COMS 4030A

Adaptive Computation and Machine Learning

8. **Unsupervised Learning**

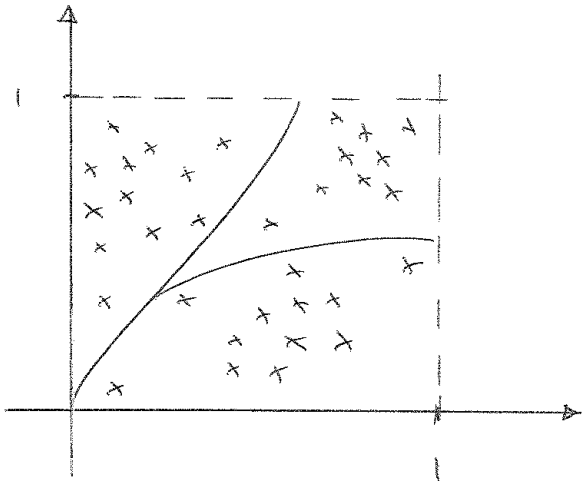
Unsupervised learning methods are applicable to datasets that have no target values, only attribute values, as in the following dataset:

$$\begin{matrix} x_1 & x_2 & \cdots & x_m \end{matrix}$$
$$\begin{bmatrix} 1 & 2 & \cdots & -3 \\ -3 & -2 & \cdots & 5 \\ 0 & 1 & \cdots & -7 \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix}$$

Each datapoint $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ is a point in $\mathbb{R}^m$, so the dataset is a subset of $\mathbb{R}^m$.

The objective of unsupervised learning is to mine the data by finding useful patterns within it. An important aspect of this is to try to find natural groupings, or **clusters**, of similar datapoints; this is referred to as **clustering** the data.

Consider the dataset in $\mathbb{R}^2$ pictured below, where the $\times$'s indicate datapoints.
The lines through the data represent a possible clustering of the datapoints into three distinct and non-overlapping clusters.



8.1. k-means Algorithm.

The first method we consider for clustering data is the k-means algorithm; this method is widely used and has many variants. We focus on the basic versions.

In the following algorithm, the value of k is a hyperparameter that is chosen by the user; k is the number of **clusters** that the data will be divided into. Each cluster will have an associated **cluster centre** that is the mean of all the datapoints in that cluster. Thus, k cluster centres are required, which are denoted by $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^k$. The cluster centres are initially chosen at random.

k-means Algorithm

The input is a dataset S consisting of points in $\mathbb{R}^m$ (usually normalised).

Choose a value for k.
Choose k random points in $\mathbb{R}^m$: $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$, $\ldots$, $\boldsymbol{\mu}^k$.

while (stopping condition is not satisfied)
for each datapoint $\boldsymbol{x}$ in S, do the following:
compute $\min\{d(\boldsymbol{x}, \boldsymbol{\mu}^1), d(\boldsymbol{x}, \boldsymbol{\mu}^2), \ldots, d(\boldsymbol{x}, \boldsymbol{\mu}^k)\}$
assign $\boldsymbol{x}$ to the cluster j for which $d(\boldsymbol{x}, \boldsymbol{\mu}^j)$ is minimum.

for each cluster j, replace $\boldsymbol{\mu}^j$ by the **mean** of all the points of cluster j:

$$\boldsymbol{\mu}^j \leftarrow \tfrac{1}{N_j} \sum_{\boldsymbol{x} \in \text{cluster } j} \boldsymbol{x}, \quad \text{where } N_j \text{ is the number of elements in cluster } j.$$

In the above k-means Algorithm, $d(\boldsymbol{x}, \boldsymbol{\mu})$ is the **Euclidean distance** between $\boldsymbol{x}$ and $\boldsymbol{\mu}$ (where $\boldsymbol{x}, \boldsymbol{\mu} \in \mathbb{R}^m$), defined by:

$$d(\boldsymbol{x}, \boldsymbol{\mu}) = \sqrt{\textstyle\sum_{i=1}^{m} (x_i - \mu_i)^2}.$$

In order to compute $\min\{d(\boldsymbol{x}, \boldsymbol{\mu}^1), d(\boldsymbol{x}, \boldsymbol{\mu}^2), \ldots, d(\boldsymbol{x}, \boldsymbol{\mu}^k)\}$, it is not necessary to compute the square roots; we need only compute the summations and take the minimum of those.

