

# Machine Learning

## Unsupervised learning

DSC 240

March 4, 2025

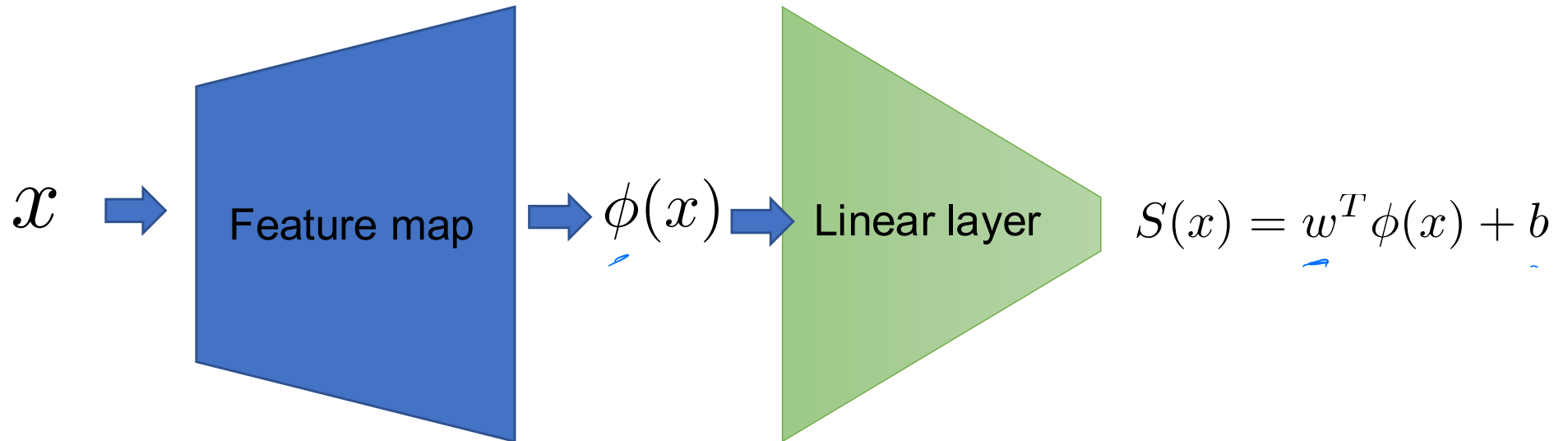
Instructor: Prof. Yu-Xiang Wang

# Announcement

- MP3 / HW4 due March 11
  - Recall: top 3 out of 4 homework determine your total credits from homeworks. But solving HW4 will help you with your learning and the final
- Final Exam time on **March 18, 9:00 am @ CENTR 105** 75 min
  - UCSD does not allow me to run the final exam before the exam week (also, course approval requires a final exam, instead of a second midterm)
- Course Evaluation form (deadline March 15)
  - Please submit your evaluation today!
  - I will keep monitoring the completion rate.
  - 2% participation bonus for everyone if we get above 75%
  - I will send another reminder if by weekend we are still not reaching 75%

# Last time

- Kernel SVM
- Feature learning and neural networks.



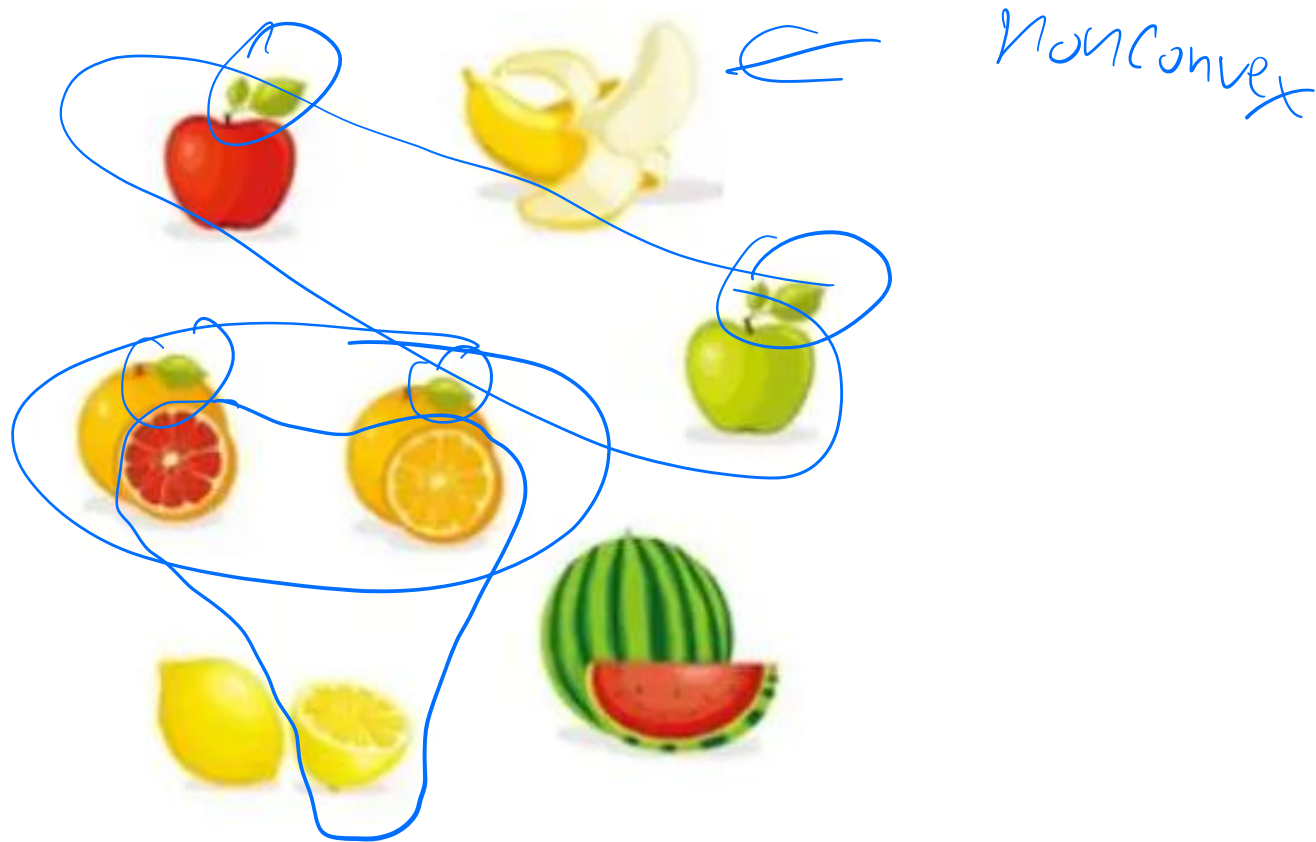
# Today

- Intro to Unsupervised learning
- Clustering
- Dimension Reduction (maybe next lecture)

# Unsupervised learning

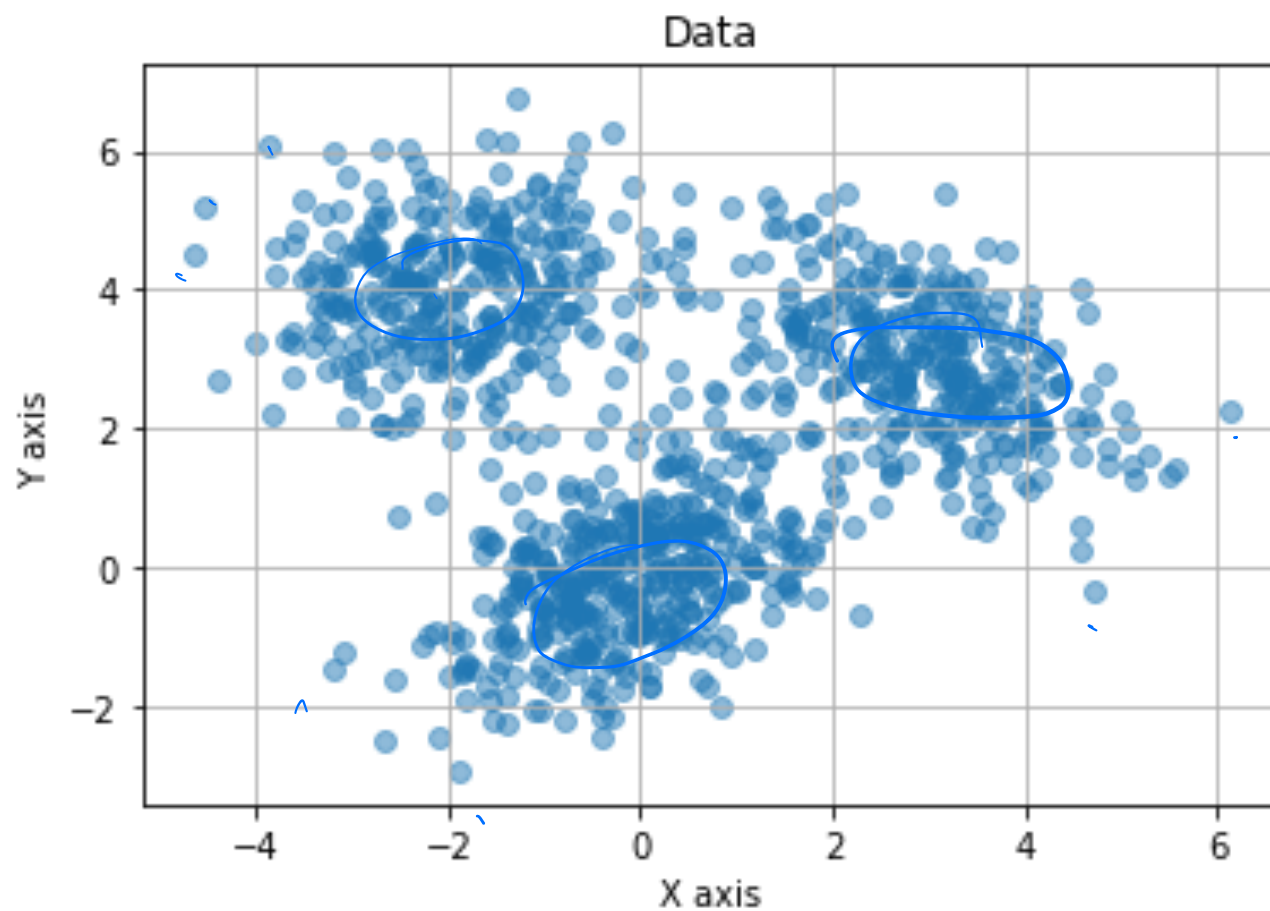
- Input space:  $\mathcal{X}$  Images, videos, text, graphs, proteins, programs, etc...
- Output space: None.
- Hypothesis space:  $\mathcal{H}$   $h: \mathcal{X} \rightarrow \text{Score}$   
↑ how typical / how normal
- Each hypothesis  $h$  is a particular way to summarize the data
- Loss function  $l: \mathcal{H} \times \mathcal{X} \rightarrow \mathbb{R}$   
↓ how much  $h$  explains  $\mathcal{X}$  (particular input)
- Goal of unsupervised learning: Discover simple hypothesis that captures interesting aspects of the data distribution.
  - Often achieved by minimizing the expected loss (i.e., Risk).

The goal of unsupervised learning is to **learn structures in data** without human annotations

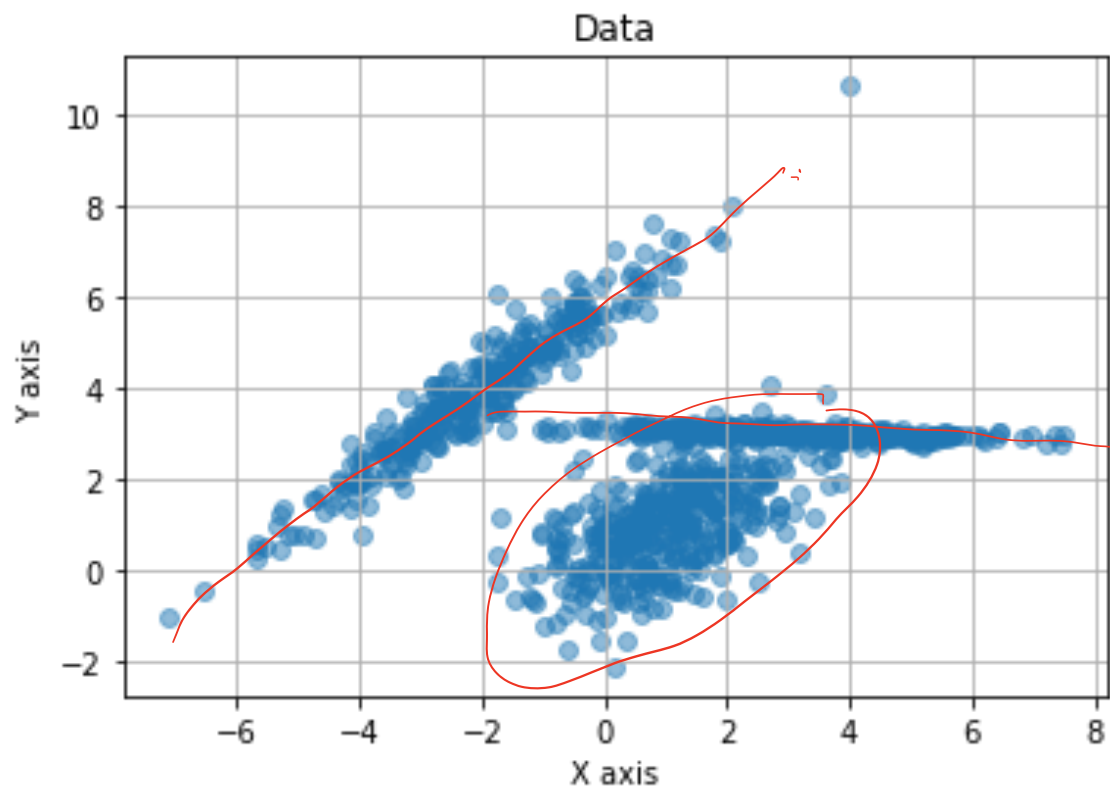


- Discussion: What kind of structures can you see?

