

Machine Learning

Unsupervised learning (Part 2)

DSC 240

March 6, 2025

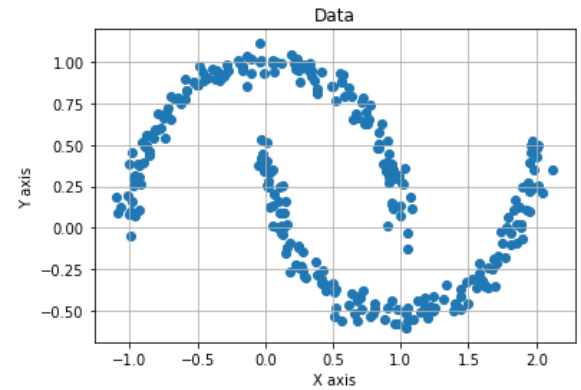
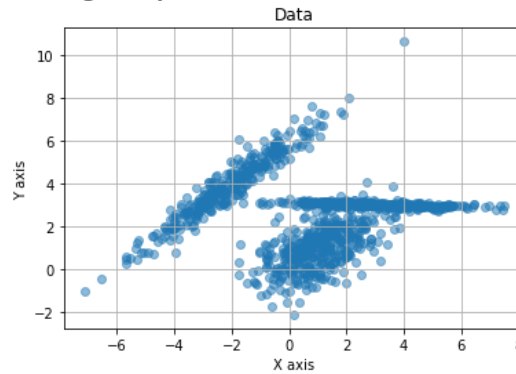
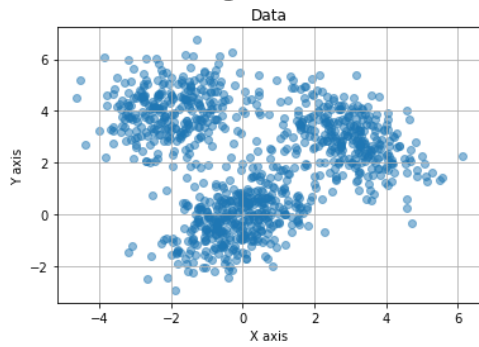
Instructor: Prof. Yu-Xiang Wang

Last time

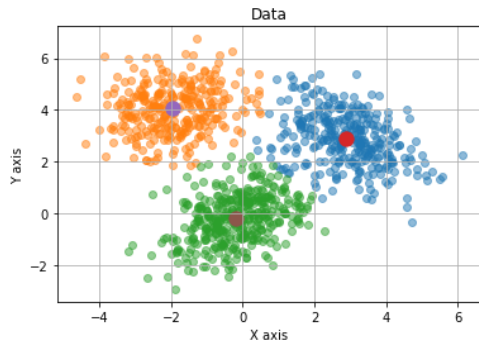
- Clustering:
 - “Loss function” for k-means
 - Lloyd’s algorithm --- Alternating Minimization.
 - Generative model for clustering: Gaussian Mixture Model
 - How to go nonlinear?
- Dimension-reduction
 - Why dimension reduction?
 - Objective function for finding the best low-dimensional subspace to approximate the data

Recap: Clustering

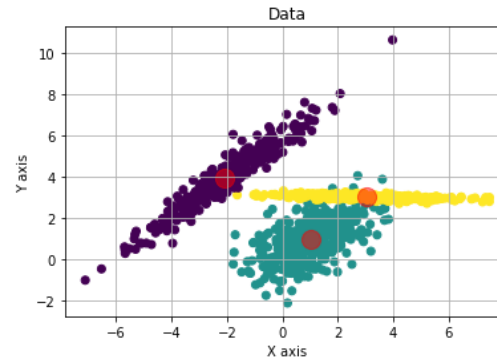
- Clustering aims at finding a partition of the data that makes sense.



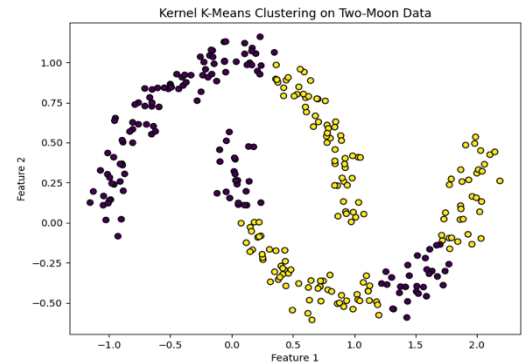
- Our solutions



K-Means

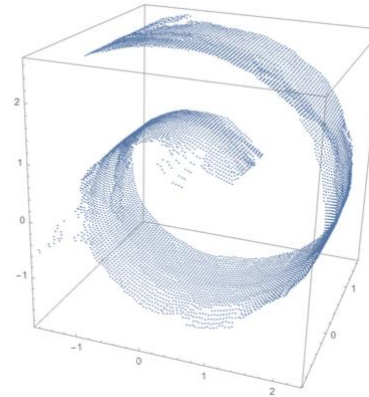
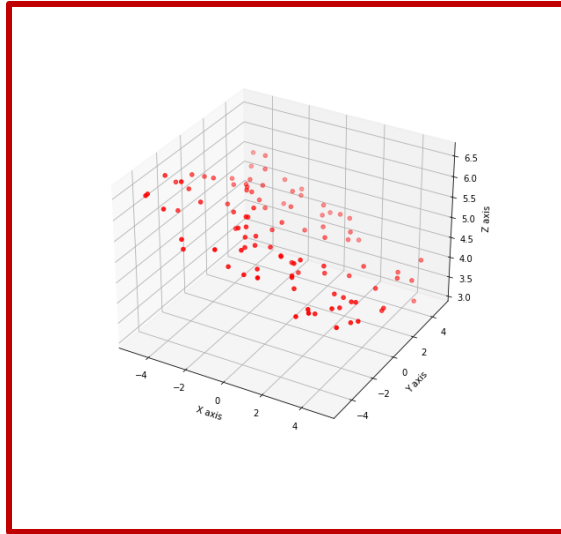


Gaussian Mixture Models



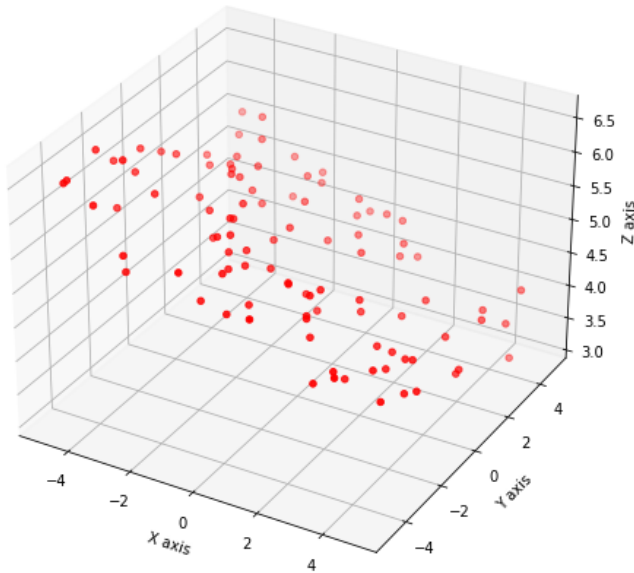
Kernel k-means? ³

Recap: Dimension Reduction



- Finding a low-dimensional subspace to represent the data by learning an “orthogonal bases” of the subspace.