In the main loop, each datapoint is assigned to a unique cluster by choosing the cluster whose cluster centre is closest to that datapoint. Once all datapoints have been assigned to a cluster, the cluster centres are recalculated by finding the mean of all the datapoints in each cluster.

If we write $\boldsymbol{\mu}^j = (\mu_1^j, \ldots, \mu_i^j, \ldots, \mu_m^j)$ and $\boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_m)$, then the update rule in the k-MEANS ALGORITHM can be written as:

$$(\mu_1^j, \ldots, \mu_i^j, \ldots, \mu_m^j) \leftarrow \tfrac{1}{N_j} \sum_{\boldsymbol{x} \in \text{cluster } j} (x_1, \ldots, x_i, \ldots, x_m).$$

In the above update, for each i, we have that μ_i^j is set to the mean of all the x_i coordinates of all $\boldsymbol{x}$ in cluster j:

$$\mu_i^j = \tfrac{1}{N_j} \sum_{\boldsymbol{x} \in \text{cluster } j} x_j$$

The output of the k-MEANS ALGORITHM is the set $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$, ..., $\boldsymbol{\mu}^k$ of cluster centres. These centre define a clustering of the dataset, whereby every datapoint $\boldsymbol{x}$ in S belongs to exactly one cluster, namely, the cluster j for which $d(\boldsymbol{x}, \boldsymbol{\mu}^j)$ is minimum. The cluster centres $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$, ..., $\boldsymbol{\mu}^k$ completely describe the clustering.

If $\boldsymbol{x}$ is a new datapoint (not from the dataset), then $\boldsymbol{x}$ can be assigned to a cluster by simply finding the cluster centre $\boldsymbol{\mu}^j$ for which $d(\boldsymbol{x}, \boldsymbol{\mu}^j)$ is minimum. The idea is that $\boldsymbol{x}$ should be assigned to a cluster of datapoints that are similar to itself.

Regarding stopping conditions, the algorithm can stop if there are no changes to any of the cluster centres $\boldsymbol{\mu}^j$ during an execution of the **for** loop, which means that no points in the dataset have moved to a new cluster.
However, it is possible that the algorithm continues to oscillate (i.e., the $\boldsymbol{\mu}^j$'s do not stabilize), so we may need to stop the algorithm after some specified time or number of iterations.

The initial choice of cluster centres can affect the final clustering.
Different clusterings of a dataset can be compared using the following measure.

For a clustering defined by a set of cluster centres $\boldsymbol{\mu}^1$, ..., $\boldsymbol{\mu}^k$, the **sum-of-squares error** is:

$$\text{sum-of-squares error} = \sum_{j=1}^{k} \sum_{\boldsymbol{x} \in \text{cluster } j} d(\boldsymbol{x}, \boldsymbol{\mu}^j)^2.$$

Given two clusterings of a dataset, the one with the smaller sum-of-squares error is preferred since the smaller error indicates that the points within each cluster are more tightly centred on their cluster centre.

The sum-of-squares error can also be used as a stopping condition. After each execution of the **for** loop, calculate the sum-of-squares error; if there is no change to the error, then stop.

Note that the sum-of-squares error will never be zero, except in the extreme case where the number of clusters centres equals the number of datapoints and each datapoint matches a cluster centre.

The number k of clusters is a hyperparameter that must be chosen by the user. Knowing what the correct number of clusters should be is a difficult question, especially if the data is high-dimensional and difficult to visualise. It may be necessary to try different values of k and see which give the best results (they can be compared by the sum-of-squares error).

Datapoints that lie far from all cluster centres are called **outliers** or **noisy datapoints**. Such points nevertheless each belong to one of the clusters and can affect the correct positioning of the cluster centre. One way to avoid this is to remove them from the dataset, or to use the **median** instead of the mean when updating cluster centres.

It is usually a good idea to normalise the data before clustering, that is, to scale each attribute's values to the same range, usually $[0, 1]$ or $[-1, 1]$. The reason is that if some attribute has large values, say in the 1000s, and the other attributes have small values, say in the range 0 to 1, then the large values will dominate the Euclidean distance and the final clusters will be based mostly on the attribute with the large values.

8.2. **Online k-means Algorithm.**

In the following online version of the k-means Algorithm, cluster centres are updated after considering each datapoint in the dataset. This version allows for new datapoints to be added to the dataset during training.