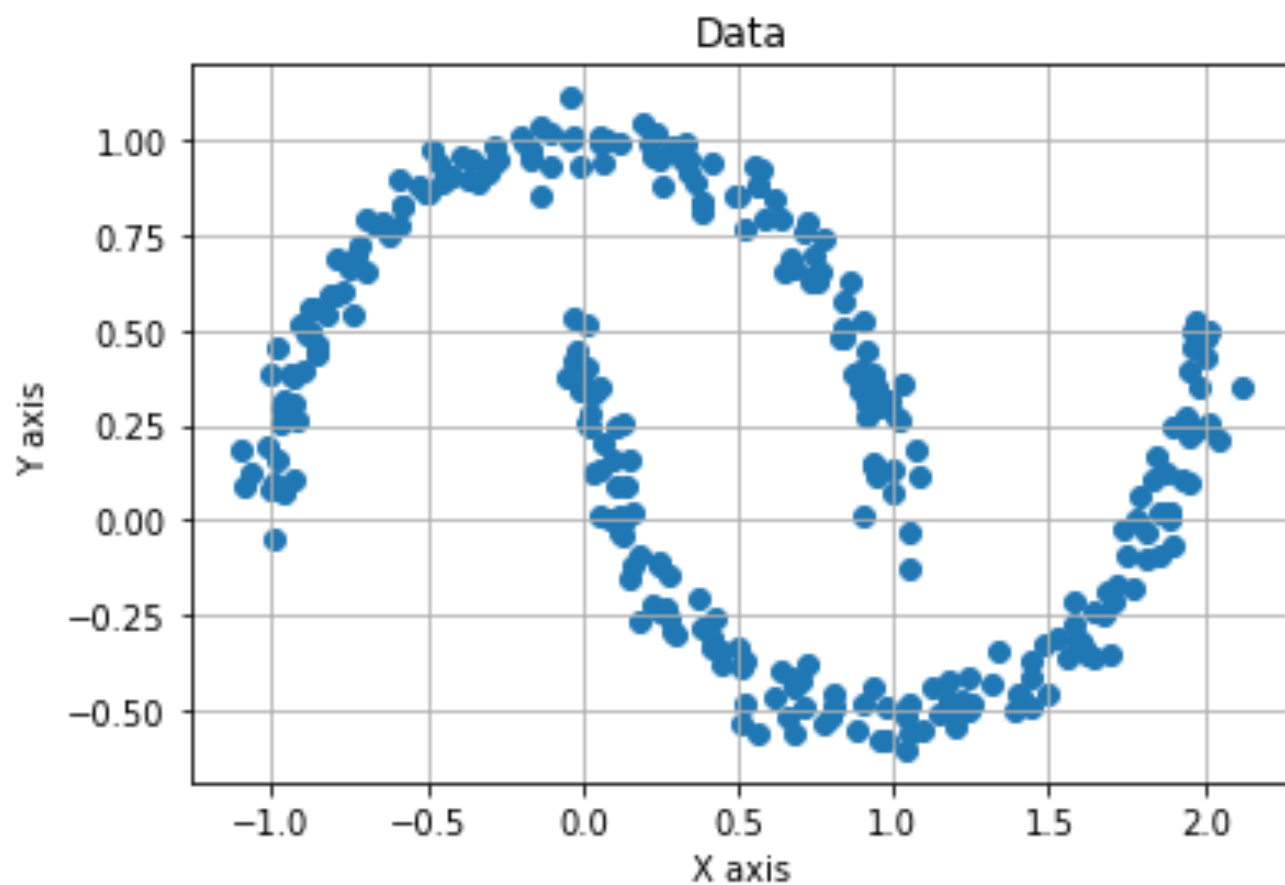
What kind of structures can you see?



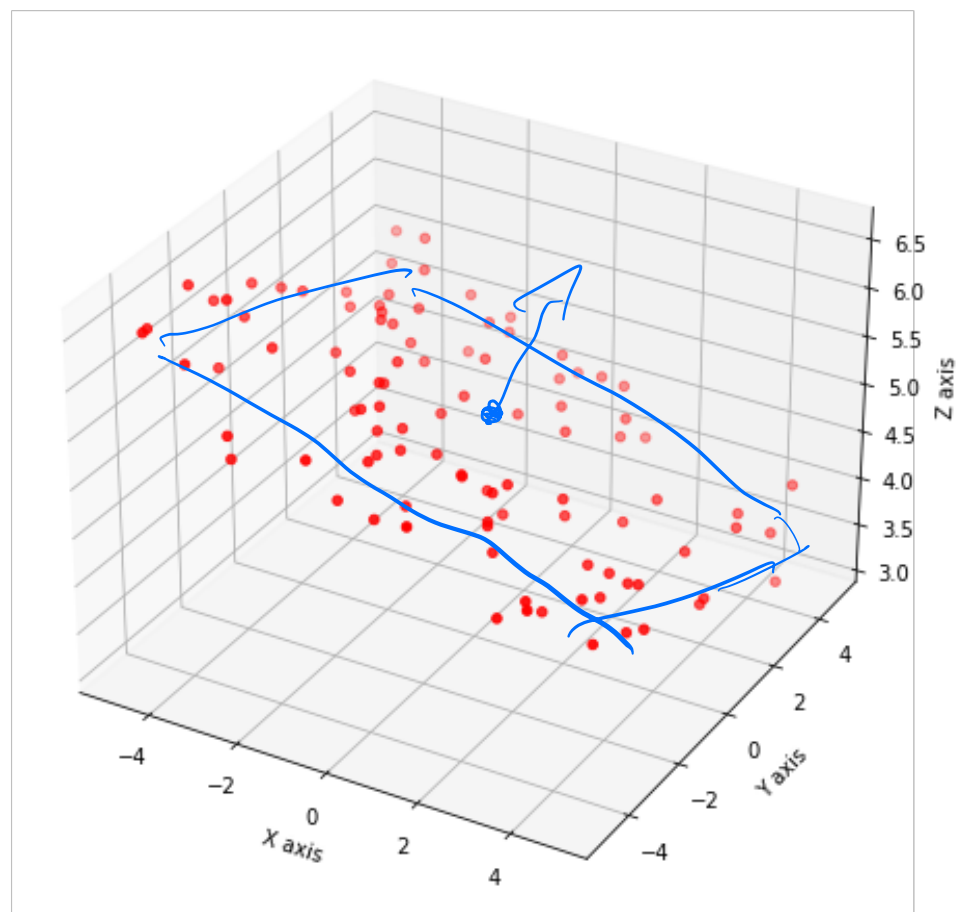
# What kind of structures can you see?



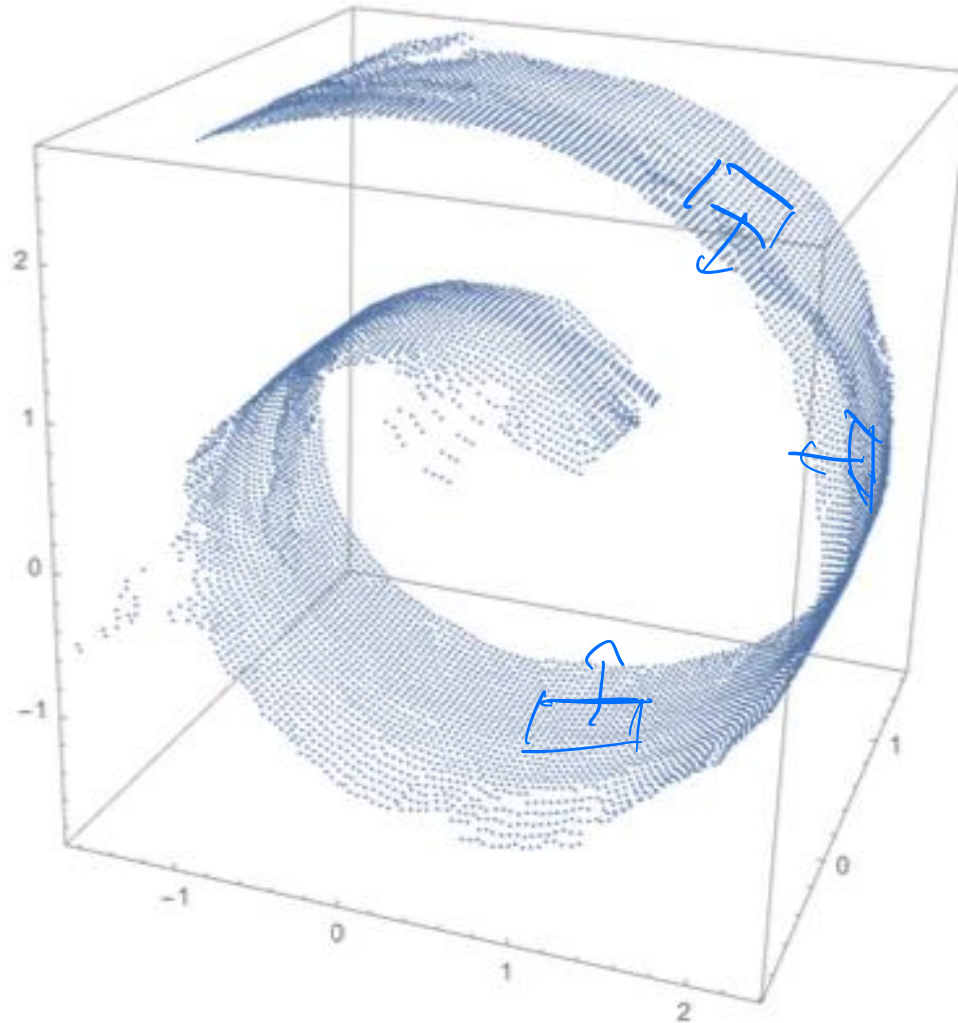
What kind of structures can you see?



What kind of structures can you see?



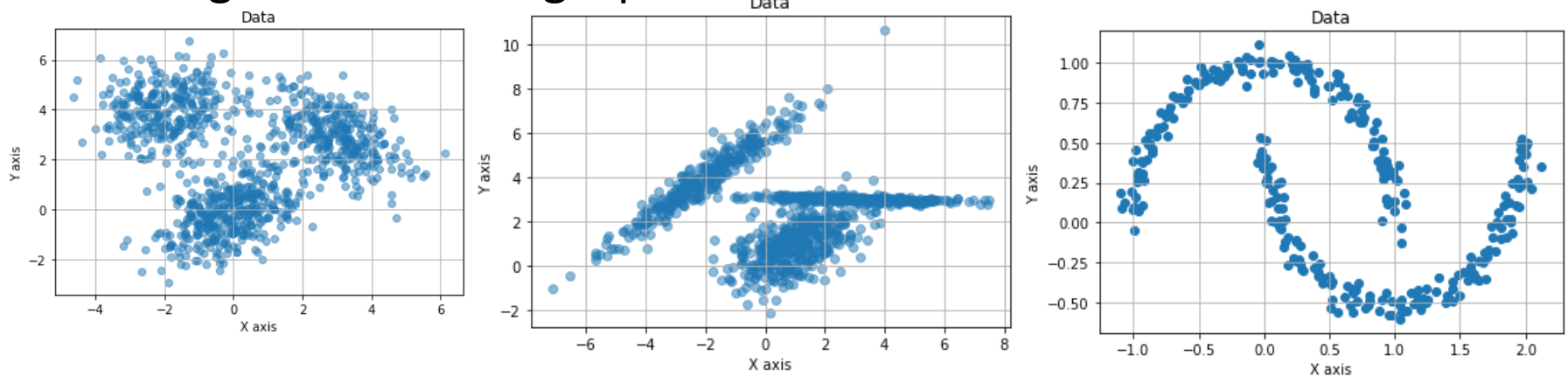
What kind of structures can you see?



