CSE7850/CX4803 Machine Learning in Computational Biology

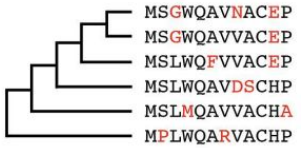
Lecture 11: Unsupervised Learning
(Clustering & Dimensionality Reduction)

Yunan Luo

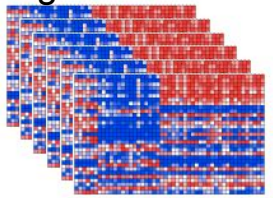
Acknowledgements: Some of the slides were adapted from lectures of Jian Peng

Week	Date	Topic	Contents
1	01/08	Introduction	Introduction & Logistics
1	01/10	Basics in computational biology	Molecular biology
2	01/15		No class (MLK day)
2	01/17		Sequence alignment I
3	01/22		Sequence alignment II
3	01/24	ML foundations	No Class (PyTorch video + exercise)
4	01/29		Regression & Gradient descent
4	01/31		Classification & Toolbox for Applied ML
5	02/05		Neural networks
5	02/07		Deep learning
6	02/12	Learning from sequence data	Deep learning for Protein/DNA sequences
6	02/14		Large language models (LLMs)
7	02/19	Learning from high-dim data	Clustering and dimensionality reduction
7	02/21		Generative AI
8	02/26	Learning from network data	Network basics & ML for graphs
8	02/28		Graph neural network
9	03/04	Learning from structure data	Protein structure prediction & generation (AlphaFold, diffusion models)

Sequence

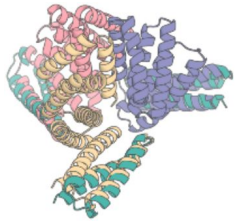


High-dim data



Network

Structure



Supervised vs Unsupervised learning

- Supervised learning: Given $(x_i, y_i), i = 1, \ldots, n$, learn a function $f : X \to Y$.
  - Categorical Y: classification
  - Continuous Y: regression
- Unsupervised learning: Given only $(x_i), i = 1, \ldots, n$, can we infer the underlying structure of X?

Slides credit: Sriram Sankararaman

Why do unsupervised learning?

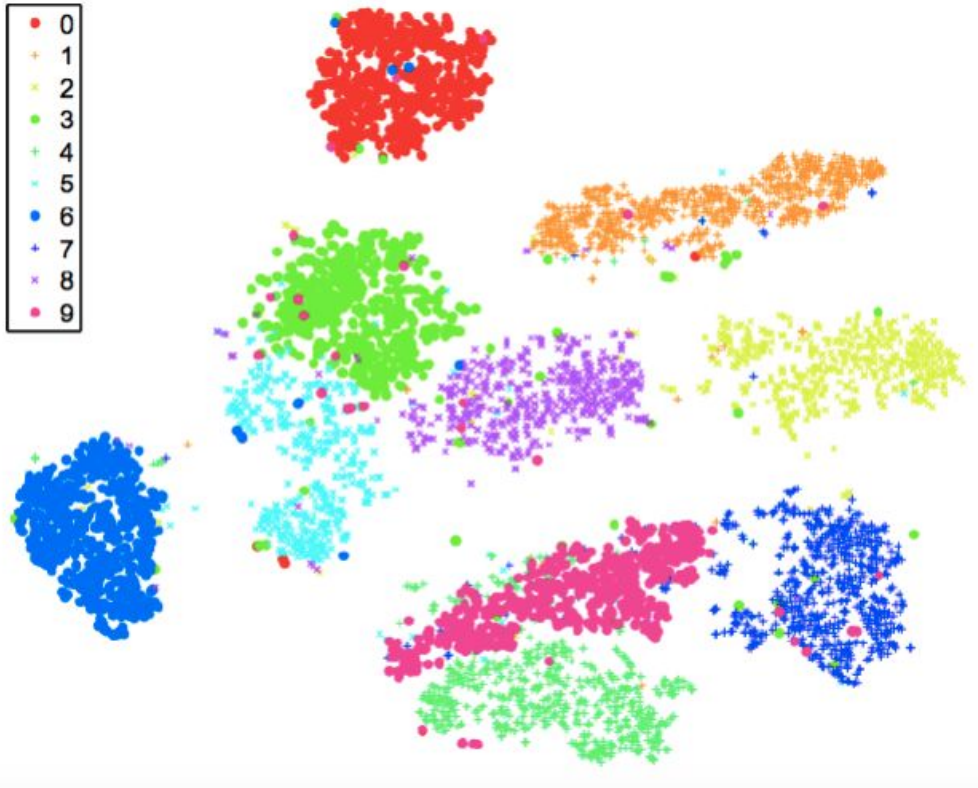
- Raw data cheap. Labeled data expensive.
- Save memory/computation.
- Reduce noise in high-dimensional data.
- Useful in exploratory data analysis.
- Often a pre-processing step for supervised learning.

Slides credit: Sriram Sankararaman

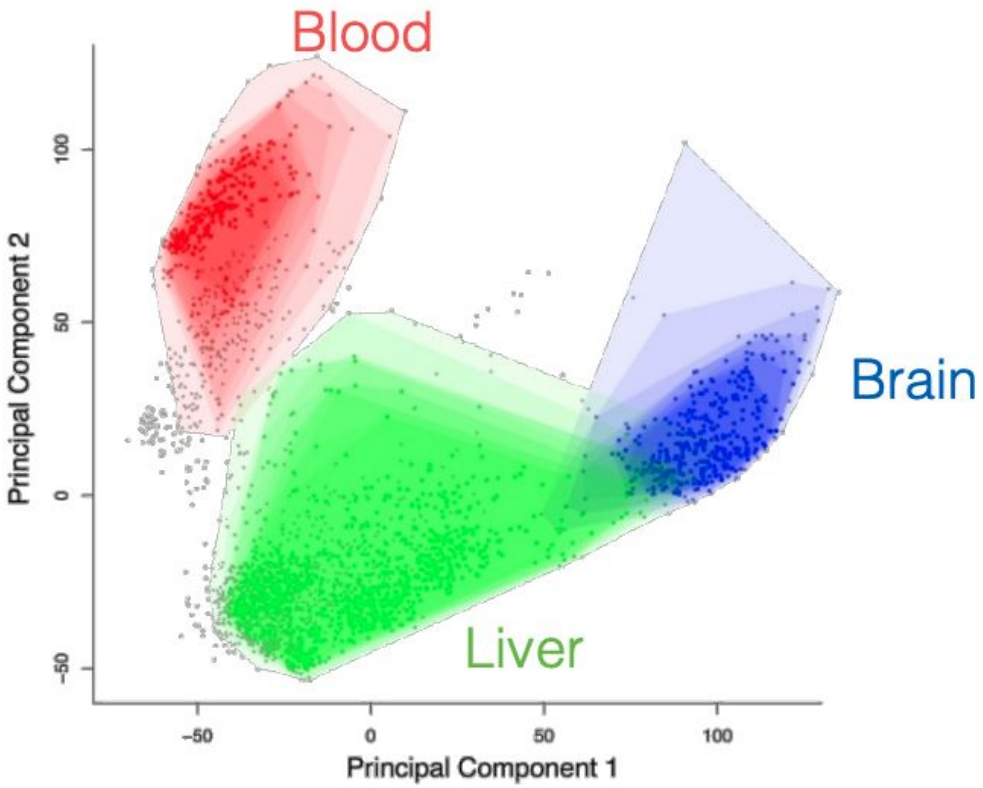
This lecture

- **Clustering**
  - K-means
- Dimensionality Reduction
  - PCA
  - Auto-encoder

Finding hidden structure in data



Expression analysis



Single-cell expression analysis

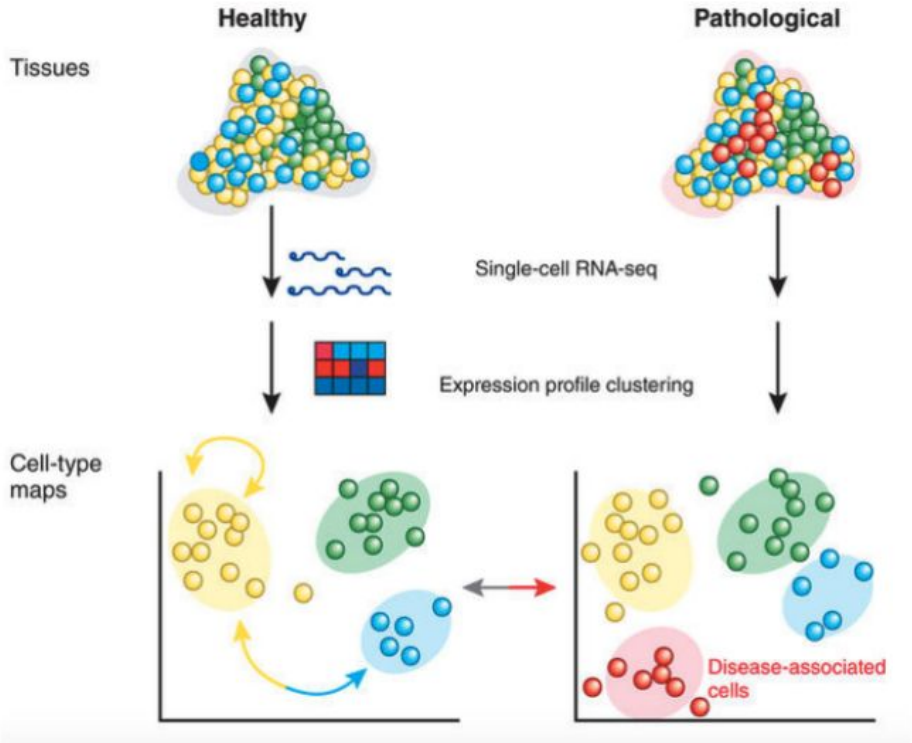
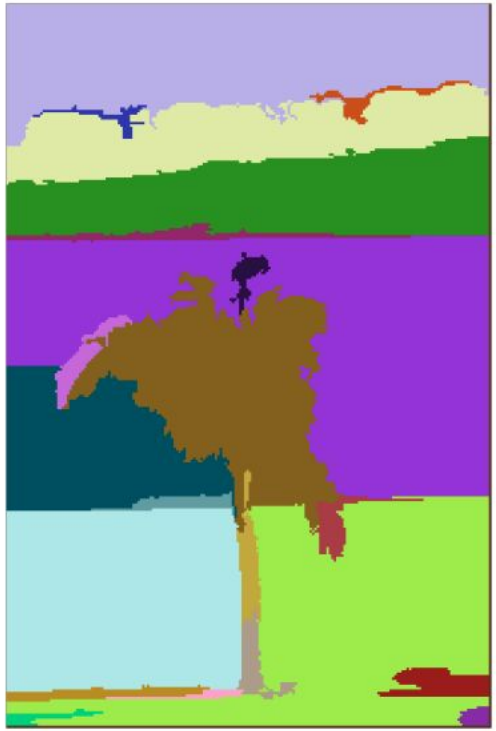
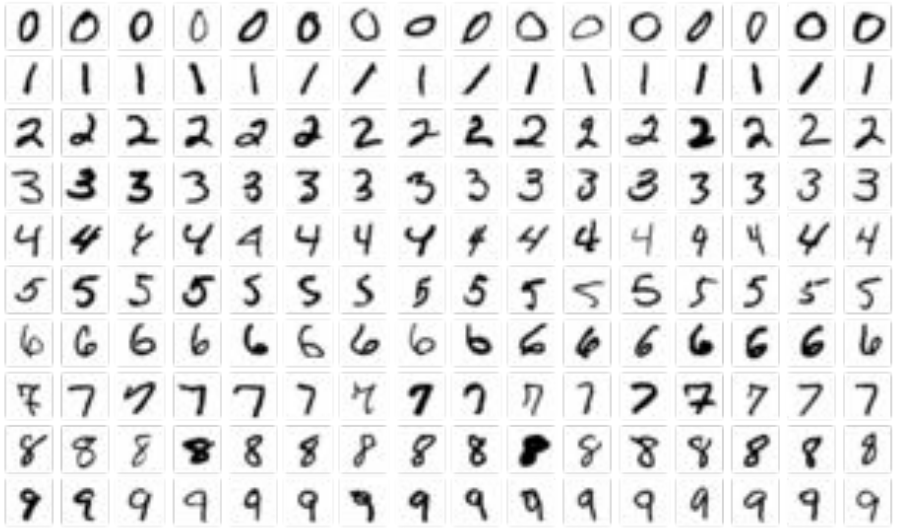


