STA 314: Statistical Methods for Machine Learning I

Lecture 12 - Unsupervised Learning: K-means clustering and PCA

Xin Bing

Department of Statistical Sciences
University of Toronto

Unsupervised learning

- Unsupervised learning is the study of learning without labels. What can we do without labels?
- How can we even define what learning means without labels?
- In some sense, the ML community does not exactly agree on what it means to do unsupervised learning, but intuitively, unsupervised learning is the task of
  - grouping (clustering)
  - explaining
  - finding structured data

Motivating Examples

Some examples of situations where you'd use unupservised learning

1. You want to understand how a scientific field has changed over time. You want to take a large database of papers and model how the distribution of topics changes from year to year. But what are the topics?

2. You're a biologist studying animal behavior, so you want to infer a high-level description of their behavior from video. You don't know the set of behaviors ahead of time.

3. You want to reduce your energy consumption, so you take a time series of your energy consumption over time, and try to break it down into separate components (refrigerator, washing machine, etc.).

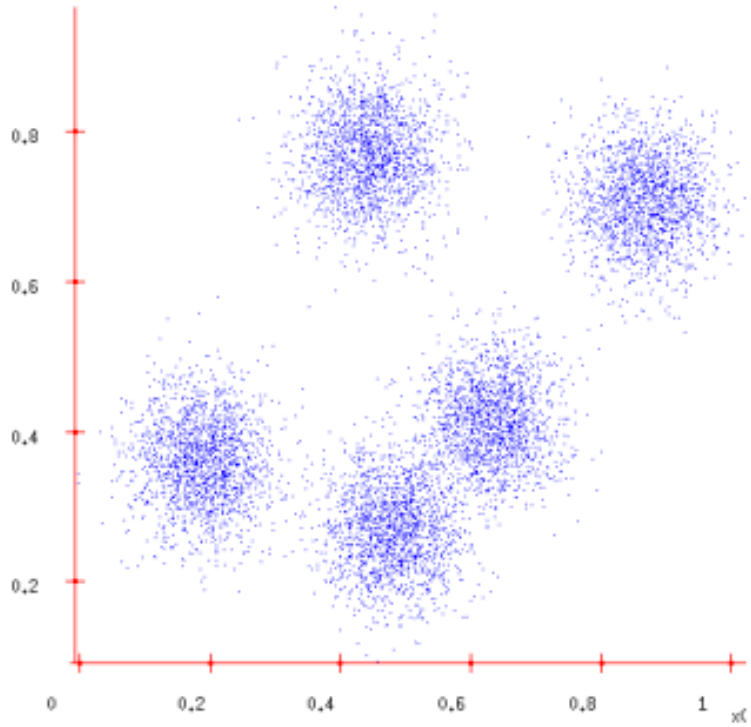
Two major unsupervised learning problems

Common themes: you have some data, and you want to infer some structure underlying the data.

- Clustering
- Low-dimensional representation

Clustering

- Sometimes the data form clusters, where samples within a cluster are similar to each other, and samples in different clusters are dissimilar:

