CS2109s Tutorial 10

by Lee Zong Xun

Recap

Unsupervised learning

- Unlabelled data - No idea what is it for
- Usually for Classification tasks
    - Derive structure by clustering data based relationships among the variables in the data

K-Means Algorithm

- K-Means is a clustering algorithm that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
  - Not **deterministic** due to random initialisation, simply converging to a "good" enough solution
  - Repeat multiple times, pick the clusters with the minimum cost
  - Value of K is **ambiguous**, usually determined by domain knowledge

K-Means algorithm

- Randomly initialize K centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K$
- Repeat until convergence:

  for $i = 1, \cdots, m$:

    $c^{(i)} \leftarrow$ index of cluster centroid $(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K)$ closest to $\boldsymbol{x}^{(i)}$

  for $k = 1, \cdots, K$:

    $\boldsymbol{\mu}_k \leftarrow$ centroid of data points $\boldsymbol{x}^{(i)}$ assigned to cluster k

Dimensional Reduction

Many common ML problems have training data with many features n.
Higher dimensional data is more computationally expensive to work with.

- **Kernel tricks** (Simulate a high dimensional environment **without** actually working in it)
- **Principal Component Analysis** (Project higher dimensions to lower dimensions, while keeping relevancy)
- Convolutions, max-pooling etc.

Principal Component Analysis

- Not a technique used for **overfitting**! PCA have no idea what the data is for. Might throw away relevant information.
- Trade accuracy for **simplicity** $\Longrightarrow$ potentially results in lower accuracy (but rarely)
- Lower dimensions $\Longrightarrow$ **significantly faster** training

Step by step

Standardization (or centering)

Standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

Calculating the covariance matrix

Understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information.

Step by step

Compute principal components using SVD

- Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables.
- Think of it as a rotation of the data, such that the new axes are the principal components (change of coordinates).
- Represent the directions of the data that explain a maximal amount of variance. The larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more information it has.

Question 1

Given that every iteration produces a partition with a lower loss, prove that this algorithm always converges.

Question 1

Given that every iteration produces a partition with a lower loss, prove that this algorithm always converges.

Solution:

Since the loss always decreases, prove there are a finite number of possible partitions.

- There are indeed a finite number of ways to partition N points into k clusters (upper bound of N^k).
- Cannot be a cycle as a cycle cannot constantly decrease in loss.

Optional bonus question: how do we prove that loss monotonically decreases?

Hint: Consider breaking down the k-means into two steps

- Assign points to cluster
- Recompute new centroids

Question 1

Although k-means always converges, it may get stuck at a bad local minimum. As mentioned in the lecture, one method of circumventing this is to run the algorithm multiple times and choose the clusters with the minimum loss. Suggest some other ways to help the algorithm get closer to the global minimum.

Solution:

The problem lies in the random initialisation. We can come up with better ways of initialising the random initial state.

- We can choose the first centroid randomly. Then, choose the second centroid to be as far as possible from the first centroid. Then, choose the next centroid to be as far as possible from the all previous centroids. Repeat until we have initialised all k centroids.
- We can also make use of a probabilistic method (k-means++). We start by choosing the first centroid randomly. To choose the rest of the centroids, we choose it randomly using a weighted probability distribution where the probability of choosing a point is proportional to $D(x)^2$, where $D(x)$ is the distance from x to the closest existing centroid.

Question 1

You are given the following data.

i	1	2	3	4	5	6
x	1	1	2	2	3	3
y	0	1	1	2	1	2

Cluster the 6 points in table 1 into **two** clusters using K-means algorithm. The two initial centroids are (0, 1) and (2.5, 2).

Iteration 1

Using first centroid = (0, 1) and second centroid = (2.5, 2), we get the table below.

Point	Distance to first centroid	Distance to second centroid	Assigned Cluster
1	2	6.25	1
2	1	3.25	1
3	4	1.25	2
4	5	0.25	2
5	9	1.25	2
6	10	0.25	2

Computing the new centroids:

- Centroid 1 = ((1, 0) + (1, 1)) / 2 = (1, 0.5)
- Centroid 2 = ((2, 1) + (2, 2) + (3, 1) + (3, 2)) / 4 = (2.5, 1.5)

Iteration 2

Using first centroid = (1, 0.5) and second centroid = (2.5, 1.5), we get the table below.

Point	Distance to first centroid	Distance to second centroid	Assigned Cluster
1	0.25	4.5	1
2	0.25	2.5	1
3	1.25	0.5	2
4	3.25	0.5	2
5	4.25	0.5	2
6	6.25	0.5	2

Computing the new centroids:

- Centroid 1 = ((1, 0) + (1, 1)) / 2 = (1, 0.5)
- Centroid 2 = ((2, 1) + (2, 2) + (3, 1) + (3, 2)) / 4 = (2.5, 1.5)

Complete!