Image segmentation

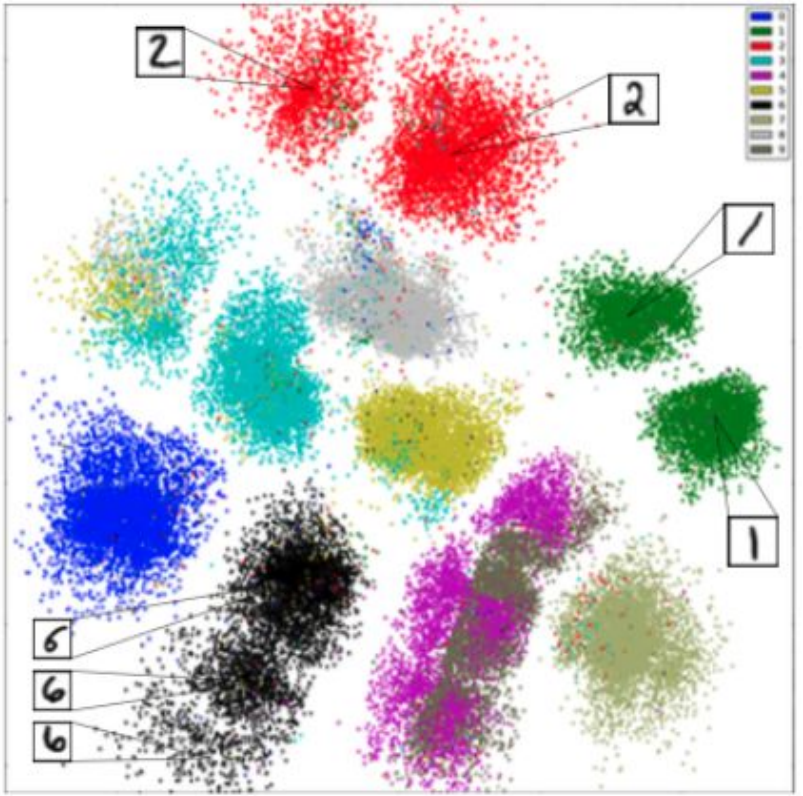


http://people.cs.uchicago.edu/ pff/segment

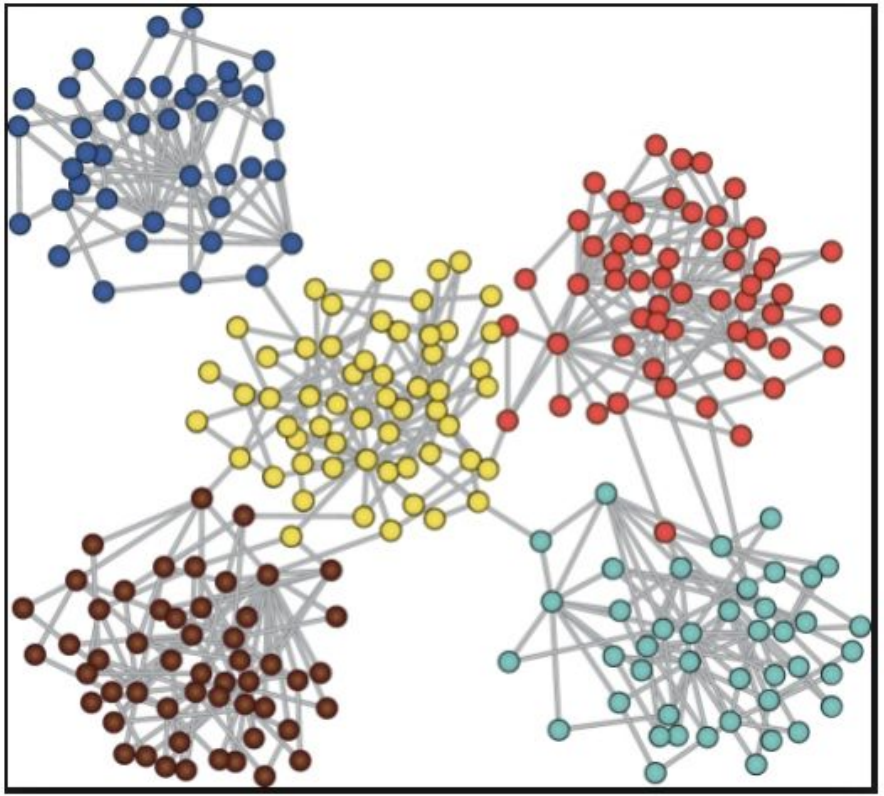
Discovering Structure in Digits



MNIST dataset

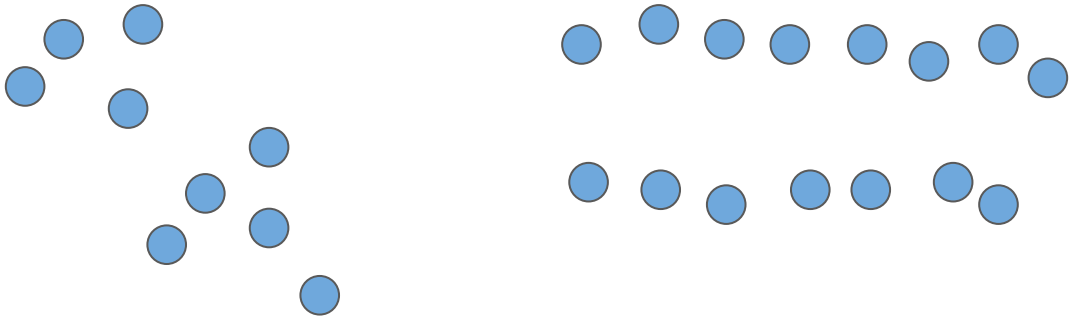


Network clustering



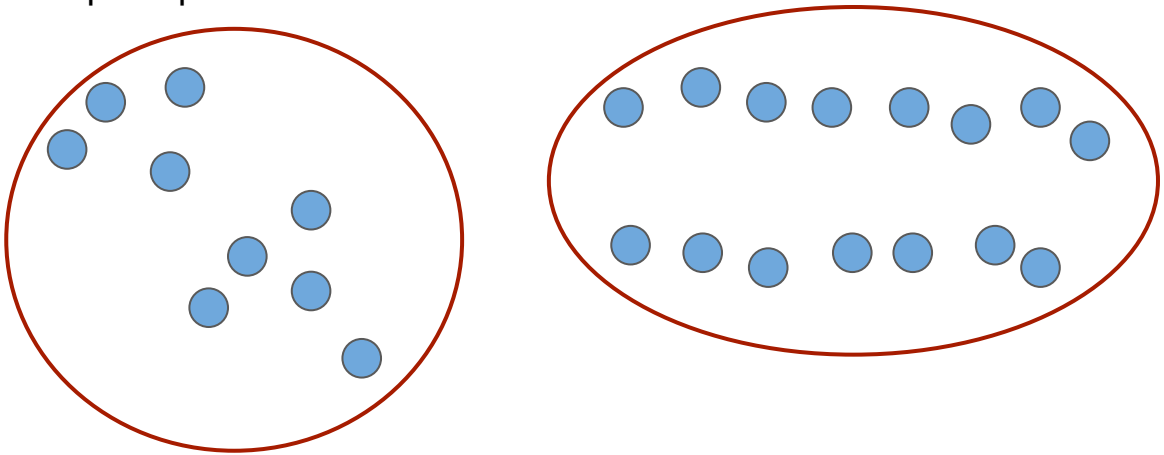
Clustering

- **Basic idea**: group together similar instances
- **Example**: 2D point patterns



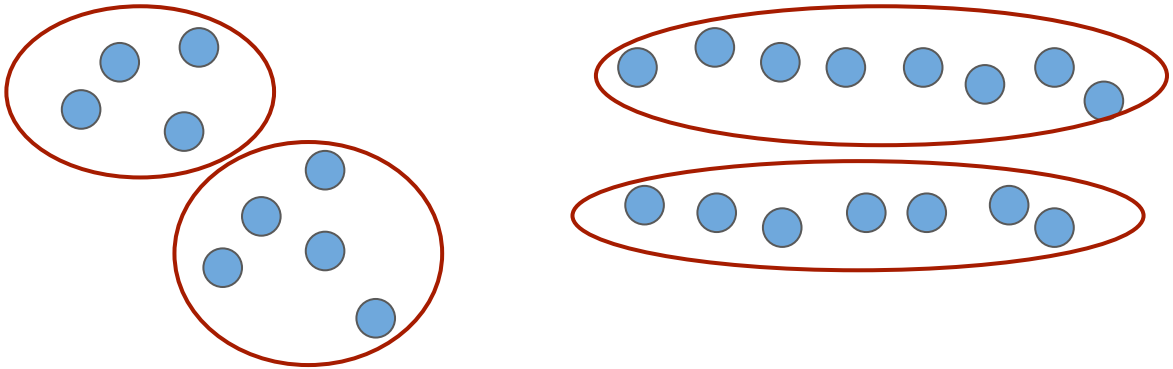
Clustering

- **Basic idea**: group together similar instances
- **Example**: 2D point patterns



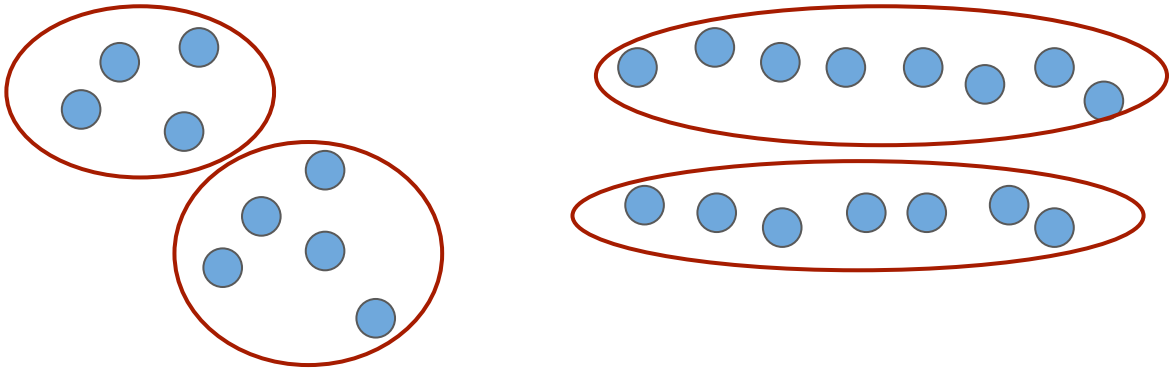
Clustering

- **Basic idea**: group together similar instances
- **Example**: 2D point patterns



Clustering

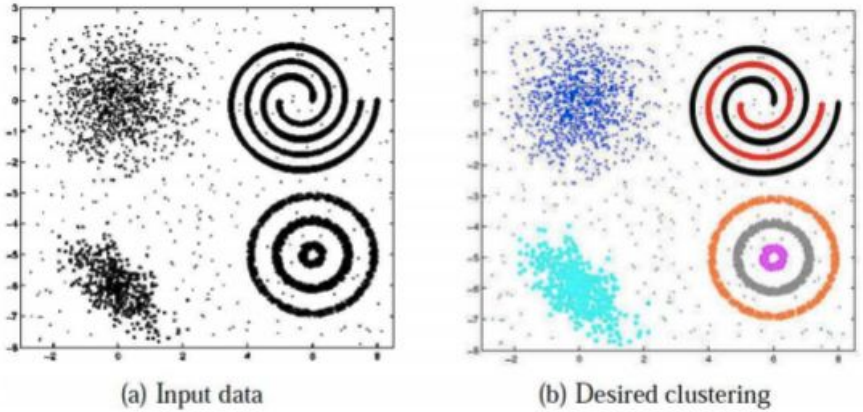
- **Basic idea**: group together similar instances
- **Example**: 2D point patterns



- What could “similar” mean?
  - One option: small Euclidean distance $\text{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$
  - Clustering results are crucially dependent on the measure of similarity (or distance) between “points” to be clustered

Clustering Problem

Given: N unlabeled examples $X = [x_1, x_2, \ldots, x_N]$; the number of partitions ***K***
Goal: Group the examples into ***K*** partitions



(a) Input data (b) Desired clustering

The only information clustering uses is the similarity between examples
Clustering groups examples based of their mutual similarities

K-Means

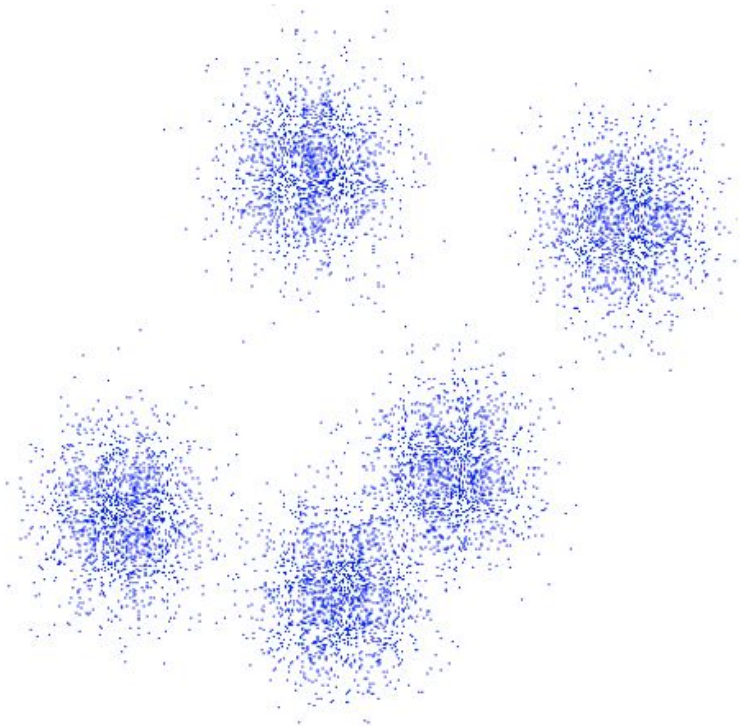
An iterative clustering algorithm

- **Initialize**: Pick K random points as cluster centers
- **Repeat**:
    a. Assign data points to closest cluster center
    b. Change the cluster center to the average of its assigned points
- **Stop** when no points' assignments change

