CSC 580 Principles of Machine Learning

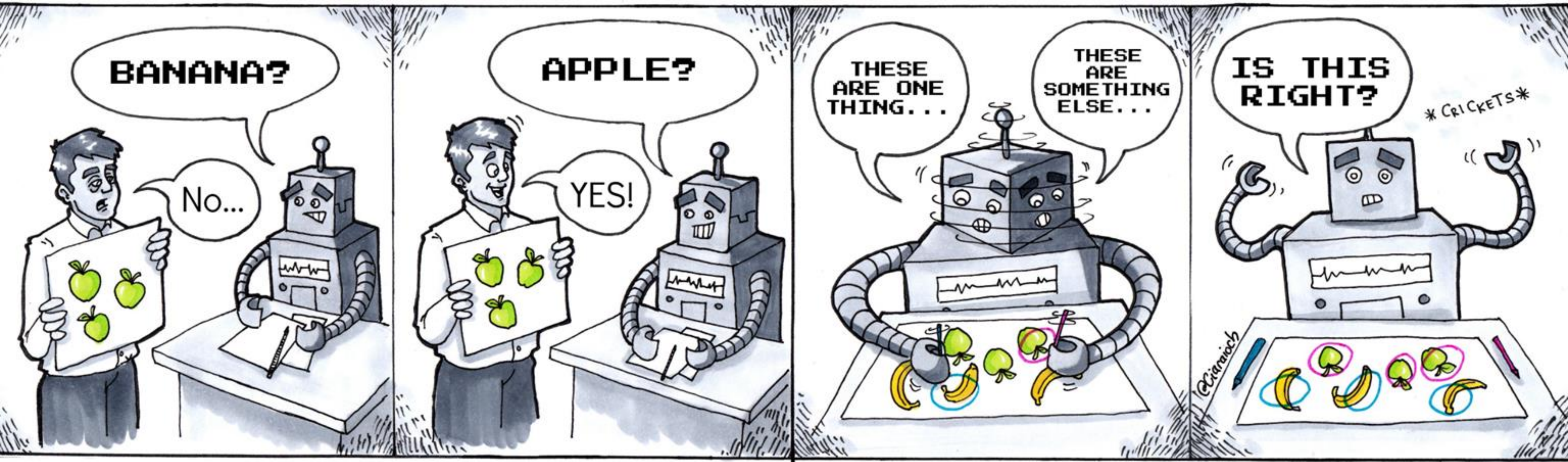
09 Unsupervised learning

Chicheng Zhang

Department of Computer Science

THE UNIVERSITY OF ARIZONA

*slides credit: built upon CSC 580 Fall 2021 lecture slides by Kwang-Sung Jun



Supervised Learning

Unsupervised Learning

Task 1 : Group These Set of Document into 3 Groups based on meaning

Doc1 : Health , Medicine, Doctor

Doc 2 : Machine Learning, Computer

Doc 3 : Environment, Planet

Doc 4 : Pollution, Climate Crisis

Doc 5 : Covid, Health , Doctor

Task 1 : Group These Set of Document into 3 Groups.

Doc1 : Health , Medicine, Doctor

Doc 2 : Machine Learning, Computer

Doc 3 : Environment, Planet

Doc 4 : Pollution, Climate Crisis

Doc 5 : Covid, Health , Doctor

Task 1 : Group These Set of Document into 3 Groups.

Doc1 : Health , Medicine, Doctor

Doc 5 : Covid, Health , Doctor

Doc 3 : Environment, Planet

Doc 4 : Pollution, Climate Crisis

Doc 2 : Machine Learning, Computer

Task 2: Topic modeling

- Provides a summary of a corpus.
- n tweets containing the keyword "bullying", "bullied", etc.
- Extracts k topics: each topic is a list of words with importance weights.
    - A set of words that co-occurs frequently throughout.

Figure 4: Selected topics discovered by latent Dirichlet allocation.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, Amy Bellmore, "Learning from Bullying Traces in Social Media"

What is unsupervised learning?

- Uncovering structures in unlabeled data
- What can we expect to learn?
  - **Clustering**: obtain partition of the data that are well-separated.
    - can be viewed as a preliminary classification without predefined class labels.
  - **Components**: extract common components that compose data points.
    - e.g., topic modeling given a set of articles: each article talks about a few topics => extract the set of topics that appears frequently.
- Usage
  - As a summary of the data
    - **Exploratory data analysis**: what are the **patterns** we can get even without labels?
  - Often used as a ‘preprocessing techniques’
    - e.g., extract useful **features** using “gaussian mixture model” (will be covered later)

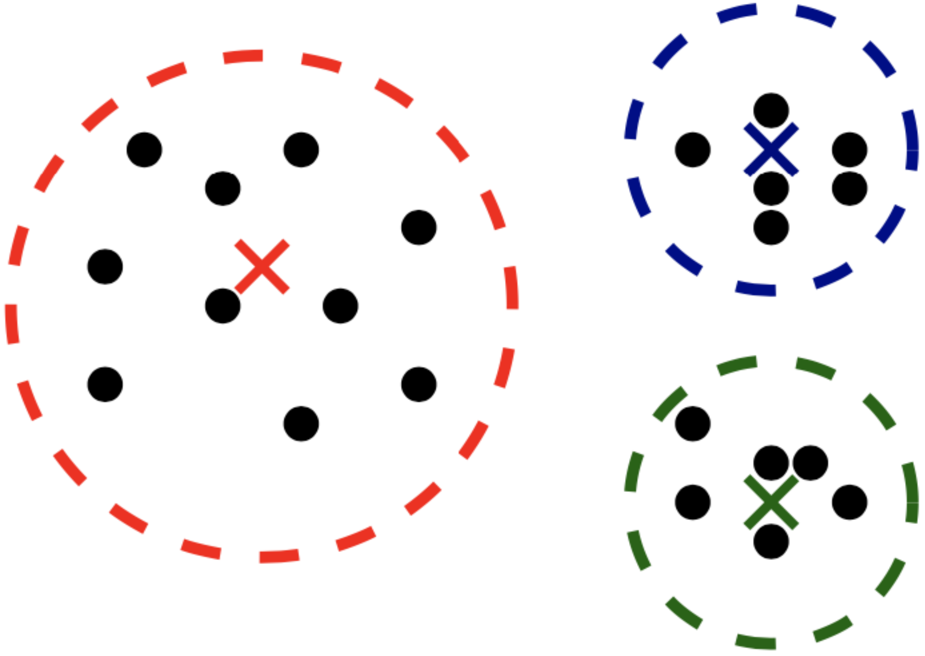
Clustering

Clustering

- Input: k: the number of clusters (hyperparameter)

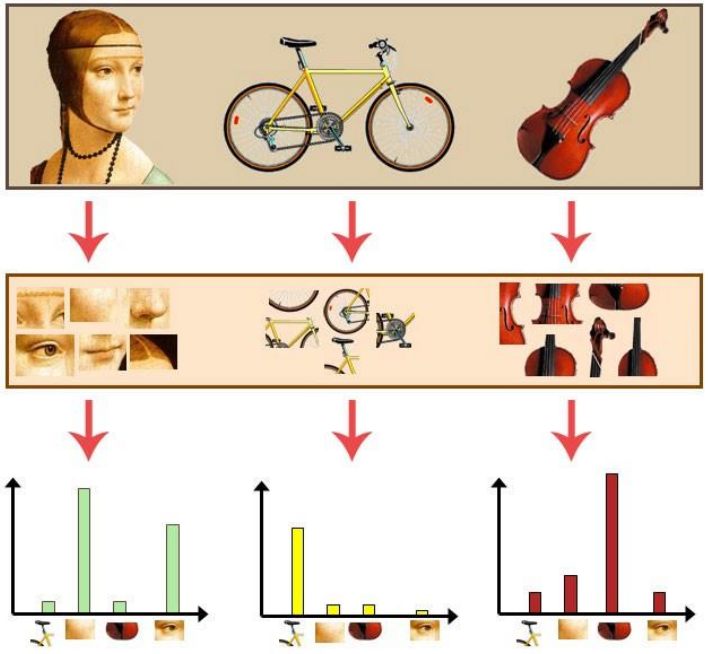
$$S = \{x_1, \dots, x_n\}$$

- Output
  - partition $\{G_i\}_{i=1}^{k}$ s.t. $S = \cup_i G_i$ (disjoint union).
  - often, we also obtain 'centroids'



- Q: what would be a reasonable definition of centroids?

Application: Clustering for feature extraction



- Feature extraction: **histogram features (bag of visual words)**
- A set of images: $S = \{x_1, \dots, x_n\}$
- Cut up each $x_i \in \mathbb{R}^d$ into different parts $x_i^{(1)}, \dots, x_i^{(m)} \in \mathbb{R}^p$
  - e.g., small (overlapping) patches of an image
- Notation: $[n] \coloneqq \{1, \dots, n\}$
- Pool all the patches together: $P \coloneqq \left\{x_i^{(j)}\right\}_{i\in[n], j\in[m]}$
- Run clustering on P with #clusters=$k \Rightarrow$ for each $x_i^{(j)}$, we have a cluster assignment $A\left(x_i^{(j)}\right) \in [k]$
- Generate the feature vector of x_i as the histogram of $\left\{A\left(x_i^{(j)}\right)\right\}_{j\in[m]}$
  - i.e., $z = (z_1, \dots, z_k)$ where z_ℓ is the count of the cluster ℓ

https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb

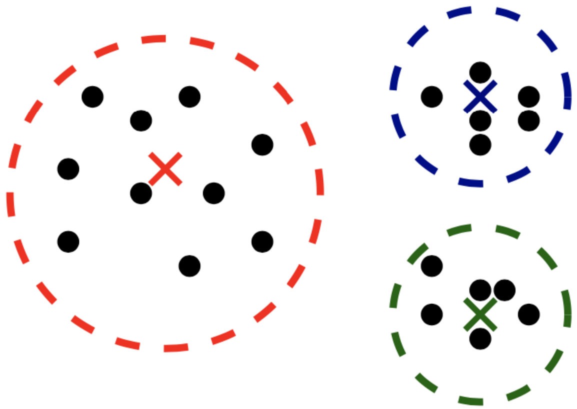
k-means clustering

- Idea: to partition the data, it would be great if someone gives us k reasonable centroids $c_1, \dots, c_k$, since then we can partition the data with them.