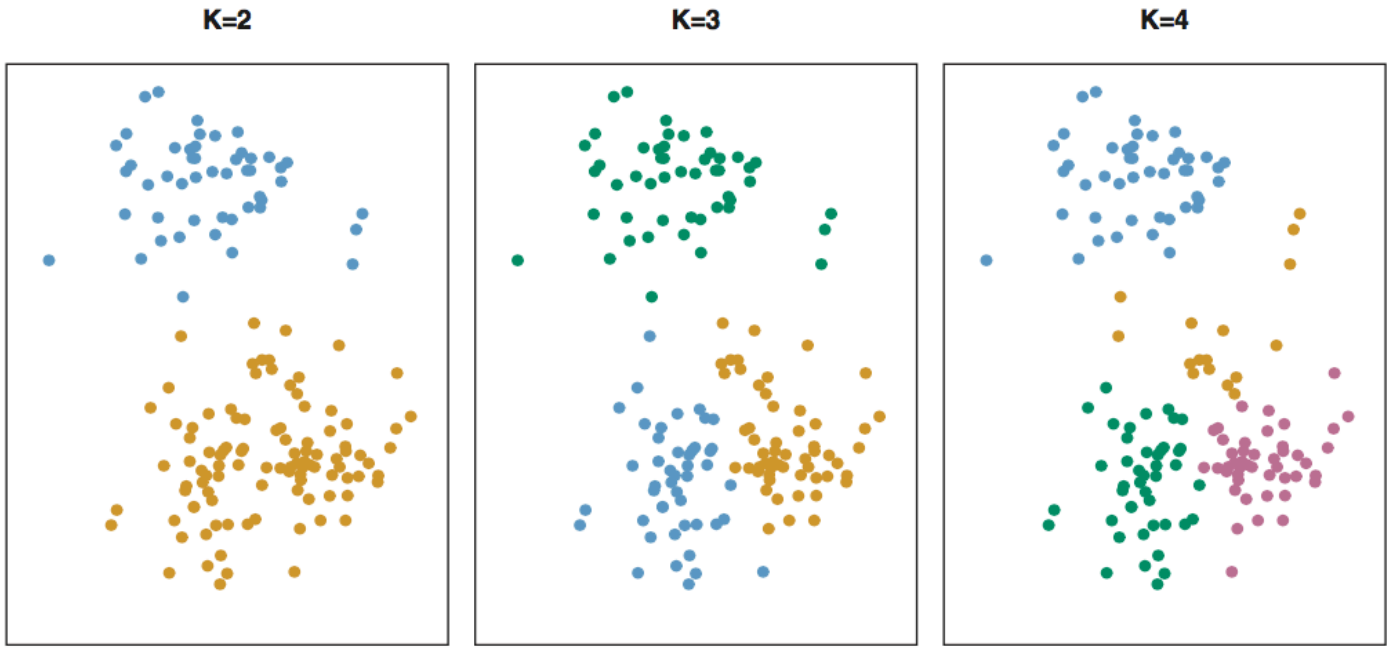


- Such a distribution is multimodal, since it has multiple modes, or regions of high probability mass.

Clustering

- Grouping data points into clusters, with no observed labels, is called clustering. E.g.
  - Clustering machine learning papers based on topic (deep learning, Bayesian models, reinforcement learning, etc.)
  - But topics are never observed (unsupervised).
- We will study the most popular clustering algorithm: K-means.

Example of results from K-means



- A simulated data set with 150 observations in two-dimensional space. Panels show the results of applying K-means clustering with different values of K.
- The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm.
- There is no ordering of the clusters, so the cluster coloring is arbitrary.

K-Means Clustering

K-means clustering is a simple approach for partitioning a data set into K distinct, non-overlapping clusters.

Let $C_1, \ldots, C_K$ denote sets that form a partition of $\{1, \ldots, n\}$:

1. Each observation belongs to at least one of the K clusters.
2. No observation belongs to more than one cluster.

$$C_k \cap C_{k'} = \emptyset, \qquad \text{for all } k \neq k'.$$

Hence every single observation belongs to one and only one cluster. The goal is to find such partition $C_1, \ldots, C_K$ for certain purpose.

K-Means Clustering

- The idea behind K-means clustering is that a good clustering ensures the within-cluster variation as small as possible.
- The within-cluster variation for cluster C_k is a measure $W(C_k)$ on the difference among observations within a cluster.
- We aim to find sets $C_1, \ldots, C_K$ by solving

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k).$$

In words, we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

Definition of the within-cluster variation

- It is common to use the Euclidean distance

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

$$= \frac{1}{|C_k|} \sum_{i,i' \in C_k} \|\mathbf{x}_i - \mathbf{x}_{i'}\|_2^2$$

  where $|C_k|$ denotes the number of observations in the kth cluster.

- Let

$$\bar{\mathbf{x}}_k = \frac{1}{|C_k|} \sum_{i \in C_k} \mathbf{x}_i.$$

  Verify that

$$W(C_k) = \frac{1}{|C_k|} \sum_{i \in C_k} \|\mathbf{x}_i - \bar{\mathbf{x}}_k\|_2^2.$$