K-Means

An iterative clustering algorithm

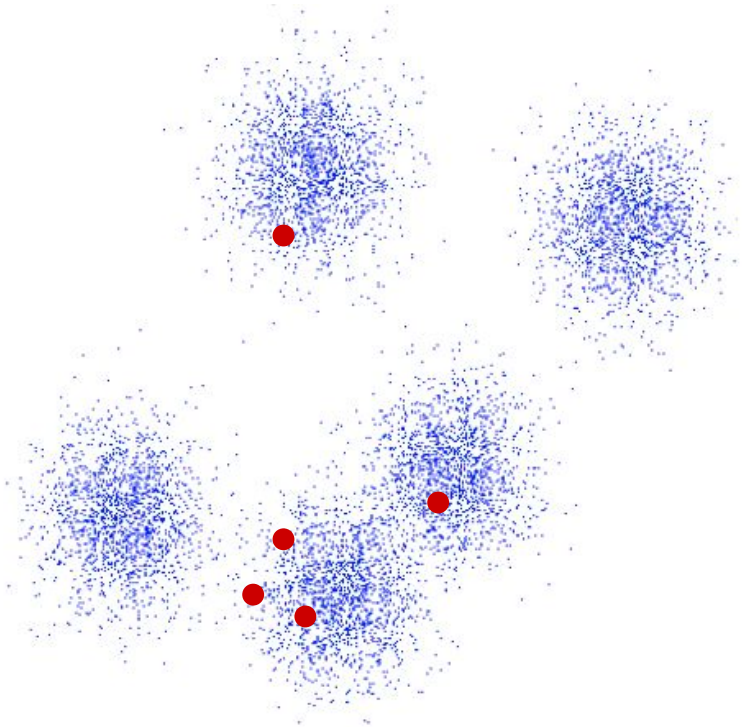
- **Initialize**: Pick K random points as cluster centers
- **Repeat**:
  a. Assign data points to closest cluster center
  b. Change the cluster center to the average of its assigned points
- **Stop** when no points' assignments change



K-Means

An iterative clustering algorithm

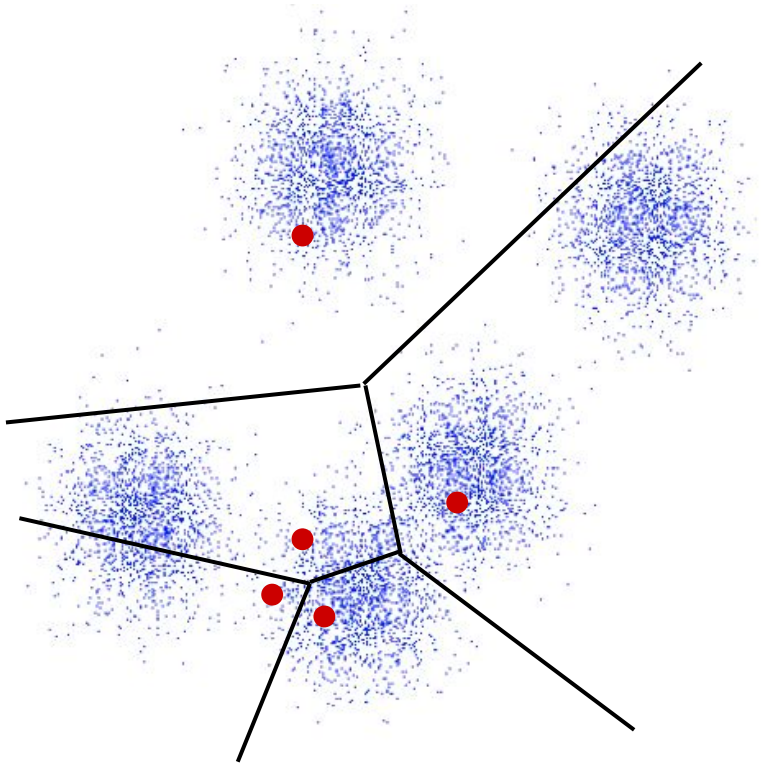
- **Initialize**: Pick K random points as cluster centers
- **Repeat**:
  a. Assign data points to closest cluster center
  b. Change the cluster center to the average of its assigned points
- **Stop** when no points' assignments change



K-Means

An iterative clustering algorithm

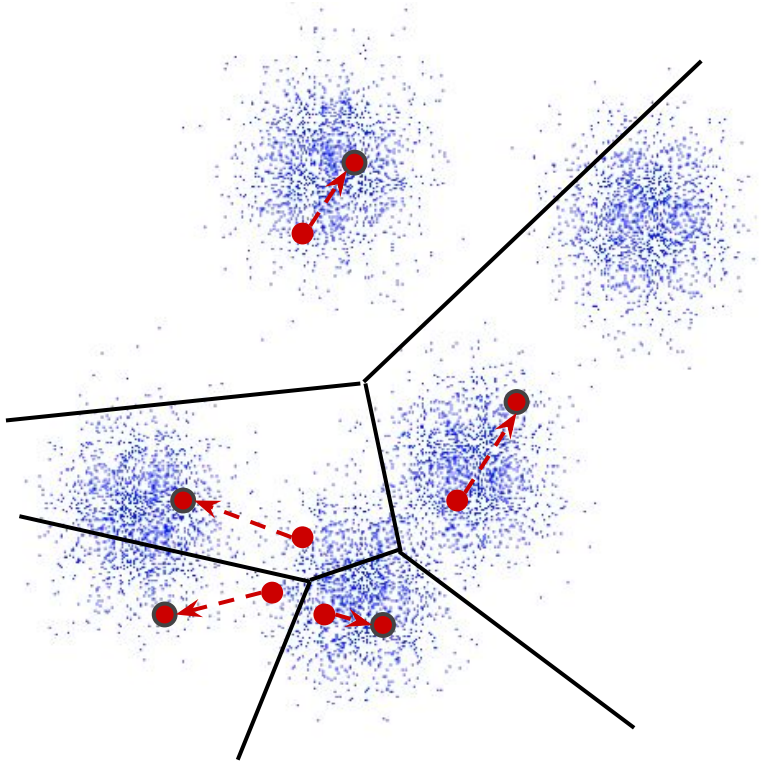
- **Initialize**: Pick K random points as cluster centers
- **Repeat**:
  a. Assign data points to closest cluster center
  b. Change the cluster center to the average of its assigned points
- **Stop** when no points' assignments change



K-Means

An iterative clustering algorithm

- **Initialize**: Pick K random points as cluster centers
- **Repeat**:
  a. Assign data points to closest cluster center
  b. Change the cluster center to the average of its assigned points
- **Stop** when no points' assignments change



K-Means

Input: N examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ ($\mathbf{x}_n \in \mathbb{R}^D$); the number of partitions K

Initialize: K cluster centers $\mu_1, \ldots, \mu_K$. Several initialization options:

- Randomly initialized anywhere in $\mathbb{R}^D$
- Choose any K examples as the cluster centers

Iterate:

- Assign each of example $\mathbf{x}_n$ to its closest cluster center

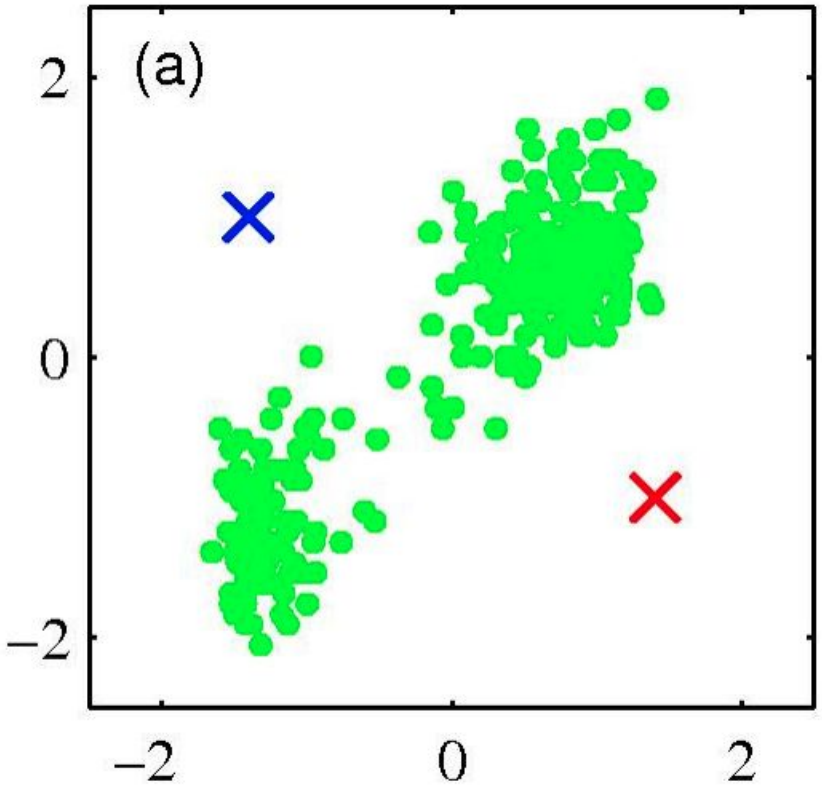
$$C_k = \{n : \quad k = \arg\min_k ||\mathbf{x}_n - \mu_k||^2\}$$

  (C_k is the set of examples closest to μ_k)
- Recompute the new cluster centers μ_k (mean/centroid of the set C_k)

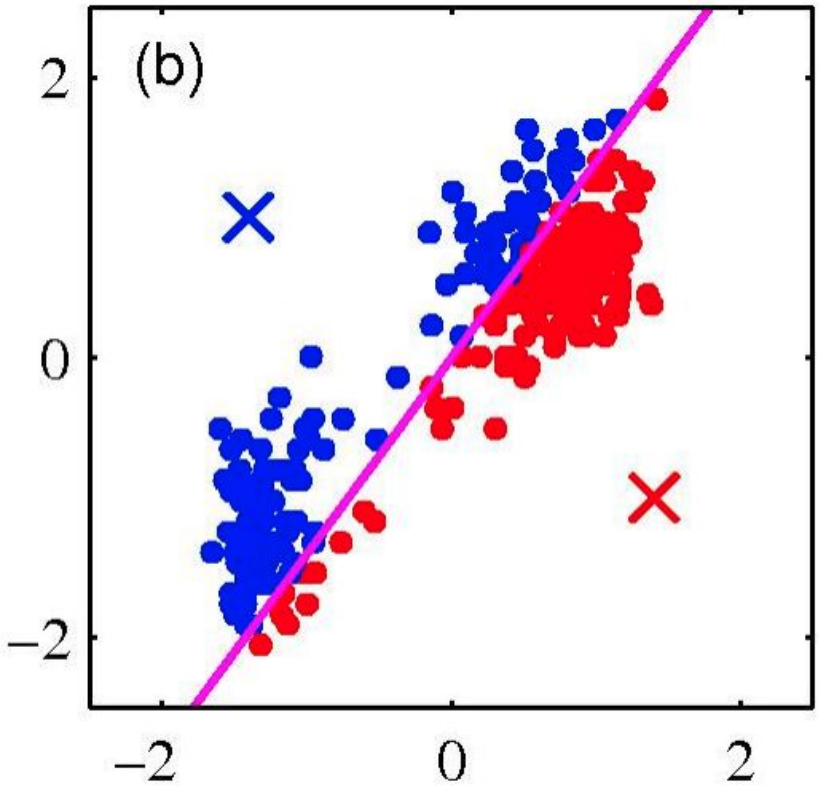
$$\mu_k = \frac{1}{|C_k|} \sum_{n \in C_k} \mathbf{x}_n$$