$$A(x) = \arg\min_{j\in[k]} \left\| x - c_j \right\|_2$$

- But we don't have those centroids => Let's find them with an optimization formulation.

$$\underset{c_1,\dots,c_k}{\text{minimize}}\, f(c_1, \dots, c_k), \text{ where } f(c_1, \dots, c_k) = \sum_{i=1}^{n} \min_{j\in[k]} \left\| x - c_j \right\|_2^2$$



Special case: k=1

- $\min\limits_{c_1,\ldots,c_k} \sum_{i=1}^{n} \min\limits_{j\in[k]} \left\|x_i - c_j\right\|_2^2$ => $\min\limits_{c} \sum_{i=1}^{n} \|x_i - c\|_2^2$

- Let $F(c) = \sum_{i=1}^{n} \|x_i - c\|_2^2$ convex; minimizer c^* satisfies that $\nabla F(c^*) = 0$

  => $\sum_{i=1}^{n}(x_i - c^*) = 0$ => $c^* = \frac{1}{n}\sum_{i=1}^{n} x_i$

For $k \geq 2$

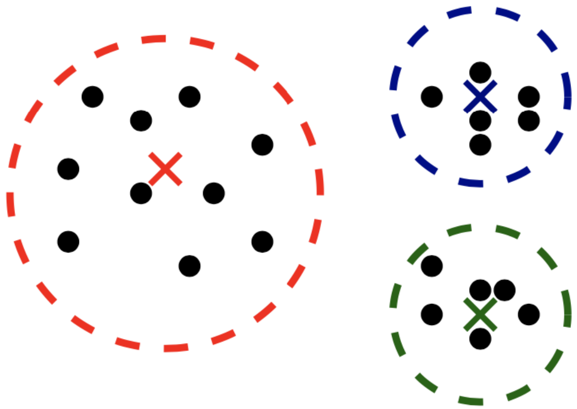
- $\underset{c_1,\dots,c_k}{\text{minimize}} f(c_1, \dots, c_k)$, where $f(c_1, \dots, c_k) = \sum_{i=1}^{n} \min_{j \in [k]} \left\| x - c_j \right\|_2^2$ => NP-hard even when $d = 2$

- **<u>Lloyd's algorithm</u>**: solve it approximately (heuristic)

(but people just say it is k-means clustering algorithm)

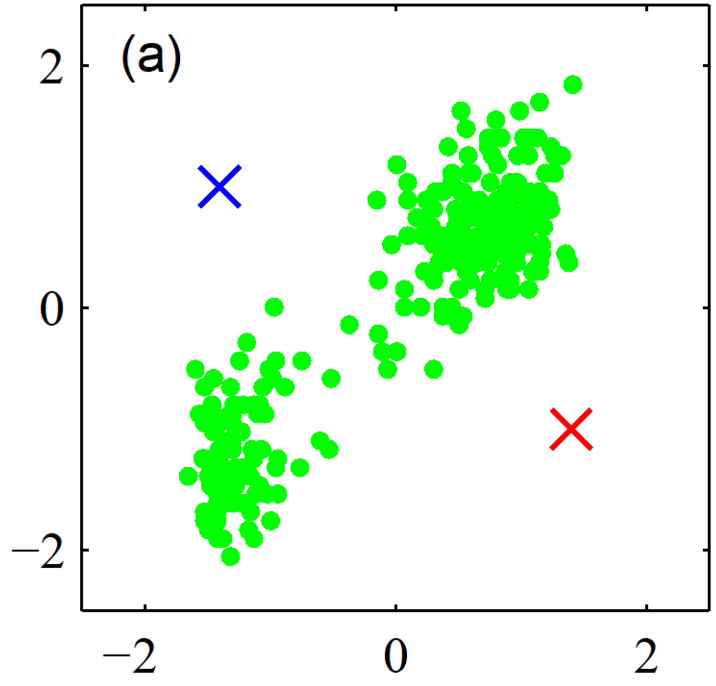
- Observation: The chicken-and-egg problem.
  - Cluster center location depends on the cluster assignment
  - Cluster assignment depends on cluster location

- Very common heuristic (that may or may not be the best thing to do)



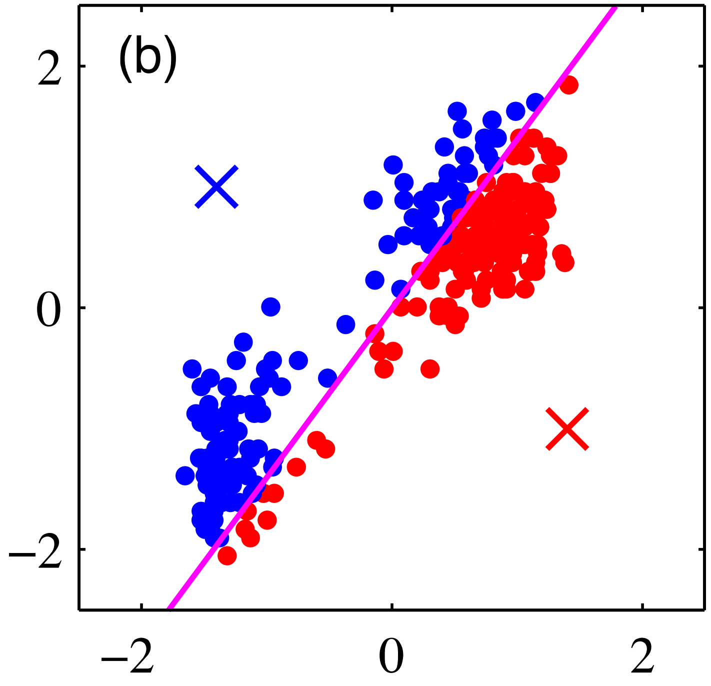
Andrea Vattani, "the hardness of k-means clustering in the plane"

Initialization

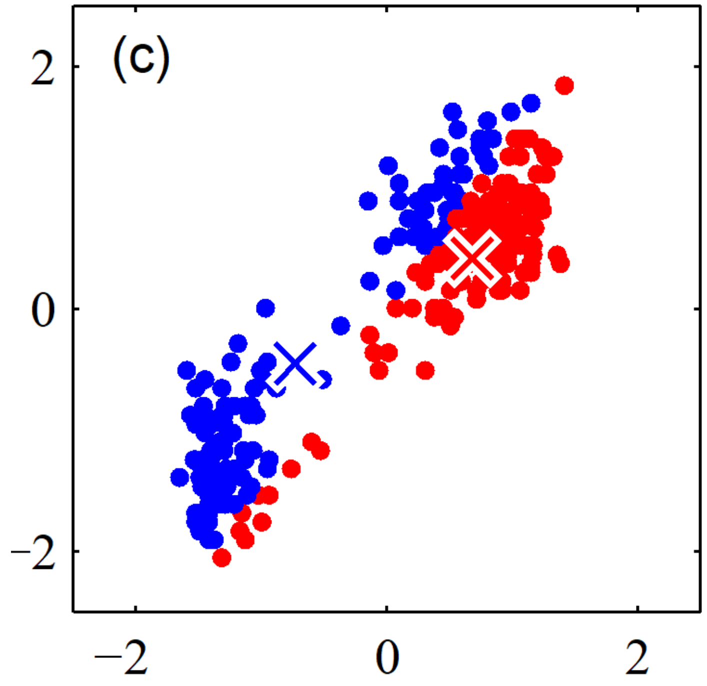


Arbitrary/random initialization of c_1 and c_2

Iteration 1

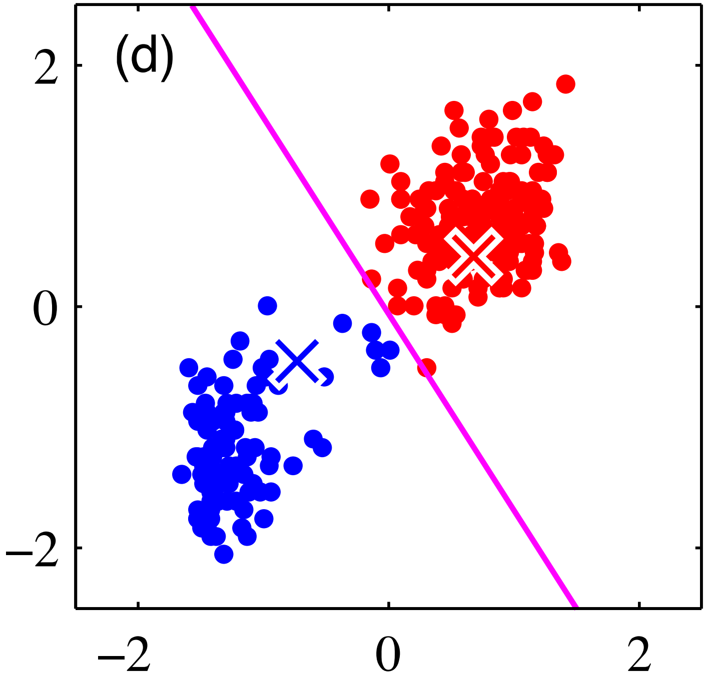


(A) update the cluster assignments.

