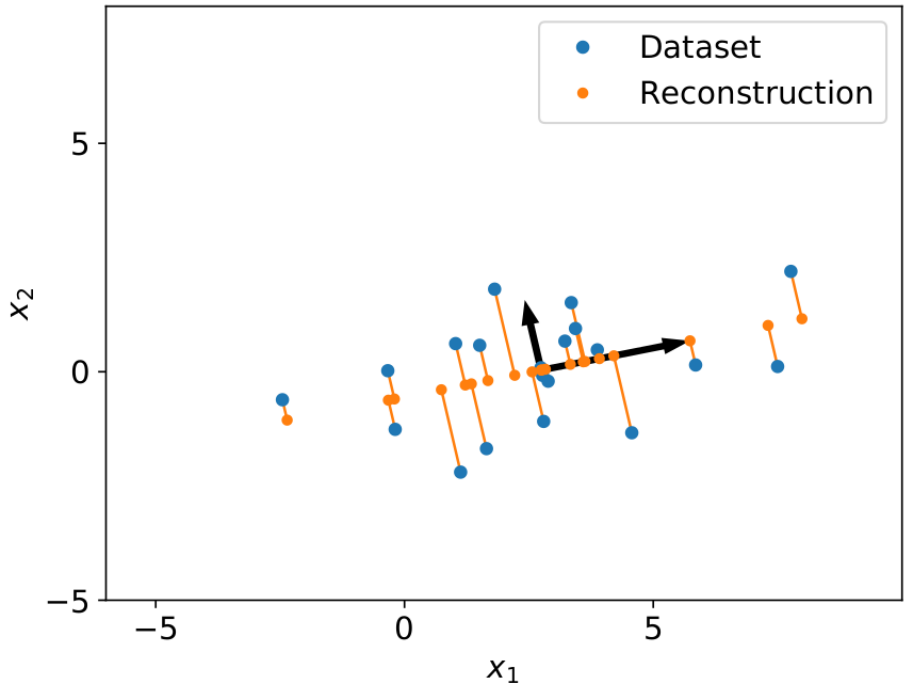
- Again assume a Q-dimensional orthonormal basis $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_Q)$
- Constrained optimization problem

$$\lambda^*, \mathbf{V}^* = \text{argmax}_{\lambda, \mathbf{V}} \sum_{j=1}^{Q} \mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j + \sum_{j=1}^{Q} \lambda_j (\mathbf{v}_j^\top \mathbf{v}_j - 1)$$

- Solution: the Q largest eigenvalues and corresponding eigenvectors of S
- Remark: $\mathbf{v}_j^\top \mathbf{S} \mathbf{v}_j = \lambda_j \mathbf{v}_j^\top \mathbf{v}_j = \lambda_j$ is the **variance** along the j'th principal component if the scatter matrix $\mathbf{S}$ is normalized by N (covariance matrix)

Principal Component Analysis (PCA)