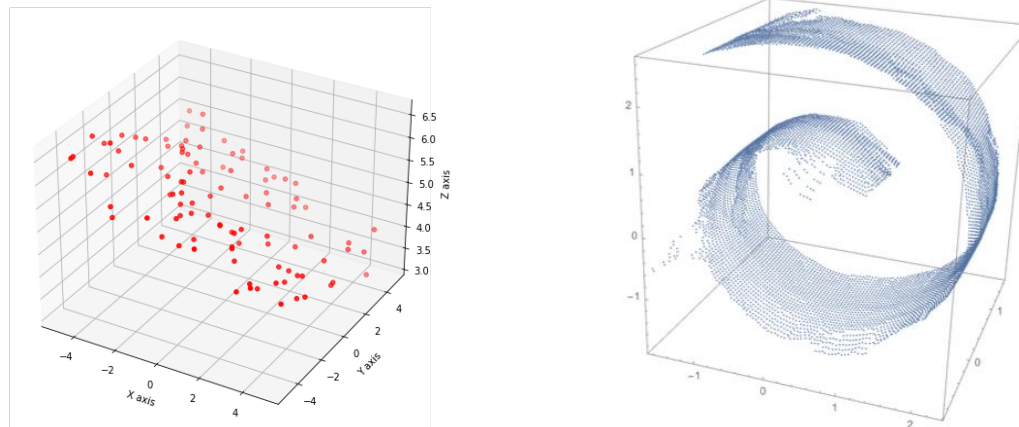
# Two broad categories of unsupervised learning

## (1) Clustering (2) Dimension reduction

- Clustering aims at finding a partition of the data that makes sense.



- Dimension reduction aims at identifying a more compact representation of data



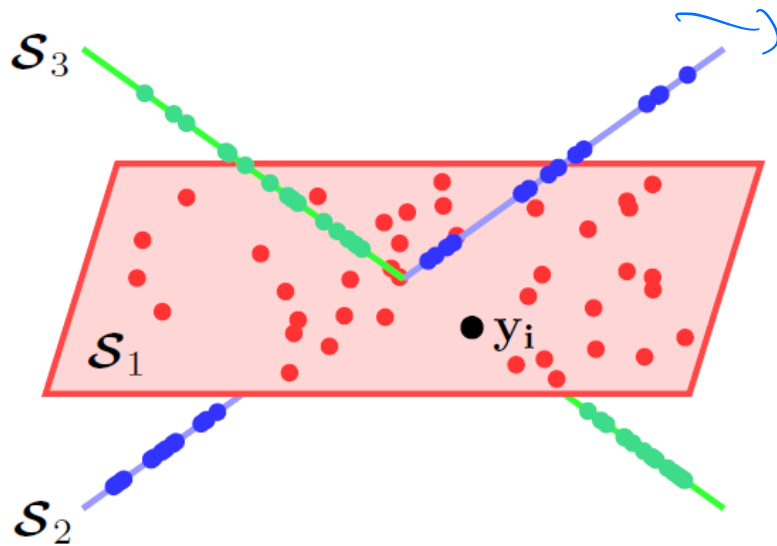
# Applications: Indexing text corpus

<u>“Arts”</u>	<u>“Budgets”</u>	<u>“Children”</u>	<u>“Education”</u>
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give \$1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. “Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services,” Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center’s share will be \$200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive \$400,000 each. The Juilliard School, where music and the performing arts are taught, will get \$250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual \$100,000 donation, too.

# Application: Motion segmentation and subspace clustering

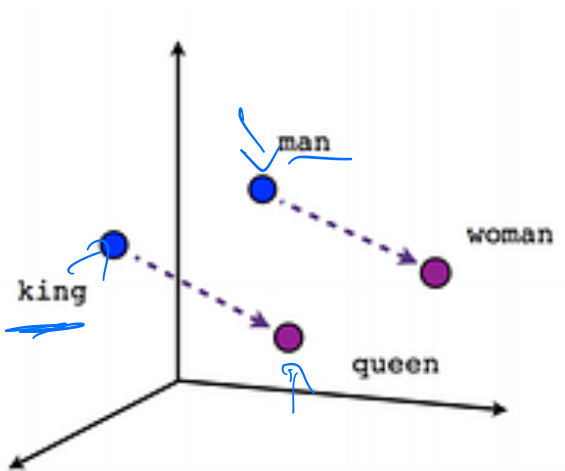
total # of frames  
 $(x_1, y_1)$   
 $(x_2, y_2)$   
 $(x_3, y_3)$   
 $(x_4, y_4)$   
 $(x_5, y_5)$   
 $(x_6, y_6)$   
 $(x_7, y_7)$   
 $(x_8, y_8)$   
 $(x_9, y_9)$   
 $(x_{10}, y_{10})$   
Length  
Video



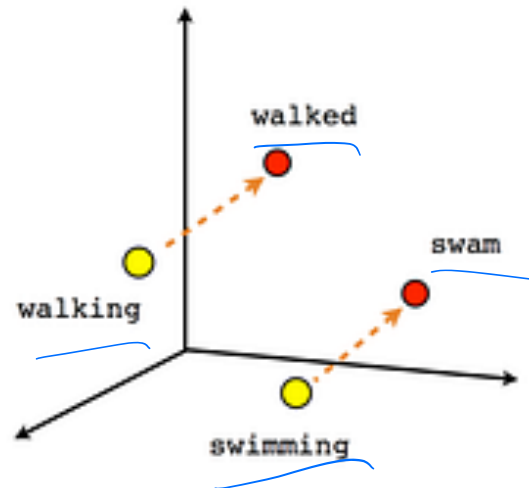
- Turns out that the tracked point trajectories of each rigid moving body captured on a video fall into a 4-dimensional subspace.
- To cluster these points, it suffices to find the subspace membership.

# Applications: learn useful vector space representation of language

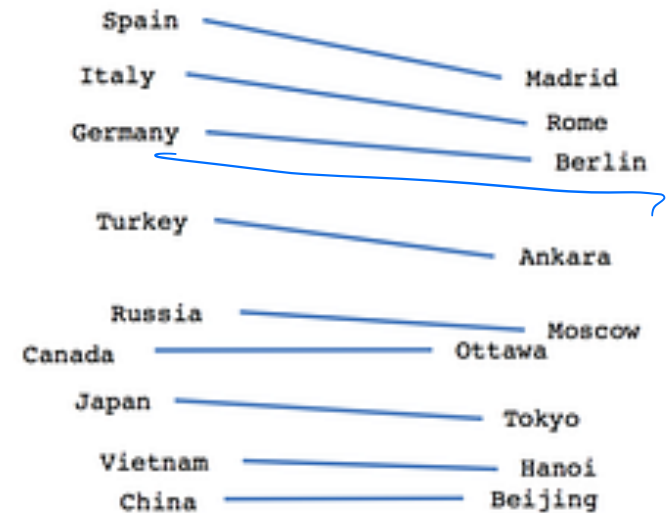
- So you can do algebra on them..



Male-Female



Verb tense

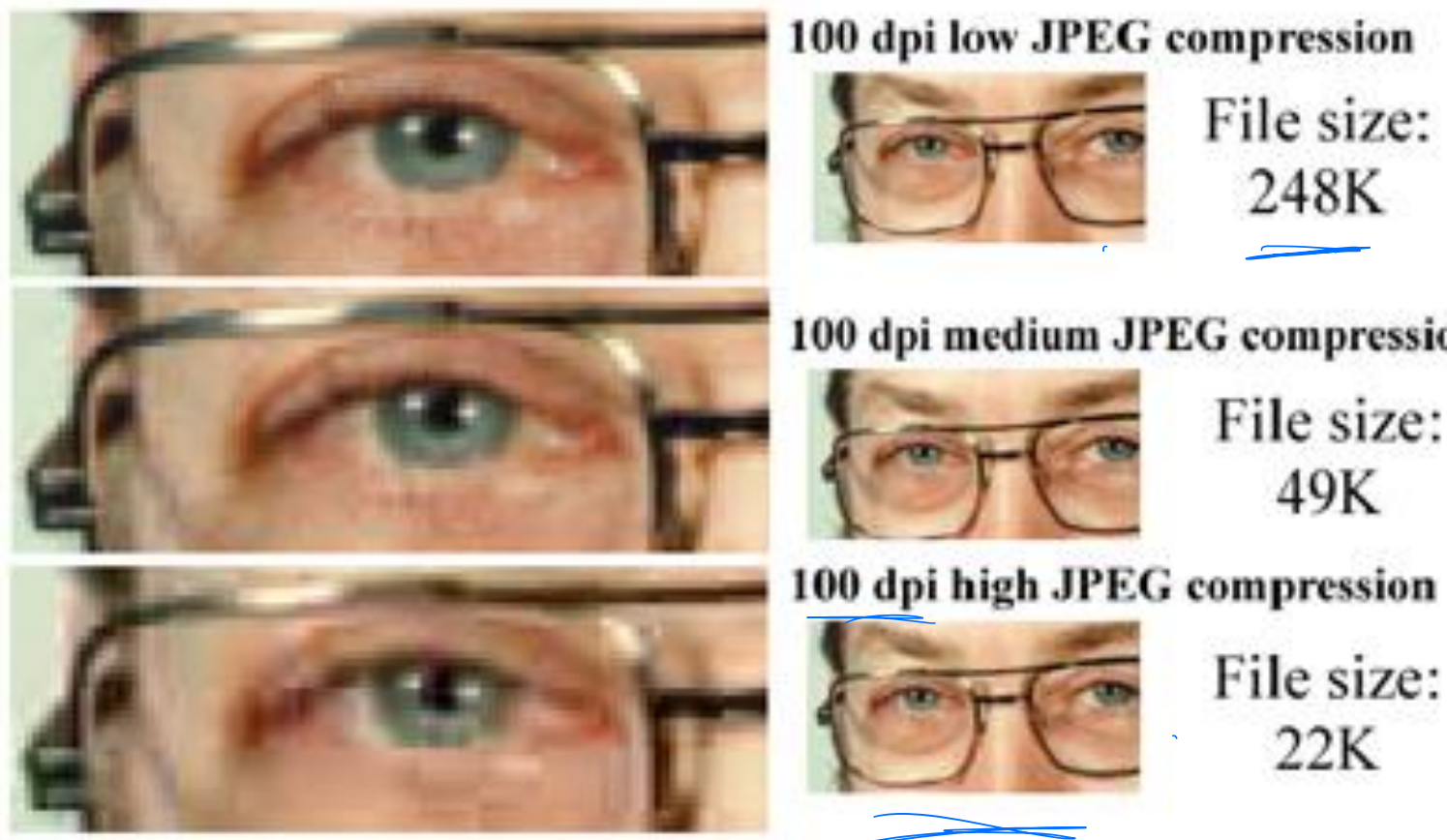


Country-Capital

man - king + queen  $\approx$  woman

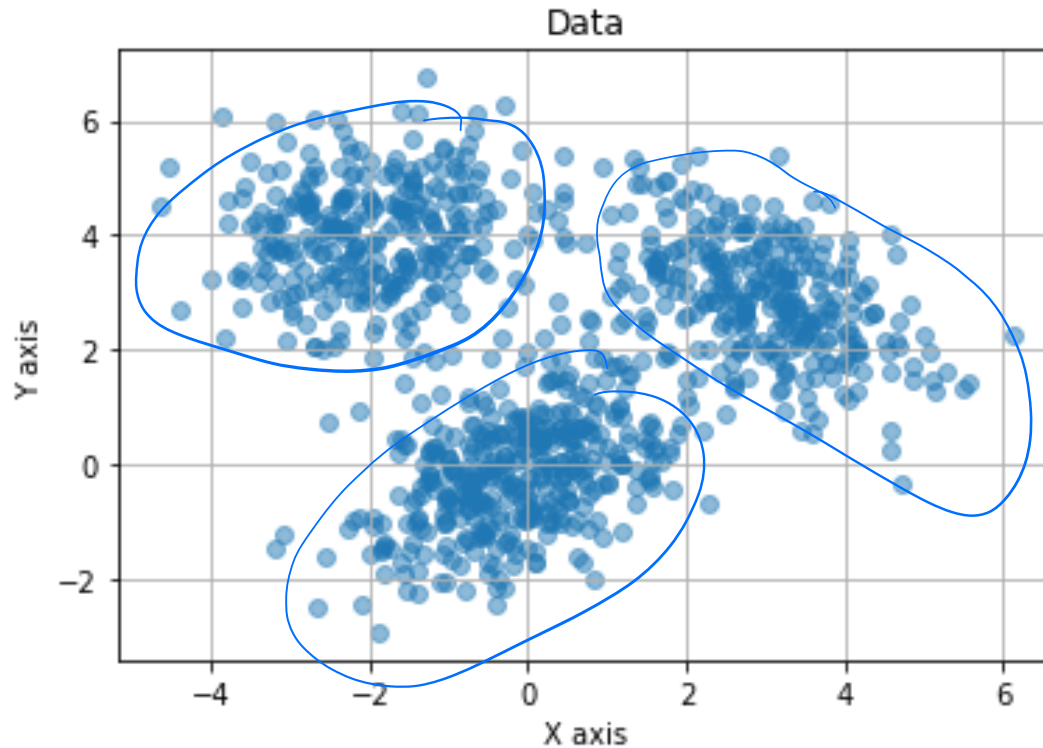
Source: <https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8>