Since the centroids are the same as the previous iteration, the K-means algorithm has converged. The first cluster has centroid (1, 0.5) and contains points 1 and 2. The second cluster has centroid (2.5, 1.5) and contains points 3, 4, 5, and 6.

Question 1

Cluster the 6 points in table 1 into two clusters using K-medoids algorithm. The initial medoids are point 1 and point 3.

Algorithm 2: K-medoids clustering

for $k = 1$ ***to*** K **do**
  $\mu_k \leftarrow$ random data point $x^{(i)}$
while *not converged* **do**
  for $i = 1$ ***to*** m **do**
    $c^{(i)} \leftarrow argmin_k ||x^{(i)} - \mu_k||^2$
  for $k = 1$ ***to*** K **do**
    $\mu_k \leftarrow \frac{1}{|\{x^{(i)}|c^{(i)}=k\}|} \sum_{x \in \{x^{(i)}|c^{(i)}=k\}} x$
  for $k = 1$ ***to*** K **do**
    $\mu_k \leftarrow argmin_{x^{(i)} \in \{x^{(i)}|c^{(i)}=k\}} ||x^{(i)} - \mu_k||^2$

Iteration 1

Using first medoid = (1, 0) and second medoid = (2, 1), we get the table below.

Point	Distance to first centroid	Distance to second centroid	Assigned Cluster
1	0	2	1
2	1	1	2
3	2	0	2
4	5	1	2
5	5	1	2
6	8	2	2

Note: Use a deterministic tie-break strategy to avoid infinite loops or excessive iterations.

Computing the new medoid:

- Centroid 1 = (1, 0)
- Centroid 2 = ((1, 1) + (2, 1) + (2, 2) + (3, 1) + (3, 2)) / 5 = (2.2, 1.4)

We need to find the closest points to each centroid.

- For centroid 1, the closest point is point 1. Hence, we set point 1 as the new medoid.
- For centroid 2, the closest point is point 3. Hence, we set point 3 as the new medoid.

Since the medoids are the same as the initial one, the K-medoids algorithm has converged.

Question 2

Hierarchical clustering requires a linkage method that defines the proximity of one cluster to another. Three linkage methods are defined as follows:

- Single linkage (or nearest neighbour):
  The proximity of two clusters is defined as the proximity of their closest items.
- Complete linkage (or furthest neighbour):
  The proximity of two clusters is defined as the proximity of their most distant items.
- Centroid linkage:
  The proximity of two clusters is defined as the proximity of their centroids.

Given this dataset:

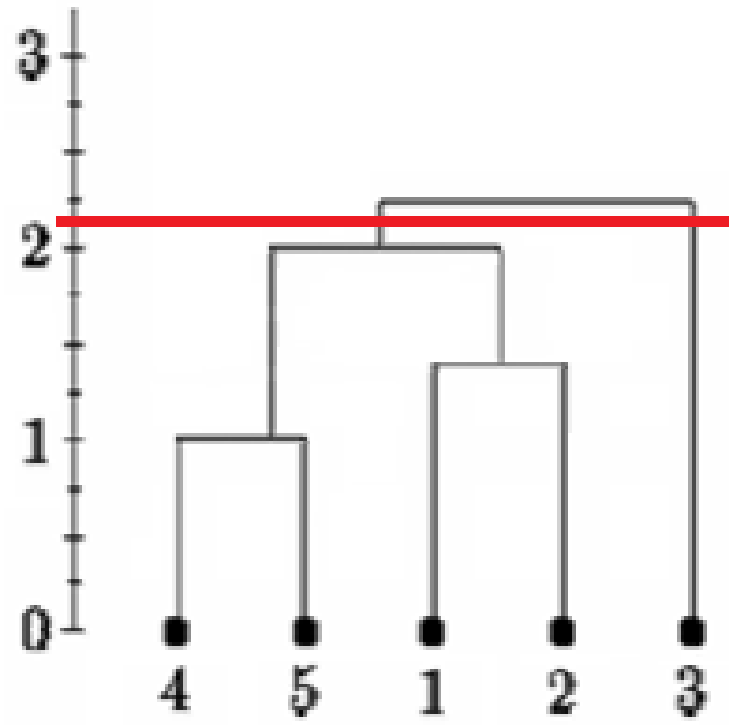
i	1	2	3	4	5
x_1	0	1	3	1	1
x_2	0	1	0	3	4

Complete the distance matrix below (using euclidean distance).