Recap: 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

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

- Come up with a loss function to minimize?

$$\min_{\mu, v_1, \dots, v_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \min_{\alpha_1, \dots, \alpha_k} \left\| \boldsymbol{x}_i - \left(\mu + \sum_{j=1}^k v_j \alpha_j \right) \right\|^2$$

Today

- Principle Component Analysis
- And variants

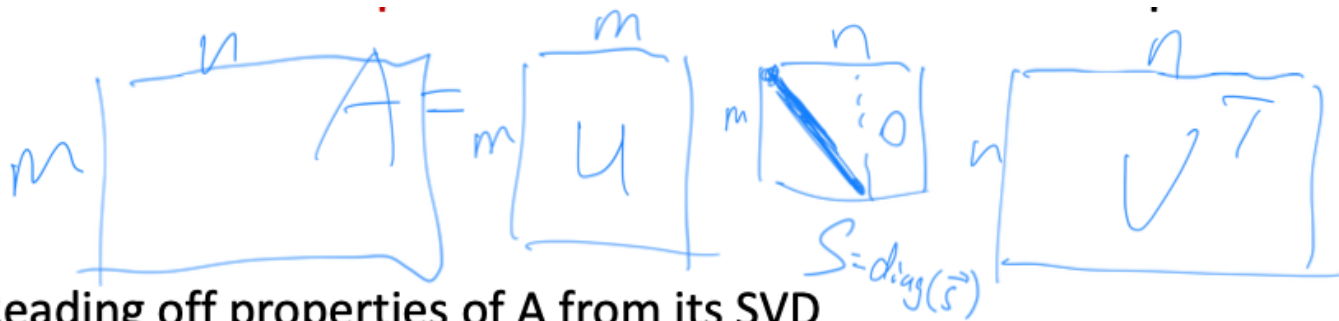
Variants of this problem and an optimization problem that minimizes the reconstruction error

$$\min_{\mu, v_1, \dots, v_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \min_{\alpha_1, \dots, \alpha_k} \left\| x_i - \left(\mu + \sum_{j=1}^k v_j \alpha_j \right) \right\|^2$$

- Best feature selection

- Best low-dimensional approximation

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) = $\sum_{i=1}^n s_i$

- Det(A) = $\prod_i s_i$

- Eigenvalue = s_1, s_2, \dots, s_n
for real symmetric matrix

- Eigenvectors = columns of U

- Matrix norm(A)

$$\|A\|_2 = \sqrt{\sum_i s_i^2}$$

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

$$A = U \Lambda U^T$$

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

Square matrix

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:

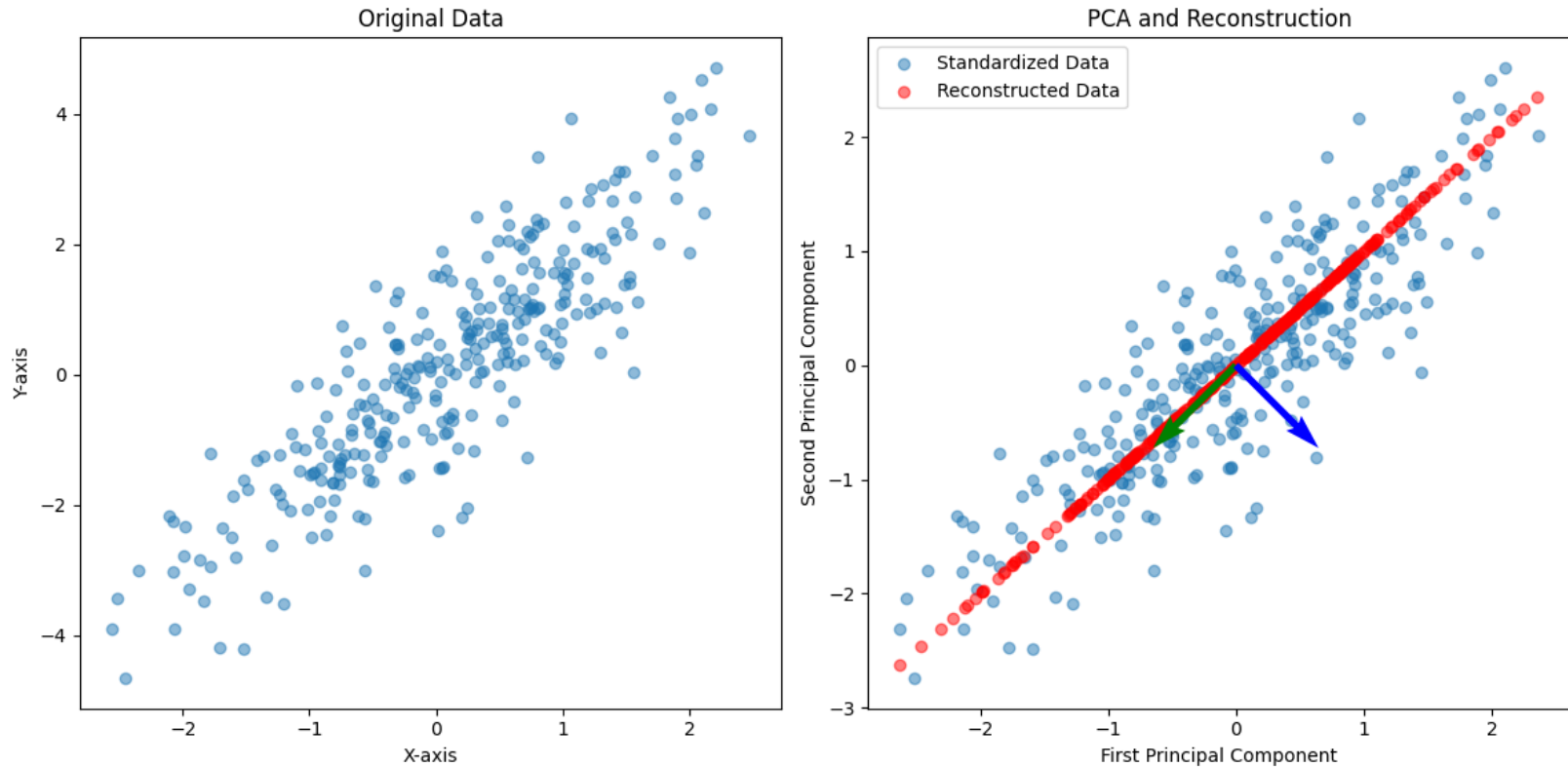
* SVD solves the best-low-rank approximation of the sample covariance matrix. PCA hopes to explain