# Application: Image / video compression



Source: <http://preservationtutorial.library.cornell.edu/intro/intro-07.html>

# How do you learn the structure you see?



- Come up with a loss function to minimize?  $\in$  deterministic model
- Come up a probabilistic model that generates the data?  
MLE

# The problem of k-means clustering

$$f = \text{Kmeans Obj Loss}(S)$$

$$\arg \min_S \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2$$

• Where  $S = \{S_1, S_2, \dots, S_k\}$  is a partition of the dataset  $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ ,

• And  $\mu_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x}$ , is called a centroid of  $S_i$

• This optimization problem is *NP-complete*, but we have very practical algorithms. Kmeans++ is guaranteed to find a solution within a factor of  $O(\log k)$ .

$$\hat{S} \text{ s.t. } \min_S f(S) \leq (C \log k) \cdot \min_S f(S)$$

The above optimization problem is equivalent to the following loss minimization

$$\min_{\mu_1, \dots, \mu_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \min_{j \in [k]} \|x_i - \mu_j\|^2$$

Handwritten notes:  $(= \sum_{i=1}^n \min_{j \in [k]} \|x_i - \mu_j\|^2)$ ,  $\mu_i = \mu_k$ ,  $\min_{\mu_1, \dots, \mu_k}$ ,  $\min_{j_1, j_2, \dots, j_n \in [k]}$ ,  $\frac{1}{n} \sum_{i=1}^n \|x_i - \mu_{j_i}\|^2$

- Once we find the **centroids**, finding the **partition of the data** is easy.

- If we have the **partition**, finding the corresponding **centroids** is also easy.

- Idea:** Alternating minimizing the **centroids** and **cluster assignments**.

$$\min_{\mu_1} \sum_{i \in C_1} \|x_i - \mu_1\|^2 \quad \nabla = \sum_{i \in C_1} 2(\mu_1 - x_i) = 0$$

$$\mu_1 = \frac{\sum_{i \in C_1} x_i}{\# \text{ of } j_i \text{ s.t. } j_i = 1}$$

# K-means clustering with Lloyd's algorithm

Clever Seeding ( $K_{means}++$ )

$K$  is a hyperparameter

---

**Algorithm**  $KMeans(D, K)$  –  $K$ -means clustering using Euclidean distance  $Dis_2$ .

---

**Input** : data  $D \subseteq \mathbb{R}^d$ ; number of clusters  $K \in \mathbb{N}$ .

**Output** :  $K$  cluster means  $\mu_1, \dots, \mu_K \in \mathbb{R}^d$ .

randomly initialise  $K$  vectors  $\mu_1, \dots, \mu_K \in \mathbb{R}^d$ ;

**repeat**

    assign each  $\mathbf{x} \in D$  to  $\operatorname{argmin}_j Dis_2(\mathbf{x}, \mu_j)$ ;  $\leftarrow$  1-Nearest neighbor assignment

**for**  $j = 1$  to  $K$  **do**

$D_j \leftarrow \{\mathbf{x} \in D \mid \mathbf{x} \text{ assigned to cluster } j\}$ ;  $\leftarrow$  Partition defined by assignment

$\mu_j = \frac{1}{|D_j|} \sum_{\mathbf{x} \in D_j} \mathbf{x}$ ;  $\leftarrow$  Re-compute the cluster mean

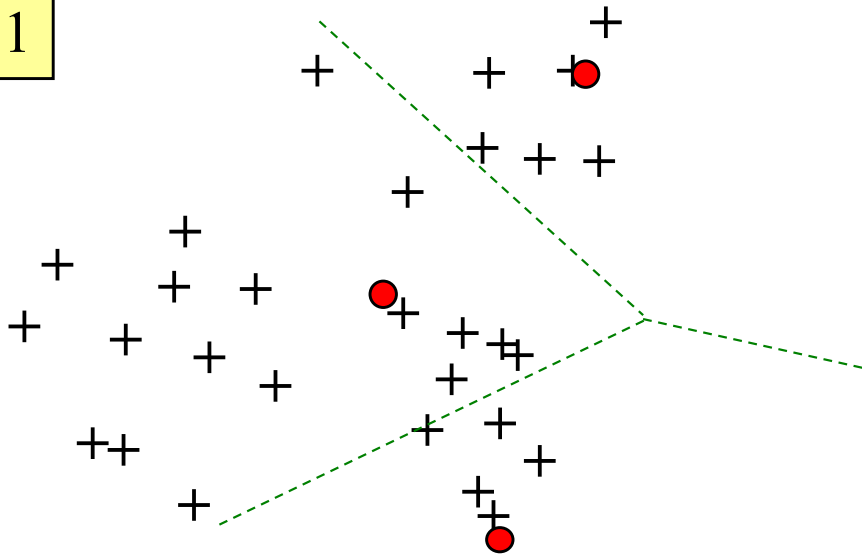
**end**

**until** no change in  $\mu_1, \dots, \mu_K$ ;

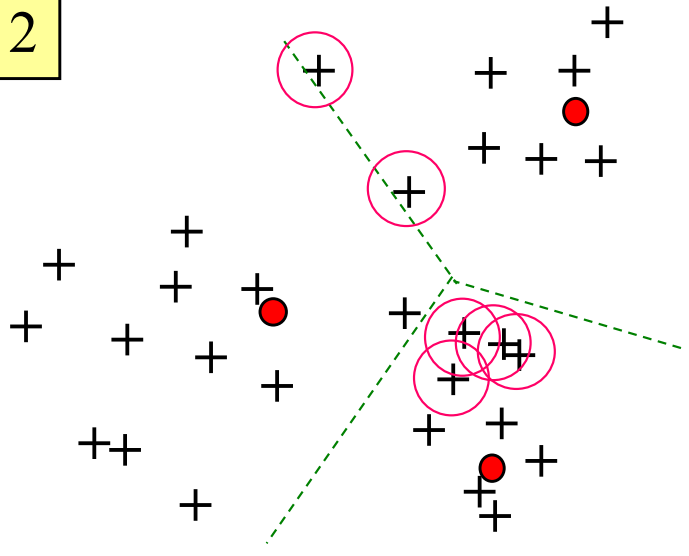
**return**  $\mu_1, \dots, \mu_K$ ;

---

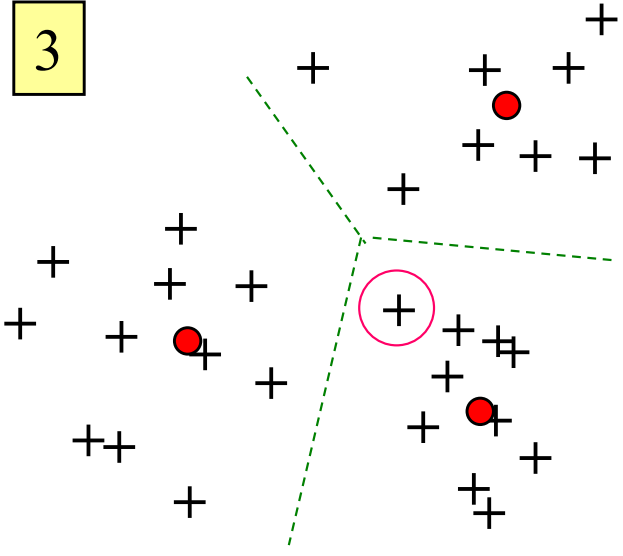
1



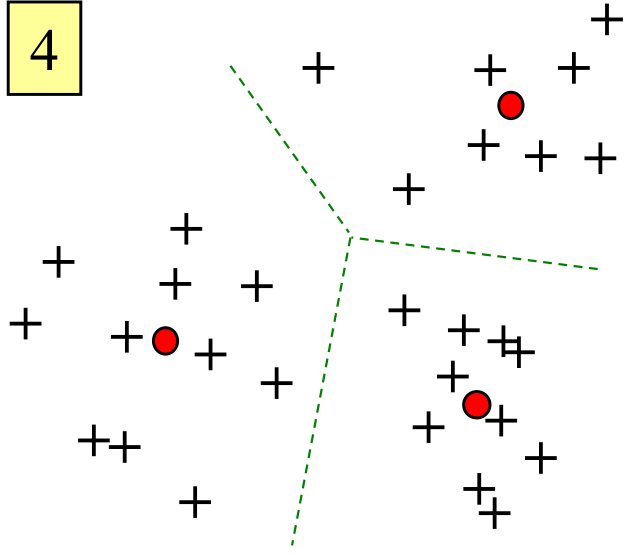
2



3



4

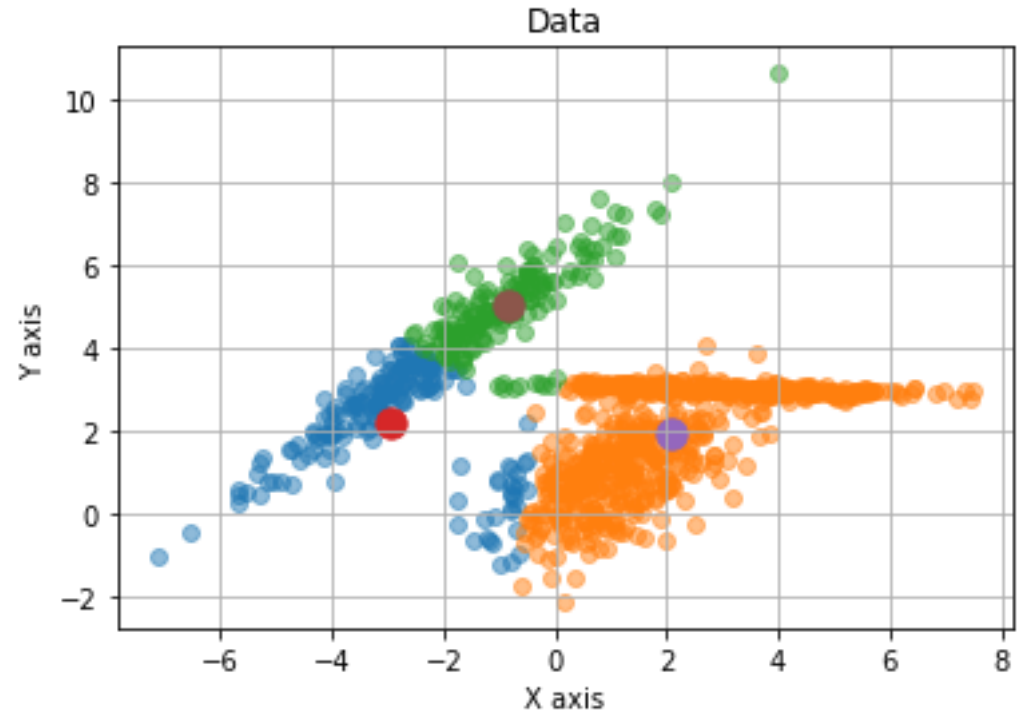
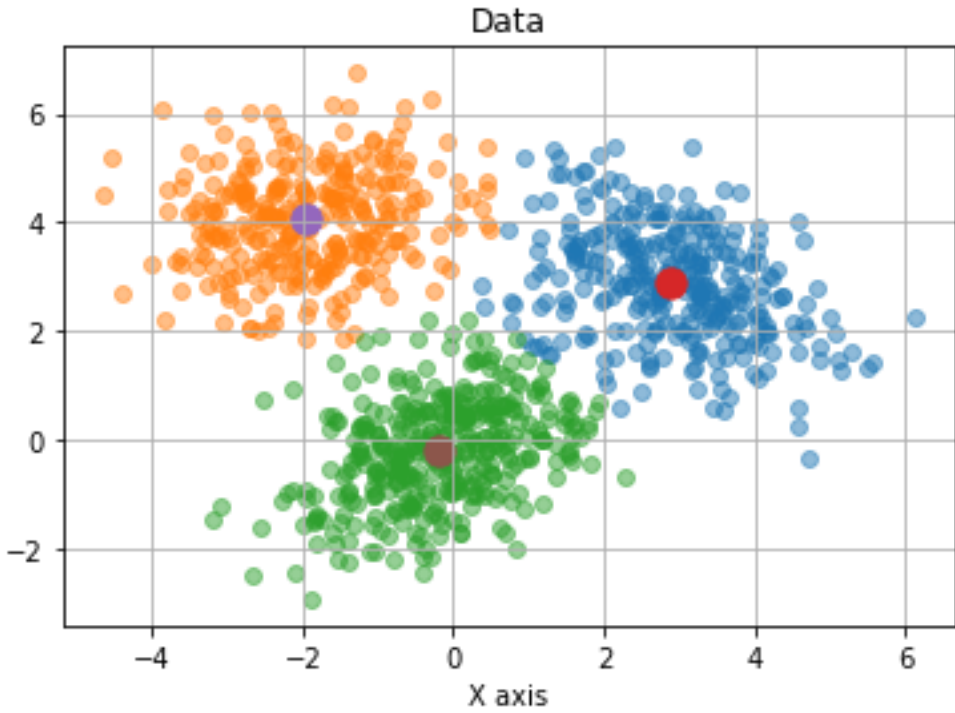


No change – finished!