- Repeat while not converged

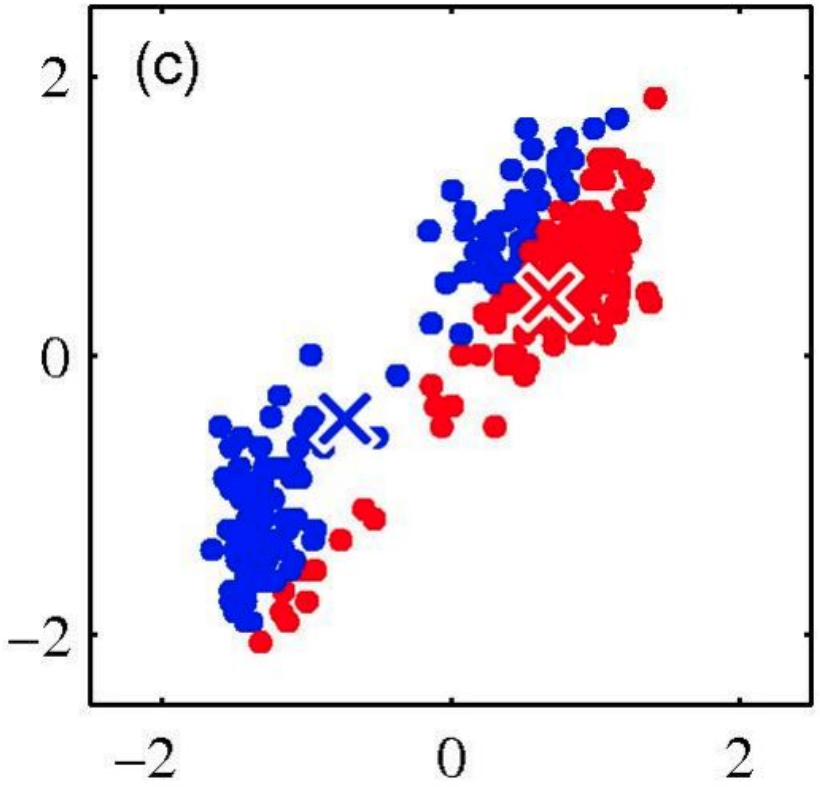
K-Means Clustering: Example



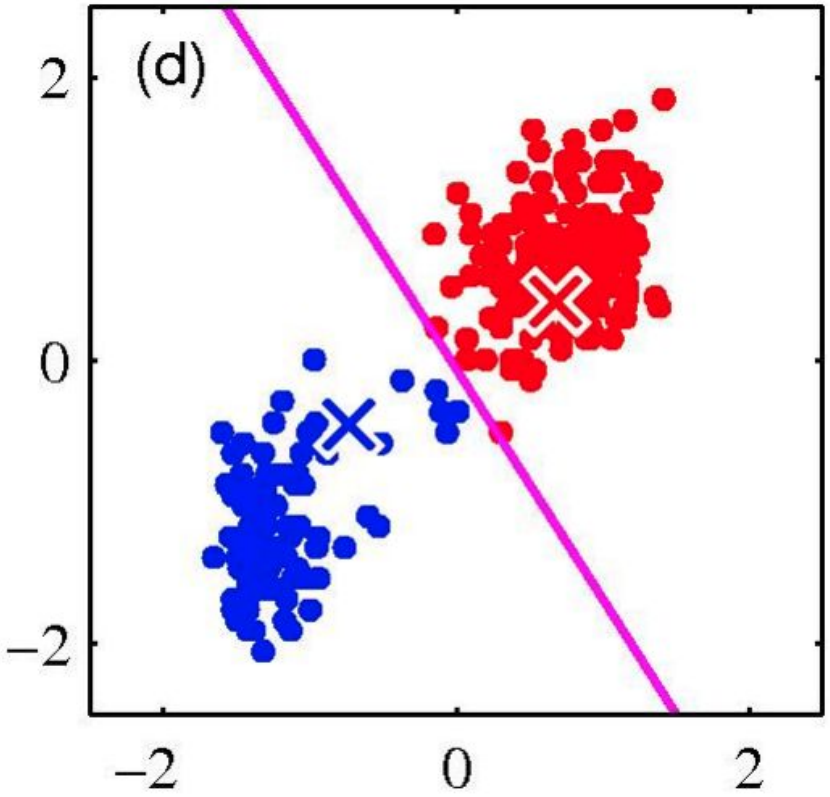
K-Means Clustering: Example



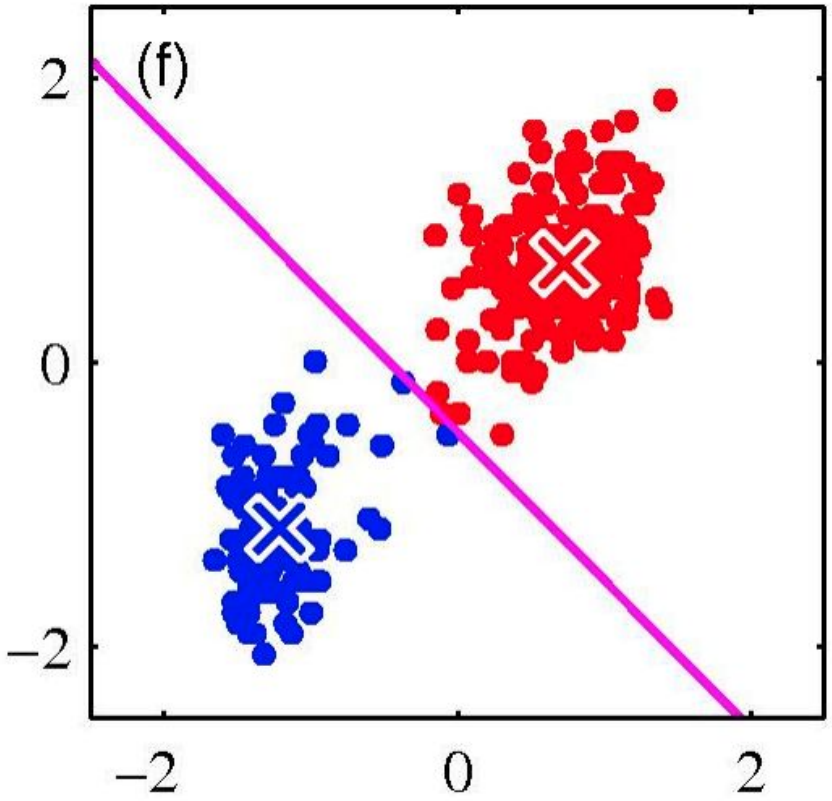
K-Means Clustering: Example



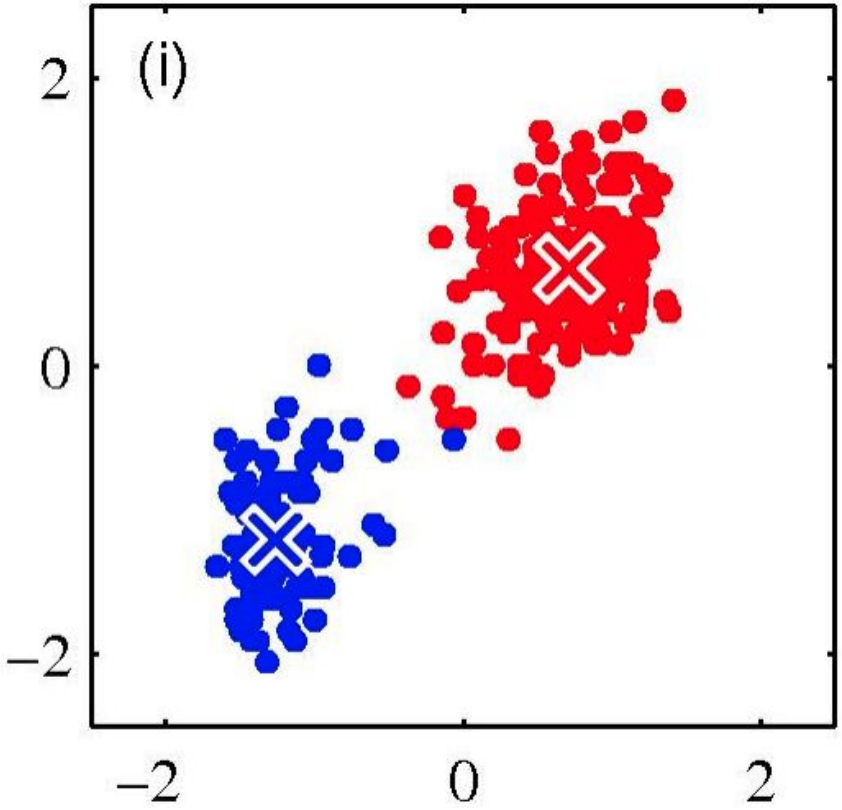
K-Means Clustering: Example



K-Means Clustering: Example



K-Means Clustering: Example



Properties of K-means algorithm

- Guaranteed to converge in a finite number of iterations
- Running time per iteration:
  - Assign data points to closest cluster center **O(KN) time**
  - Change the cluster center to the average of its assigned points **O(N) time**

K-Means Convergence

Objective

$$\min_{\mu}\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix μ, optimize C:

$$\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 = \min_{c} \sum_{i}^{n} \left|x_i - \mu_{x_i}\right|^2$$

Step 1 of kmeans

Assign data points to closest cluster center

2. Fix C, optimize μ:

$$\min_{\mu} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

– Take partial derivative of μ_i and set to zero, we have

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

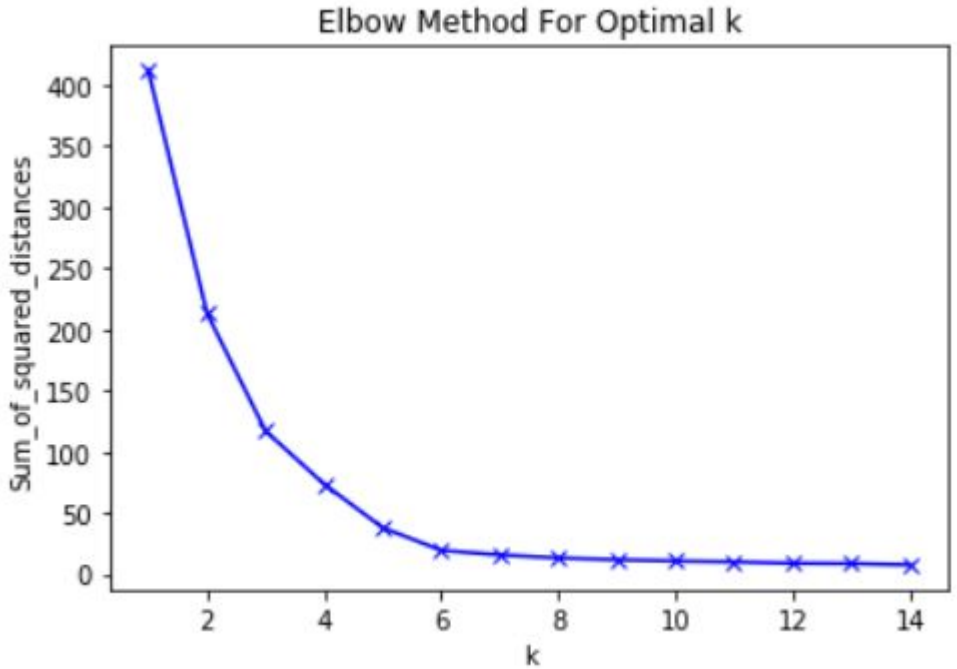
Step 2 of kmeans

Change the cluster center to the average of its assigned points

Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

How to choose *K*?

$$\min_{\mu}\min_{C}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2$$



How to choose *K*?

- Gap statistic
- Cross-validation: Partition data into two sets. Estimate prototypes on one and use these to compute the loss function on the other.
- Stability of clusters: Measure the change in the clusters obtained by resampling or splitting the data.
- Non-parametric approach: Place a prior on K.

K-Means for image compression

K=2 K=3 K=10 Original

K-Means for image compression

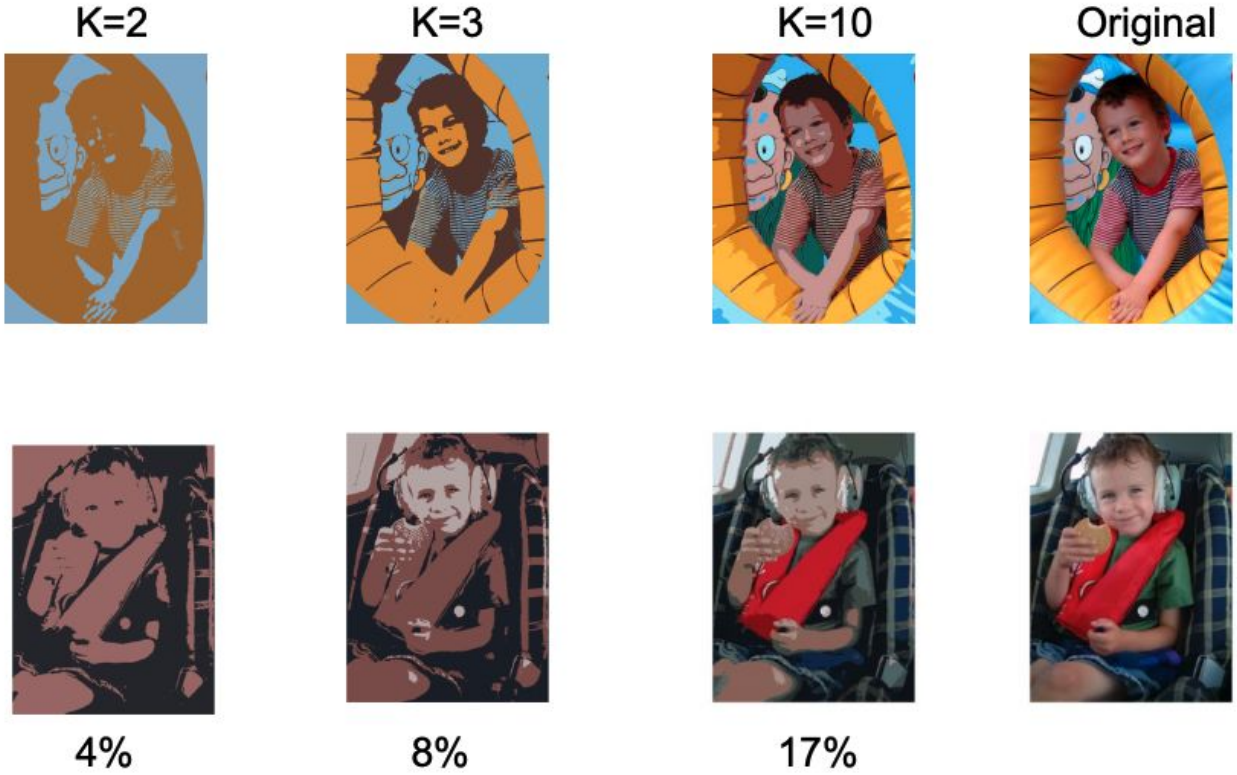
K=2 K=3 K=10 Original




K-Means for image compression

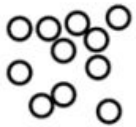
K-Means for image compression

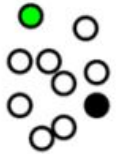
K-Means for image compression

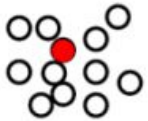


Impact of initialization

- **Initialize**: Pick K random points as cluster centers
- **Repeat**:
  a. Assign data points to closest cluster center
  b. Change the cluster center to the average of its assigned points
- **Stop** when no points' assignments change