Target of the K-means algorithm

- Thus, K-means aims to solve

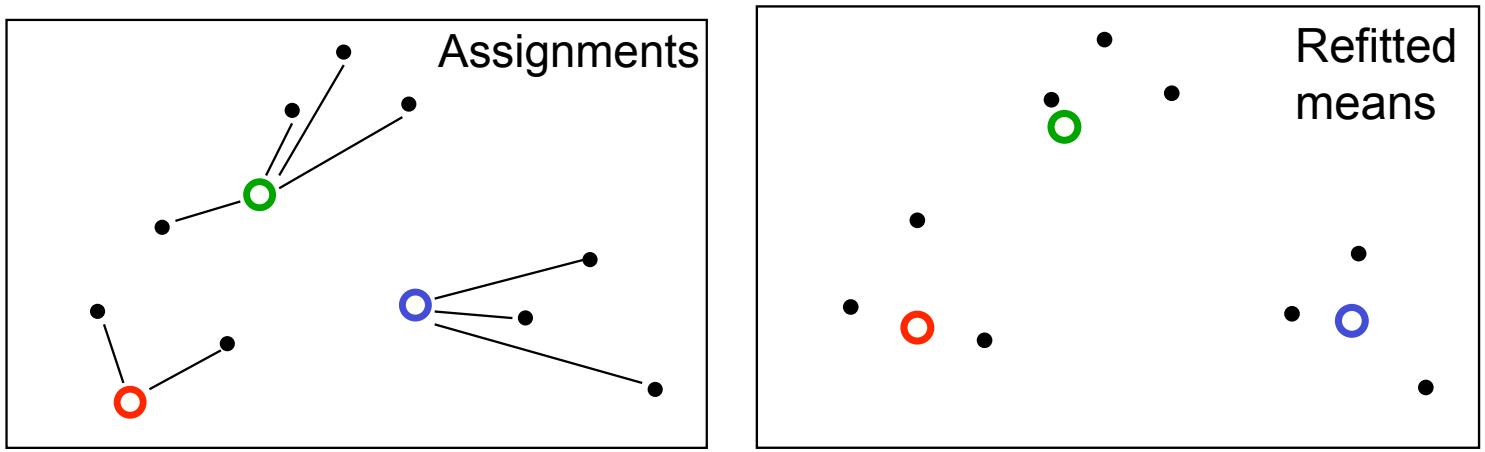
$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k) \;=\; \min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i \in C_k} \|\mathbf{x}_i - \bar{\mathbf{x}}_k\|_2^2.$$

- This is, however, a very difficult problem to solve exactly. There are almost K^n ways to partition n observations into K clusters.

A pratical alternating algorithm

High level overview of algorithm:

- Initialization: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - Assignment step: Assign each data point to the closest cluster
  - Re-center step: Move each cluster center to the mean of the data assigned to it