Nice demo of k-means variants for  
you to play with

<https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

# K-means on our previous examples

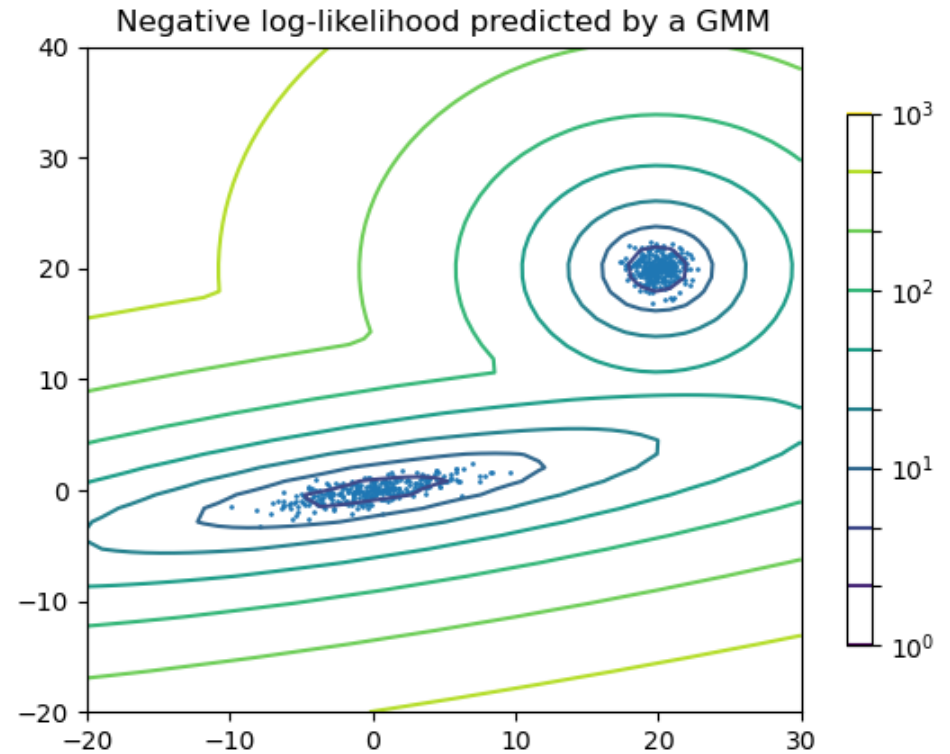
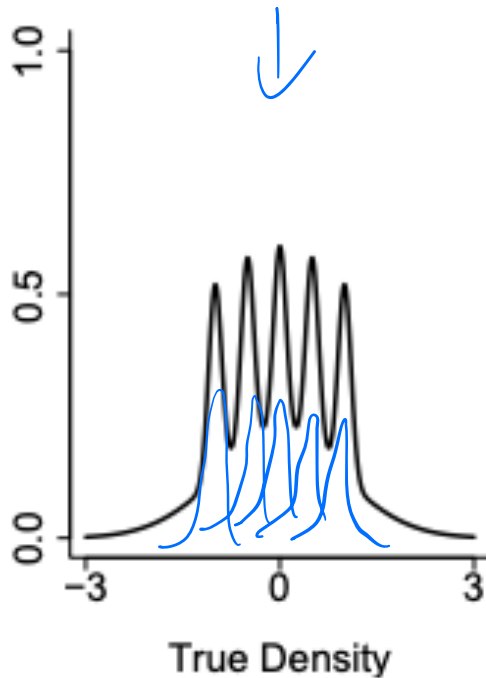


hyperparameter  $\rightarrow K =$

# Gaussian mixture models

$(\mu_j, \Sigma_j)$  for  $j \in \{1, \dots, K\}$   
 $P_j \in \Delta^K$

- Assume the data is generated from a mixture of Gaussian distribution



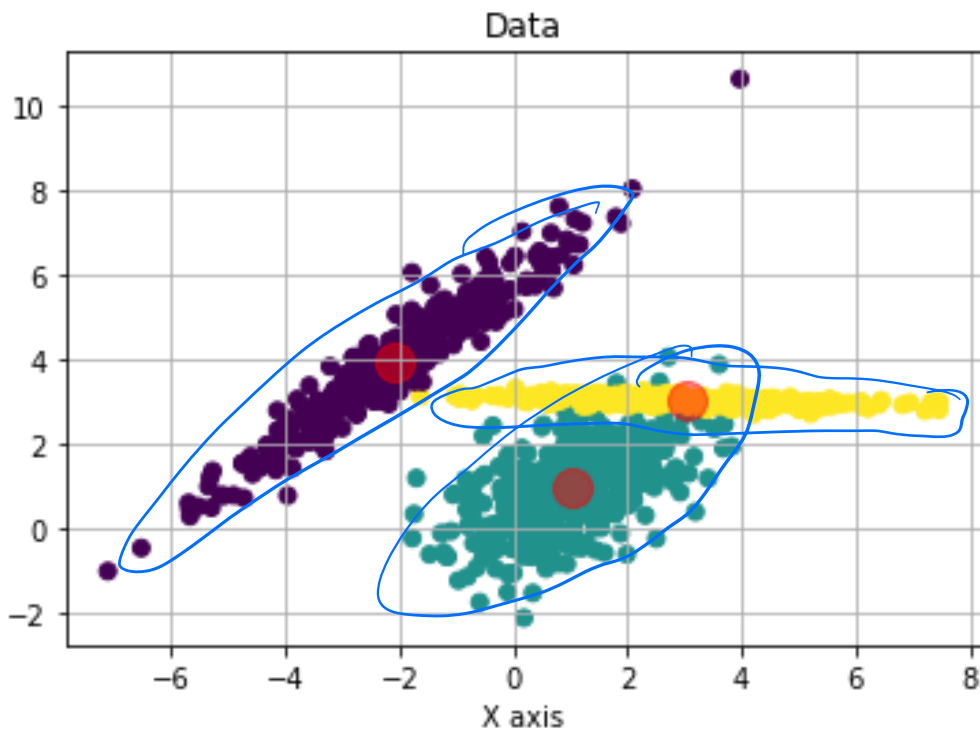
- Data generating process

for  $i=1, \dots, n$   
 $c_i \sim \text{uniform}(\{1, 2, \dots, k\})$   
 $X_i \sim \mathcal{N}(\mu_{c_i}, \Sigma_{c_i})$

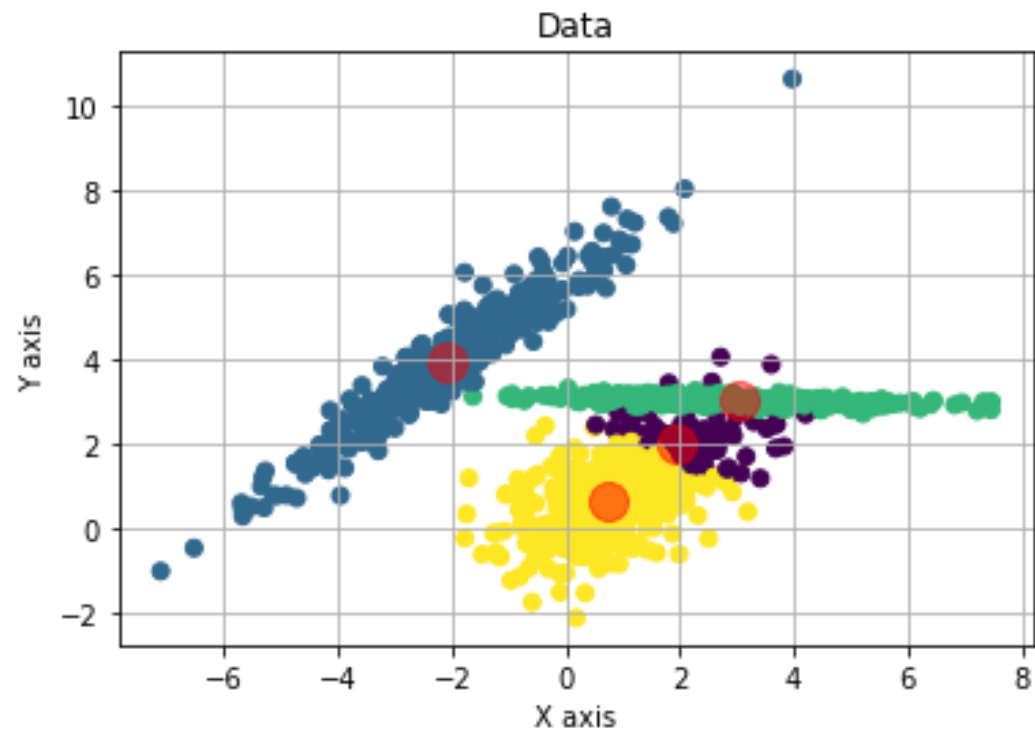
← replace with categorical dist  
 $(P_1, P_2, \dots, P_k)$   
 $\sum_{i=1}^k P_i = 1$

# Fitting Mixture of Gaussian model

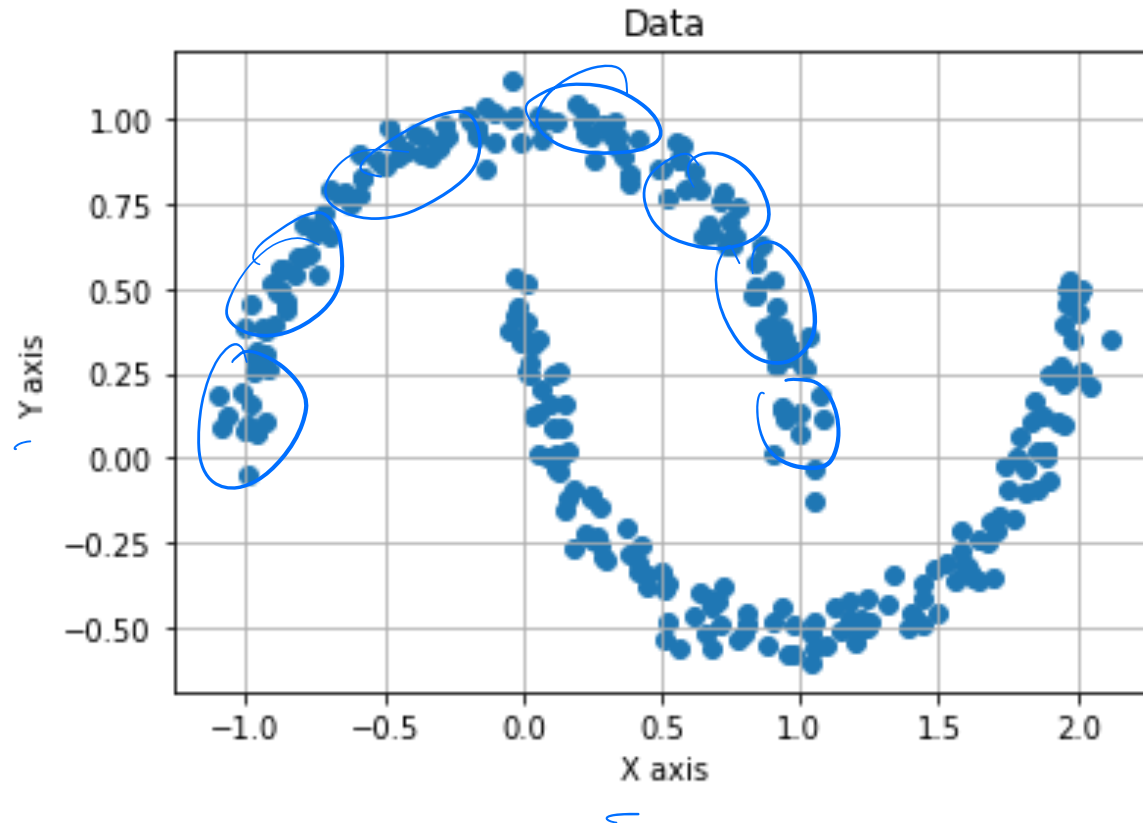
$k=3$



$k=4$



Discussion: what can we do for this?