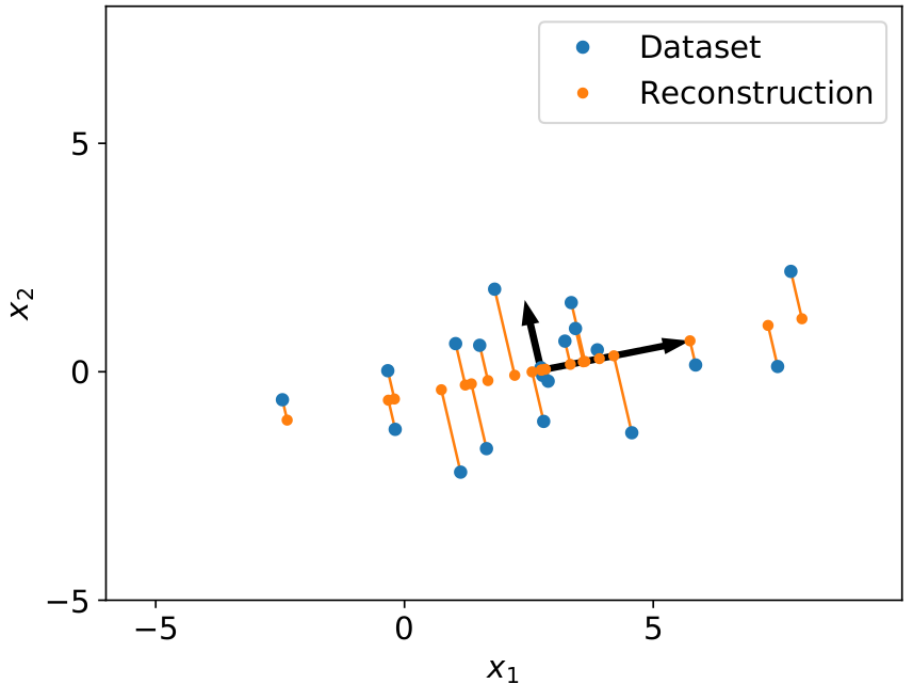
Example



- PCA on a dataset $N = 20, D = 2, Q = 1$ (projection to 1D)

Principal Component Analysis (PCA)

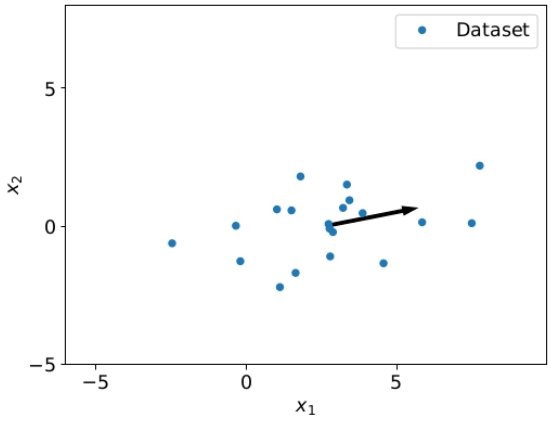
Example

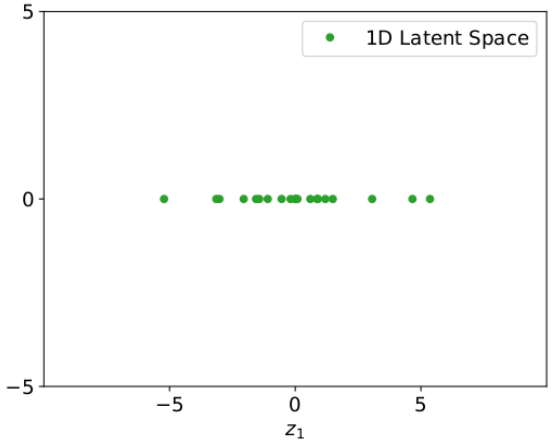


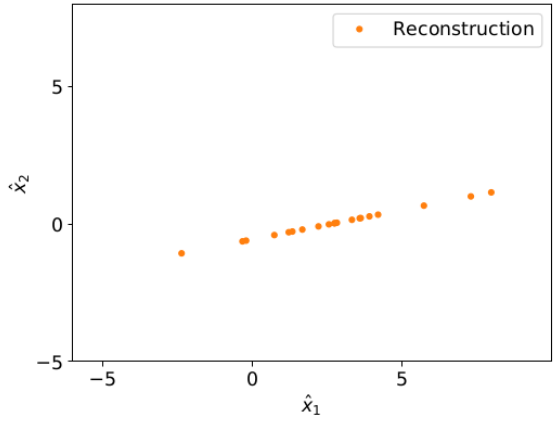
- PCA on a dataset $N = 20, D = 2, Q = 1$ (projection to 1D)
- The two eigenvectors $\mathbf{v}_j$ are shown in black and scaled by $\sqrt{\lambda_j}$

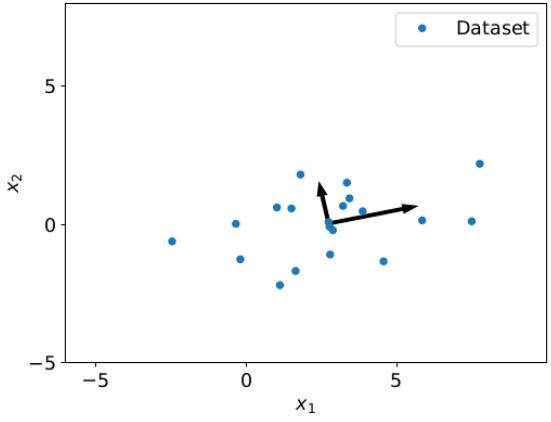
Principal Component Analysis (PCA)