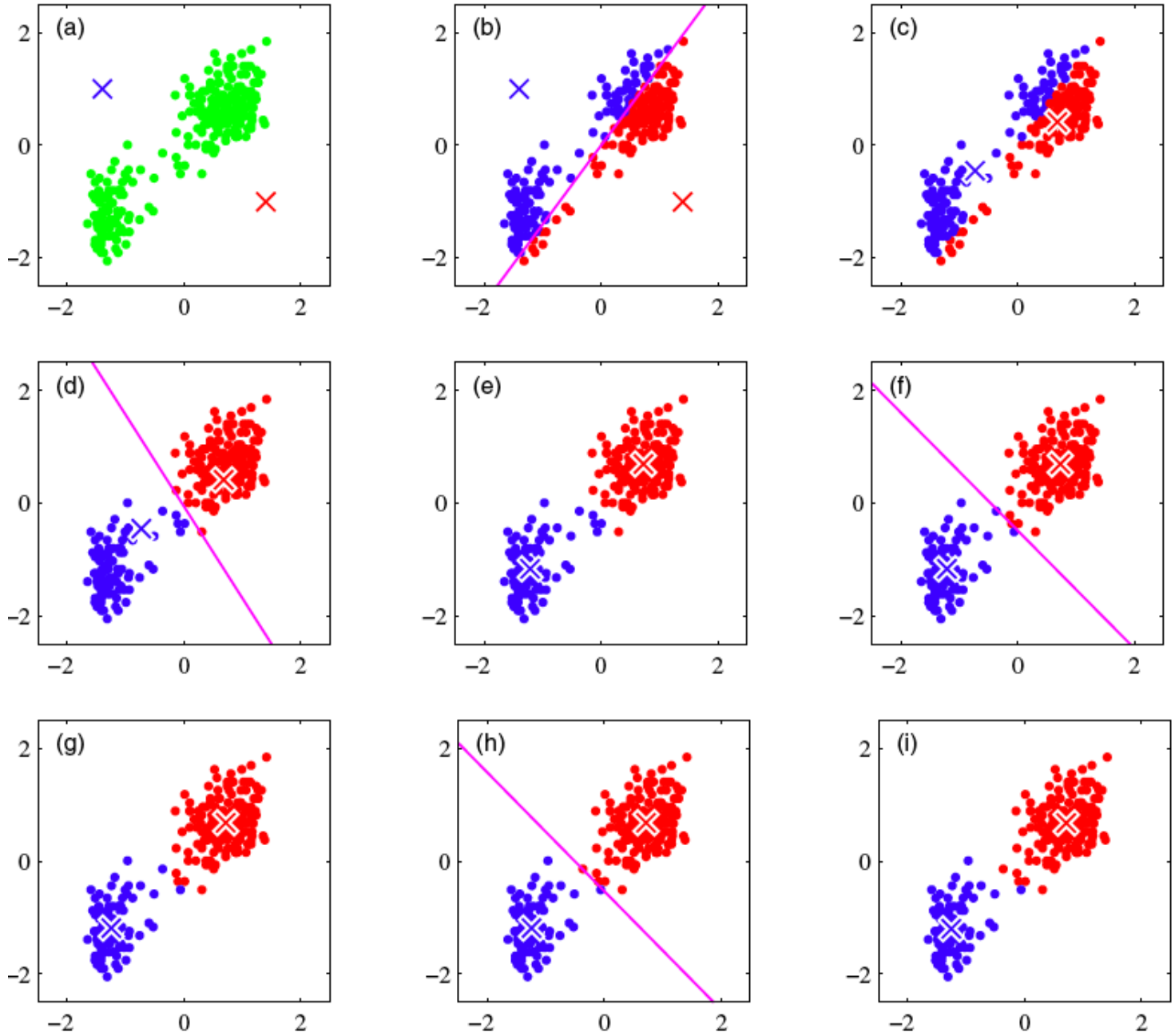


Figure from Bishop

Animation of the algorithm: `http://shabal.in/visuals/kmeans/5.html`.

K-means algorithm

Algorithm 10.1 *K-Means Clustering*

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.

2. Iterate until the cluster assignments stop changing:

   (a) For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.

   (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

Why K-means Converges

- K-means algorithm reduces the cost at each iteration.

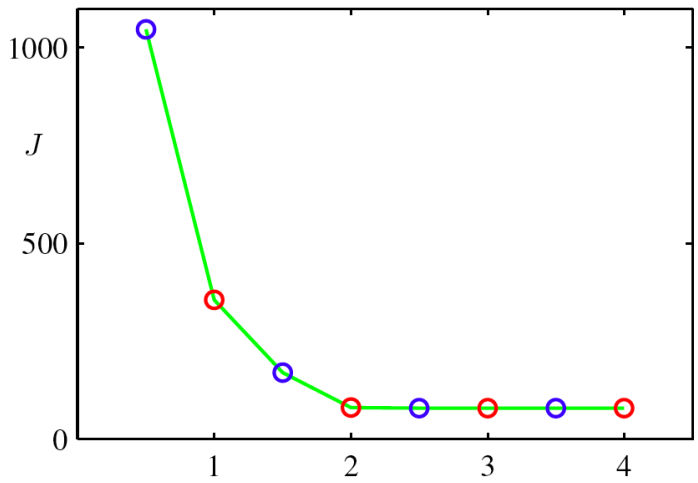
$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i \in C_k} \|\mathbf{x}_i - \bar{\mathbf{x}}_k\|_2^2.$$

  - Assignment step: fixing the centers, re-assignment will decrease the total within-cluster variation.
  - Re-centering step: fixing the assignments, re-centering the data within clusters will reduce the total within-cluster variation.

- Stopping criterion for convergence: when the assignments do not change in the assignment step, we have converged (to at least a local minimum).

Convergence of the K-means algorithm

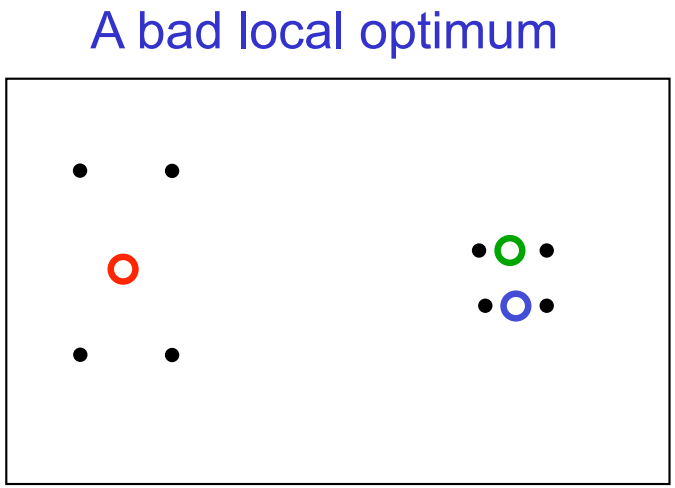
- Convergence will always happen after a finite number of iterations, since the number of possible cluster assignments is finite



- K-means cost function after each assignment step (blue) and refitting step (red). The algorithm has converged after the third refitting step.