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

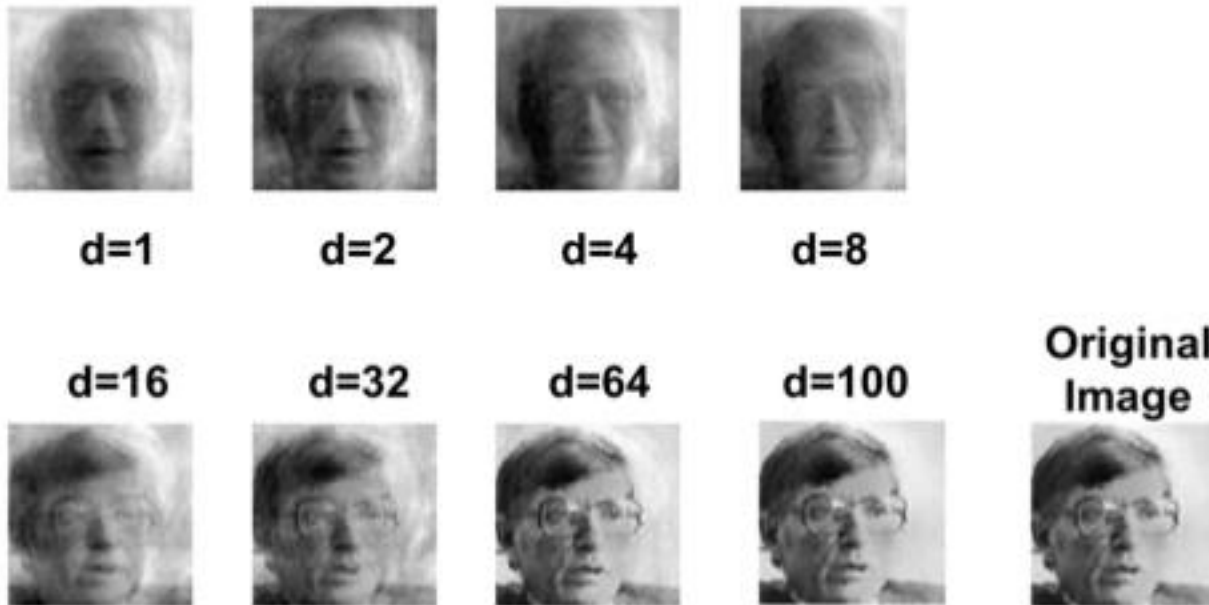
Principal Component Analysis

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



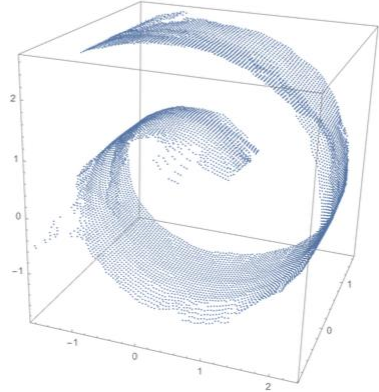
Applications of PCA to image compression



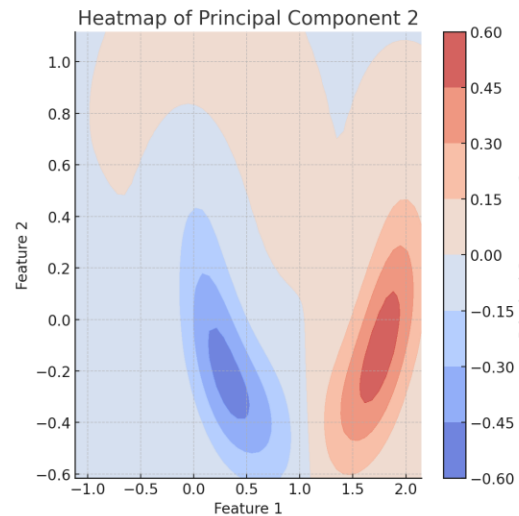
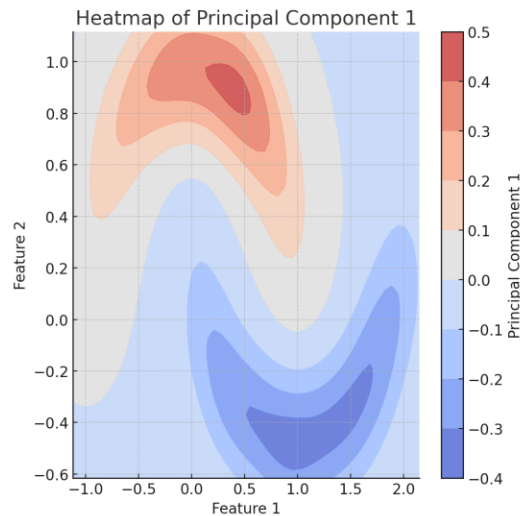
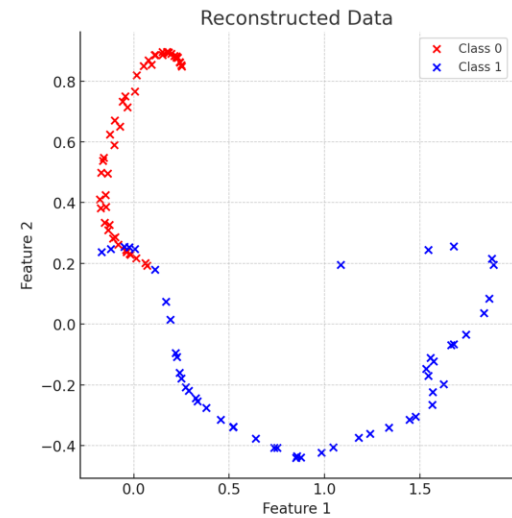
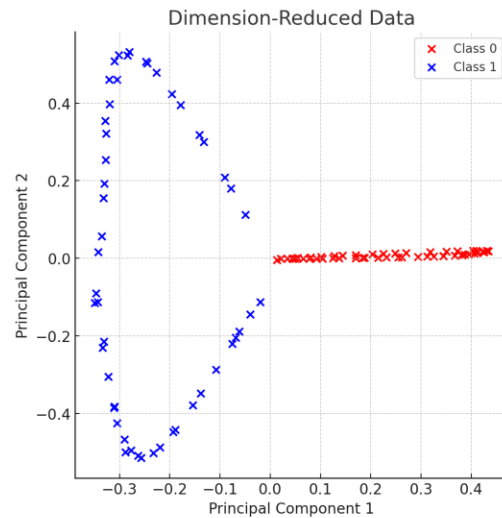
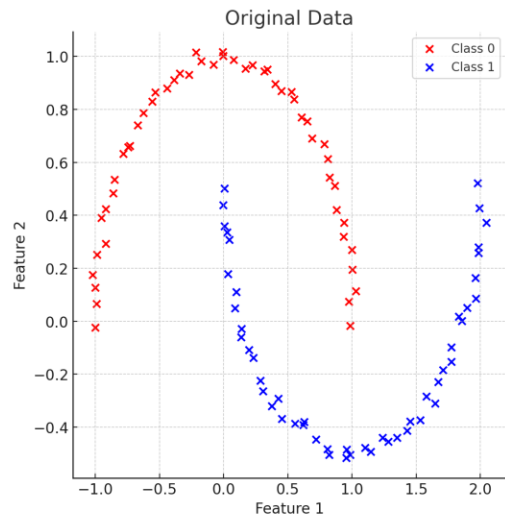
You will learn how to do this in HW4 Q3!