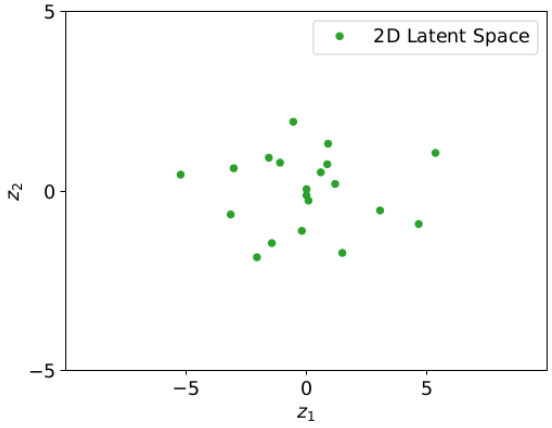
Example

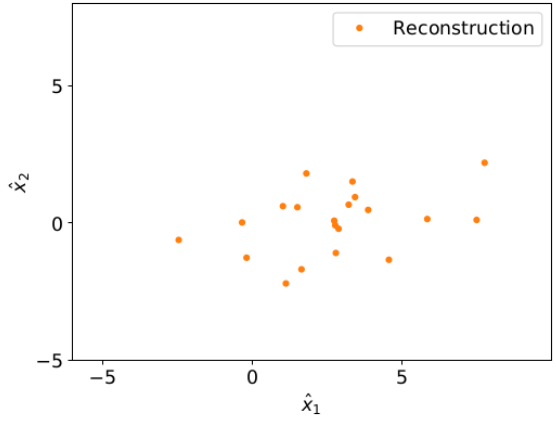






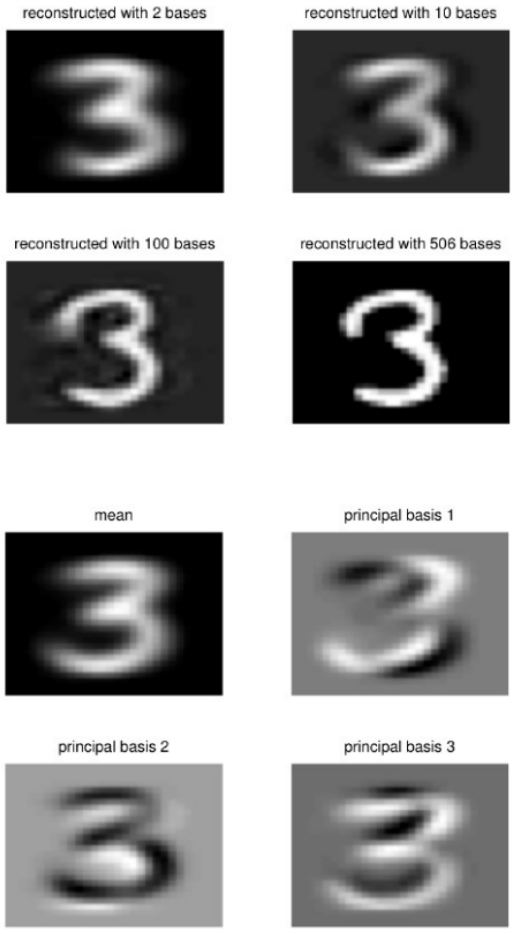






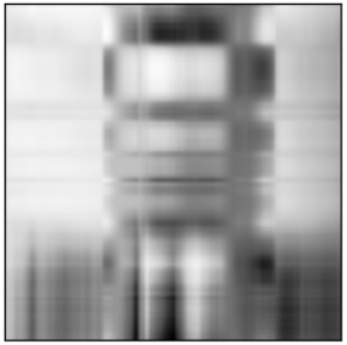
Principal Component Analysis (PCA)

Example: MNIST digits



Principal Component Analysis (PCA)