Online k-means Algorithm

The input is a dataset S consisting of points in $\mathbb{R}^m$ (usually normalised).

Choose a value for k.

Choose k random points in $\mathbb{R}^m$: $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots, \boldsymbol{\mu}^k$.

Choose a learning rate η.

while (stopping condition is not satisfied)

randomize the order of the datapoints in S

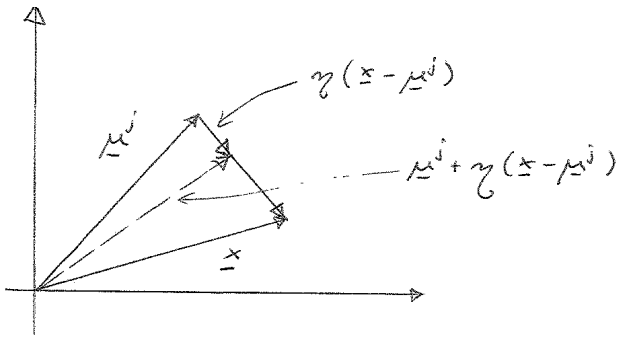
for each datapoint $\boldsymbol{x}$ in S, do the following:

compute $\min\{d(\boldsymbol{x}, \boldsymbol{\mu}^1), d(\boldsymbol{x}, \boldsymbol{\mu}^2), \ldots, d(\boldsymbol{x}, \boldsymbol{\mu}^k)\}$

update the cluster centre $\boldsymbol{\mu}_j$ for which $d(\boldsymbol{x}, \boldsymbol{\mu}^j)$ is the minimum:

$$\boldsymbol{\mu}^j \leftarrow \boldsymbol{\mu}^j + \eta(\boldsymbol{x} - \boldsymbol{\mu}^j)$$

The update rule for the cluster centres in the Online k-means Algorithm is based on the vector geometry illustrated in the diagram below. If $\boldsymbol{x}$ is closest to centre $\boldsymbol{\mu}^j$, then $\boldsymbol{\mu}^j$ moves incrementally closer to $\boldsymbol{x}$ in a straight line.



The cluster centres trace a path through space until they find stable positions.

The algorithm can stop once the movements of the cluster centres become very small. However, the cluster centres may oscillate, so a good idea is decay the learning rate η after each iteration.

8.3. **Hierarchical clustering.**

The Hierarchical Clustering Algorithm described below is an example of **agglomerative hierarchical clustering**. In such a clustering algorithm, each datapoint begins as a cluster of one point, and clusters are then agglomerated into larger clusters by merging nearby clusters. The process of agglomeration continues until there is only one cluster left, which contains all the datapoints. Then the user may choose a natural clustering from those that were generated by the algorithm.

There also exist **divisive hierarchical clustering** algorithms which start with one cluster containing all datapoints and repeatedly divide the clusters into smaller clusters until a suitable clustering is obtained. This method is not commonly used as it is difficult to determine how best to divide a cluster into two smaller clusters.

Hierarchical Clustering Algorithm

The input is a dataset S consisting of points in $\mathbb{R}^m$ (usually normalised).

Assign each datapoint to a cluster containing only itself.
Compute the **proximity matrix** for all clusters using a **proximity metric**.

while there are more than one cluster
    find the two clusters that are closest to each other with respect to the
        proximity matrix and merge these two clusters into one new cluster.
    Update the proximity matrix.

In the above algorithm, the set of clusters obtained at each step is recorded, so that once the algorithm is complete, a clustering into any chosen number of clusters can be obtained. Alternatively, you can stop the algorithm when the required number of clusters is obtained.

Note that there may be ties when determining the two closest clusters; in this case, some random choice can be made.

The **proximity matrix** used in the above algorithm requires a **proximity metric** that computes the distance between two clusters, i.e., the distance between two finite subsets of $\mathbb{R}^m$. There are many choices of proximity metric, also called **linkage**, between two sets; we describe some of them below.

Let $d(\boldsymbol{x}, \boldsymbol{y})$ denote the Euclidean distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^m$.
Let A and B be finite subsets of $\mathbb{R}^m$:

The **single linkage** (or **minimum distance**) between A and B is:
$\min\{d(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x} \in A,\, \boldsymbol{y} \in B\}$.