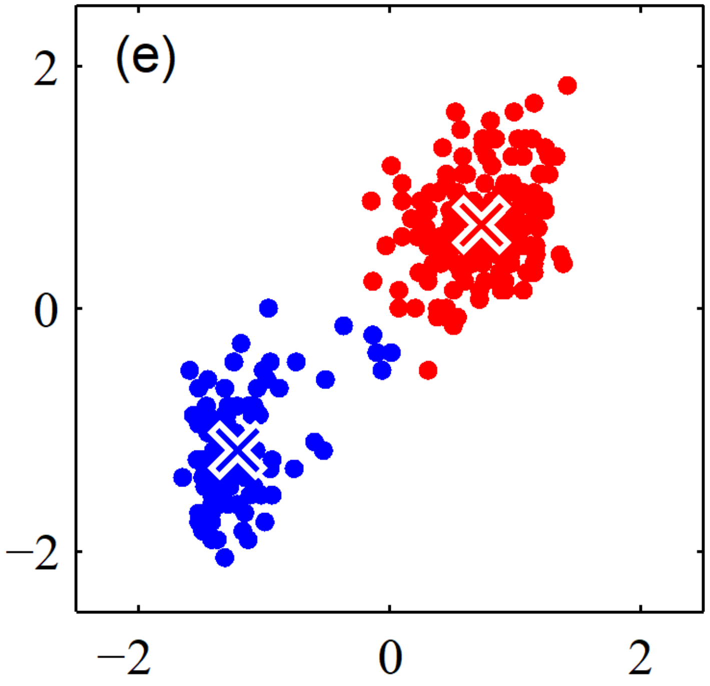


(B) Update the centroids $\{c_j\}$

Iteration 2

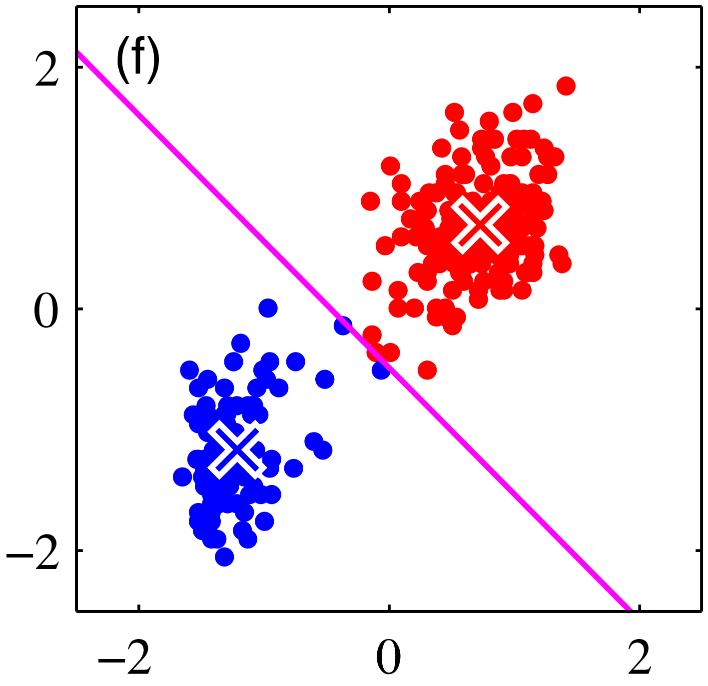


(A) update the cluster assignments.

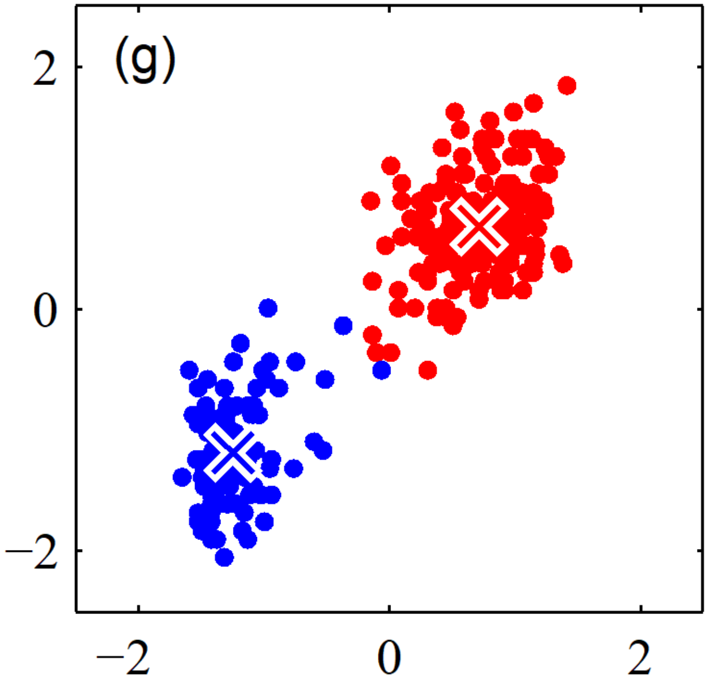


(B) Update the centroids $\{c_j\}$

Iteration 3

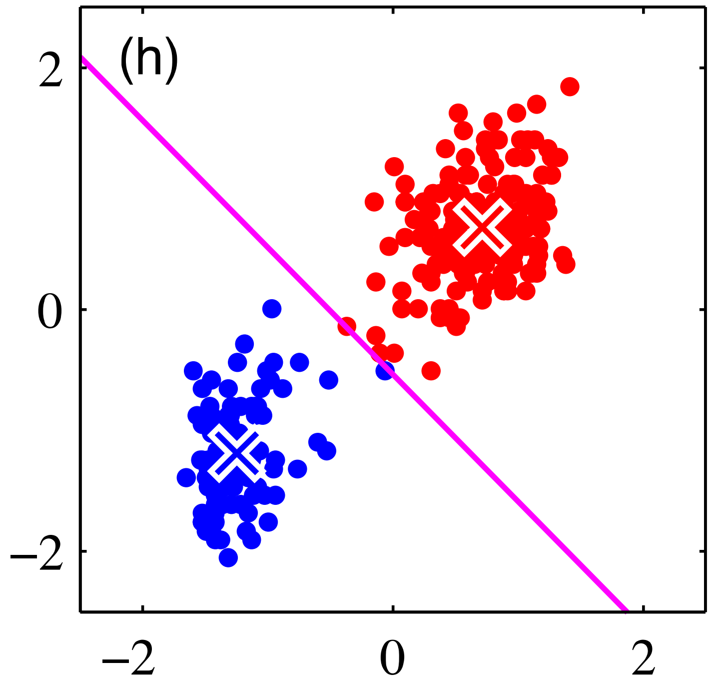


(A) update the cluster assignments.

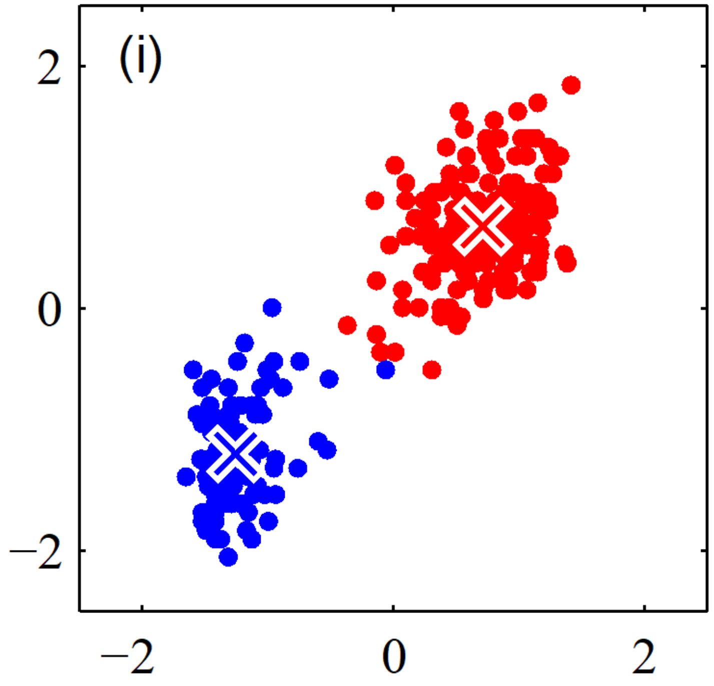


(B) Update the centroids $\{c_j\}$

Iteration 4



(A) update the cluster assignments.



(B) Update the centroids $\{c_j\}$

Next lecture (10/10)

- Dimensionality reduction; Principal component analysis (PCA)

- Probabilistic machine learning; naïve Bayes algorithm

- Assigned reading: CIML Chap. 15

Lloyd's algorithm for k-means clustering

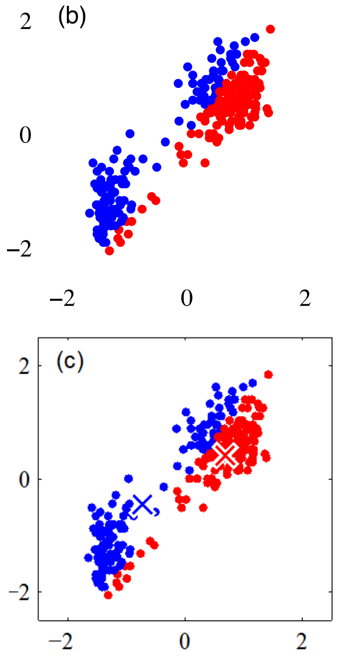
Input: k: num. of clusters, $S = \{x_1, \dots, x_n\}$

[Initialize] Pick $c_1, \dots, c_k$ as randomly selected points from S (see next slides for alternatives)

For t=1,2,…,max_iter

- ***[Assignments]*** $\forall x \in S, \quad a_t(x) = \arg\min_{j \in [k]} \left\| x - c_j \right\|_2^2$
- If t $\neq 1$ AND $a_t(x) = a_{t-1}(x), \forall x \in S$
  - break
- ***[Centroids]*** $\forall j \in [k], \quad c_j \leftarrow \text{average}(\{x \in S: a_t(x) = j\})$

Output: $c_1, \dots, c_k$ and $\{a_t(x_i)\}_{i \in [n]}$



Lloyd's algorithm: cost minimization perspective

- Key idea: solving the optimization problem by *reformulation* and *alternating minimization*:

- Reformulation: denote by $\vec{c} := (c_1, \dots, c_k)$, $\vec{z} := (z_1, \dots, z_n)$;

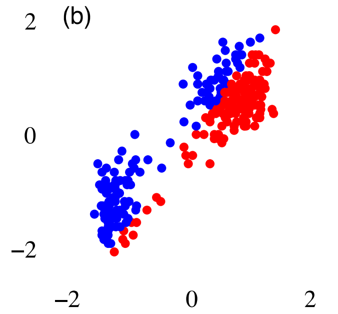
  $f(\vec{c}) = \min_{\vec{z}} g(\vec{c}, \vec{z})$, where $g(\vec{c}, \vec{z}) = \sum_{i=1}^{n} \left\| x_i - c_{z_i} \right\|_2^2$

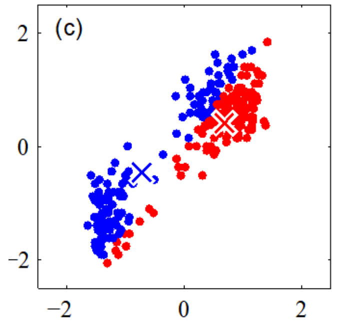
  suffices to solve

$$\min_{\vec{c},\vec{z}} g(\vec{c}, \vec{z})$$

- For $t = 1,2, \dots, T$:
  - Update the cluster assignments: $\vec{z}_t \leftarrow \mathrm{argmin}_{\vec{z}}\, g(\vec{c}_{t-1}, \vec{z})$
  - Update the centroids: $\vec{c}_t \leftarrow \mathrm{argmin}_{\vec{c}}\, g(\vec{c}, \vec{z}_t)$

- Observation: objective function $g(\vec{c}_t, \vec{z}_t)$ decreases *monotonically in t*





Issue 1: Unreliable solution

- You usually get suboptimal solutions
- You usually get different solutions every time you run.
- <u>**Standard practice**</u>: Run it 50 times and take the one that achieves the smallest objective function
  - Recall: $\underset{c_1,\ldots,c_k}{\text{minimize}}\, f(c_1, \ldots, c_k)$, where $f(c_1, \ldots, c_k) = \sum_{i=1}^{n} \min_{j\in[k]} \left\| x - c_j \right\|_2^2$
- Or, change the initialization (next slide)
  - Idea: ensure that we pick a widespread $c_1, \ldots, c_k$

Two alternative initializations.