Example: Images



K = 2

K = 5

K = 20

K = 50

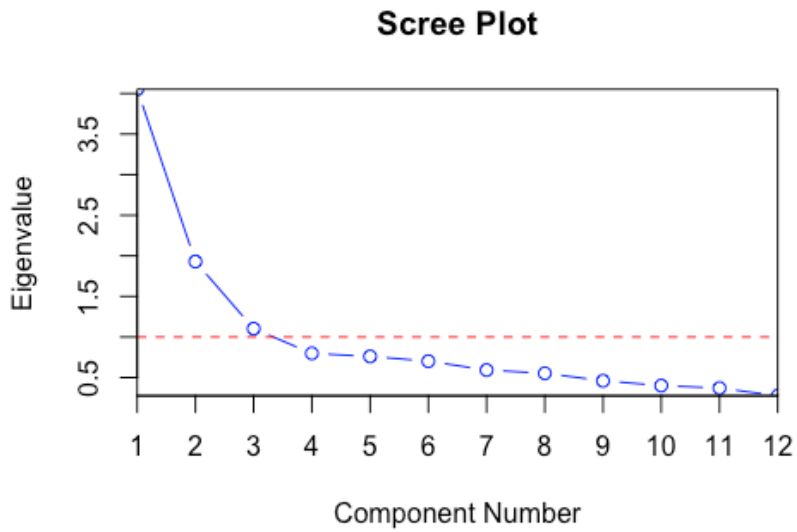
K = 100

K = 300

Principal Component Analysis (PCA)

Eigenspectrum

- The **eigenspectrum** shows how the variance is distributed across dimensions
- Scree plot: shows eigenvalues (monotonically decreases)



- Equivalently, proportion of variance per component

References

Additional references:

- Lecture on PCA. Prof. Andreas Geiger.
  Deep Learning - Lecture 11.2. [Link]
- Other dimensionality reduction resources: [Link]

Content

Exercise 1

k-means

- Given clusters $C_1, \ldots, C_k$, each centroid μ_i is computed as

$$\mu_i = \mu(C_i) = \arg \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} d(x, \mu')^2$$

Exercise 1

k-means

- Given clusters $C_1, \ldots, C_k$, each centroid μ_i is computed as

$$\mu_i = \mu(C_i) = \arg \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} d(x, \mu')^2$$

- Show that when $d(x, y) = \|x - y\|$, the solution is given by

$$\mu_i = \mu(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Exercise 1

k-means

- The optimization problem is given by

$$\min_{\mu' \in \mathcal{X}} f(\mu') = \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu'\|^2$$

Exercise 1

k-means

- The optimization problem is given by

$$\min_{\mu' \in \mathcal{X}} f(\mu') = \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu'\|^2$$

- The gradient is given by

$$\begin{cases} \nabla f(\mu') = \sum_{x \in C_i} 2(x - \mu')(-1) = \sum_{x \in C_i} 2(\mu' - x) \end{cases}$$

Exercise 1

k-means

- The optimization problem is given by

$$\min_{\mu' \in \mathcal{X}} f(\mu') = \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu'\|^2$$

- The gradient is given by

$$\begin{cases} \nabla f(\mu') = \sum_{x \in C_i} 2(x - \mu')(-1) = \sum_{x \in C_i} 2(\mu' - x) \\ \qquad\qquad = 2\left(|C_i|\mu' - \sum_{x \in C_i} x\right) \end{cases}$$

Exercise 1

k-means

- The optimization problem is given by

$$\min_{\mu' \in \mathcal{X}} f(\mu') = \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu'\|^2$$

- The gradient is given by

$$\begin{cases} \nabla f(\mu') = \sum_{x \in C_i} 2(x - \mu')(-1) = \sum_{x \in C_i} 2(\mu' - x) \\ \qquad\qquad = 2\left(|C_i|\mu' - \sum_{x \in C_i} x\right) \end{cases}$$

Exercise 1

k-means

- The optimization problem is given by

$$\min_{\mu' \in \mathcal{X}} f(\mu') = \min_{\mu' \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu'\|^2$$

- The gradient is given by

$$\begin{cases} \nabla f(\mu') = \sum_{x \in C_i} 2(x - \mu')(-1) = \sum_{x \in C_i} 2(\mu' - x) \\ \qquad\qquad = 2\left(|C_i|\mu' - \sum_{x \in C_i} x\right) \end{cases}$$

- Setting the gradient to 0: $\nabla f(\mu') = 0 \implies \mu_i^* = \frac{1}{|C_i|} \sum_{x \in C_i} x$

Exercise 2

k-means

k-means clustering

1. Input: $S = (x_1, \ldots, x_m)$
2. Randomly choose k points as centroids
3. Repeat until convergence:
   - Recompute clusters $C_1, \ldots, C_k$ given the centroids
   - Recompute centroids $\mu_1, \ldots, \mu_k$ given the clusters

Show that each iteration does not increase the *k*-means cost

Exercise 2

k-means

- Recall that the k-means cost is

$$G_{k-\text{means}}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu(C_i))^2$$

Exercise 2

k-means

- Recall that the k-means cost is

$$G_{k-\text{means}}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu(C_i))^2$$