The **complete linkage** (or **maximum distance**) between A and B is:
$\max\{d(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x} \in A,\, \boldsymbol{y} \in B\}$.

The **average linkage** between A and B is:
$\frac{1}{|A|\,|B|} \sum_{\boldsymbol{x} \in A} \sum_{\boldsymbol{y} \in B} d(\boldsymbol{x}, \boldsymbol{y})$ where $|\cdot|$ denotes the number of elements in a set.

The **centroid linkage** distance between A and B is:
$d(\boldsymbol{\mu}_A, \boldsymbol{\mu}_B)$ where $\boldsymbol{\mu}_A$ is the cluster centre of A and $\boldsymbol{\mu}_B$ is the cluster centre of B.

Recall that the **cluster centre** of a set A is the mean of all points in A.

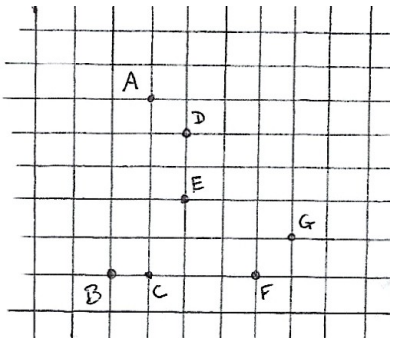
In each of the above functions, if A and B each contain just one point, say $A = \{\boldsymbol{x}\}$ and $B = \{\boldsymbol{y}\}$, then the linkage between A and B is just $d(\boldsymbol{x}, \boldsymbol{y})$, the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. Thus, the initial proximity matrix will look like the following table:

	$\{\boldsymbol{x}_1\}$	$\{\boldsymbol{x}_2\}$	$\{\boldsymbol{x}_3\}$	$\cdots\cdots$	$\{\boldsymbol{x}_n\}$
$\{\boldsymbol{x}_1\}$	0	$d(\boldsymbol{x}_1, \boldsymbol{x}_2)$	$d(\boldsymbol{x}_1, \boldsymbol{x}_3)$	$\cdots\cdots$	$d(\boldsymbol{x}_1, \boldsymbol{x}_n)$
$\{\boldsymbol{x}_2\}$	$d(\boldsymbol{x}_2, \boldsymbol{x}_1)$	0	$d(\boldsymbol{x}_2, \boldsymbol{x}_3)$	$\cdots\cdots$	$d(\boldsymbol{x}_2, \boldsymbol{x}_n)$
$\{\boldsymbol{x}_3\}$	$d(\boldsymbol{x}_3, \boldsymbol{x}_1)$	$d(\boldsymbol{x}_3, \boldsymbol{x}_2)$	0	$\cdots\cdots$	$d(\boldsymbol{x}_3, \boldsymbol{x}_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots\cdots$	$\vdots$
$\{\boldsymbol{x}_n\}$	$d(\boldsymbol{x}_n, \boldsymbol{x}_1)$	$d(\boldsymbol{x}_n, \boldsymbol{x}_2)$	$d(\boldsymbol{x}_n, \boldsymbol{x}_3)$	$\cdots\cdots$	0

Once two clusters are combined into one, the two clusters are replaced in the proximity matrix by one new cluster. Then the proximity between the new cluster and all the other clusters is calculated using the selected proximity metric. Keeping a proximity matrix saves computational time as it avoids recalculating proximities between clusters that were not changed in the current step (which is a kind dynamic programming approach).

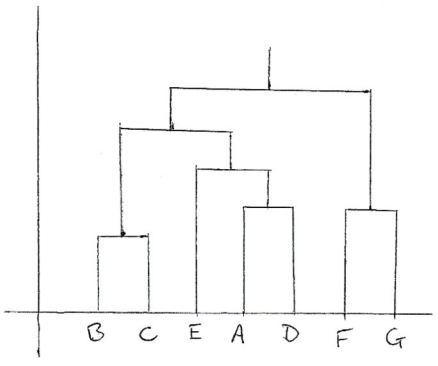
Only the upper triangle of the matrix is required (assuming the proximity metric is symmetric). A suitable data-structure is required to store the proximity values and allow for the combining of two clusters into one.

Example: We apply the HIERARCHICAL CLUSTERING ALGORITHM to the set of datapoints $\{ A, B, C, D, E, F, G \}$ indicated in the diagram below.