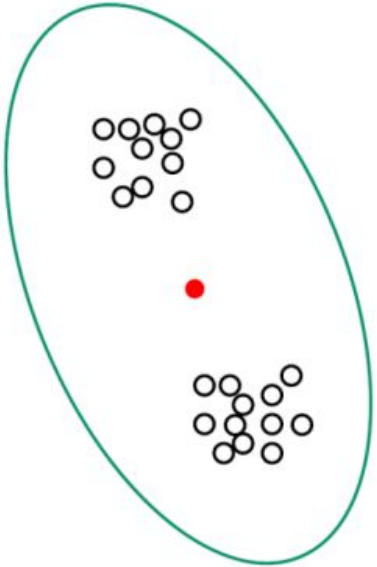






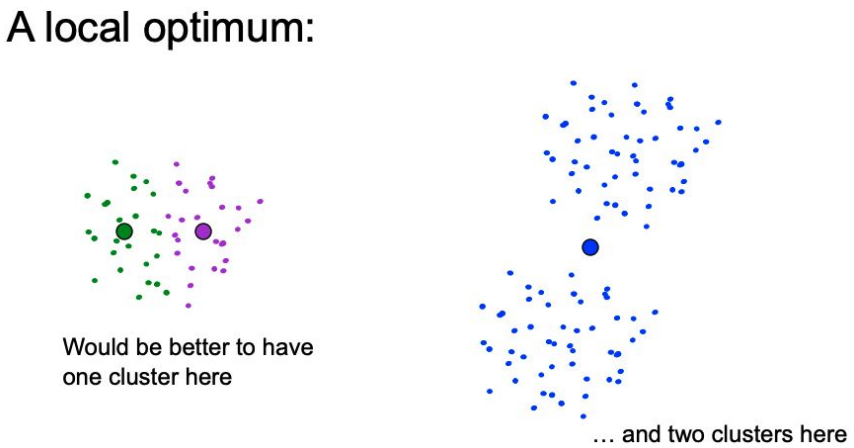
Impact of initialization

- **Initialize**: Pick K random points as cluster centers
- **Repeat**:
  a. Assign data points to closest cluster center
  b. Change the cluster center to the average of its assigned points
- **Stop** when no points' assignments change



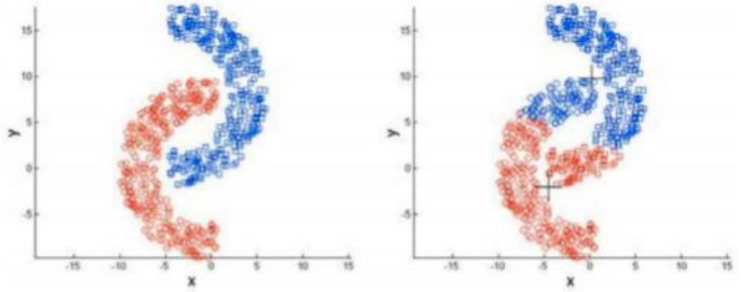
Trying to find good optima

- Idea 1: Be careful about where you start
  - K-Means++
- Idea 2: Do many runs of k-means, each from a different random start configuration

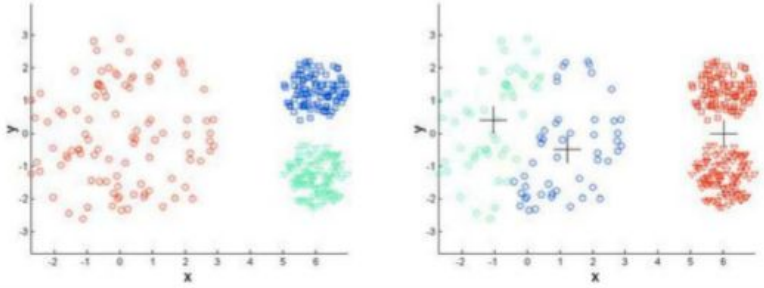


When will K-Means fail?

Non-convex/non-round-shaped clusters: Standard K-means fails!

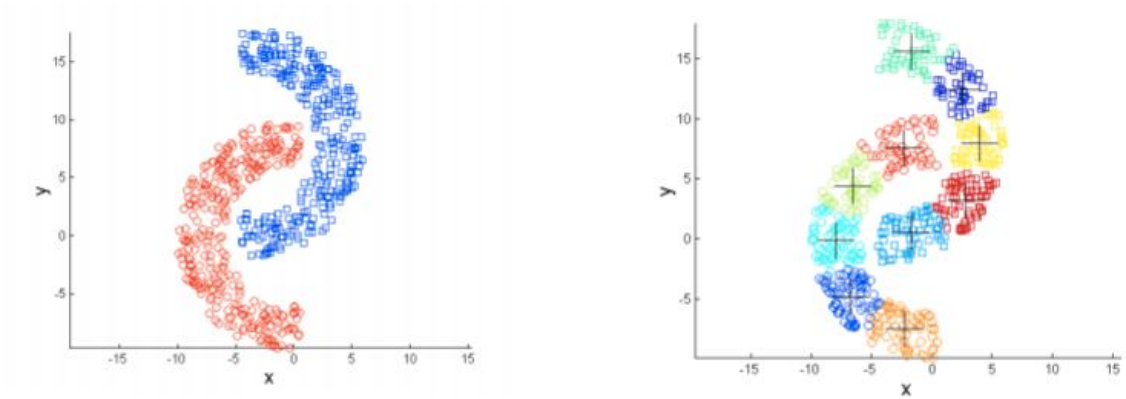


Clusters with different densities



Hierarchical clustering

A hierarchical approach can be useful when considering versatile cluster shapes:

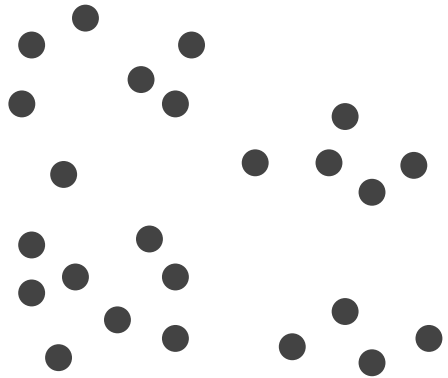


10-means

By first detecting many small clusters, and then merging them, we can uncover patterns that are challenging for partitional methods.

Agglomerative Clustering

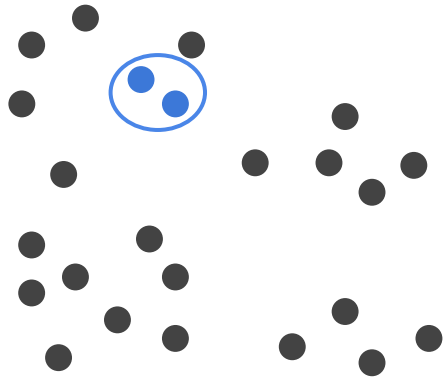
- Start with every point as a single cluster
- Repeat
  - Find “most similar” pair of clusters
  - Merge them into a “super point”
- Stop when there is only one cluster left



Produces not one clustering, but a family of clusterings represented by a **dendrogram**

Agglomerative Clustering

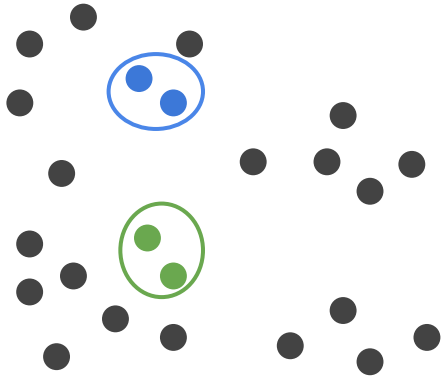
- Start with every point as a single cluster
- Repeat
  - Find “most similar” pair of clusters
  - Merge them into a “super point”
- Stop when there is only one cluster left



Produces not one clustering, but a family of clusterings represented by a **dendrogram**

Agglomerative Clustering

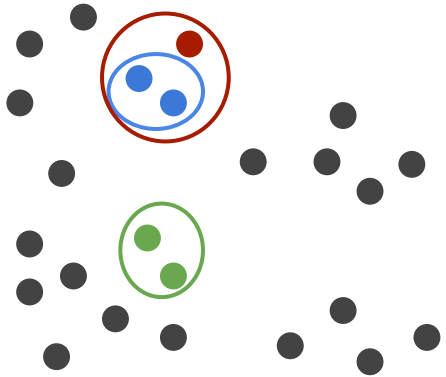
- Start with every point as a single cluster
- Repeat
  - Find “most similar” pair of clusters
  - Merge them into a “super point”
- Stop when there is only one cluster left



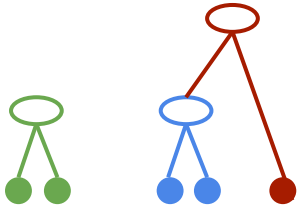
Produces not one clustering, but a family of clusterings represented by a **dendrogram**

Agglomerative Clustering

- Start with every point as a single cluster
- Repeat
  - Find “most similar” pair of clusters
  - Merge them into a “super point”
- Stop when there is only one cluster left

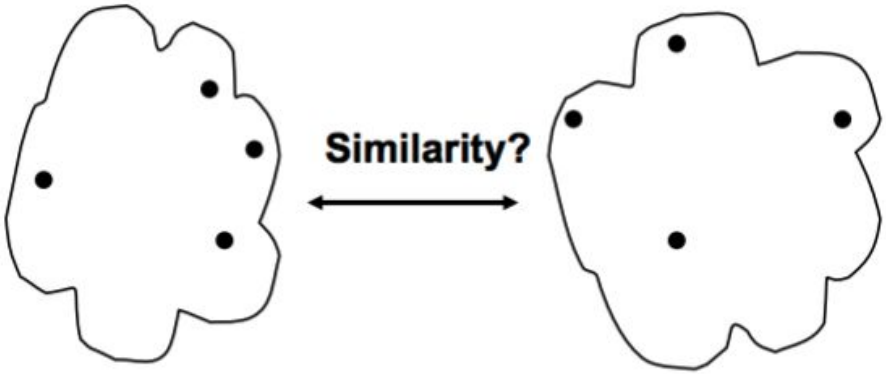


Produces not one clustering, but a family of clusterings represented by a **dendrogram**



Agglomerative Clustering

- Start with every point as a single cluster
- Repeat
  - Find “most similar” pair of clusters
  - Merge them into a “super point”
- Stop when there is only one cluster left



We need a notion of similarity between clusters.

Cluster Distance

Closest pair (single-link)

$$CD(X,Y)=\min_{x \in X,\, y \in Y} D(x,y)$$

Furthest pair (complete-link)

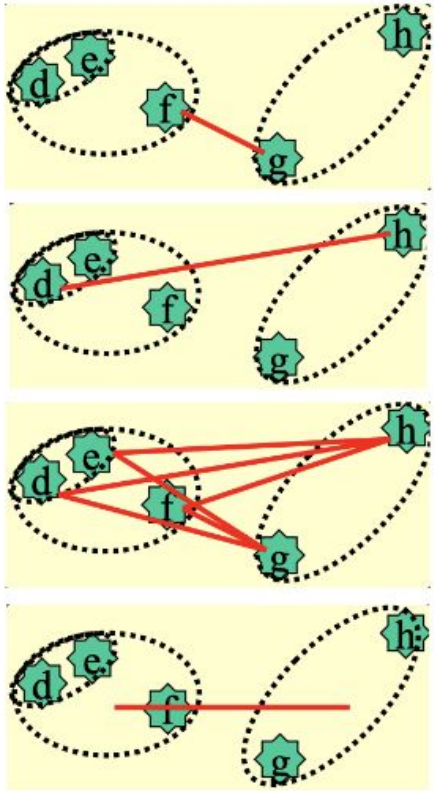
$$CD(X,Y)=\max_{x \in X,\, y \in Y} D(x,y)$$

average-link

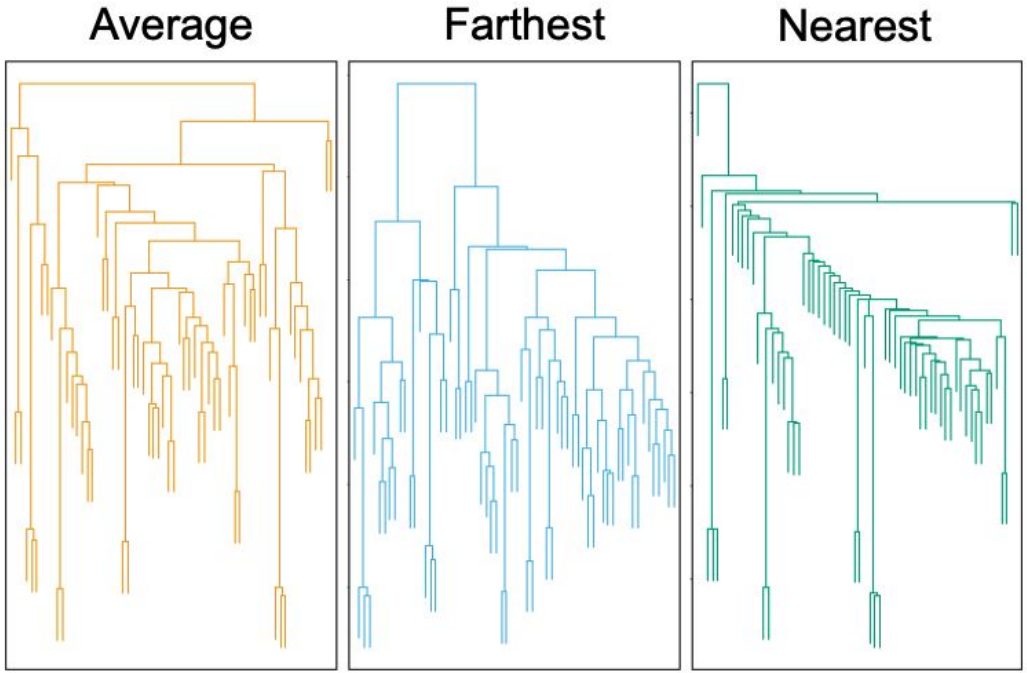
$$CD(X,Y)=\text{avg}_{x \in X,\, y \in Y} D(x,y)$$