- Note that we can rewrite this as

$$G_{k-\text{means}}(C_1, \ldots, C_k) = \sum_{x \in S} d(x, \mu(C(x)))^2,$$

where $C(x) \in \{C_1, \ldots, C_k\}$ is the cluster to which x belongs

Exercise 2

k-means

- Given centroids $\mu_1, \ldots, \mu_k$, each cluster is recomputed as

$$C_i' = \left\{ x \in S : i = \arg\min_{j=1}^{k} \left\{ d(x, \mu_j)^2 \right\} \right\}$$

Exercise 2

k-means

- Given centroids $\mu_1, \ldots, \mu_k$, each cluster is recomputed as

$$C'_i = \left\{ x \in S : i = \arg\min_{j=1}^{k} \left\{ d(x, \mu_j)^2 \right\} \right\}$$

- Hence for each $x \in S$, it holds that

$$d(x, \mu(C'(x)))^2 \leq d(x, \mu(C(x)))^2$$

$C'(x) \in \{C'_1, \ldots, C'_k\}$: new cluster to which x belongs
$\mu(C'_i) = \mu(C_i) = \mu_i$: old centroid of C_i, $\forall i$

Exercise 2

k-means

- Given clusters $C'_1, \ldots, C'_k$, each centroid μ'_i is recomputed as

$$\mu'_i = \arg \min_{\mu' \in \mathcal{X}} \sum_{x \in C'_i} d(x, \mu')^2$$

Exercise 2

k-means

- Given clusters $C'_1, \ldots, C'_k$, each centroid μ'_i is recomputed as

$$\mu'_i = \arg \min_{\mu' \in \mathcal{X}} \sum_{x \in C'_i} d(x, \mu')^2$$

- Hence for each C'_i, it holds that

$$\sum_{x \in C'_i} d(x, \mu'_i)^2 \leq \sum_{x \in C'_i} d(x, \mu_i)^2$$

Exercise 2

k-means

Putting everything together, it follows that

$$\begin{cases} G_{k-\text{means}}(C'_1, \ldots, C'_k) = \sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu'(C'_i))^2 = \sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu'_i)^2 \end{cases}$$

Exercise 2

k-means

Putting everything together, it follows that

$$\begin{cases} G_{k-\text{means}}(C'_1, \ldots, C'_k) = \sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu'(C'_i))^2 = \sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu'_i)^2 \\ \qquad \leq \sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu_i)^2 = \sum_{x \in S} d(x, \mu(C'(x)))^2 \end{cases}$$

Exercise 2

k-means

Putting everything together, it follows that

$$
\begin{cases}
G_{k-\text{means}}(C'_1, \ldots, C'_k) = \displaystyle\sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu'(C'_i))^2 = \sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu'_i)^2 \\
\qquad \leq \displaystyle\sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu_i)^2 = \sum_{x \in S} d(x, \mu(C'(x)))^2 \\
\qquad \leq \displaystyle\sum_{x \in S} d(x, \mu(C(x)))^2 = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu(C_i))^2
\end{cases}
$$

Exercise 2

k-means

Putting everything together, it follows that

$$
\begin{cases}
G_{k-\text{means}}(C'_1, \ldots, C'_k) = \displaystyle\sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu'(C'_i))^2 = \sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu'_i)^2 \\
\qquad \leq \displaystyle\sum_{i=1}^{k} \sum_{x \in C'_i} d(x, \mu_i)^2 = \sum_{x \in S} d(x, \mu(C'(x)))^2 \\
\qquad \leq \displaystyle\sum_{x \in S} d(x, \mu(C(x)))^2 = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu(C_i))^2 \\
\qquad = G_{k-\text{means}}(C_1, \ldots, C_k)
\end{cases}
$$

Exercise 3

Mixture of Gaussians

- For $\Sigma_1 = \cdots = \Sigma_k = I$, the maximization step is given by

$$\arg \max_{c,\mu_1,\ldots,\mu_k} \sum_{j=1}^{m} \sum_{i=1}^{k} P(i|x_j) \left(\log(c_i) - \frac{1}{2} \|x_j - \mu_i\|^2 \right)$$

- Show that the analytical solution for c is given by

$$c_i = \frac{\sum_{j=1}^{m} P(i|x_j)}{\sum_{i'=1}^{k} \sum_{j=1}^{m} P(i'|x_j)}$$