- **Furthest-first traversal** ⇒ Sequentially choose c_j that are the farthest from the previously-chosen.
  - Pick $c_1 \in \{x_1, \dots, x_n\}$ arbitrarily (or randomly)
  - For $j = 2, \dots, k$
    - Pick $c_j \in \mathbb{R}^d$ as a point in $\{x_1, \dots, x_n\}$ that maximizes the squared distances to $c_1, \dots, c_{j-1}$.

$$c_j = \arg\max_{i \in [n]} \min_{j'=1,\dots,j-1} \left\| x_i - c_{j'} \right\|_2^2$$

- ***k*-means++ (Arthur and Vassilvitskii, 2007)**
  - Pick $c_1 \in \{x_1, \dots, x_n\}$ uniformly at random
  - For $j = 2, \dots, k$
    - Define a distribution $\forall i \in [n],\ \mathbb{P}\left(c_j = x_i\right) \propto \min_{j'=1,\dots,j-1} \|x_i - c_{j'}\|_2^2$
    - Draw c_j from the distribution above.

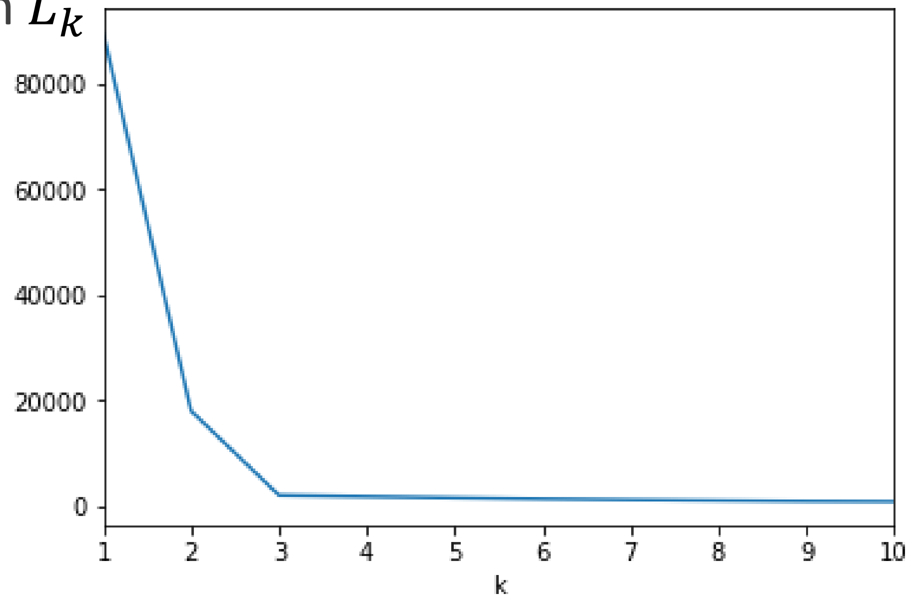
More likely to choose x_i that is farthest from already-chosen centroids.

=> has a mathematical guarantee that it will be better than an arbitrary starting point!

Issue 2: Choosing k

- $\hat{L}_k = f(c_1, \dots, c_k)$ for $c_1, \dots, c_k$ obtained by any k-means clustering algorithm

Objective function $\hat{L}_k$



- Elbow method: see where you get saturation.
- Akaike information criterion (AIC): $\operatorname{argmin}_k \left(\hat{L}_k + 2kd\right)$
- Bayesian information criterion (BIC): $\operatorname{argmin}_k \left(\hat{L}_k + kd \cdot \log n\right)$

https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb

Kernelizing Lloyd’s algorithm

How to perform clustering with feature transformations $\phi: \mathcal{X} \rightarrow \mathbb{R}^D$?

<u>Input</u>: k: num. of clusters, $S = \{x_1, \dots, x_n\}$, kernel function K with feature map ϕ

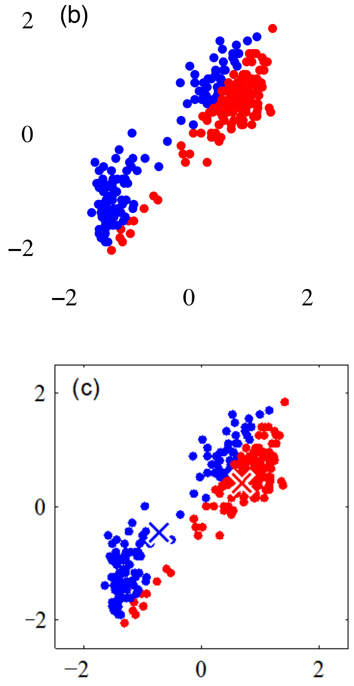
Idea: perform clustering over $\tilde{S} = \{\phi(x_1), \dots, \phi(x_n)\}$ without explicitly evaluating ϕ

[Initialize] Pick $c_1, \dots, c_k$ as randomly selected points from $\tilde{S}$

For t=1,2,…,max_iter

- ***[Assignments]*** $\forall x \in S, \quad a_t(x) = \arg\min_{j \in [k]} \left\| \phi(x) - c_j \right\|_2^2$
- If $t \neq 1 \quad \text{AND} \quad a_t(x) = a_{t-1}(x), \forall x \in S$
  - break
- ***[Centroids]*** $\forall j \in [k], \quad c_j \leftarrow \text{average}(\{\phi(x) : x \in S, a_t(x) = j\})$

<u>Output</u>: $c_1, \dots, c_k$ and $\{a_t(x_i)\}_{i \in [n]}$



Kernelizing Lloyd's algorithm (cont'd)

- How to calculate $\left\|\phi(x) - c_j\right\|_2^2$ without explicitly evaluating ϕ?

- Key observation: c_j always takes the form $c_j = \frac{1}{|S|}\sum_{i\in S}\phi(x_i)$ for some S , and therefore has the form $c_j = \sum_{i=1}^{n}\alpha_i\phi(x_i)$

- Therefore,

$$\begin{aligned}\left\|\phi(x) - c_j\right\|_2^2 &= \langle\phi(x), \phi(x)\rangle - 2\left\langle\phi(x), \sum_{i=1}^{n}\alpha_i\phi(x_i)\right\rangle + \left\langle\sum_{i=1}^{n}\alpha_i\phi(x_i), \sum_{i=1}^{n}\alpha_i\phi(x_i)\right\rangle \\ &= K(x,x) - 2\sum_{i=1}^{n}K(x,x_i) + \sum_i\sum_j\alpha_i\alpha_j K(x_i,x_j)\end{aligned}$$

- Efficiently computable: only requires evaluating K now

Clustering as cost minimization: additional remarks

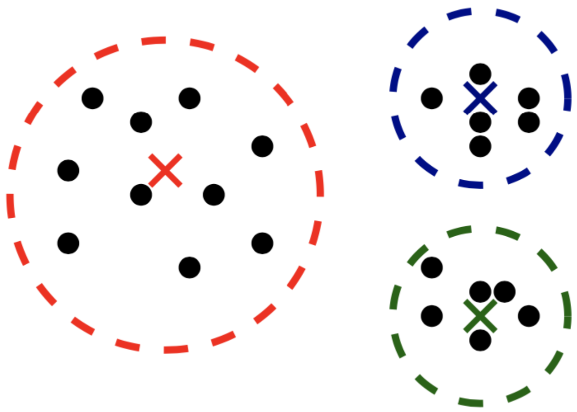
- k-means objective function is not the only one used in practice

$$f(c_1, \dots, c_k) = \sum_{i=1}^{n} \min_{j \in [k]} \left\| x - c_j \right\|_2^2$$

- Alternative popular cost functions:

k-median: $f(c_1, \dots, c_k) = \sum_{i=1}^{n} \min_{j \in [k]} \left\| x - c_j \right\|_2$

k-center: $f(c_1, \dots, c_k) = \max_{i} \min_{j \in [k]} \left\| x - c_j \right\|_2$