Going non-linear dimension reduction

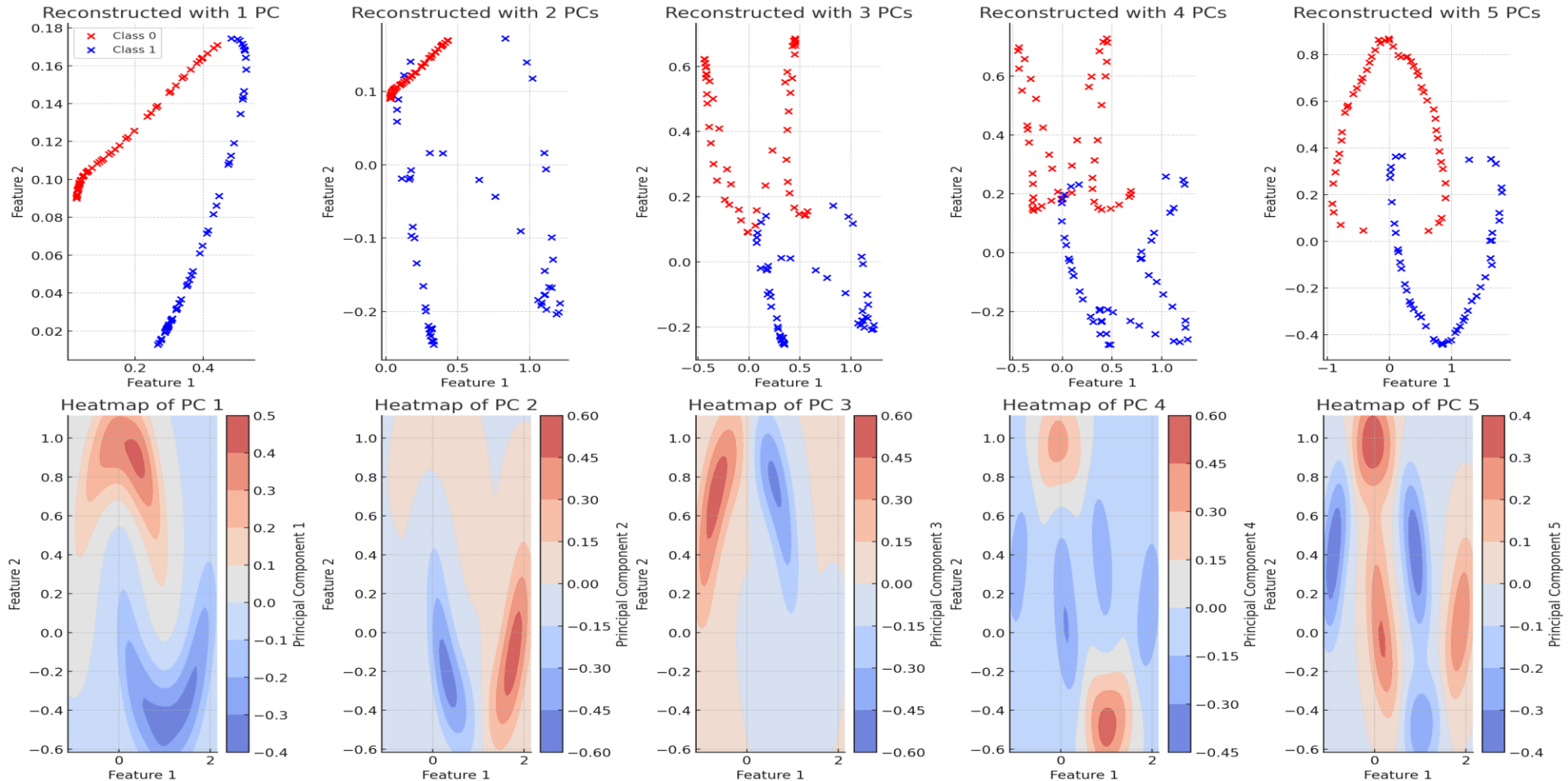
- Mixture of linear subspaces:
 - Subspace clustering
 - Mixture of probabilistic PCAs
 - A combination
- Kernel PCA
 - Run PCA on the Gram matrix instead of the covariance matrix
- Laplace Isomapx, Laplacian Eigenmaps
 - First construct a nearest neighbor graph
 - Then run SVD on the Laplacian matrix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