Exercise 3

Mixture of Gaussians

- For $\Sigma_1 = \cdots = \Sigma_k = I$, the maximization step is given by

$$\arg \max_{c,\mu_1,\ldots,\mu_k} \sum_{j=1}^{m} \sum_{i=1}^{k} P(i|x_j) \left(\log(c_i) - \frac{1}{2} \|x_j - \mu_i\|^2 \right)$$

- To maximize with respect to c, we can simplify to

$$\arg \max_{c} \sum_{i=1}^{k} z_i \log(c_i),$$

where $z_i = \sum_{j=1}^{m} P(i|x_j) > 0, \forall i$

Exercise 3

Mixture of Gaussians

- Since c is a probability distribution, we have two constraints:

$$c_i \geq 0 \quad \forall i,$$

$$\sum_{i=1}^{k} c_i = 1.$$

- Hence the problem is a constrained optimization problem:

$$\max_{c} \sum_{i=1}^{k} z_i \log(c_i),$$

$$\text{s.t.} \quad c_i \geq 0 \quad \forall i,$$

$$\sum_{i=1}^{k} c_i = 1.$$

Exercise 3

Mixture of Gaussians

- The Lagrangian is given by

$$\mathcal{L}(c, \lambda, \beta) = \sum_{i=1}^{k} z_i \log(c_i) + \lambda(1 - \sum_{i=1}^{k} c_i) + \sum_{i=1}^{k} \beta_i c_i$$

- The KKT conditions are given by

$$\frac{\partial \mathcal{L}}{\partial c_i} = \frac{z_i}{c_i} - \lambda + \beta_i = 0 \quad \forall i,$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \sum_{i=1}^{k} c_i = 0,$$

$$\beta_i c_i = 0 \quad \forall i.$$

Exercise 3

Mixture of Gaussians

- Since $z_i > 0$, $c_i = 0$ yields $z_i \log(0) = -\infty$
- Since we are maximizing, better choose $c_i > 0 \Rightarrow \beta_i = 0$

Exercise 3

Mixture of Gaussians

- Since $z_i > 0$, $c_i = 0$ yields $z_i \log(0) = -\infty$
- Since we are maximizing, better choose $c_i > 0 \Rightarrow \beta_i = 0$
- Hence we have

$$0 = \frac{z_i}{c_i} - \lambda + \beta_i = \frac{z_i}{c_i} - \lambda \quad \Leftrightarrow \quad c_i^* = \frac{z_i}{\lambda} \quad \forall i$$

Exercise 3

Mixture of Gaussians

- Since $z_i > 0$, $c_i = 0$ yields $z_i \log(0) = -\infty$
- Since we are maximizing, better choose $c_i > 0 \Rightarrow \beta_i = 0$
- Hence we have

$$0 = \frac{z_i}{c_i} - \lambda + \beta_i = \frac{z_i}{c_i} - \lambda \quad \Leftrightarrow \quad c_i^* = \frac{z_i}{\lambda} \quad \forall i$$

- To obtain λ^*, insert the expression for c_i^* into the constraint:

$$1 = \sum_{i=1}^{k} c_i^* = \sum_{i=1}^{k} \frac{z_i}{\lambda} = \frac{1}{\lambda} \sum_{i=1}^{k} z_i \quad \Leftrightarrow \quad \lambda^* = \sum_{i=1}^{k} z_i$$

Exercise 3

Mixture of Gaussians

In conclusion, the optimal value of c is given by

$$\begin{cases} c_i^* = \dfrac{z_i}{\lambda^*} \end{cases}$$

Exercise 3

Mixture of Gaussians

In conclusion, the optimal value of c is given by

$$\begin{cases} c_i^* = \dfrac{z_i}{\lambda^*} \\ \qquad = \dfrac{z_i}{\sum_{i'=1}^{k} z_{i'}} \end{cases}$$

Exercise 3

Mixture of Gaussians

In conclusion, the optimal value of c is given by

$$\begin{cases} c_i^* = \dfrac{z_i}{\lambda^*} \\ \quad = \dfrac{z_i}{\sum_{i'=1}^{k} z_{i'}} \\ \quad = \dfrac{\sum_{j=1}^{m} P(i|x_j)}{\sum_{i'=1}^{k} \sum_{j=1}^{m} P(i'|x_j)} \end{cases}$$

Exercise

PCA

- See file `ex_pca.pdf` in Aula Global