centroids method

$$CD(X,Y)=D(\text{avg}(X), \text{avg}(Y))$$

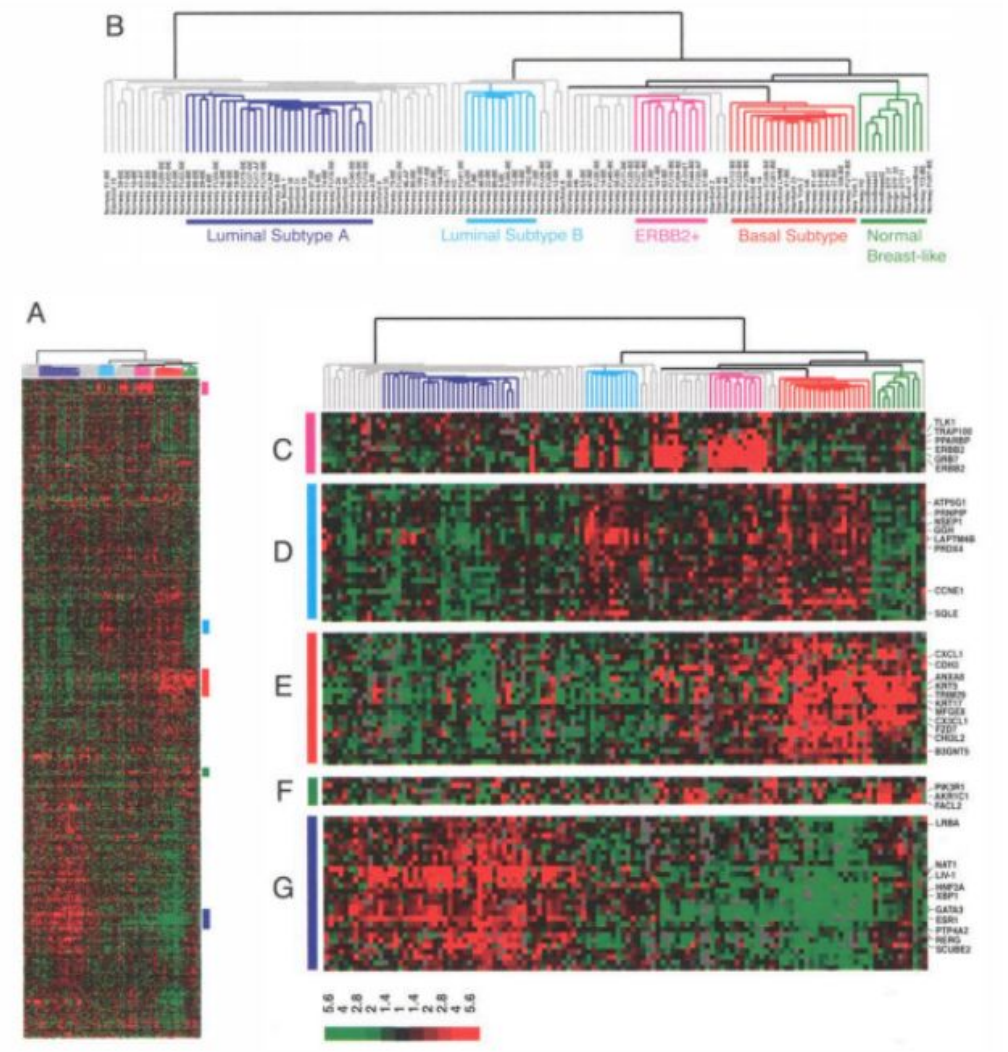


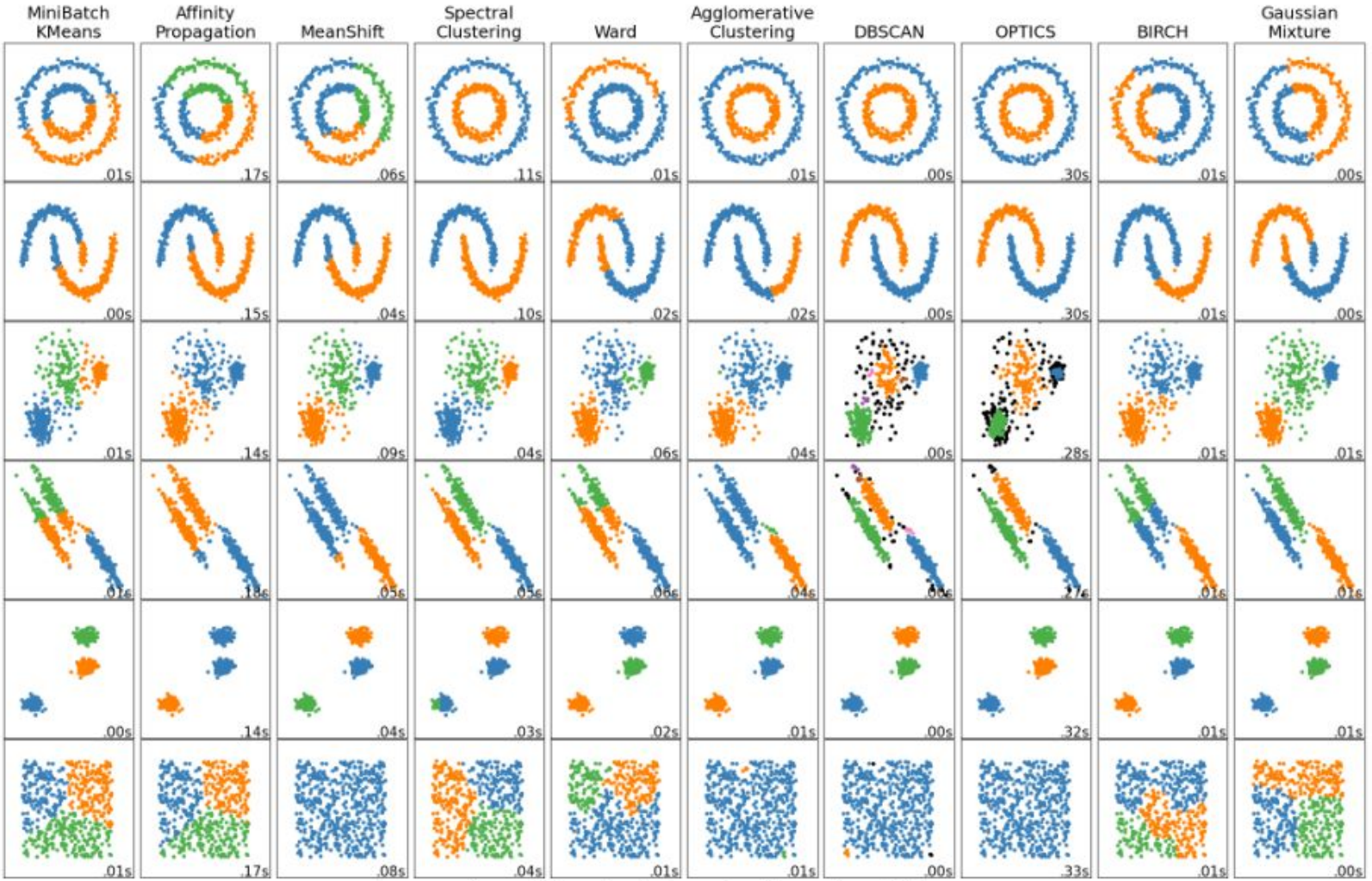
Clustering Behavior



Mouse tumor data from [Hastie *et al.*]

Application to breast cancer expression data





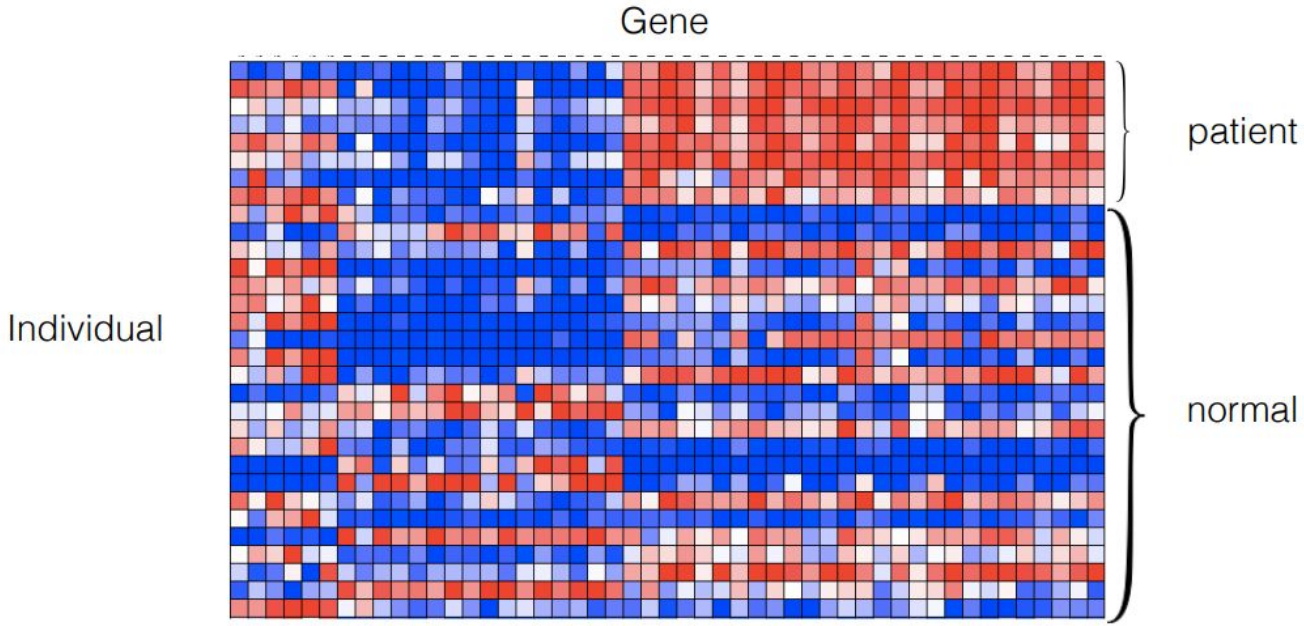
https://scikit-learn.org/stable/modules/clustering.html

A comparison of the clustering algorithms in scikit-learn

This lecture

- Clustering
  - K-means
- **Dimensionality Reduction**
  - PCA
  - Auto-encoder

Gene expression matrix



dim(features) >> num(samples)

Slides credit: CS598JP, UIUC, 2020

High-dimensional data

High-dimensional data

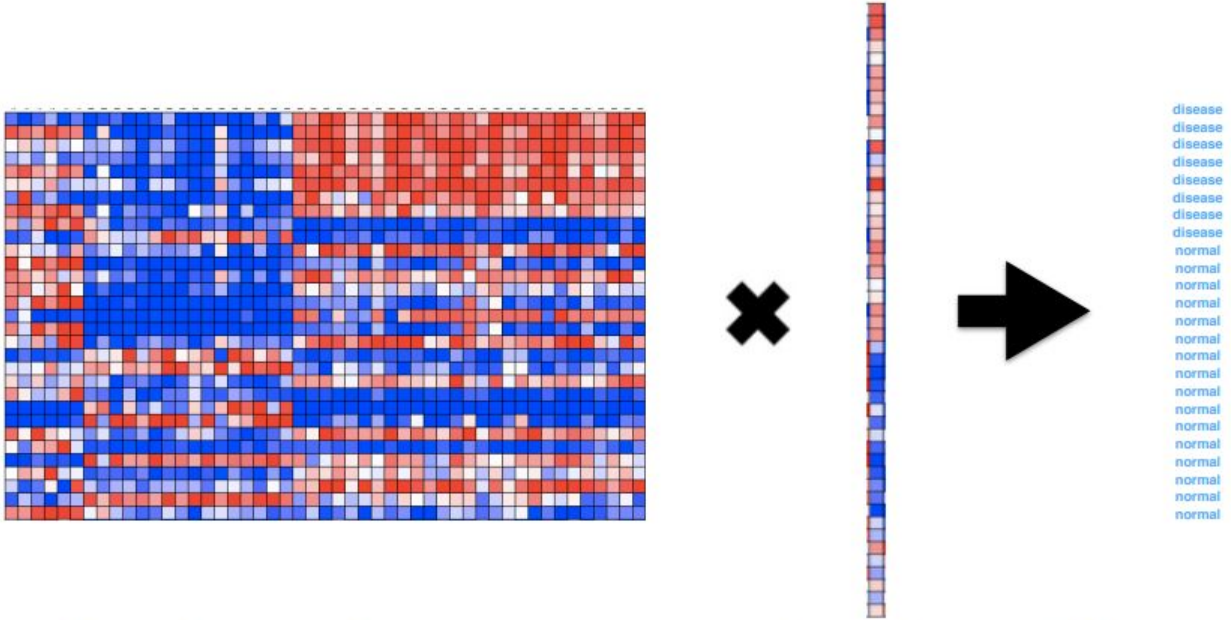
- Each sample has a large number of features/attributes