Local Minima

- The cost function is non-convex (so convergence is not equivalent to the global minimum)
- There is nothing to prevent K-means getting stuck at local minima.
- Possible remedy: could try many random starting points

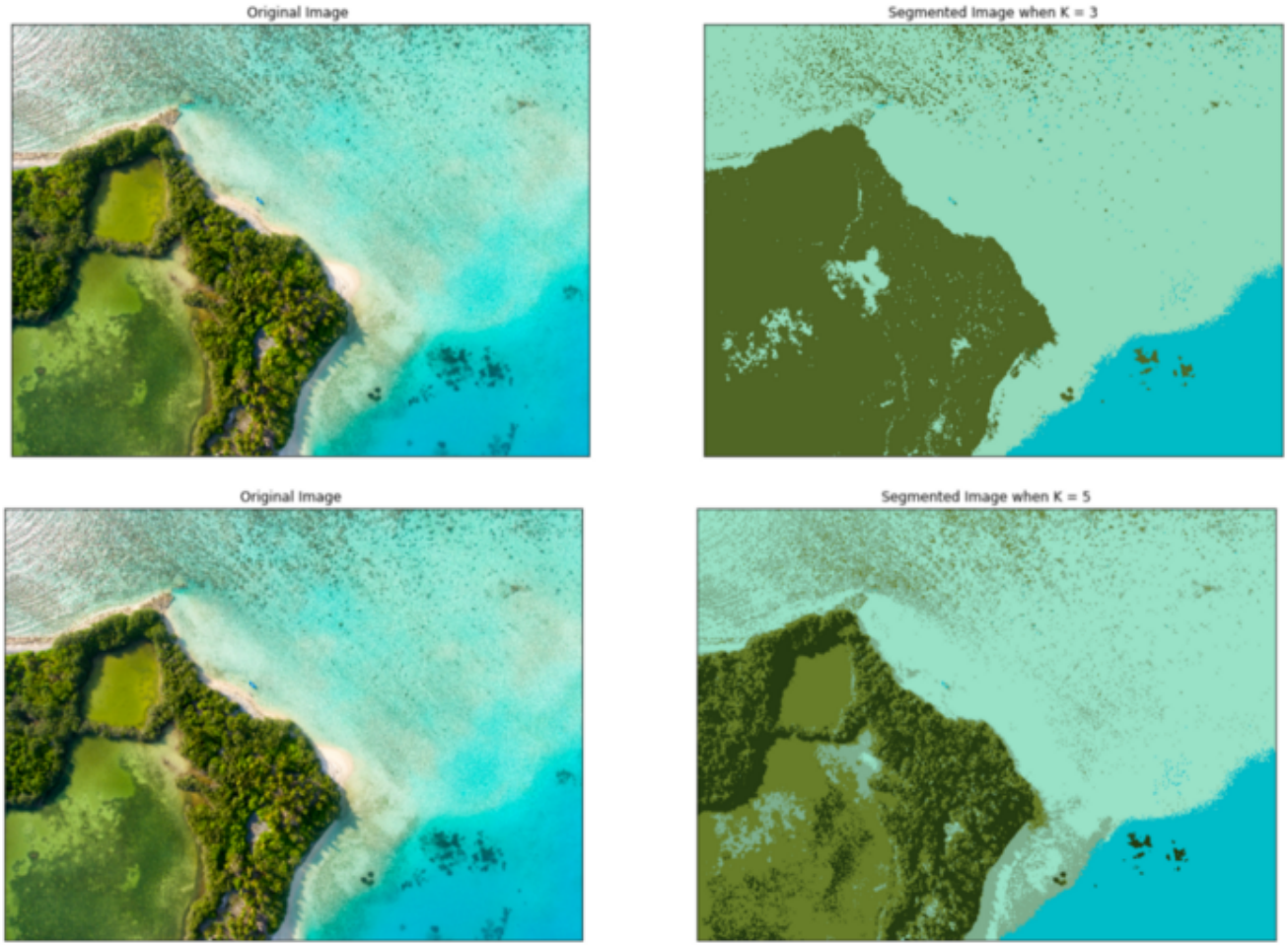


K-means for Vector Quantization

Figure from Bishop

- Given image, construct "dataset" of pixels represented by their RGB pixel intensities
- Run K-means, replace each pixel by its cluster center

K-means for Image Segmentation



- Given image, construct "dataset" of pixels, represented by their HSV pixel intensities
- Run K-means to get superpixels

Extensions

- Non-exhaustive clustering. Allow some of the data points not to belong to any cluster.
- Overlapping clustering. Allow some of the data points to belong to more than one clusters.
  - Soft K-means
- Clustering features rather than data points. Previously, we consider the clustering for data points.

PCA Overview

- We now turn to the second unsupervised learning algorithm for this course: **principal component analysis** (PCA)
- PCA is used for dimensionality reduction: map data to a lower dimensional space
- PCA finds linear low-dimensional representations of the data by preserving as much variation (in the original data) as possible.
- PCA is useful for understanding lots of other algorithms.
  - Autoencoders
  - Matrix factorizations

Low dimensional representation

- In practice, even though data is very high dimensional, its important features can be accurately captured in a low dimensional subspace.