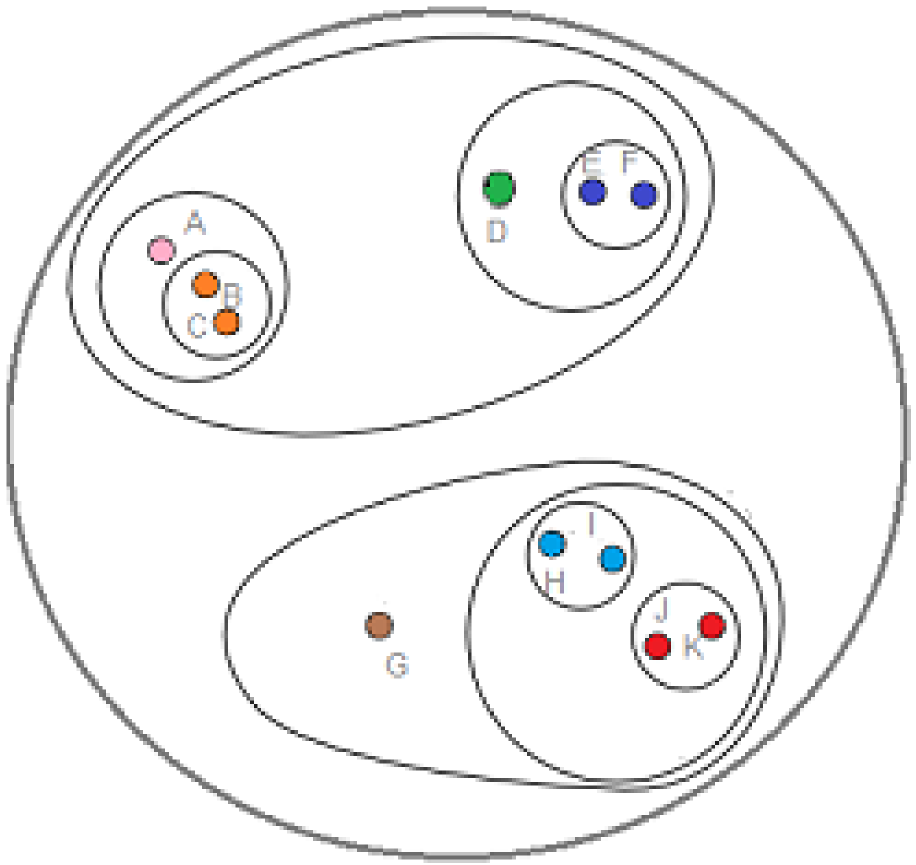


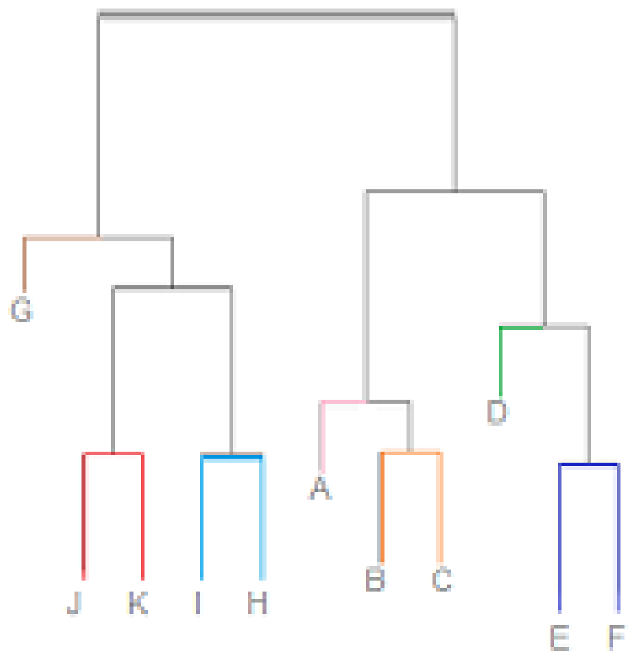
- Furthermore, we don't have to restrict to using the ℓ_2 metric

Hierarchical clustering

Hierarchical clustering – getting rid of tuning k

- Idea: produce a tree structure over objects
- Can prune the tree appropriately to fit application needs (e.g. cluster radius / size requirements)

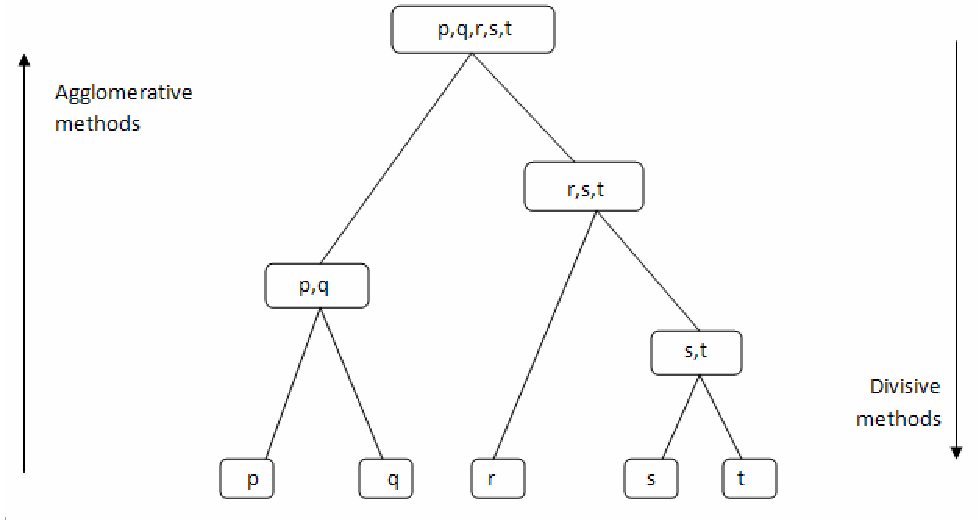




Hierarchical clustering

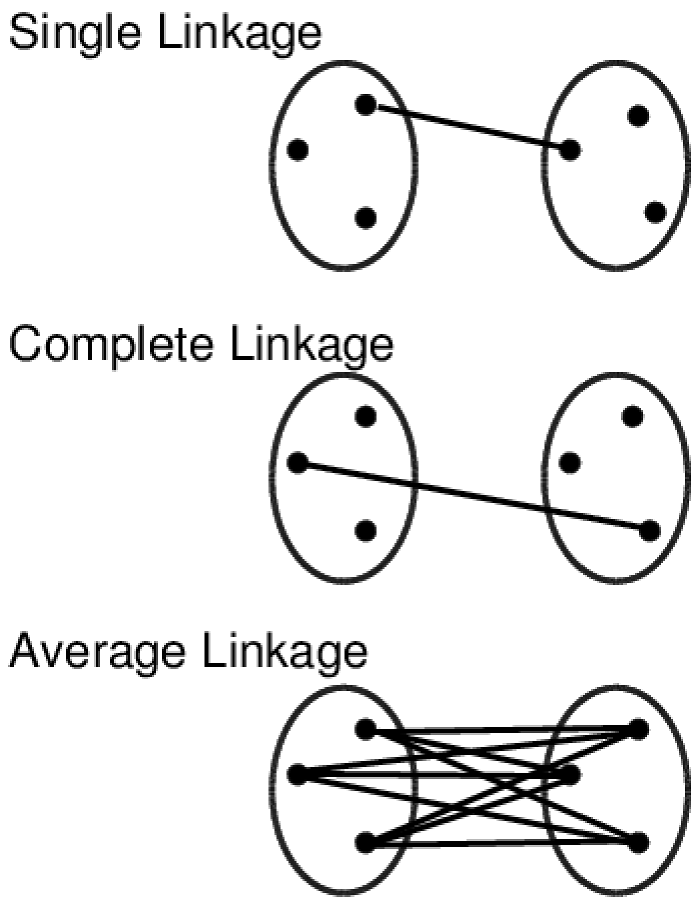
- Method 1: Top-down (divisive)
  - k-means clustering with k=2
  - Do this recursively on each resulting cluster (no more recursion when there is only one point in a cluster)
  - You now have a binary tree.

- Method 2: bottom-up (agglomerative, more popular)
  - Start with every point x_i being a singleton cluster
  - Repeatedly pick a pair of clusters with the smallest 'distance'
  - How do we define a distance between two clusters?



Agglomerative clustering: Distance between two clusters

- Single linkage
  - $\text{dist}(C, C') = \min_{x \in C, x' \in C'} \|x - x'\|_2$
- Complete linkage
  - $\text{dist}(C, C') = \max_{x \in C, x' \in C'} \|x - x'\|_2$
- Average linkage
  - $\text{dist}(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{x \in C} \sum_{x' \in C} \|x - x'\|_2$

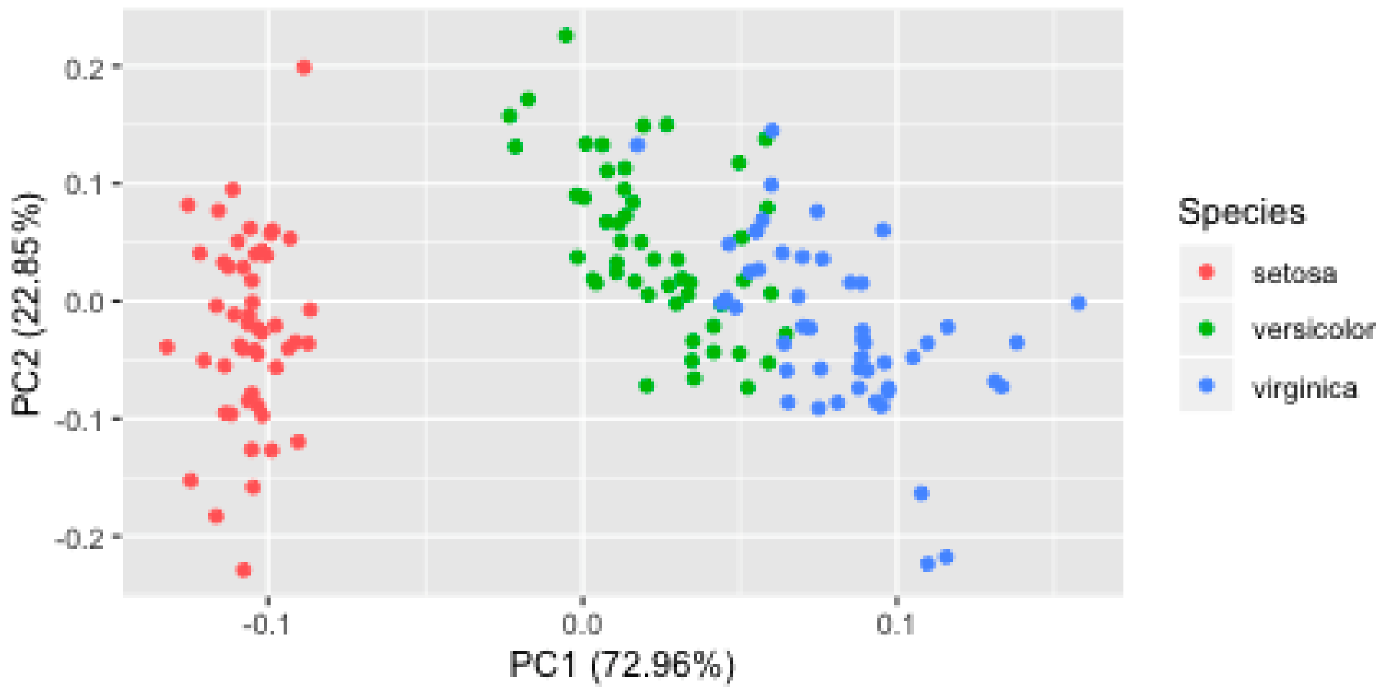


https://girke.bioinformatics.ucr.edu/GEN242/tutorials/rclustering/rclustering/

Dimensionality Reduction
and Principal Component Analysis (PCA)