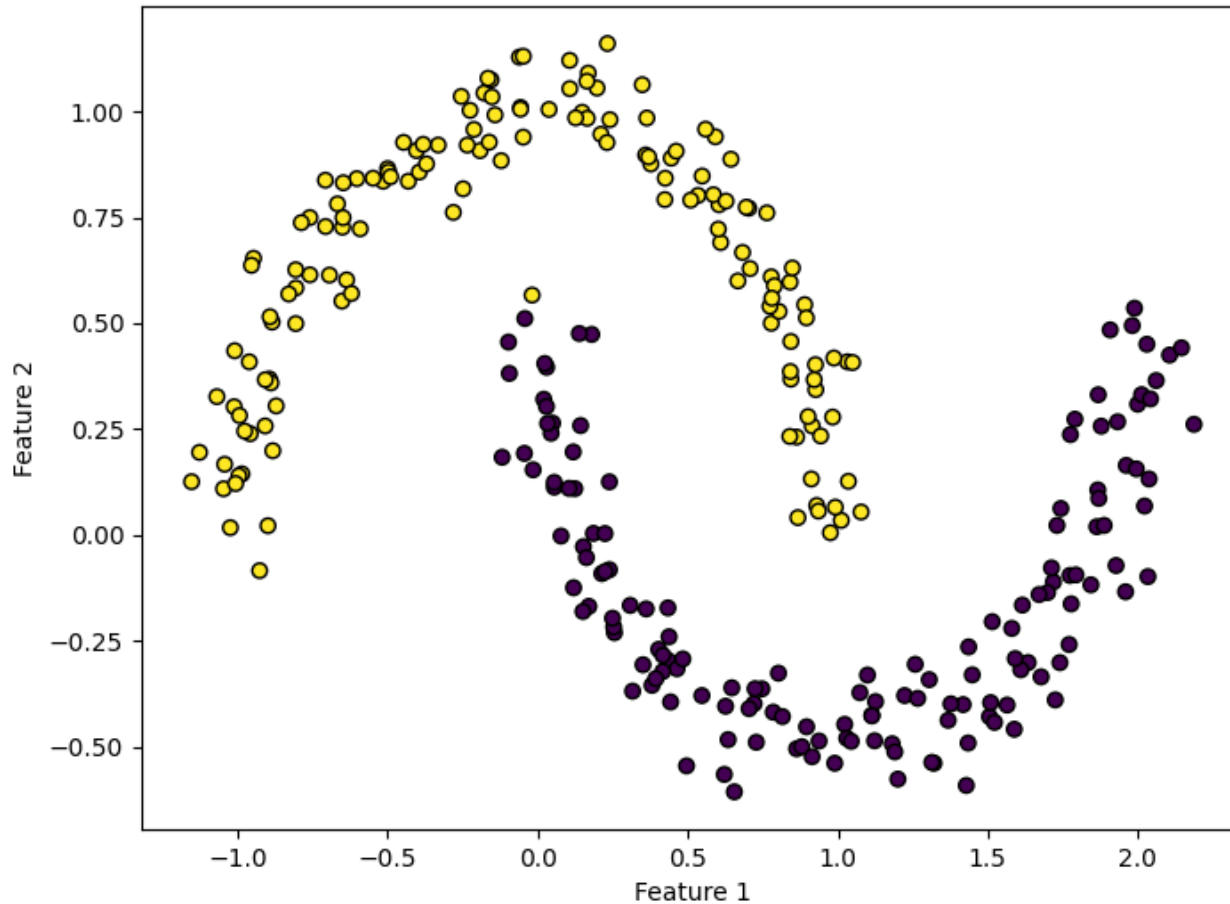
Kernel PCA on the two-moon example



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?

Recap: Optimization problems to solve

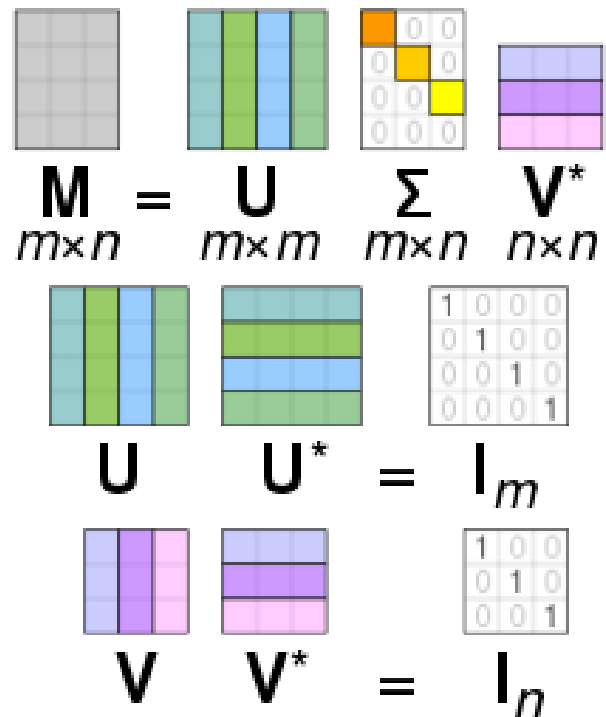
- K-means:
$$\min_{\mu_1, \dots, \mu_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \min_{j \in [k]} \|x_i - \mu_j\|^2$$

- Principal Component Analysis:

$$\min_{\mu, v_1, \dots, v_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \min_{\alpha_1, \dots, \alpha_k} \left\| x_i - \left(\mu + \sum_{j=1}^k v_j \alpha_j \right) \right\|^2$$

What are the similarities and differences of these two problems?

Magical algorithm from Numerical Linear Algebra --- Singular Value Decomposition



Recap: Principal Component Analysis

Input: data matrix X , number of principal components k

1. Calculate the mean $\hat{\mu}$ of the rows of X by $\hat{\mu} = \frac{1}{n} \sum_i x_i$
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 the data by focusing on the most prominent directions of the variance in the data distribution.