$$k = \frac{\exp(-\|x - \mu\|^2 / 2\sigma^2)}{\int \exp(-\|x - \mu\|^2 / 2\sigma^2) dx}$$

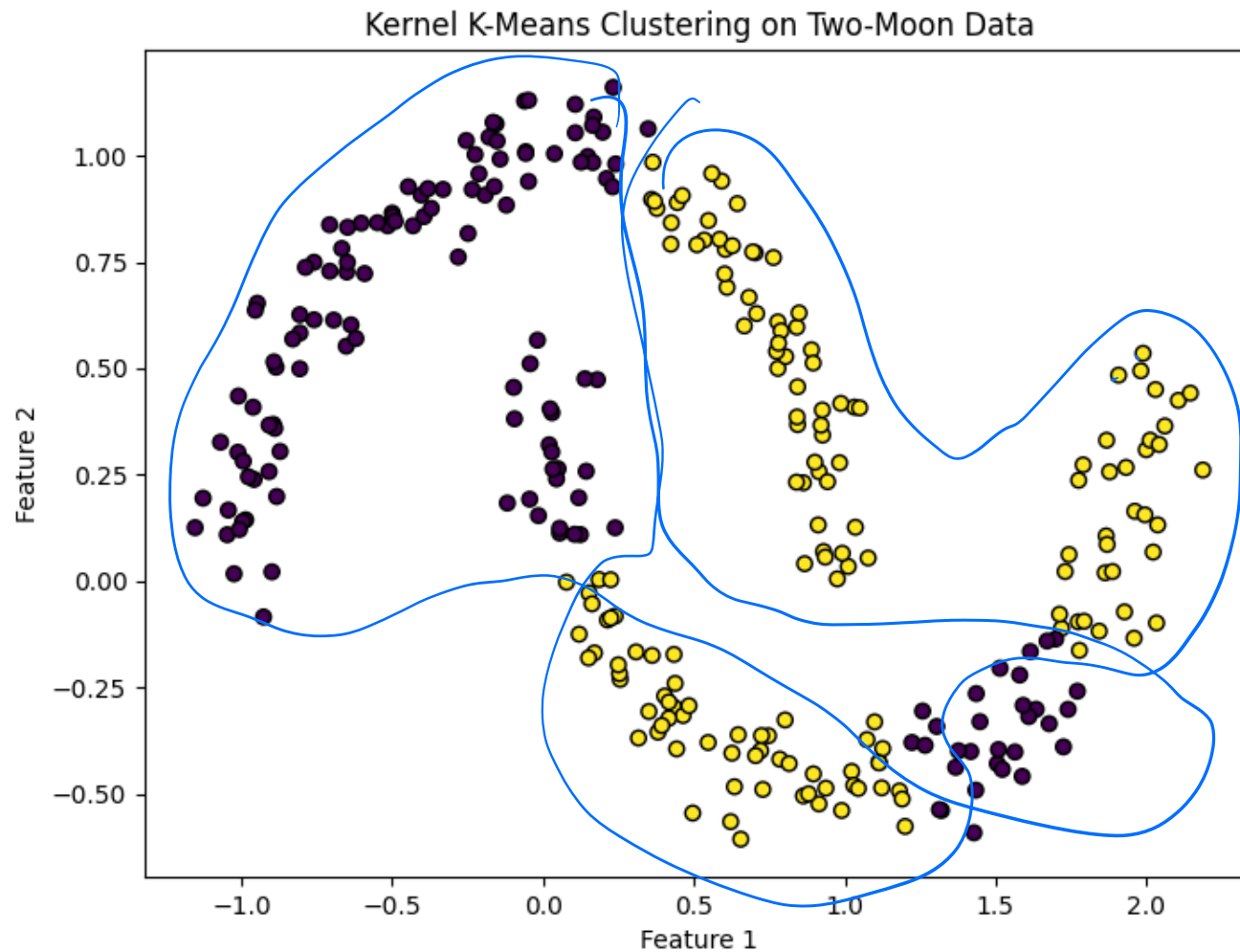
$$\frac{k(x_i)}{\int \phi(x)}$$

$k$  means on  $\phi(x)$

GMM on  $\phi(x)$

- Hint: we have learned some useful tricks last week.

Results for fitting kernel k-means using RBF-kernel with  $\gamma = 12$ . Tweaking bandwidth doesn't quite help.

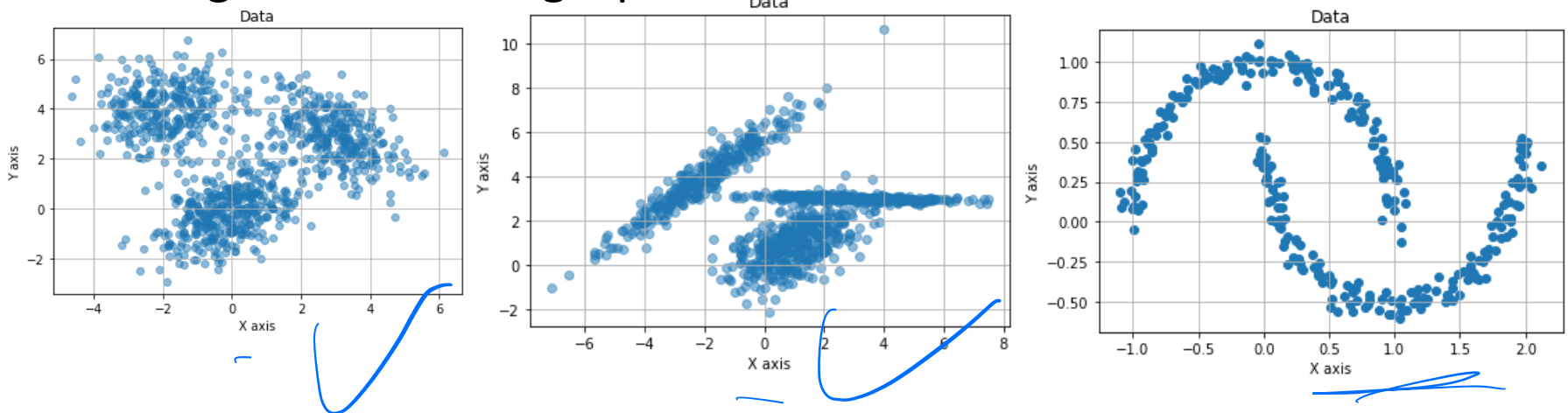


# Checkpoint: Unsupervised Learning

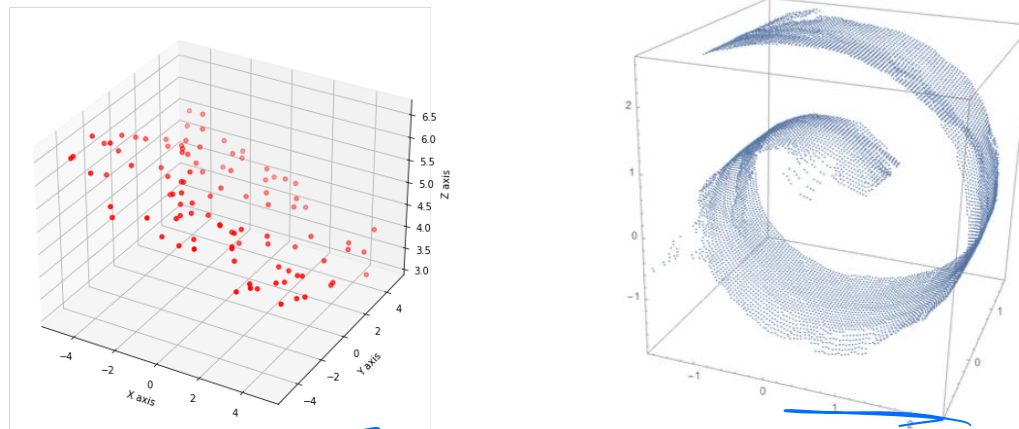
- K-means problem
  - Lloyd's algorithm for solving k-means problem
  - Which distance function to use?
  - How many cluster centers (centroids) to choose?
  - How to initialize the centroids?
  - **Implement the Lloyd's algorithm in HW4 Q2**
- Gaussian Mixture models
  - A probabilistic model for clustering.
  - Soft assignment of observed data points.
  - When the covariance matrix is small and isotropic it is very similar to k-means.
  - **What's the difference from Gaussian Naïve Bayes model?**

# Recap: Two broad categories of unsupervised learning (1) Clustering (2) Dimension reduction

- Clustering aims at finding a partition of the data that makes sense.



- Dimension reduction aims at identifying a more compact representation of data

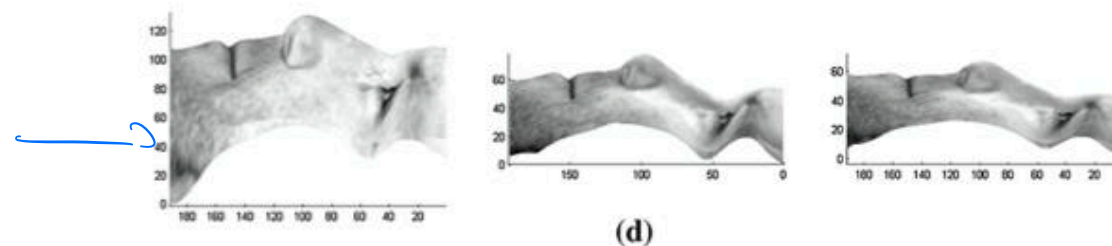
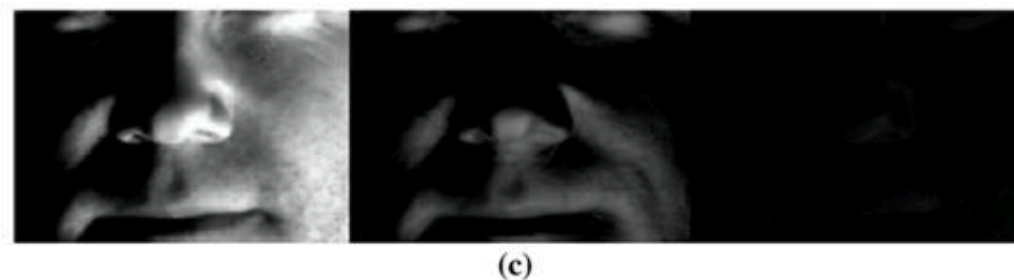
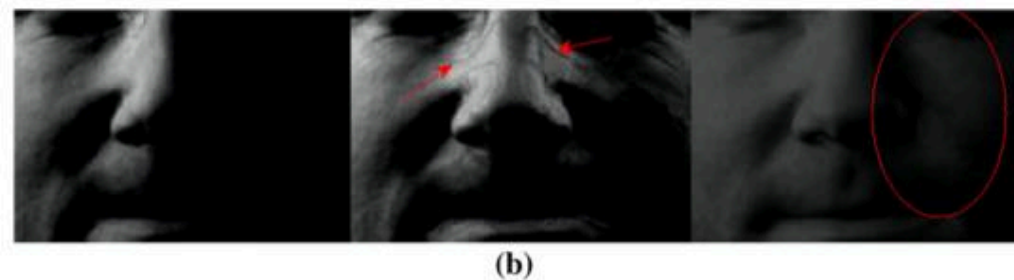
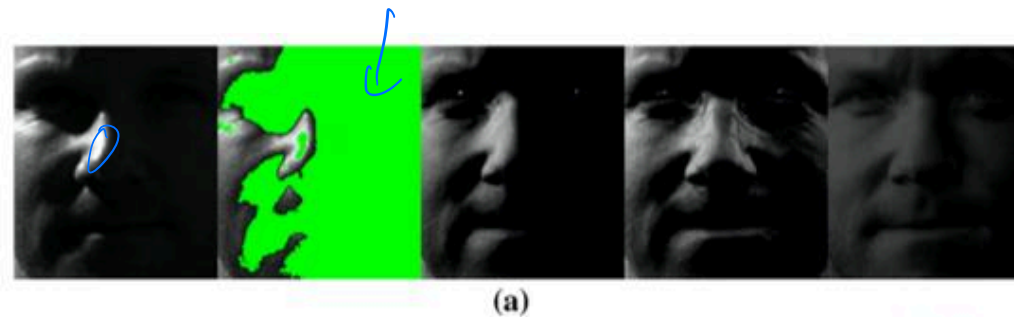


# Why dimension reduction

- Computation, memory efficiency
- Statistical efficiency (fewer samples to learn):
  - “Curse of dimensionality”
- Interpretability: fewer features are easier to understand. Dimension reduction can help identifying hidden causes factors.
- Often data are high-dimensional but the physics mandate that they should be lying on a low-dimensional subspace.