Why is high-dimension a problem? The curse of dimensionality:

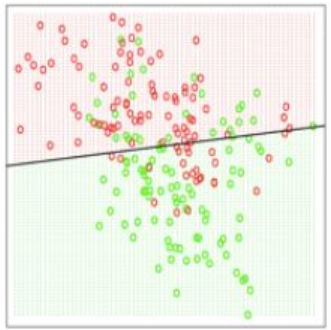
- Volume of space increases exponentially so data becomes very sparse; sparsity
- Increases the effort of searching drastically
- Makes it harder to calculate (accurate) distances between samples
- Redundancy of data
- A large number of training data samples is required to train a model for high-dim data
- Overfitting

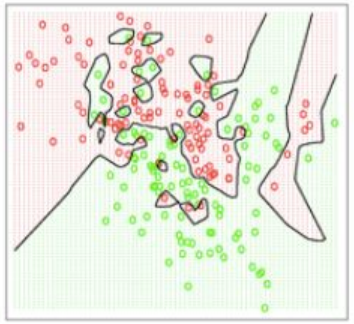
Slides credit: Xiuwei Zhang

Overfitting



$$p(\text{number of parameters}) \gg n(\text{number of data points})$$





Slides credit: CS598JP, UIUC, 2020

A solution: dimensionality reduction

Benefits:

- Reducing redundancy of data
- Identifying the most relevant information (find and filter noise) & Clean the data
- Reducing computational complexity & Speeding up subsequent learning task
- Building simpler model later
- Visualizing, exploring and understanding the data

Slides credit: Xiuwei Zhang

Dimensionality reduction: approaches

- Linear transformation:
  - PCA
  - NMF
- Non-linear transformation
  - Autoencoder, VAE
  - MDS
  - tSNE
  - UMAP
- Different methods have different *objectives*

Slides credit: Xiuwei Zhang

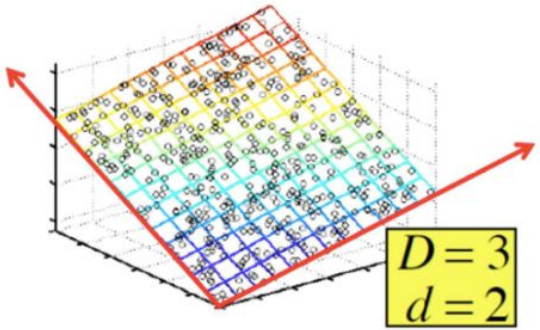
Principal Component Analysis (PCA)

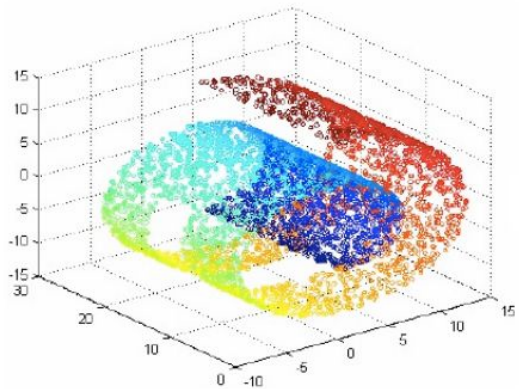
Component analysis

How to understand the main signals from the data?

Key assumptions

1. Low-rank assumption: High-dimensional data lies on a lower dimensional space (a.k.a, manifold)
2. Projections in the lower-dimensional space describes major properties of the data





Slides credit: CS598JP, UIUC, 2020

Principal Component Analysis (PCA)

Goal: Find a projection of the data onto directions that maximize variance of the original data

- **Intuition**: those are directions in which most information is encoded

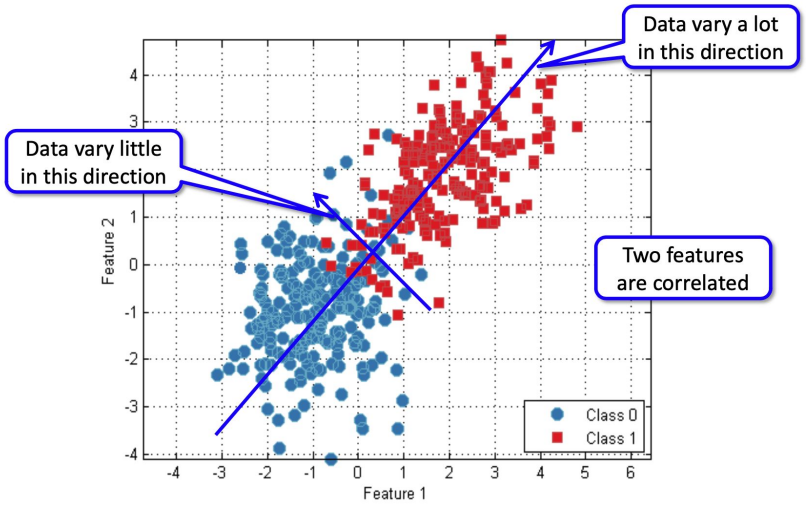


Figure credit: Le Song

Definition: Principal Components (PC) are orthogonal directions that capture most of the variance in the data

Adapted from CMSC422-0101

PCA: Finding principal components

- 1st PC:
  - Projection of data points along 1st PC discriminates data most along any one direction
- 2nd PC:
  - Next orthogonal direction of greatest variability
- 3rd PC…

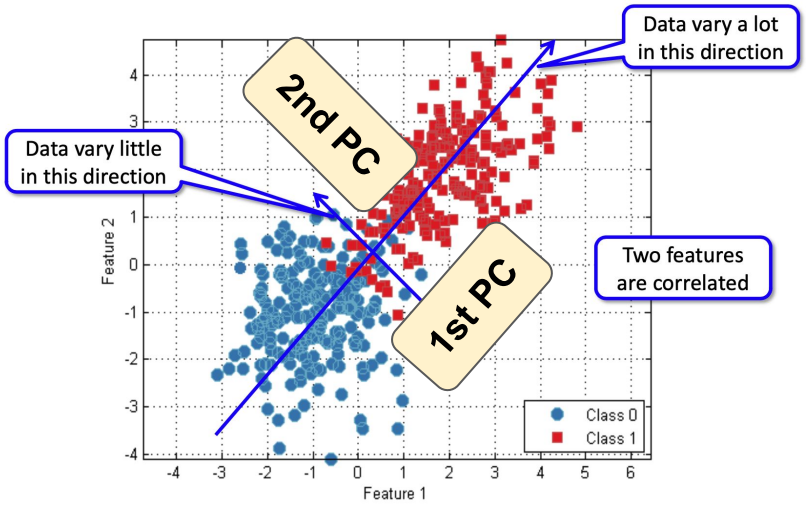


Figure credit: Le Song

PCA notation

- Input data points: matrix $X = [x_1, x_2, \ldots, x_N]$ of size $D \times N$
- x_i is the i-th column, i.e., the i-th example
- x_{ij} is the j-th feature of example i
- We assume the data is centered, i.e., $\frac{1}{N}\sum_{i=1}^{N} x_i = \vec{0}$
  - If not centered, replace x_i by $x_i - \mu$, where $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$

Finding the 1st PC

Given N data points, $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]_{D \times N}$, $\boldsymbol{x}_i \in \mathbb{R}^D$, find a direction $\boldsymbol{w}$ where $\|\boldsymbol{w}\| = 1$, such that the variation of the data along direction $\boldsymbol{w}$ is maximized.

- The sample variance on the projected on vector $\boldsymbol{w}$ is $\sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_i)^2 = \boldsymbol{w}^T \boldsymbol{X} \boldsymbol{X}^T \boldsymbol{w}$

Find the 1st PC by solving the following optimization problem

$$\max_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{X} \boldsymbol{X}^T \boldsymbol{w}$$

such that: $\|\boldsymbol{w}\| = 1$

Finding the 1st PC

$$\max_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{w}$$

such that: $\|\boldsymbol{w}\| = 1$

- Construct Lagrange multiplier

$$\max_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{w} - \lambda(\|\boldsymbol{w}\| - 1)$$

- Take the derivative with respect to $\boldsymbol{w}$ and set it to 0 => solutions are vectors $\boldsymbol{w}$ such that

$$\boldsymbol{X}\boldsymbol{X}^T \boldsymbol{w} = \lambda \boldsymbol{w}$$

Eigenvector of $\boldsymbol{X}\boldsymbol{X}^T$!

$$\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w} = \lambda\boldsymbol{w}$$

The eigenvalue problem

- For a given matrix ***A***

  $$\boldsymbol{Aw} = \boldsymbol{\lambda w}$$

  w is the eigenvector and ***λ*** is the eigenvalue

- There are multiple solutions $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, …,
  with different (or same) eigenvalues λ_1, λ_2, …

- The eigenvectors are orthonormal (symmetric,
  positive semi-definite)
  - $\boldsymbol{w}_i^T \boldsymbol{w}_j = 0$, $\boldsymbol{w}_i^T \boldsymbol{w}_i = 1$

- Let $\boldsymbol{A}=\boldsymbol{X}\boldsymbol{X}^T$ and find the eigenvectors and eigenvalues of ***A***

PCA formally

- If we rank eigenvalues from large to small
  - The 1st PC is the eigenvector of $\boldsymbol{XX^T}$ associated with the largest eigenvalue
  - The 2nd PC is the eigenvector of $\boldsymbol{XX^T}$ associated with the 2nd largest eigenvalue
  - …
- The eigenvalue $\lambda_i / \sum\lambda_i$ denotes the percentage of variance accounted for by the i-th PC $\boldsymbol{w_i}$

Adapted from CMSC422-0101

Q1: how to find eigenvalues/eigenvectors?

Singular value decomposition (SVD)

SVD in Python:

```
from scipy import linalg
U, s, Vh = linalg.svd(A)
```

The SVD is a factorization of a $m \times n$ matrix into

$$\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^{\boldsymbol{T}}$$

where $\boldsymbol{U}$ is a $m \times m$ orthogonal matrix, $\boldsymbol{V}^{\boldsymbol{T}}$ is a $n \times n$ orthogonal matrix and $\boldsymbol{\Sigma}$ is a $m \times n$ diagonal matrix.

$$\boldsymbol{A} = \begin{pmatrix} \vdots & \cdots & \vdots \\ \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_n \\ \vdots & \cdots & \vdots \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} \begin{pmatrix} \cdots & \mathbf{v}_1^T & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \mathbf{v}_n^T & \cdots \end{pmatrix}$$

Singular value decomposition (SVD)

Theorem: if a square matrix S is a real and symmetric matrix, then its SVD can be represented as

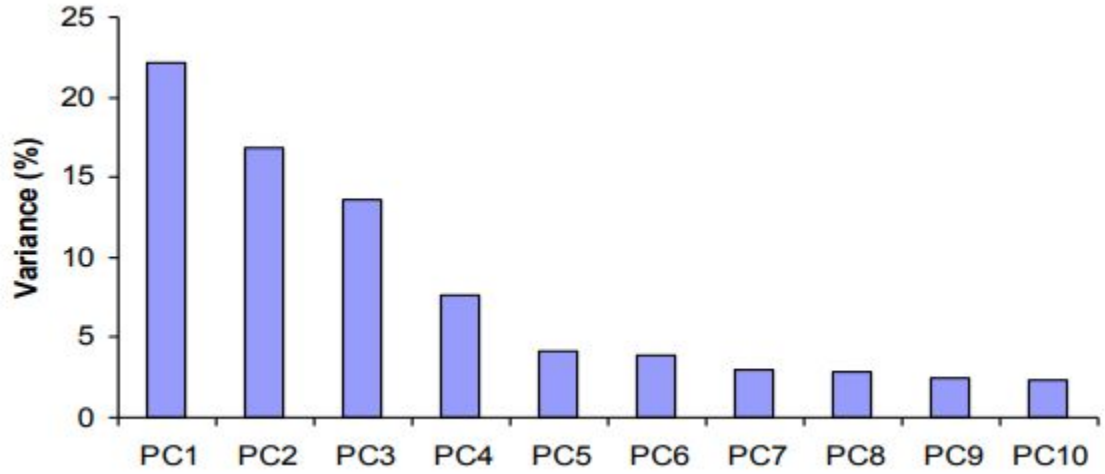
$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{T}}$$

where columns of **V** are eigenvectors of **S** and diagonal elements of **Λ** are eigenvalues of **S**

$$\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

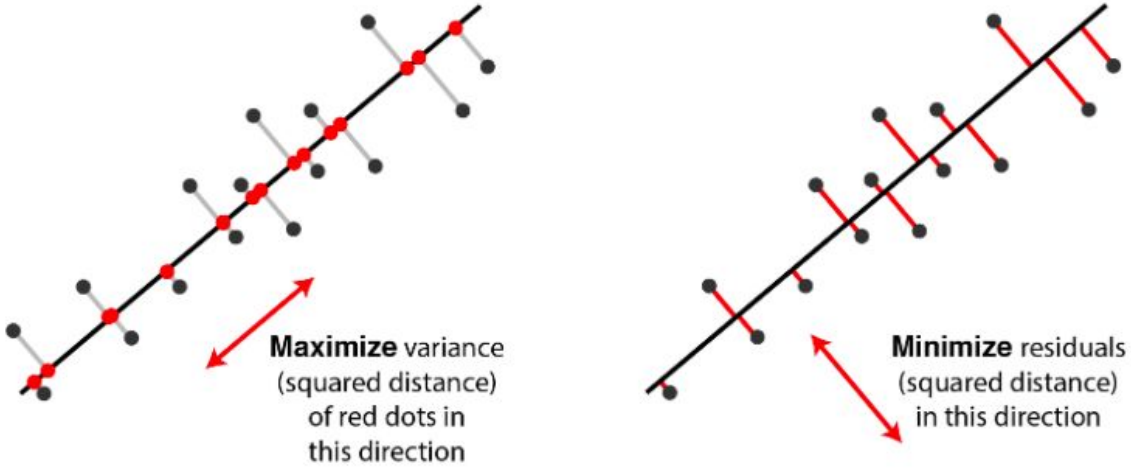
Q2: how many PCs

- The eigenvalue λ_i denotes the amount of variability captured along dimension $\boldsymbol{w_i}$
- Can ignore the components of lower variance (less significant)
- Choose the first K PCs such that the majority of variance (90%) can be explained
  - $\Sigma^{K}_{i=1} \lambda_i / \sum \lambda_i \geq 0.9$