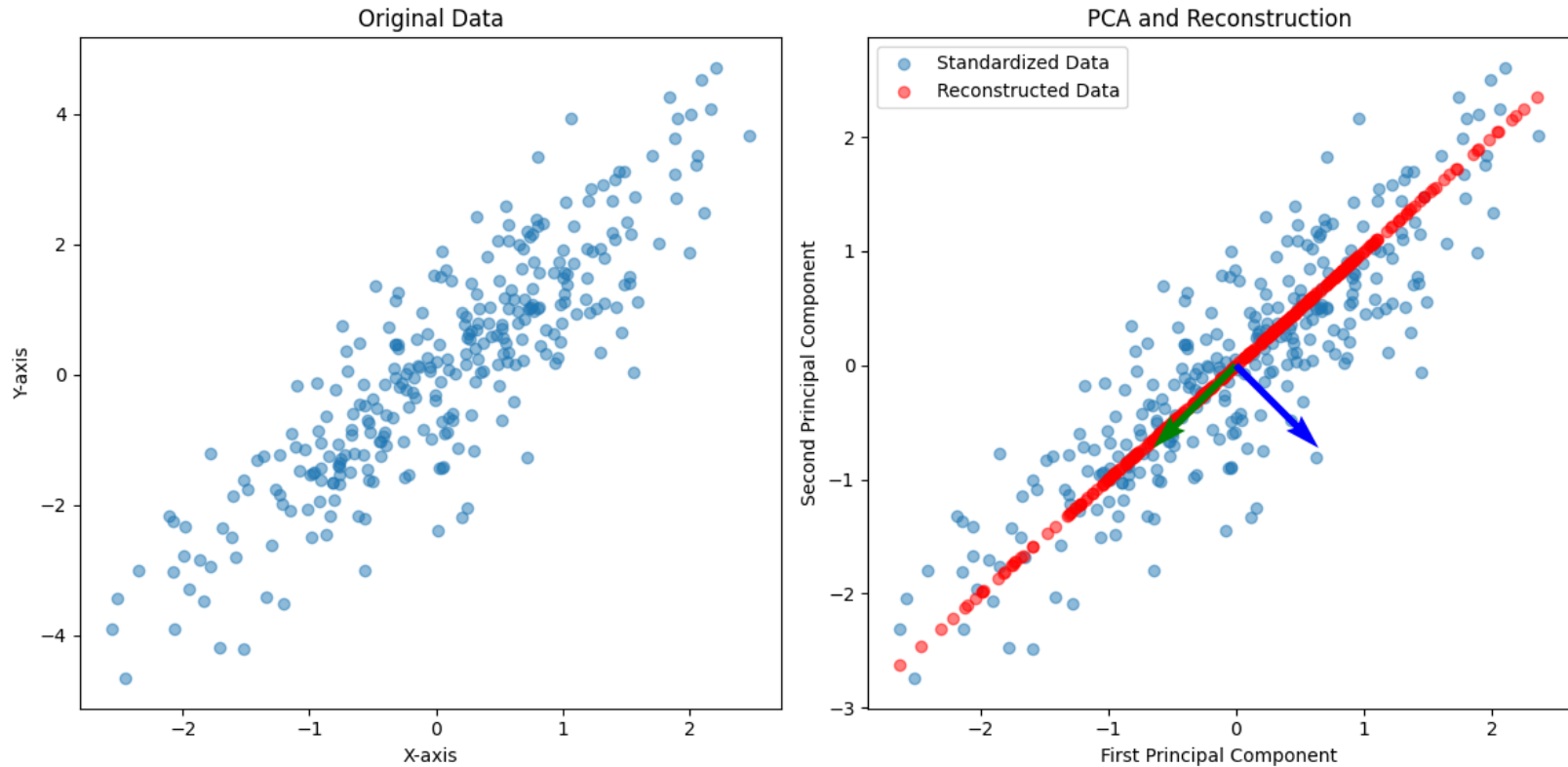
** Improved computational efficiency when k is small: replace full-SVD with skinny SVD that takes k as an input. (Use [scipy.sparse.linalg.svds](#))

Principal Component Analysis

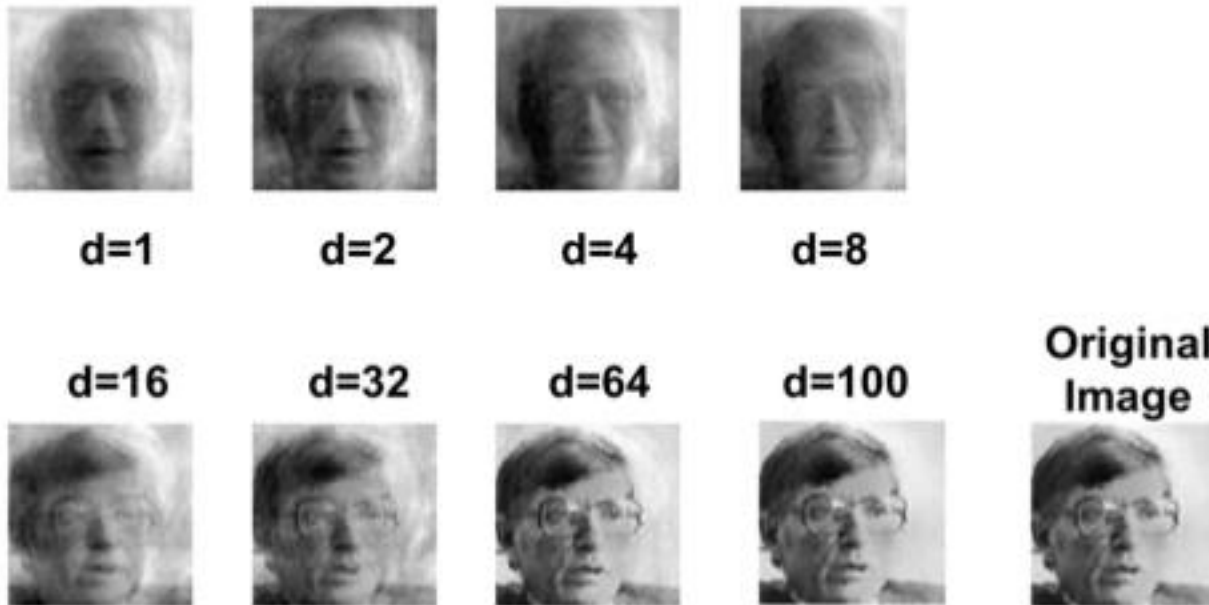
$$\min_{\mu, v_1, \dots, v_k \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \min_{\alpha_1, \dots, \alpha_k} \left\| x_i - \left(\mu + \sum_{j=1}^k v_j \alpha_j \right) \right\|^2$$

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



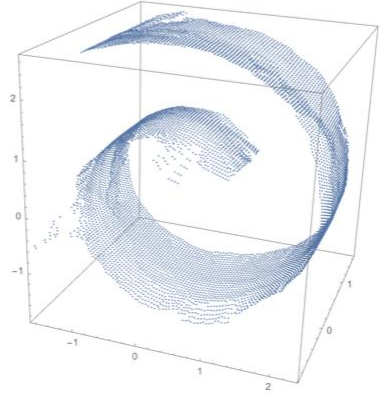
Applications of PCA to image compression