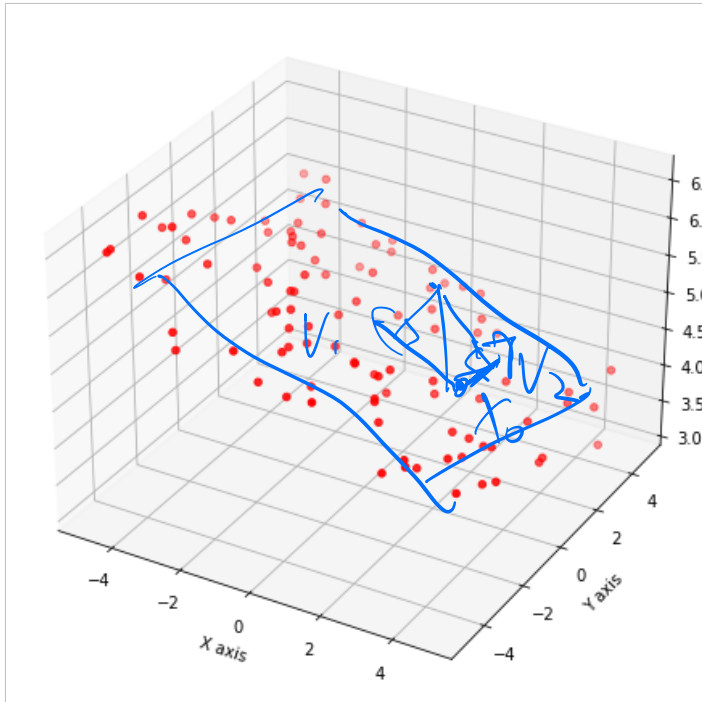
# Shape from shading: Low-dimensional structures allow us to reject gross corruption and reconstruct 3D shape from 2D images.

**Fig. 16** Qualitative comparison of algorithms on Subject 3. From left to right, the results are respectively for PARSuMi, BALM and ALM-RPCA. In **a**, they are preceded by the original image and the image depicting the missing data in *green*. **a** Comparison of the recovered image. **b** Comparison of the recovered image (details). **c** Taking the negative of **b** to see the filled-in missing pixels. This is as if the lighting direction is inverted. **d** Comparison of the reconstructed 3D surfaces (albedo rendered) (Color figure online)



$$\underline{w^T x + b = 0}$$

Discussion: If we know the data is approximately on a hyperplane, how to find that hyperplane?



Hint: all points on a hyperplane can be written as a linear combination of

$$\vec{x} = \vec{x}_0 + \vec{v}_1 \alpha_1 + \vec{v}_2 \alpha_2$$

Parameters of my "h"

- Come up with a loss function to minimize?

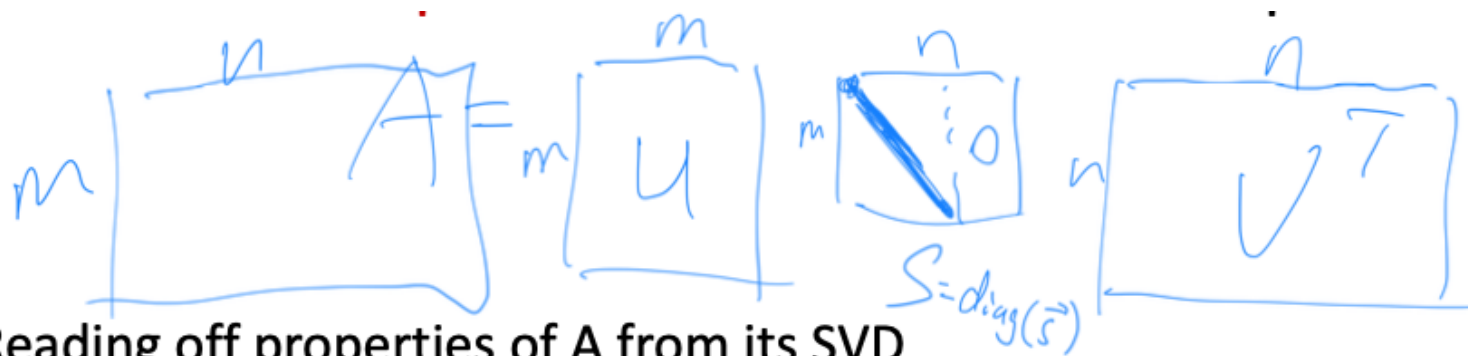
$$\min_{w, b} \frac{1}{n} \sum_i \text{"distance from the plane" of } x_i$$

$$\left( \frac{|w^T x_i + b|}{\|w\|_2} \right)^2$$

$$\min_{x_0, v_1, v_2} \frac{1}{n} \sum_{i=1}^n \min_{\alpha_1, \alpha_2} \|x_0 + \vec{v}_1 \alpha_1 + \vec{v}_2 \alpha_2 - x_i\|^2$$



# Recap (from Lecture 4/5): For any matrix there is a **Singular Value Decomposition**



## • Reading off properties of A from its SVD

- Rank(A) = # of nonzero  $s_i$

- Trace(A) =  $\sum_{i=1}^n s_i$

$$A = U \Sigma U^T$$

Square matrix

- Det(A) =  $\prod s_i$

- Eigenvalue =  $s_1, s_2, \dots, s_n$   
for Symmetric matrix

- Eigenvectors = Columns of U

$$\begin{aligned} A^{-1} &= (U \Sigma V^T)^{-1} = (V^T)^{-1} \Sigma^{-1} U^{-1} \\ &= (V^T)^T \Sigma^{-1} U^T \\ &= V \Sigma^{-1} U^T \end{aligned}$$

- Matrix norm(A)

$$\|A\|_F = \sqrt{\sum s_i^2}$$

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2 = s_1$$

# Principal Component Analysis

**Input:** data matrix  $X$ , number of principal components  $k$

1. Calculate the mean  $\hat{\mu}$  of the rows of  $X$  by  $\hat{\mu} = \frac{1}{n} \sum_i x_i$

• Use

2. Run SVD on the de-meaned data matrix:  $X - \mathbf{1} \cdot \hat{\mu}^T = U S V^T$   
(Use [scipy.linalg.svd](#))

**Output:** the mean  $\hat{\mu}$ , and **the first  $k$  columns of  $V$**  as the principal components.

## Note:

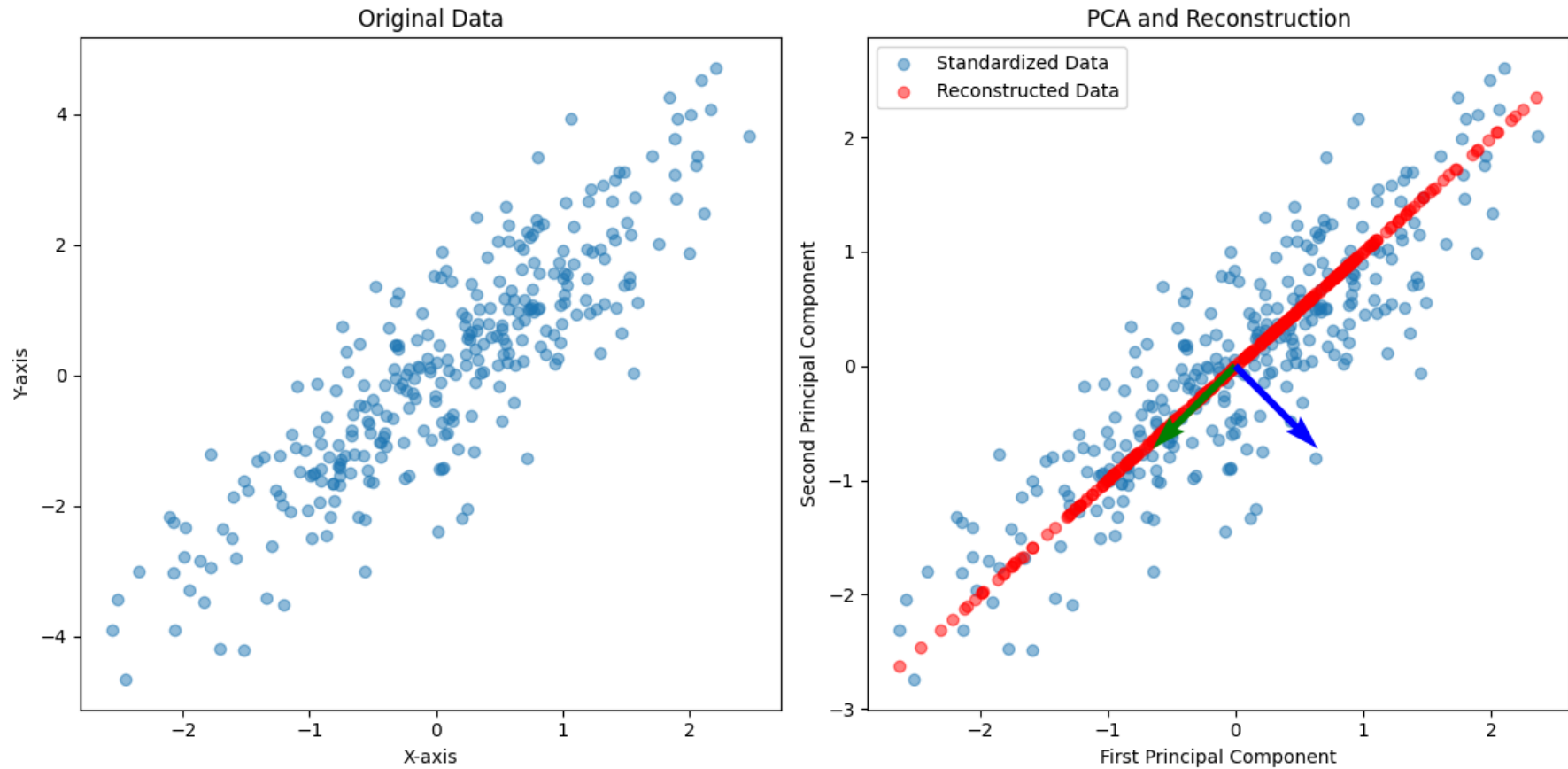
\* SVD solves the best-low-rank approximation of the sample covariance matrix. PCA hopes to explain

\*\* Improved computational efficiency when  $k$  is small: replace full-SVD with skinny SVD that takes  $k$  as an input. (Use [scipy.sparse.linalg.svds](#))

# Principal Component Analysis

1. How to perform dimension reduction for new data points after finding the principal components?
2. How to reconstruct the original data using the coefficients?