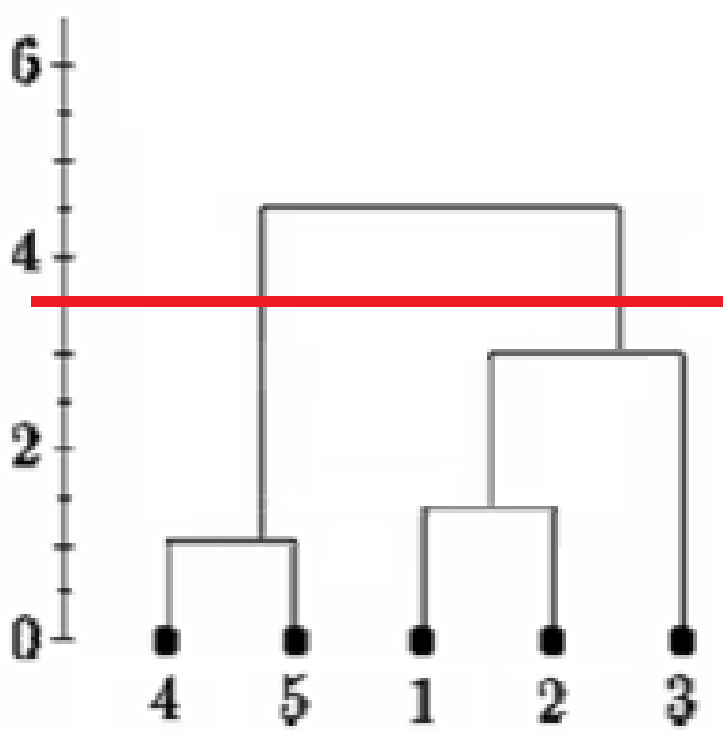
	1	2	3	4	5
1	0				
2	$\sqrt{2}$	0			
3	$\sqrt{9}$	$\sqrt{5}$	0		
4	$\sqrt{10}$	$\sqrt{4}$	$\sqrt{13}$	0	
5	$\sqrt{17}$	$\sqrt{9}$	$\sqrt{20}$	$\sqrt{1}$	0

Draw the dendrogram for the three linkage methods.

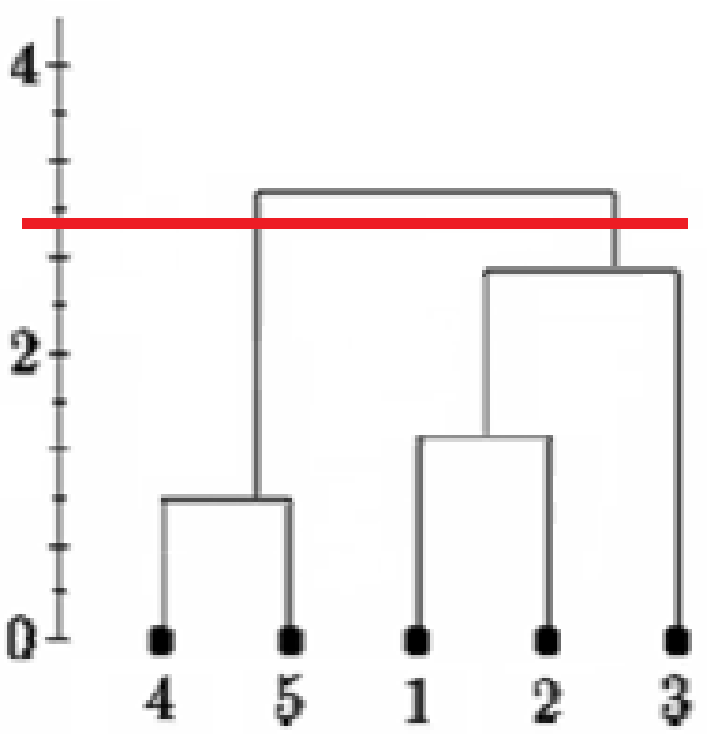
Single



Complete



Centroid



Question 3

PCA can be thought of as a form of lossy compression. In this question, we will compress some data, provided in the form of an image for better visualisation. We will be making use of a classic test image mandrill.png, which can be downloaded from https://upload.wikimedia.org/wikipedia/commons/a/ab/Mandrill-k-means.png.

The current choice of k = 9 does not produce a very nice output.
What is a good value for k? Justify your answer.

A good value for k is 286.

When $k = 286$, the explained variance is $\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2} \geq 0.99$.

For the value of k you select in (a), what is the space saved by doing this compression?

When using $k = 286$, the (512×1536) 2D-array, is now represented by U_{reduce} (512×286) and Z (286×1536) and the row means (512×1).
This demonstrates an approximate 25% space saved.

$$\frac{(512 \times 286) + (286 \times 1536) + 512}{512 \times 1536} = \frac{586240}{786432} = 0.745$$

If we wish to have more compression, we can choose a smaller k, but sacrifice image quality.

What are the drawbacks of this form of compression?

This is a lossy compression as we do not regain 100% of the variance. For data with less redundancy, the value of k might be too large such that we would actually end up requiring more space as the decompression matrix U_{reduce} needs to be stored as well. In practice, PCA is done over a dataset of images so that U_{reduce} can be reused for every image for better compression.

Any (final) Questions?