Dimensionality reduction: motivation

- Data compression: Identifies important components that can reconstruct data points
- Identify informative feature transformations
- Visualization & visual analytics: high-dim data -> 2d => easy to plot



Iris flower dataset (4 features)

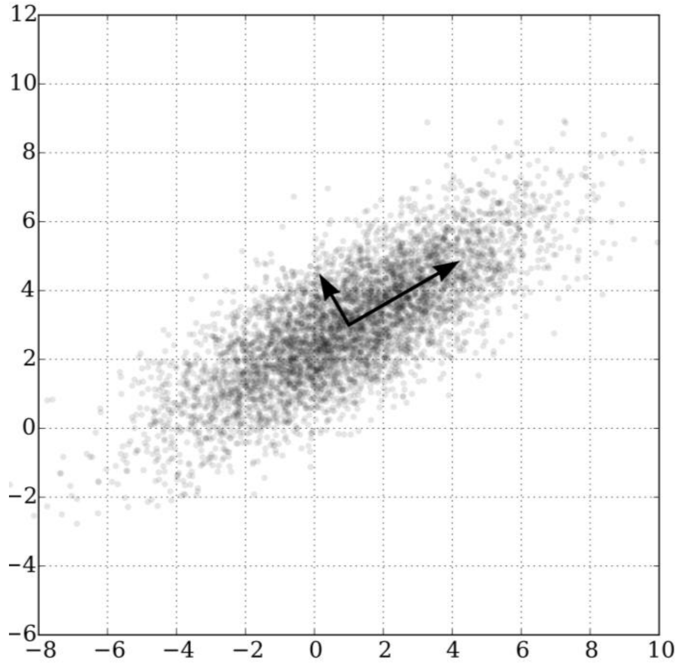
https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html

PCA: Introduction

- Task:
  - Given: raw feature vectors $x_1, \dots, x_n \in \mathbb{R}^d$, target dimension k
  - Output: a k-dimensional **subspace** represented by an *orthonormal* basis $q_1, \dots, q_k \in \mathbb{R}^d$ that the projections of datapoints with it would maximally preserve the ``spread''.

- Application: dimensionality reduction

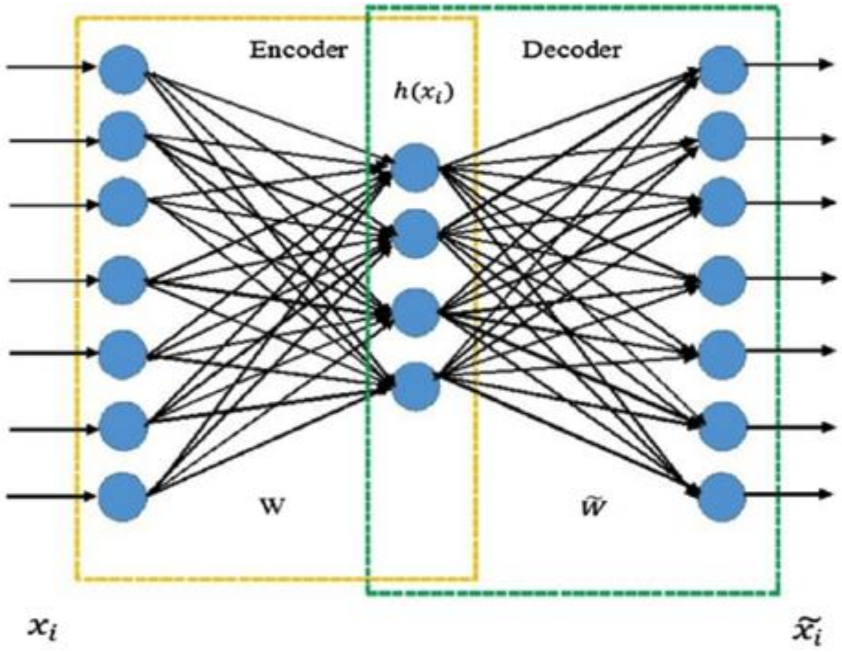
- Closely related to projections



if k=1, which basis should we choose?

Principal components: usage

- Compressing the data:
  - Let $Q = \begin{pmatrix} - \; q_1 \; - \\ \cdots \\ - \; q_k \; - \end{pmatrix} \in \mathbb{R}^{d \times k}$
  - Every $x_i \in \mathbb{R}^d$ gets mapped to $z_i = Qx_i = \begin{pmatrix} q_1^\top x_i \\ \dots \\ q_k^\top x_i \end{pmatrix} \in \mathbb{R}^k$

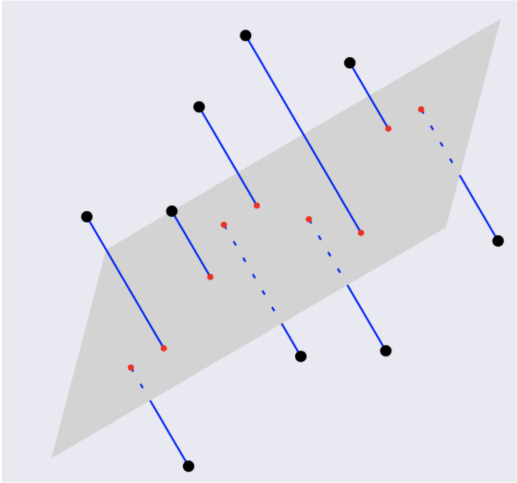


- Resconstructing the data
  - Given z_i, reconstruct x_i with $\widetilde{x_i} = \begin{pmatrix} | & \cdots & | \\ q_1 & \dots & q_k \\ | & \cdots & | \end{pmatrix} z_i = Q^\top z_i$
  - Reconstruction error: $x_i - \widetilde{x_i} = x_i - Q^\top Q x_i$
  - If $k = d$, then perfect reconstruction ($\widetilde{x_i} = x_i$)

Projection

- Why reconstructing using $Q^\top z_i$?
- Given orthonormal $Q = \begin{pmatrix} - & q_1 & - \\ & \cdots & \\ - & q_k & - \end{pmatrix}$,

$$Q^\top Q x = \underbrace{\begin{pmatrix} | & \cdots & | \\ q_1 & \cdots & q_k \\ | & \cdots & | \end{pmatrix} \cdot \begin{pmatrix} - & q_1 & - \\ & \cdots & \\ - & q_k & - \end{pmatrix}}_{\text{projection matrix } \Pi = \sum_{i=1}^{k} q_i q_i^\top} x = \sum_i (q_i^\top x) q_i$$



is also the *projection* (nearest neighbor) of x to the linear span of $q_1, \ldots, q_k$

- **Projection Objective**: find a k-dimensional **projection matrix** Π s.t. the average residual squared error (reconstruction error) is minimized:

Q: why square the distance?

$$\frac{1}{n}\left(\sum_{i=1}^{n} \|x_i - \Pi x_i\|_2^2\right)$$

Projection when k=1

- Objective:

$$\arg\min_{q:\|q\|=1} \sum_{i=1}^{n} \left\| x_i - qq^\top x_i \right\|_2^2$$

$$\frac{1}{n}\left(\sum_{i=1}^{n} x_i^\top x_i - 2x_i^\top qq^\top x_i + x_i^\top qq^\top qq^\top x_i\right)$$

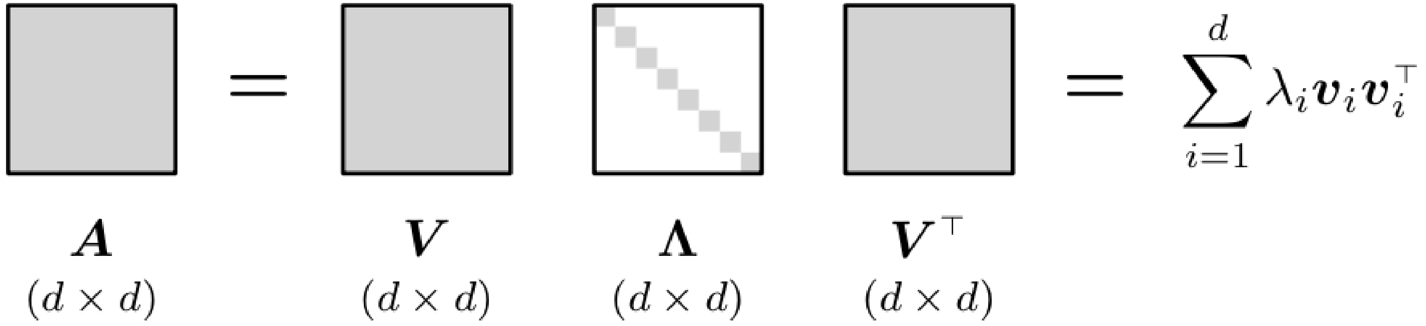
$$\propto -\frac{1}{n}\sum_i x_i^\top qq^\top x_i = q^\top \left(\frac{1}{n}X^\top X\right) q$$

where $X \in \mathbb{R}^{n\times d}$ is the design matrix

Thus, equivalent to $\arg\max_{q:\|q\|=1} q^\top \left(\frac{1}{n}X^\top X\right) q$

Eigendecomposition for real symmetric matrices

- Every **<u>Symmetric</u>** **<u>real</u>** matrix A is guaranteed to have eigen decomposition with real eigenvalues:



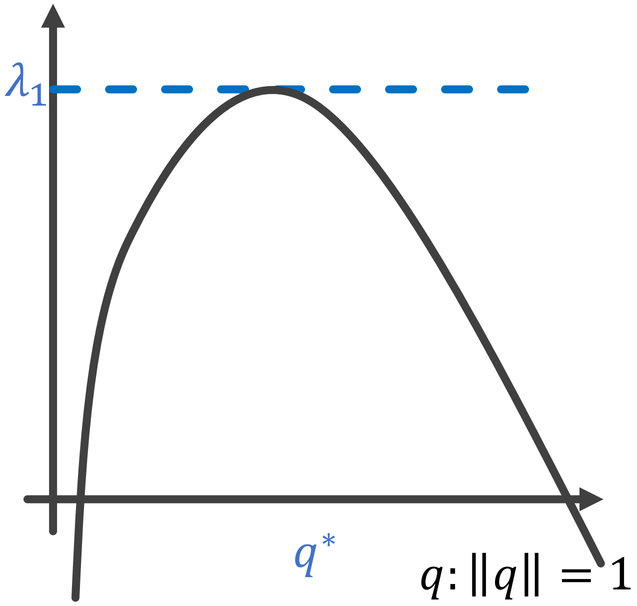
- Convention: $\lambda_1 \geq \cdots \geq \lambda_d$
- For positive semi-definite A, $\lambda_i \geq 0$ for all i
- Recall the definition: $Av_i = \lambda_i v_i \ \forall i \in [d]$
- Here, $V = \begin{pmatrix} | & \cdots & | \\ v_1 & \dots & v_d \\ | & \cdots & | \end{pmatrix}$ has orthonormal columns, i.e. $v_i^\top v_j = I(i = j)$

Variational characterization of the top eigenvector

- Claim: $\max_{q:\|q\|=1} q^\top A q$ has a maximizer $q^* = v_1$, with optimal objective λ_1
- Proof: recall $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\top$
  - (Optimal objective upper bound): For any unit vector q,

$$q^\top A q = \sum_{i=1}^{d} \lambda_i \left(v_i^\top q\right)^2 \le \lambda_1,$$

since $\left(a_i = \left(v_i^\top q\right)^2\right)_{i=1}^{d}$ is such that $\sum_{i=1}^{d} a_i = 1$ and $a_i \ge 0$ for all i



  - (The upper bound is achievable) $q^* = v_1$ satisfies that ${q^*}^\top A q^* = \lambda_1$

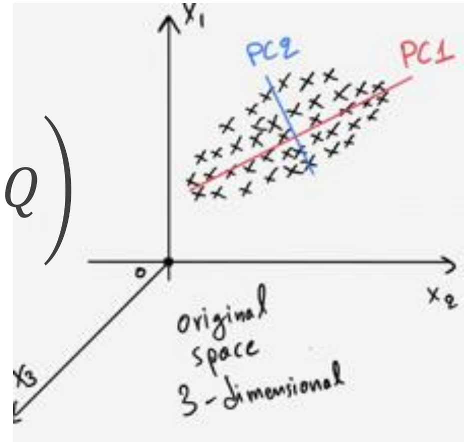
- Connection to the variance

$$A = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top \Rightarrow q^\top A q = \frac{1}{n}\sum_{i=1}^{n} q^\top x_i x_i^\top q = \frac{1}{n}\sum_{i=1}^{n} (q^\top x_i)^2 = \mathrm{E}_S[(q^\top x)^2]$$

equal to $\mathrm{var}_S[x^\top q] = \mathrm{E}_S[(x^\top q - \mathrm{E}_S[x^\top q])^2]$ if data is centralized

PCA with $k \geq 2$

$$\underset{Q \in \mathbb{R}^{d \times k}, Q^\top Q = I}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left\| x_i - Q Q^\top x_i \right\|_2^2 = \underset{Q \in \mathbb{R}^{d \times k}, Q^\top Q = I}{\operatorname{argmax}} \operatorname{tr}\left(Q^\top \left(\frac{1}{n} X^\top X \right) Q \right)$$



where for $B \in \mathbb{R}^{d \times d}$, $\operatorname{tr}(B) = \sum_{i=1}^{d} B_{ii}$

Interpretation: $Q^\top \left(\frac{1}{n} X^\top X \right) Q = \frac{1}{n} \sum_{i=1}^{n} (Q^\top x_i)(Q^\top x_i)^\top$ is the covariance matrix of $\{Q^\top x_i\}$'s

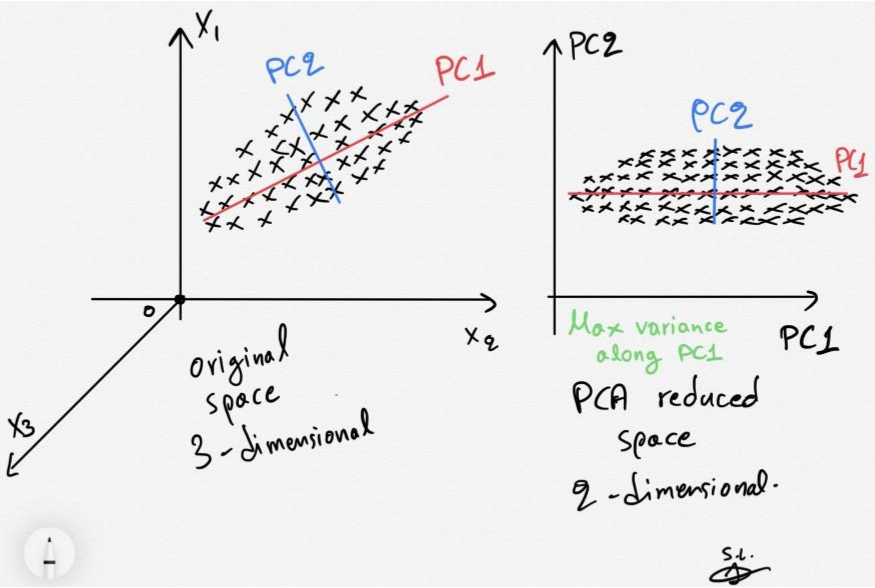
Fact: the optimal Q has the form $Q^* = \begin{pmatrix} | & \cdots & | \\ v_1 & \dots & v_k \\ | & \cdots & | \end{pmatrix}$, where $A = \frac{1}{n} X^\top X$ has eigendecomposition

$A = \sum_i^d \lambda_i v_i v_i^\top$

=> k-dimensional subspace with the maximum total variance = top-k eigenvectors!

https://en.wikipedia.org/wiki/Low-rank_approximation

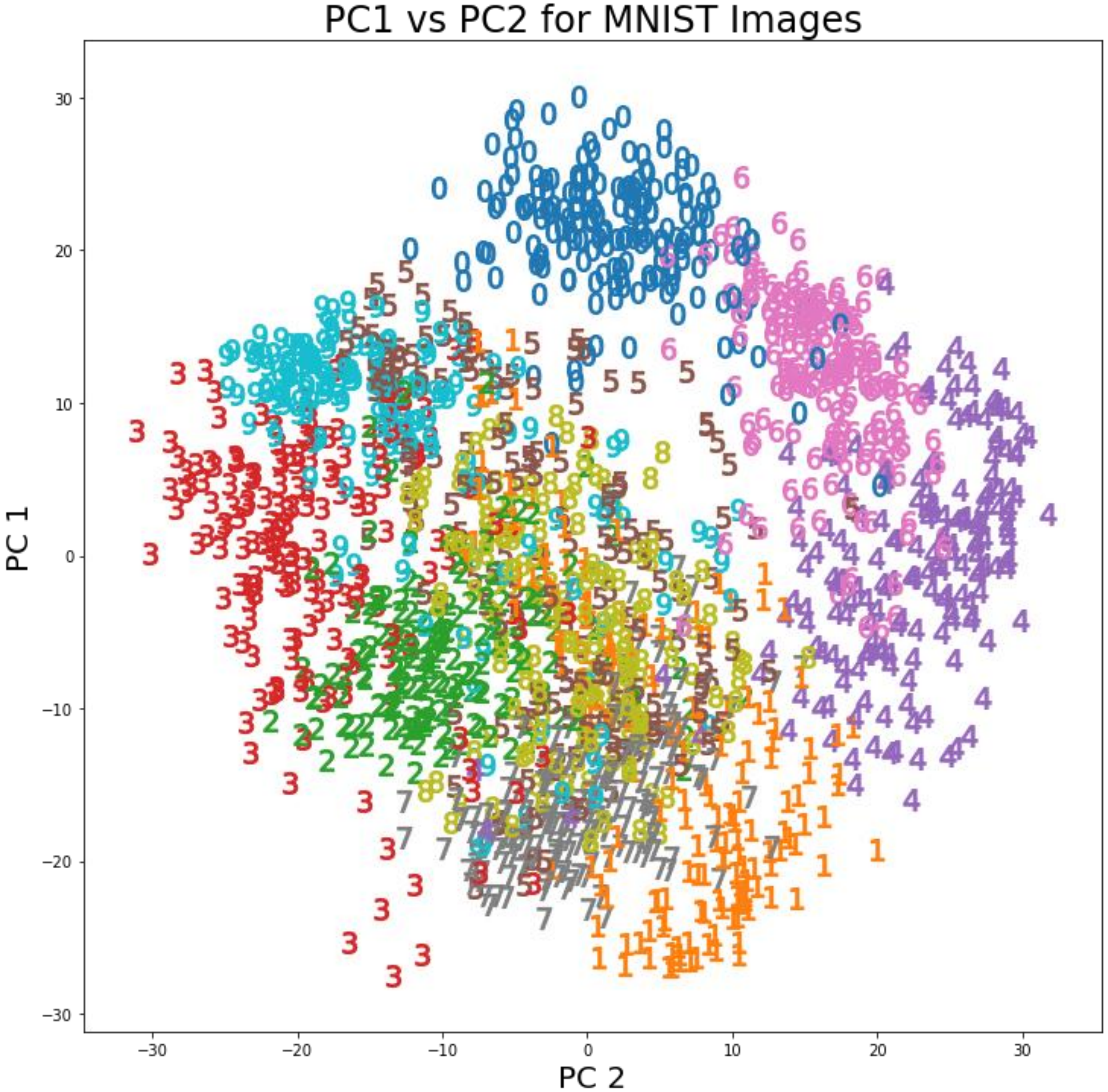
PCA pseudocode (with centering)