# Example of PCA on Gaussian data



# Applications of PCA to image compression



**d=1**



**d=2**



**d=4**



**d=8**

**d=16**



**d=32**



**d=64**



**d=100**



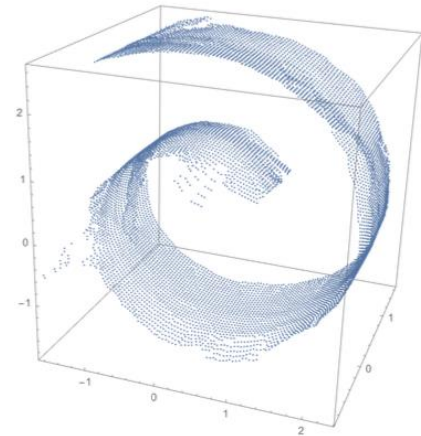
**Original  
Image**



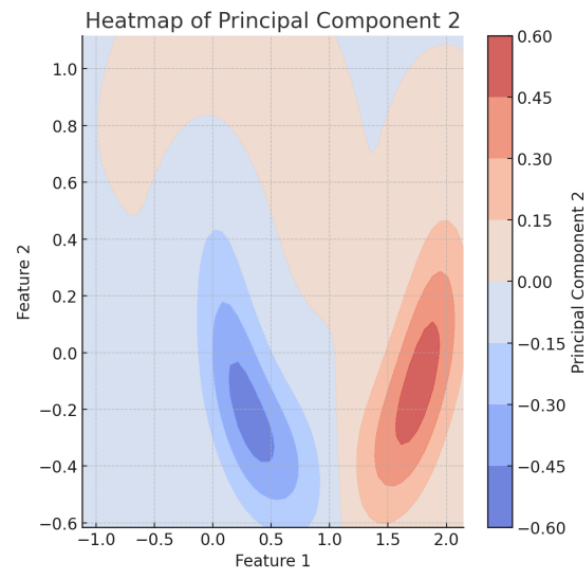
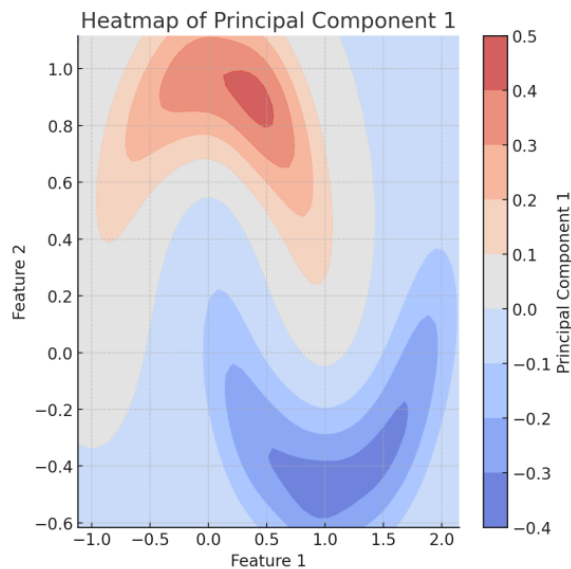
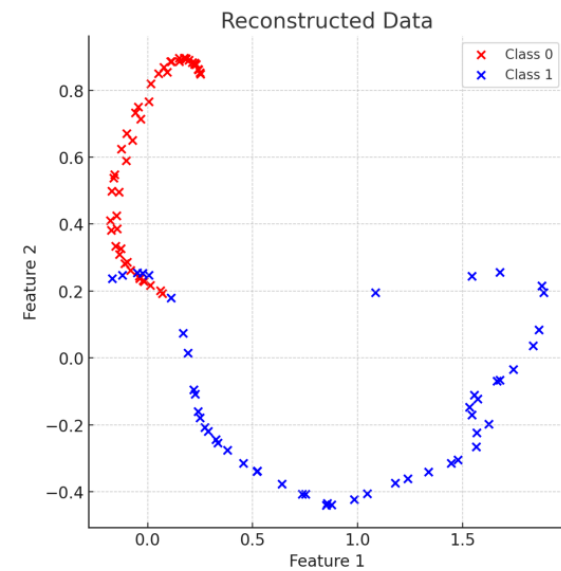
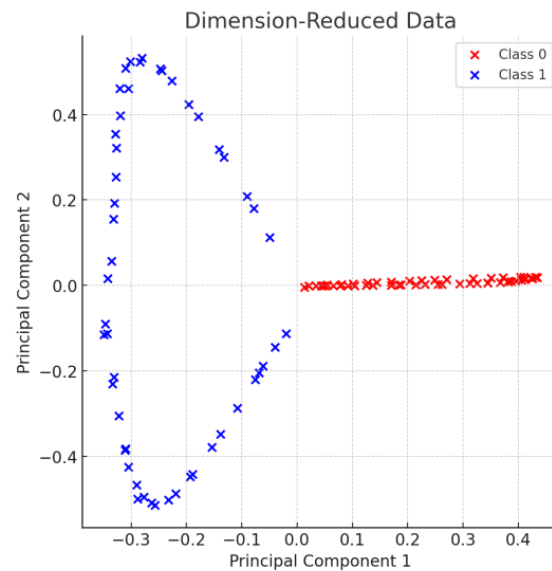
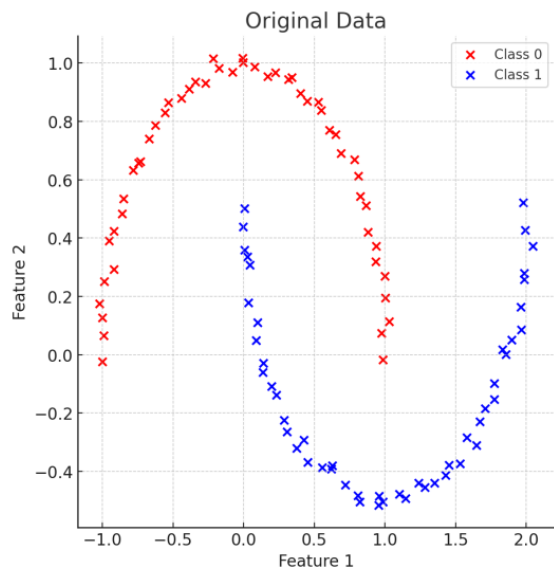
You will learn how to do this in HW4 Q3!

# Going non-linear dimension reduction

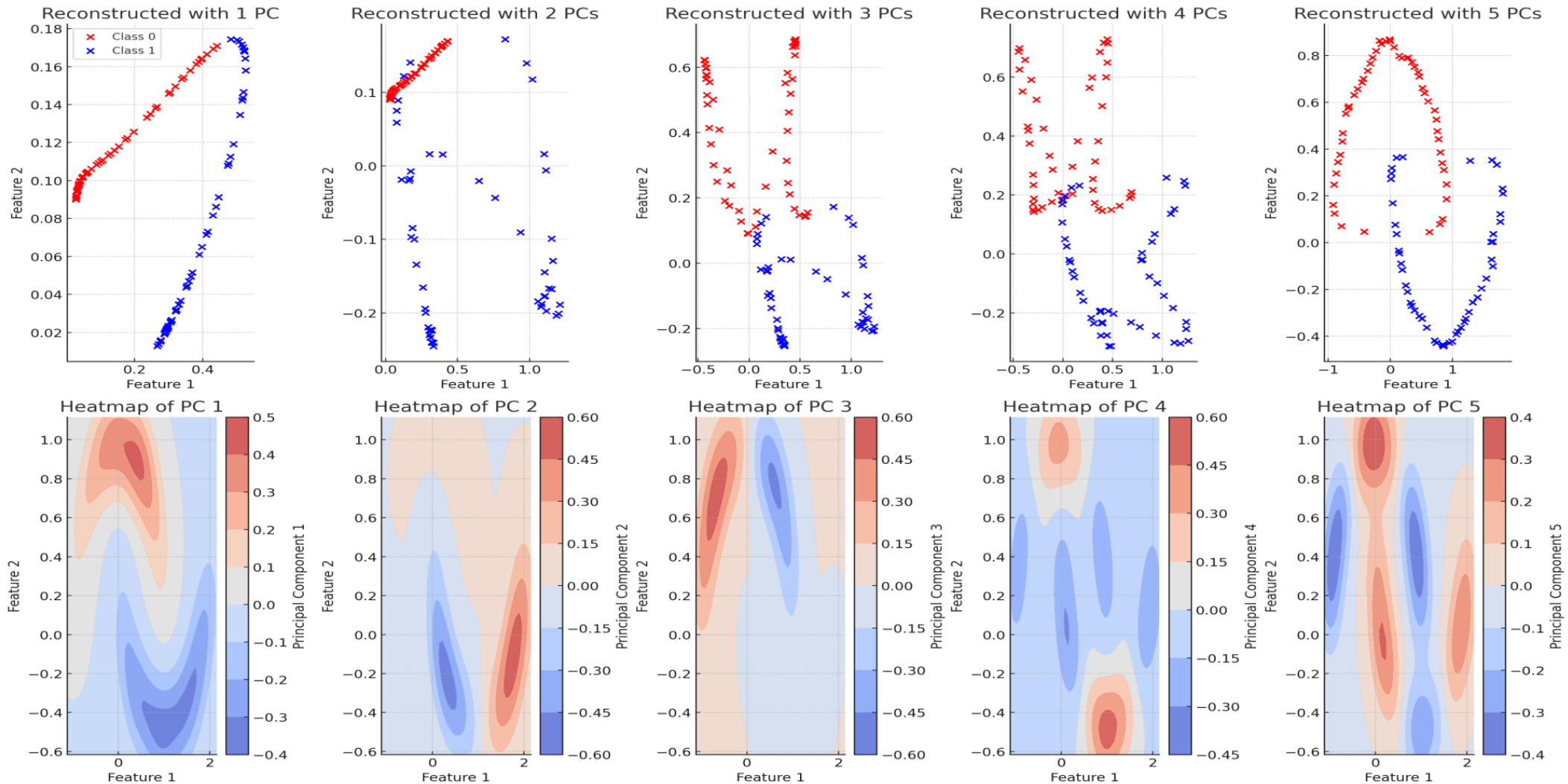
- Mixture of linear subspaces:
  - Subspace clustering
  - Mixture of probabilistic PCAs
  - A combination
- Kernel PCA
  - Run PCA on the Gram matrix instead of the covariance matrix
- Laplace Isomapx, Laplacian Eigenmaps
  - First construct a nearest neighbor graph
  - Then run SVD on the Laplacian matrix of the graph
- Neural approaches:
  - Autoencoders / variational autoencoders
  - Transformers (for data-reconstruction)



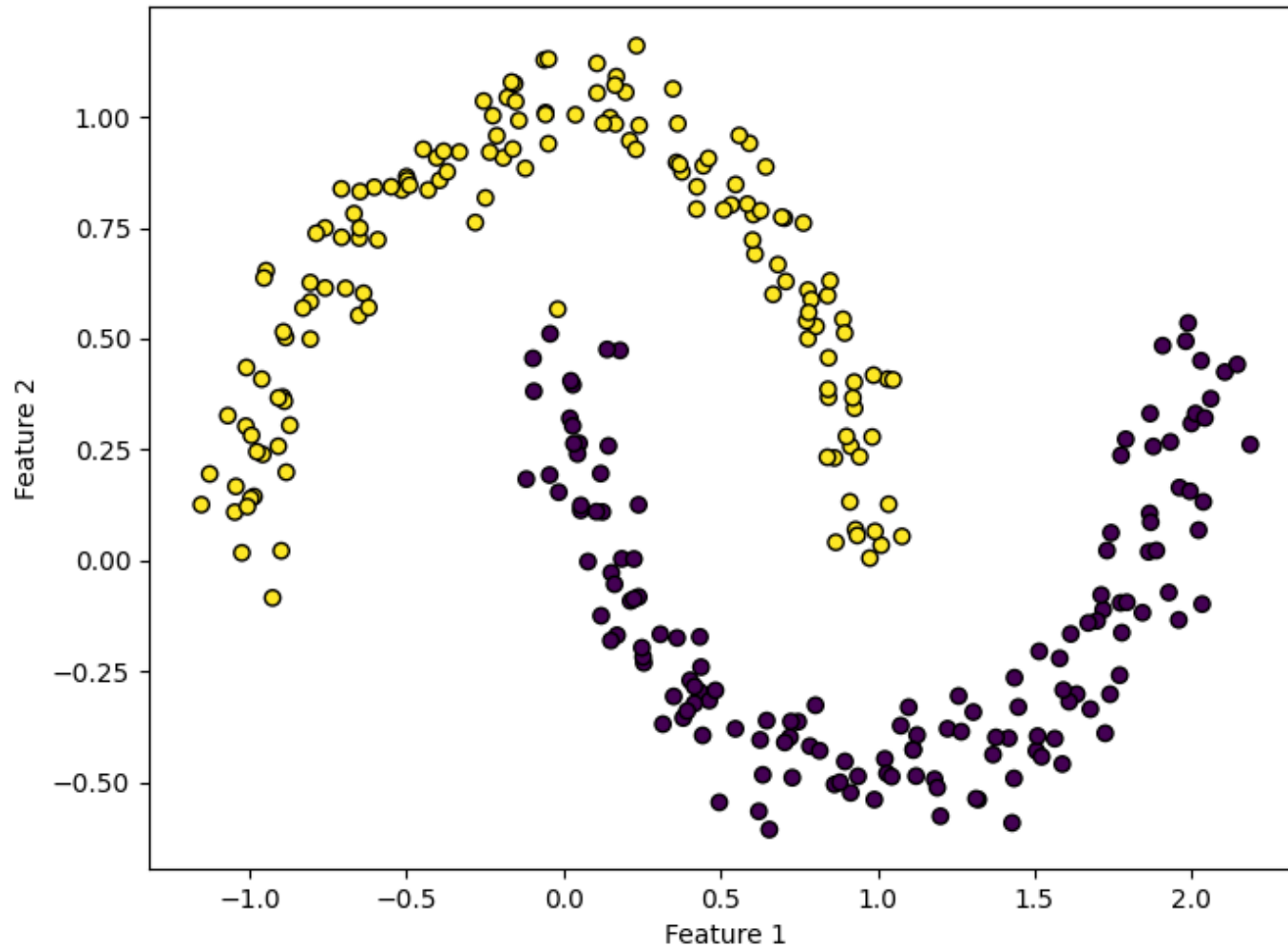
# Kernel PCA on the two-moon example



# Increasing the number of principal components in kernel PCA improves the reconstruction



Back to the clustering example. Let's do dimension-reduction on kernel PCA before running k-means... Now it works!



# Advance topics: Which topic should I cover?

- Generalization?
- Deep Learning?
- Online Learning?
- Reinforcement Learning?