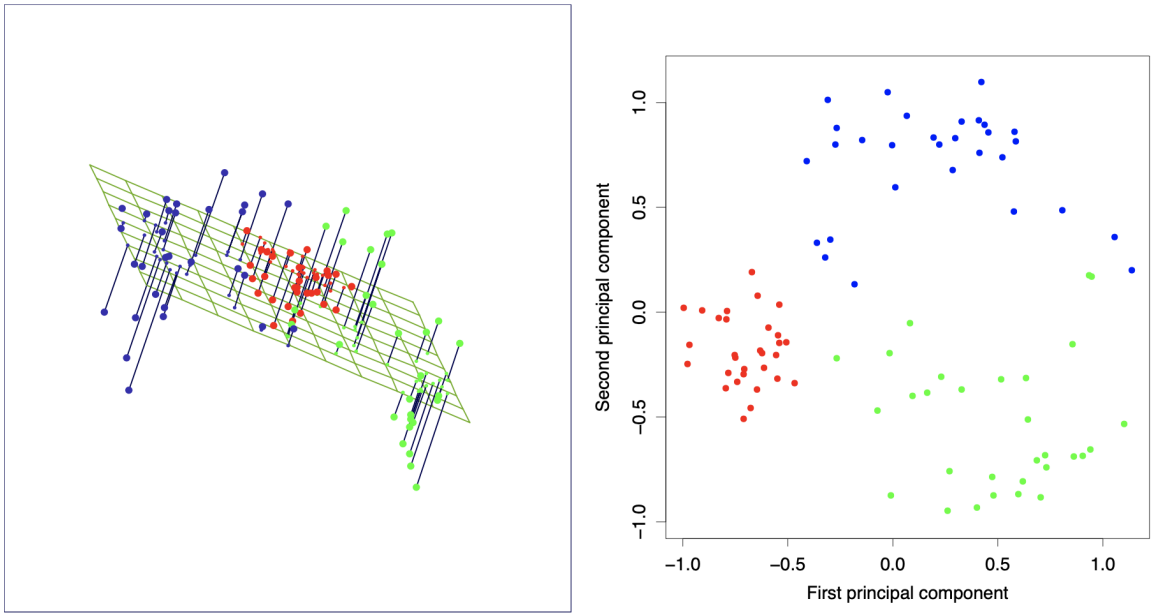


Image credit: Elements of Statistical Learning

- Find a low dimensional representation of your data.
  - Computational benefits
  - Interpretability, visualization
  - Generalization

Principal Components Analysis

- PCA constructs linear combinations of features $X_1, X_2, \ldots, X_p$.
- The first principal component(PC) is the linear combination of the features

$$Z_1 = u_{11}X_1 + u_{21}X_2 + \cdots + u_{p1}X_p$$

where the coefficients

$$\mathbf{u}_1 = \begin{bmatrix} u_{11} \\ u_{21} \\ \vdots \\ u_{p1} \end{bmatrix}$$

are chosen such that

  - Z_1 has the largest variance;
  - $\mathbf{u}_1$ is normalized such that $\sum_{j=1}^{p} u_{j1}^2 = 1$, i.e. $\|\mathbf{u}_1\|_2 = 1$.

- We refer to the coefficients $\mathbf{u}_1$ as the **loading** of the first PC.

More Principal Components

- The second PC is again the linear combination of $X_1, \ldots, X_p$

$$Z_2 = u_{12}X_1 + u_{22}X_2 + \cdots + u_{p2}X_p$$

where the loading

$$\mathbf{u}_2 = \begin{bmatrix} u_{12} \\ u_{22} \\ \vdots \\ u_{p2} \end{bmatrix}$$

are chosen such that

  - Z_2 has the largest variance,
  - $\|\mathbf{u}_2\|_2 = 1$,
  - $\mathbf{u}_2^\top \mathbf{u}_1 = 0$. This implies Z_2 is uncorrelated with Z_1.