- Input: data matrix $X \in \mathbb{R}^{n \times d}$
- Preprocess: Let $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$. Compute $x_i' = x_i - \mu, \forall i \in [n]$
- Compute the top k eigenvectors $V = [v_1, \dots, v_k]$ of $\frac{1}{n}\sum_{i=1}^{n} x_i'(x_i')^\top$
- Feature map: $\phi(x) = \left(v_1^\top(x-\mu), \dots, v_k^\top(x-\mu)\right) \in \mathbb{R}^k$ (k-dimensional embedding)
- (thm) Decorrelating property (aka "whitening")
  - $\frac{1}{n}\sum_{i=1}^{n} \phi(x_i) = 0$
  - $\frac{1}{n}\sum_{i=1}^{n} \phi(x_i)\phi(x_i)^\top = \mathrm{diag}(\lambda_1, \dots, \lambda_k)$ λ_i is the eigen value (paired with v_i)
- (optional) Reconstruction (the actual projection): apply $\mu + V\phi(x) \in \mathbb{R}^d$
  - can be used as a ``denoising'' procedure.

https://towardsdatascience.com/pca-clearly-explained-how-when-why-to-use-it-and-feature-importance-a-guide-in-python-7c274582c37e

Example: MNIST dataset



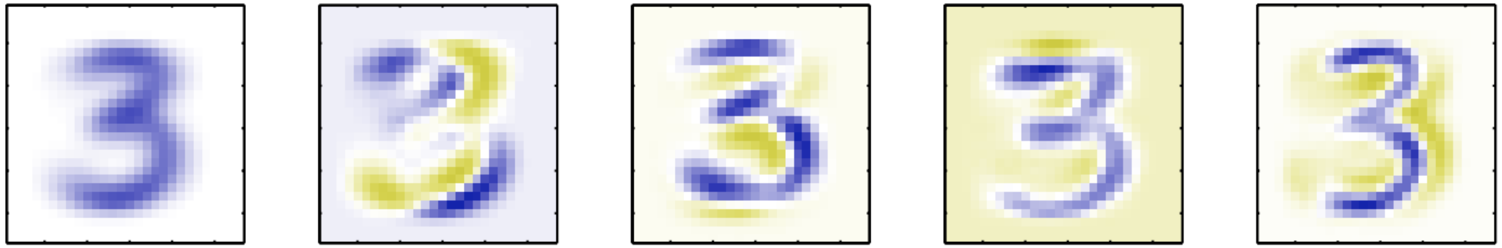
https://stats.stackexchange.com/questions/340175/why-is-t-sne-not-used-as-a-dimensionality-reduction-technique-for-clustering-or

Example: data compression

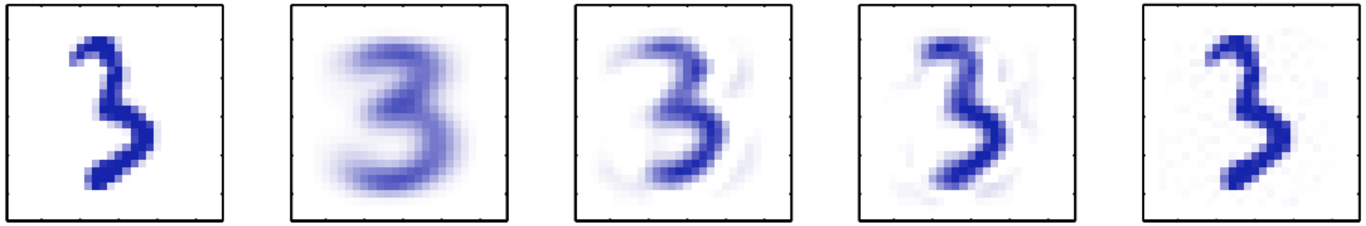
16 $\times$ 16 pixel images of handwritten 3s (as vectors in $\mathbb{R}^{256}$)

Mean $\boldsymbol{\mu}$ and eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$

Mean $\quad \lambda_1 = 3.4 \cdot 10^5 \quad \lambda_2 = 2.8 \cdot 10^5 \quad \lambda_3 = 2.4 \cdot 10^5 \quad \lambda_4 = 1.6 \cdot 10^5$



Reconstructions:

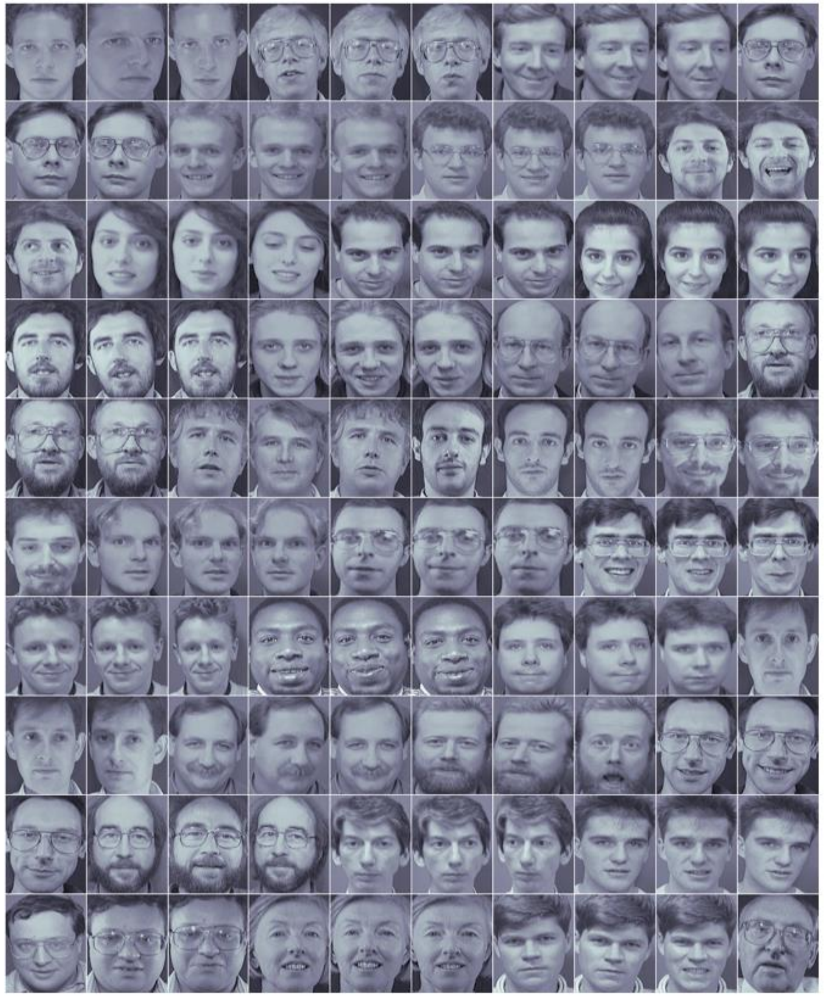


$\boldsymbol{x} \quad k = 1 \quad k = 10 \quad k = 50 \quad k = 200$

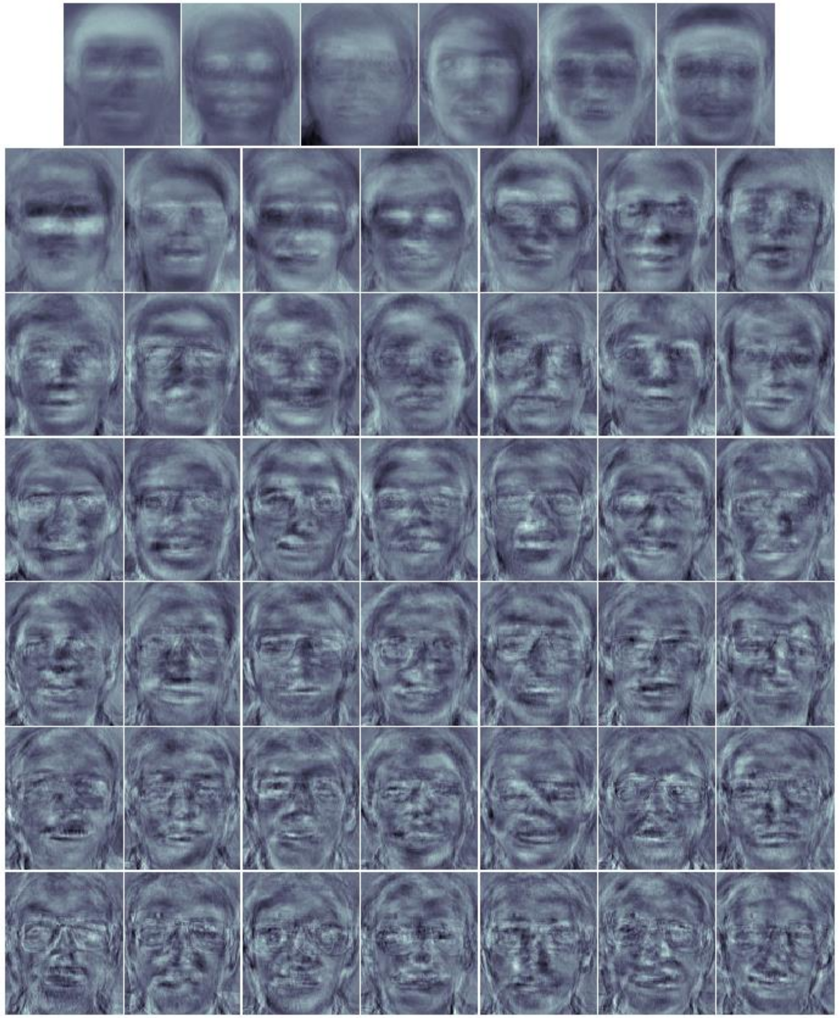
Only have to store k numbers per image,
along with the mean $\boldsymbol{\mu}$ and k eigenvectors ($256(k+1)$ numbers)

Example: eigenfaces

92×112 pixel images of faces (as vectors in $\mathbb{R}^{10304}$)



100 example images



top $k = 48$ eigenvectors

PCA caveat

- The direction of maximizing variance is not necessarily useful for classification!