Alternative interpretation 1: residual minimization

PCA finds vectors ***v*** such that projection on to these vectors minimizes reconstruction error



(image source)

Alternative interpretation 2: low-rank approximation

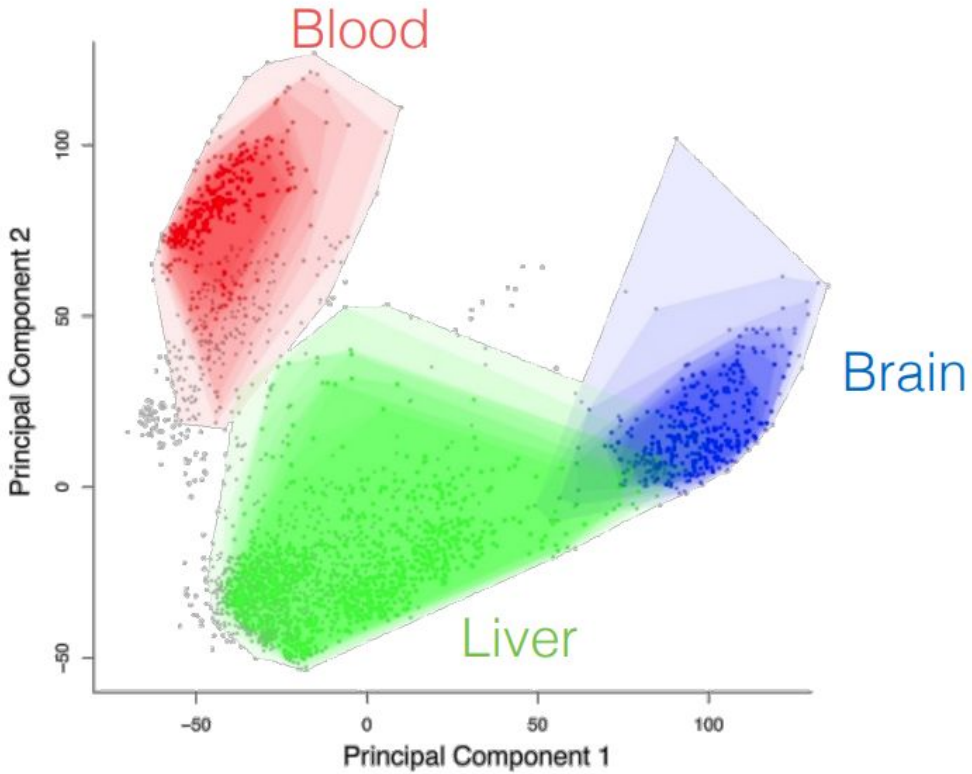
PCA seeks the best rank-k approximation to the matrix A in the least-squares sense, by solving

$$\begin{array}{ll} \text{minimize} & \|A - Z\|_F^2 \\ \text{subject to} & \text{Rank}(Z) \leq k, \end{array}$$

with variable $Z \in \mathbf{R}^{m \times n}$. Here, $\|\cdot\|_F$ is the Frobenius norm of a matrix, *i.e.*, the square root of the sum of the squares of the entries.

https://stanford.edu/~boyd/papers/pdf/glrm.pdf

Example: Tissue-specific gene expression



Slides credit: CS598JP, UIUC, 2020

Summary: PCA

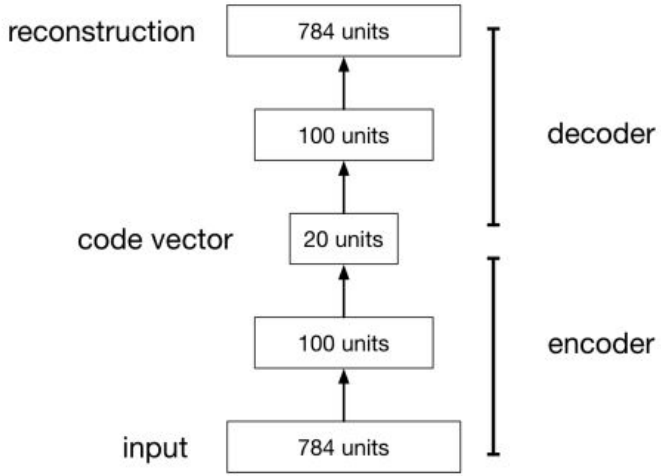
- What you should know:
  - Goal: Find a projection of the data onto directions that maximize variance of the original data
  - Optimization objective & algorithm
- Pros
  - Eigenvector method
  - No tuning of parameters
  - No local optima
- Cons
  - Only based on covariance (2nd order statistics)
  - Limited to linear projections
- Next: Nonlinear dimensionality reduction

Adapted from CMSC422-0101

Autoencoder

Autoencoders

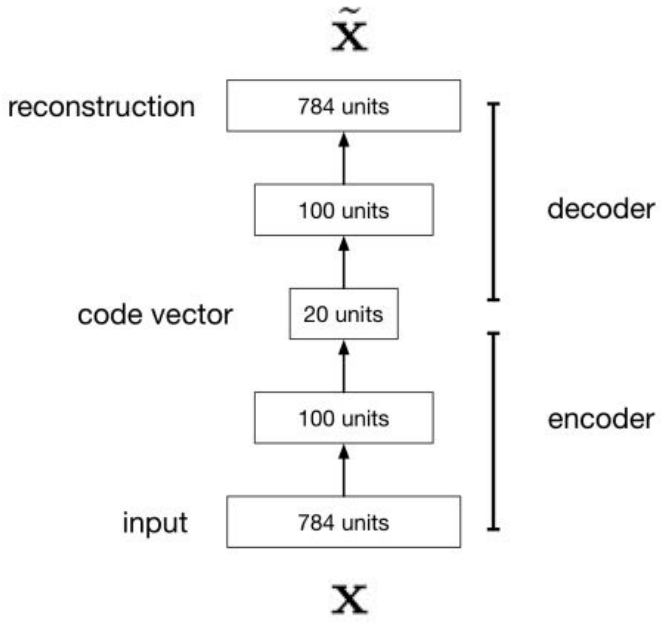
- A neural network to find latent space representation of the original data
  - Unsupervised method (with no labeled training data)
- To make this non-trivial, we add a bottleneck layer whose dimension is much smaller than the input



Adapted from slides of Roger Grosse and Jimmy Ba

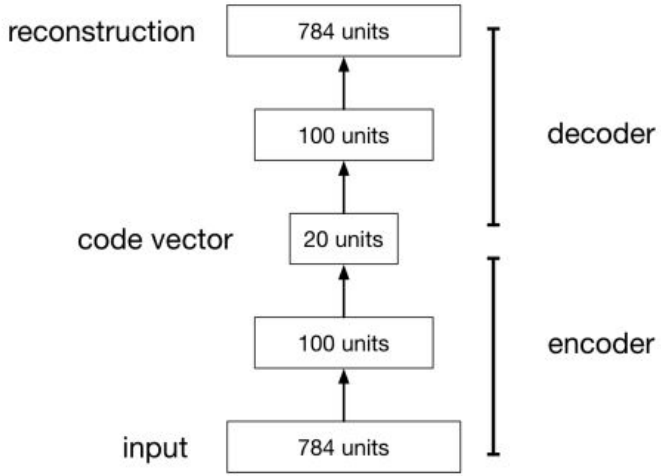
Autoencoders: approach

- **Goal**: Find the latent space representation that *best* represent the important information in the original data
  - Recall PCA: maximize the variance
- **Approach**: bottleneck layer
  - Forces the network to create a compressed representation of the input data (dimensionality reduction)
  - Forces the network to remove redundancy and noise
- **Objective**: reconstruction error $\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$
  - Can add regularization term to avoid overfitting (identity mapping)



Why autoencoders?

- Map high-dimensional data to 2D for visualization
- Compression
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
- Learn a semantically meaningful representation where you can, e.g., interpolate between different images.



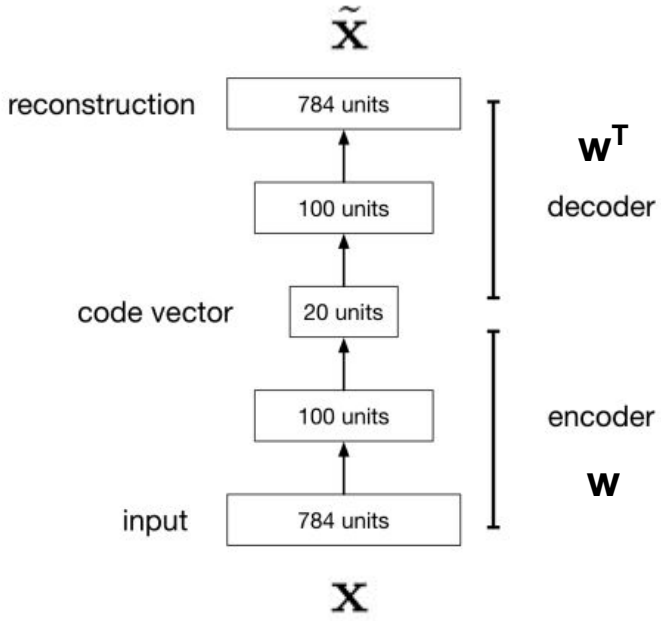
Adapted from slides of Roger Grosse and Jimmy Ba

Autoencoders: connection to PCA

Loss function: $\mathcal{L}(x, \widetilde{x})$ (reconstruction error)

Mean square error (MSE):

$$\frac{1}{D}\sum_{i}\|x_i - \widetilde{x}_i\|^2$$



What if we remove non-linearity in NN?

When we remove the non-linearity term in neural network (activation function), (and force encoder and decoder to have the same weights) autoencoder is equivalent to PCA :

$$\widehat{w} = \arg\min_{w} \mathbb{E}[\|x - \boxed{w^T w x}\|^2]$$

$$\boxed{\widetilde{x}}$$

PCA

$\mathbf{Z}_{(rxN)} = \mathbf{W}^T_{(rxD)}\,\mathbf{X}_{(DxN)}$

A different objective function:

min || X - WZ ||^2

= min || X - WW^{T}X ||^2

Autoencoders: connection to PCA

Autoencoders learn to project the data, not onto a subspace, but onto a nonlinear manifold

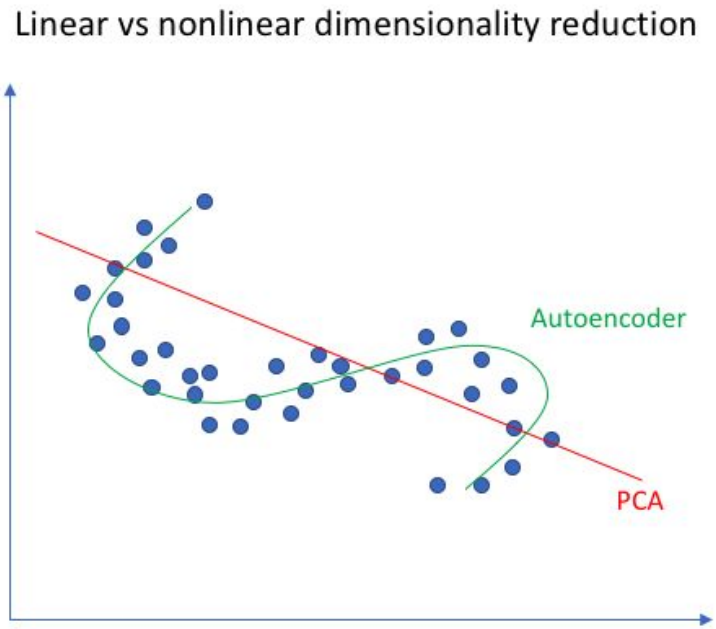


Image source: https://www.jeremyjordan.me/autoencoders/

Adapted from slides of Roger Grosse and Jimmy Ba

Conclusion

- Unsupervised learning
  - No labels
  - Aim to discover underlying structure of data
- Clustering
  - K-Means
- PCA
  - Linear dimensionality reduction
  - Maximize variance
- Autoencoders
  - Nonlinear dimensionality reduction
  - Minimize reconstruction error