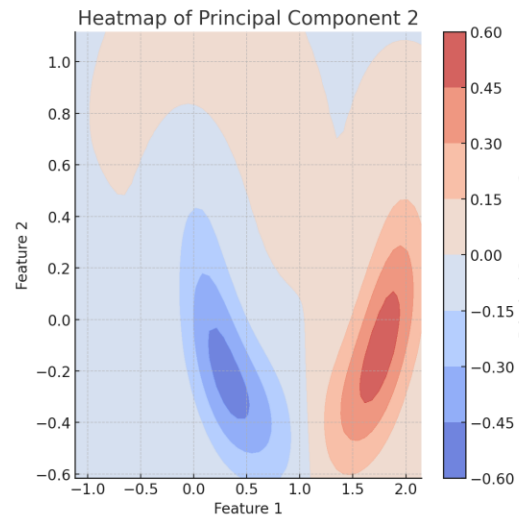
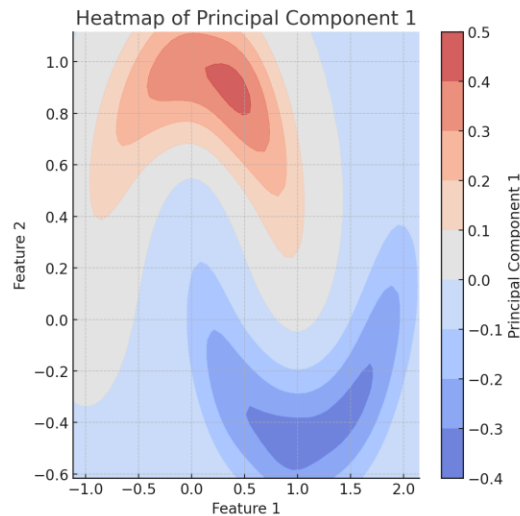
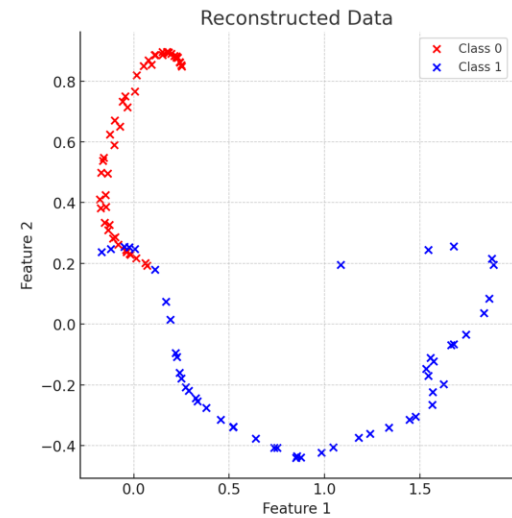
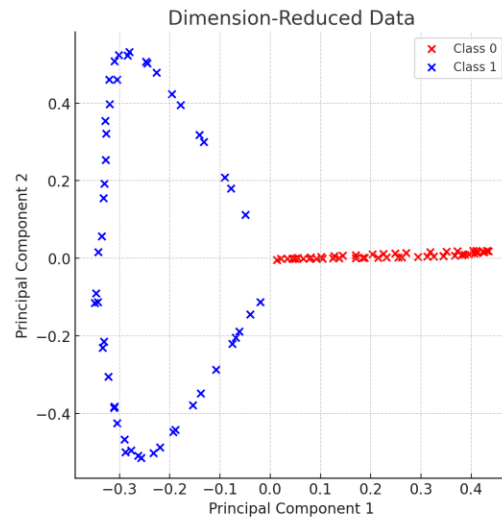
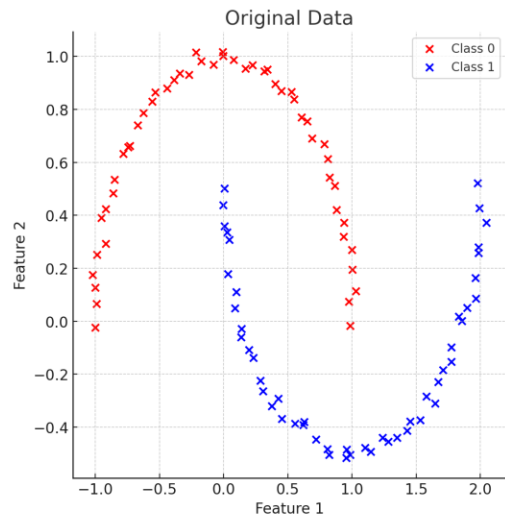
You will learn how to do this in HW4 Q3!