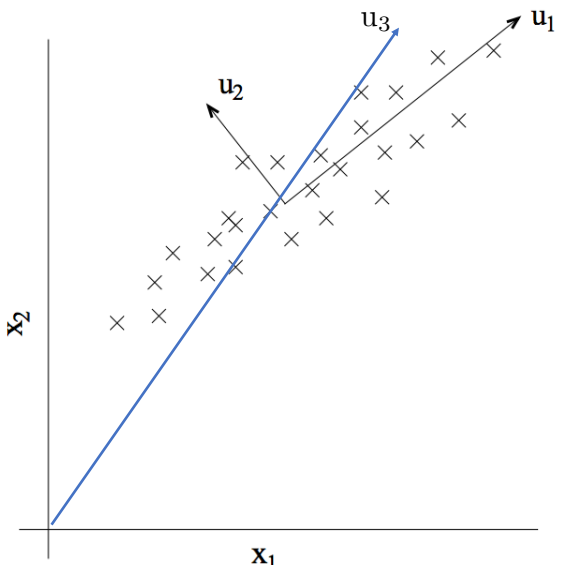
- This successively defines the first K PCs, $Z_1, \ldots, Z_K$, with corresponding loadings $\mathbf{u}_1, \ldots, \mathbf{u}_K$.

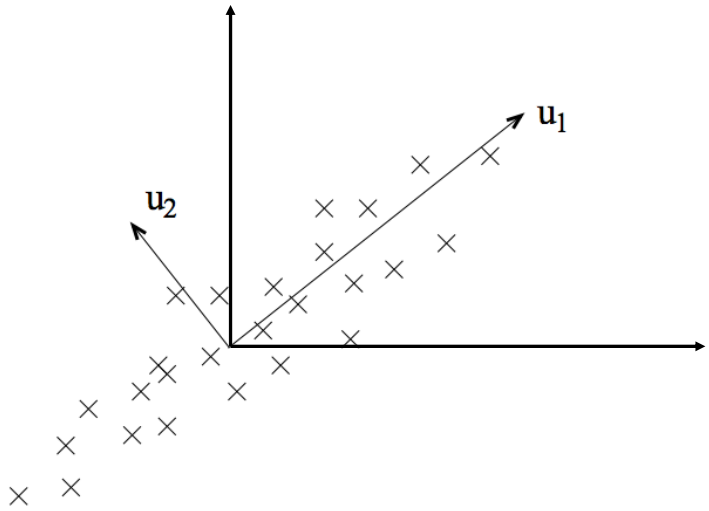
Computation

Suppose we have a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ and we want to construct K PCs with $K \in \{1, 2, \ldots, p\}$.

1. Center $\mathbf{X}$ such that the columns have zero mean, that is,
$$\tilde{\mathbf{X}} = \mathbf{X} - 1_n \bar{\mathbf{X}}^\top.$$
2. Compute the first K loadings
$$\mathbf{U}_K = (\mathbf{u}_1, \ldots, \mathbf{u}_K)$$
from the centered data, $\tilde{\mathbf{X}}$.
3. Obtain the first K PCs
$$\tilde{\mathbf{Z}} = \tilde{\mathbf{X}}\mathbf{U}_K \in \mathbb{R}^{n \times K}.$$
4. Add the centers back to the PCs
$$\mathbf{Z} = \tilde{\mathbf{Z}} + 1_n \bar{\mathbf{X}}^\top \mathbf{U}_K = (\tilde{\mathbf{X}} + 1_n \bar{\mathbf{X}}^\top)\mathbf{U}_K = \mathbf{X}\mathbf{U}_K.$$

Centering the data





- Directions we compute will pass through origin, and should represent the direction of the highest variation.
- We need to center our data since we don't want location of data to influence calculation of the loadings. That is, we are **not** interested in $\mathbf{u}_3$.

Computation of the first loading

Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be the centered data.

- The first loading vector $\mathbf{u}_1$ is obtained via the optimization problem

$$\mathbf{u}_1 = \arg\max_{\mathbf{u}} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} u_j x_{ij} \right)^2, \quad \text{subject to} \quad \sum_{j=1}^{p} u_j^2 = 1.$$

- In the matrix notation,

$$\mathbf{u}_1 = \arg\max_{\mathbf{u}} \frac{1}{n} \mathbf{u}^\top \mathbf{X}^\top \mathbf{X} \mathbf{u}, \quad \text{subject to} \quad \mathbf{u}^\top \mathbf{u} = 1.$$

- The problem can be solved via the eigen decomposition of $\hat{\mathbf{\Sigma}} := \mathbf{X}^\top \mathbf{X}/n$, a standard technique in linear algebra. More specifically, $\mathbf{u}_1$ is nothing but the first eigenvector of $\hat{\mathbf{\Sigma}}$.

Computation of the second loading

Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be the centered data and $\mathbf{u}_1$ be the first loading.