The initial clusters are $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{F\}$, $\{G\}$.
Using Euclidean distance $d(\boldsymbol{x}, \boldsymbol{y})$ between the points and single linkage between clusters, the clusters are formed in the following order.
First, $\{B\}$ and $\{C\}$ will be joined into a cluster $\{B, C\}$ as they are the closest pair.
Next, pairs $\{A\}$, $\{D\}$ and $\{F\}$, $\{G\}$ are equally close, so one of them is chosen, say $\{A, D\}$.
In the next step $\{F\}$, $\{G\}$ are closest and $\{F, G\}$ is formed.
Next, the closest pair of clusters is $\{E\}$ and $\{A, D\}$ which combine to get $\{E, A, D\}$;
then $\{B, C\}$ and $\{A, D, E\}$ are joined to get $\{B, C, E, A, D\}$;
lastly, the clusters $\{B, C, E, A, D\}$ and $\{F, G\}$ are joined to form $\{B, C, E, A, D, F, G\}$.

The **dendrogram** below shows the clusters and the order in which they were formed.
By making a horizontal cut across the diagram, a clustering with a selected number of clusters can be obtained.



If a different proximity metric is used, the clustering can be different.
As an exercise, try the complete linkage and use a proximity matrix for the above example.

EXERCISES

(1) Suppose you want to cluster the following dataset of points in $\mathbb{R}^2$ into 2 clusters.

$$\begin{bmatrix} 1 & 1 \\ 1 & 4 \\ 2 & 1 \\ 4 & 1 \\ 4 & 6 \\ 5 & 4 \\ 5 & 5 \end{bmatrix}$$

(a) Starting with $\boldsymbol{\mu}^1 = (3, 3)$ and $\boldsymbol{\mu}^2 = (3, 4)$ do two iterations of the k-means algorithm. Plot the datapoints in $\mathbb{R}^2$ and also the changes to $\boldsymbol{\mu}^1$ and $\boldsymbol{\mu}^2$.

(b) Starting with $\boldsymbol{\mu}^1 = (3, 3)$ and $\boldsymbol{\mu}^2 = (3, 4)$ do two iterations of the online k-means algorithm. Use $\eta = 0.1$. Plot the datapoints in $\mathbb{R}^2$ and also the changes to $\boldsymbol{\mu}^1$ and $\boldsymbol{\mu}^2$.

(c) For each of the questions (a), (b), compute the sum-of-squares error for the final cluster centres obtained.

(2) Use hierarchical clustering on the dataset in exercise (1); do it with single linkage and another linkage of your choice.

(3) Suppose you want to cluster the following dataset of points in $\mathbb{R}^2$ into 2 clusters.

$$\begin{bmatrix} 0.1 & 0.4 \\ 0.6 & 0.5 \\ 0.7 & 0.7 \\ 0.3 & 0.6 \\ 0.4 & 0.55 \\ 0.8 & 0.6 \end{bmatrix}$$

(a) Starting with $\boldsymbol{\mu}^1 = (0.5, 0.5)$ and $\boldsymbol{\mu}^2 = (0.7, 0.7)$ do two iterations of the k-means algorithm. Plot the datapoints in $\mathbb{R}^2$ and also the changes to $\boldsymbol{\mu}^1$ and $\boldsymbol{\mu}^2$.

(b) Starting with $\boldsymbol{\mu}^1 = (0.5, 0.5)$ and $\boldsymbol{\mu}^2 = (0.7, 0.7)$ do two iterations of the online k-means algorithm. Use $\eta = 0.1$. Plot the datapoints in $\mathbb{R}^2$ and also the changes to $\boldsymbol{\mu}^1$ and $\boldsymbol{\mu}^2$.

(c) For each of the questions (a), (b), compute the sum-of-squares error for the final cluster centres obtained.

(4) Use hierarchical clustering on the dataset in exercise (3); do it with single linkage and another linkage of your choice.

(5) A network administrator wants to develop an automated intruder detection system for a computer network. A large amount of data on users is available, including: day and time of login, length of time logged in, type of programs used and amount of data downloaded. The dataset **does not** contain information on whether a user is an intruder or not, however, the network administrator does have a record of a few cases in which a user was positively identified as an intruder. Describe how you could use **clustering** to predict if a current user is a suspected intruder. What other information could you get from the clustering?