Going non-linear dimension reduction

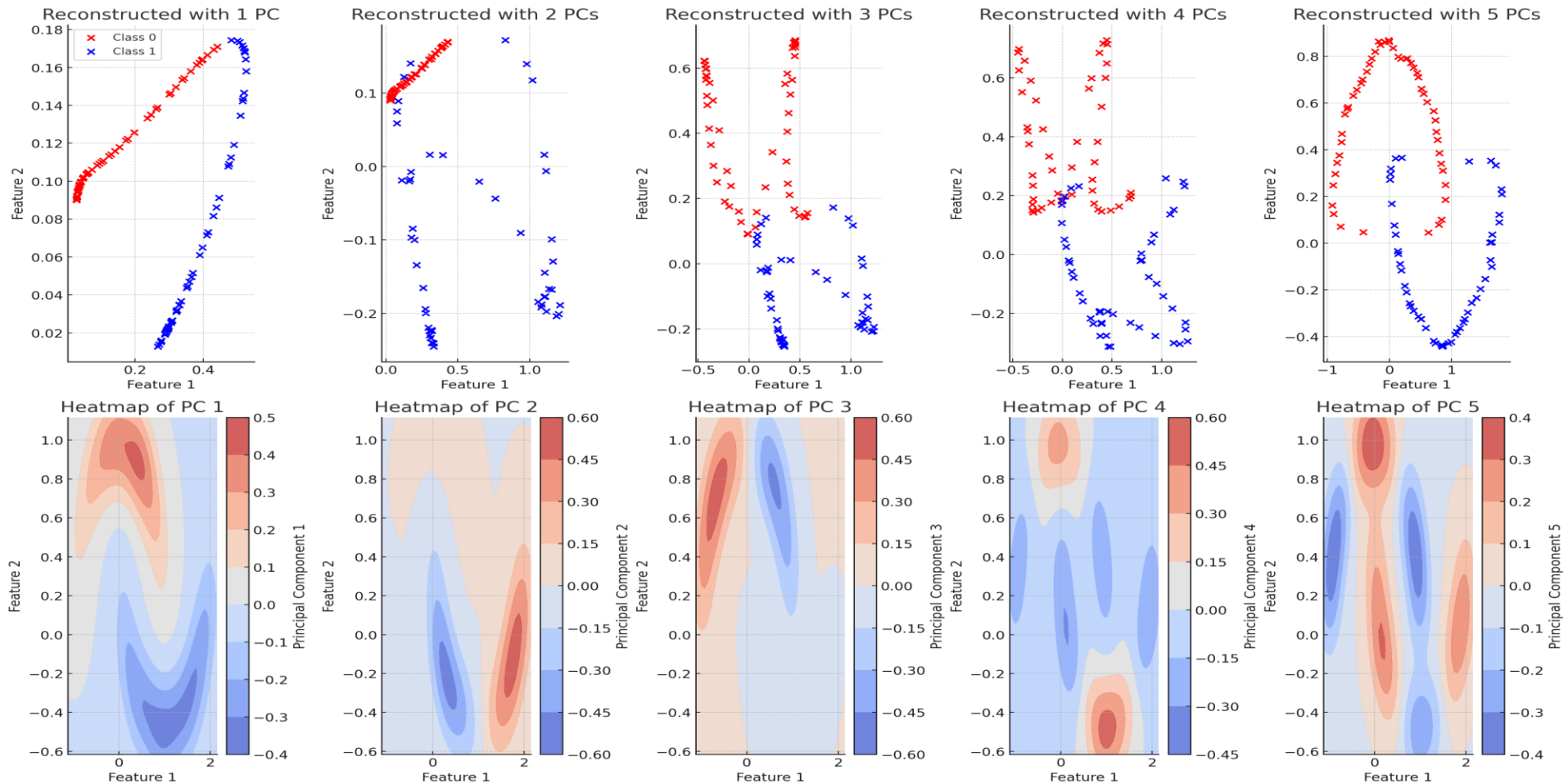
- Mixture of linear subspaces:
 - Subspace clustering
 - Mixture of probabilistic PCAs
 - A combination clustering and dim-reduction
- Kernel PCA
 - Run PCA on the kernel matrix instead of the covariance matrix
- Laplacian Eigenmaps (also the related Isomap)
 - First construct a nearest neighbor graph
 - Then run SVD on the Laplacian matrix of the graph
- Neural approaches:
 - Autoencoders / variational autoencoders
 - Transformers (for data-reconstruction)



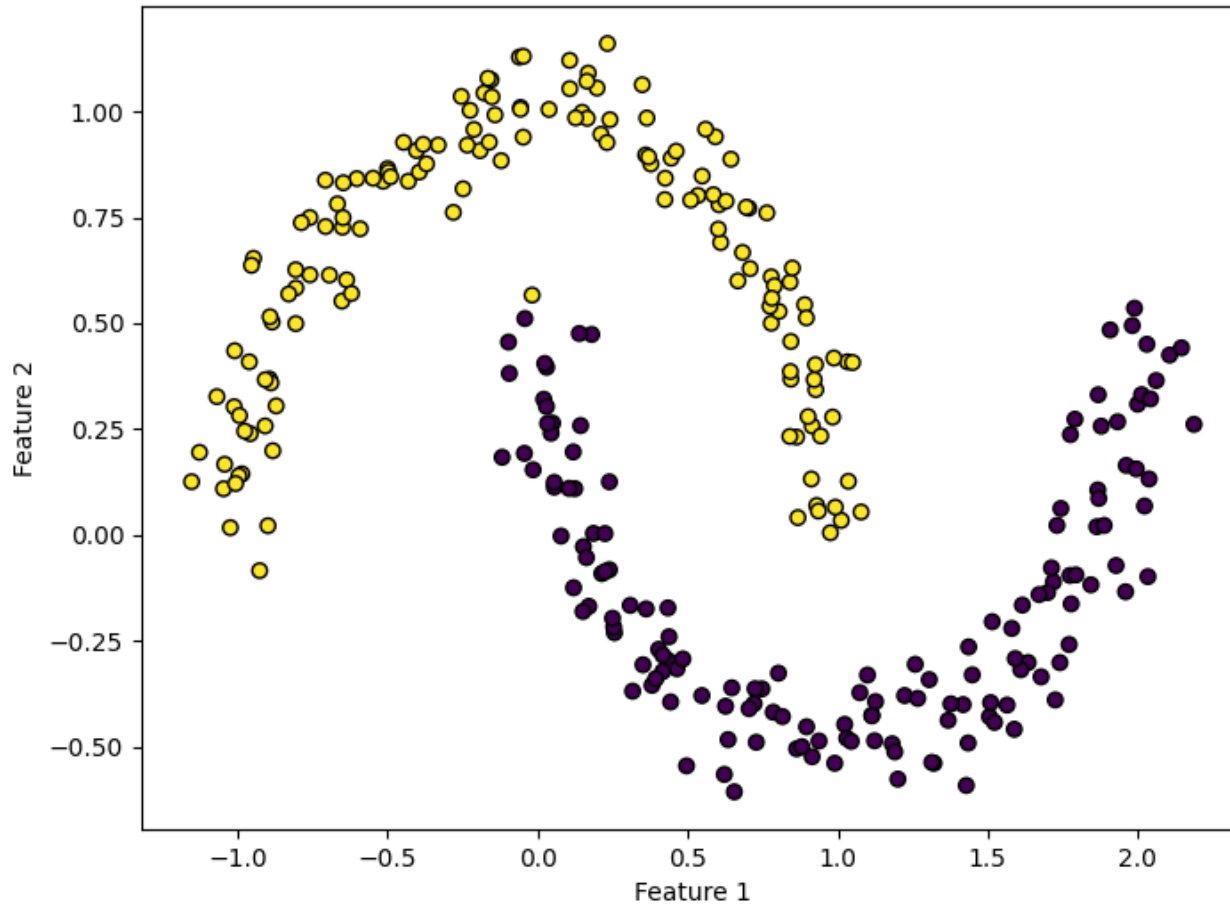
Kernel PCA on the two-moon example



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?