- The second loading vector $\mathbf{u}_2$ is obtained via the optimization problem

$$\mathbf{u}_2 = \arg\max_{\mathbf{u}} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} u_j x_{ij} \right)^2, \quad \text{s.t.} \quad \sum_{j=1}^{p} u_j^2 = 1, \sum_{j=1}^{p} u_j u_{j1} = 0.$$

- In the matrix notation,

$$\mathbf{u}_2 = \arg\max_{\mathbf{u}} \frac{1}{n} \mathbf{u}^\top \mathbf{X}^\top \mathbf{X} \mathbf{u}, \quad \text{s.t.} \quad \mathbf{u}^\top \mathbf{u} = 1, \mathbf{u}^\top \mathbf{u}_1 = 0.$$

- $\mathbf{u}_2$ is simply the second eigenvector of $\hat{\mathbf{\Sigma}}$.
- Similarly, $\mathbf{u}_1, \ldots, \mathbf{u}_K$ are the first K eigenvectors of $\hat{\mathbf{\Sigma}}$.

Another Interpretation of PCA

- The first K loadings

$$\mathbf{U} \in \mathbb{R}^{p \times K}$$

are obtained via

$$\arg\max_{\mathbf{U}} \ \frac{1}{n}\mathrm{tr}\left(\mathbf{U}^\top \mathbf{X}^\top \mathbf{X}\mathbf{U}\right), \quad \text{s.t.} \quad \mathbf{U}^\top \mathbf{U} = \mathbf{I}_K.$$

- It can also be obtained by

$$\arg\min_{\mathbf{U}} \ \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top \mathbf{x}_i\|^2, \quad \text{s.t.} \quad \mathbf{U}^\top \mathbf{U} = \mathbf{I}_K.$$

- PCA finds the subspace such that the projected data points in this subspace are closest to the original data points.

Some practical considerations

- In general, in addition to centering, standardizing each variable to have unit standard deviation is recommended.
- Each principal component loading vector is unique, up to a sign flip.
- How many PCs to retain?
  - No simple answer to this question, as cross-validation is not available for this purpose.
  - There are several ad-hoc procedures.

Applying PCA to faces

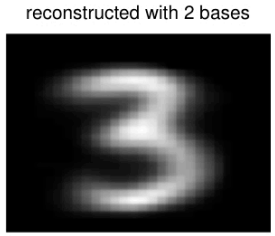
- Consider running PCA on 2429 19x19 grayscale images (CBCL data)
- Can get good reconstructions with only 3 components

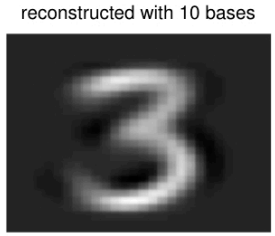
- PCA for pre-processing: can apply classifier to latent representation
  - Original data is 361 dimensional
  - For face recognition PCA with 3 components obtains 79% accuracy on face/non-face discrimination on test data vs. 76.8% for a Gaussian mixture model (GMM) with 84 states. (We'll cover GMMs later in the course.)
- Can also be good for visualization

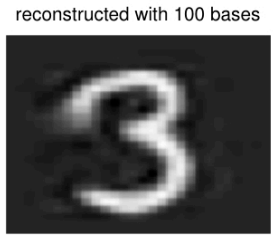
Applying PCA to faces: Learned basis

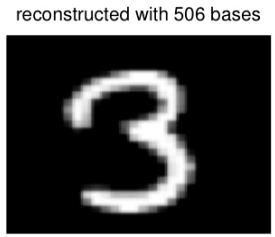
Principal components of face images ("eigenfaces")

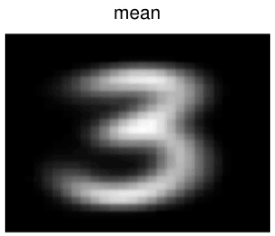
Applying PCA to digits

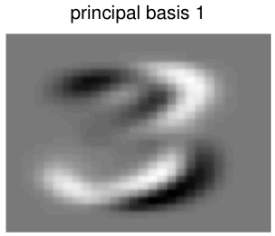


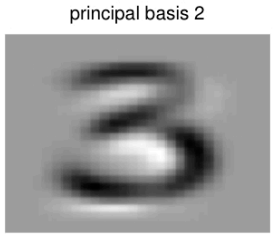


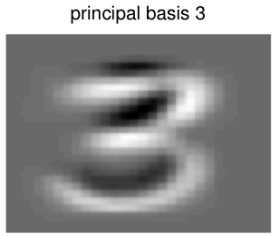












Summary

- Dimensionality reduction aims to find a low-dimensional representation of the data.
- PCA projects the data onto a subspace which maximizes the projected variance, or equivalently, minimizes the reconstruction error.
- The optimal subspace is given by the top eigenvectors of the empirical covariance matrix.
- PCA gives a set of decorrelated features (linear) in the original features.