Unsupervised Learning

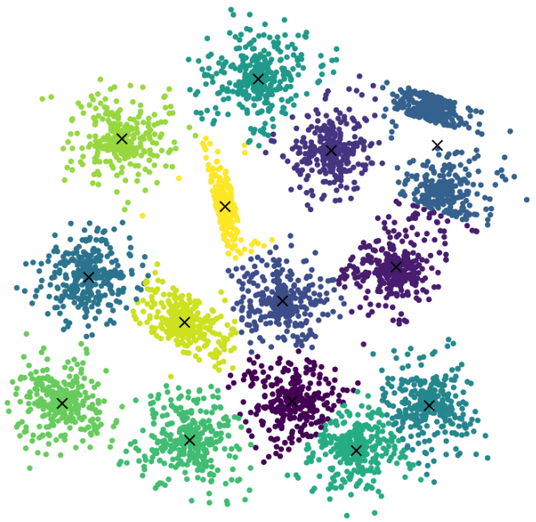
Introduction

Until now, all models we've seen require training data to tweak the model such that it becomes representative, this is called supervised learning. Unsupervised learning does not require labelled data such that it can "learn what to do to", but rather draws its own conclusions.

In this poster, two unsupervised learning techniques will be explained: K-Means clustering and Principal Component Analysis (PCA).

K-Means Clustering

K-Means clustering is an algorithm whereby input data are classified into groups. Unlike logistic regression or a neural network, these groups are not specified in advance (however the number of groups is), instead the model groups data that are close in vector space.



In data that has distinct groups like the one above this allows them to be formally classified, which is useful for two things:

1. Getting a meaningful intuition of the structure of the data – Clustering allows us to see the sizes and characteristics of groups within the data.
2. Grouping for further analyses – Once clustered, differences between groups can be analysed.

Even when data aren't classified into discrete groups, clustering can still be useful because it allows classification based on criteria that aren't intuitive to a person. For example, it can be used to group colours together for image compression, group customers together for market segmentation or identify different airline behaviours (e.g. cluster into short/long haul business models based on network route lengths).

It's worth noting K-Means can be performed in more than 2 dimensions!

Principal Component Analysis (PCA)

PCA is a method of reducing the dimensionality of numeric data. This involves reducing the number of columns in a dataset while maintaining as much of the variation between rows as possible, by creating a new dataset with a smaller number of columns (dimensions) containing dimensionless values that are calculated from all the columns in the original dataset.

Original Dataset			
Index	**ESAD range**	**Great Circle (nm)**	**ESAD + Wind (nm)**
0	3249	2823	3462
1	2342	1930	2412
2	3180	2684	3100
3	1892	1609	2121
4	2403	1904	2467
5	1500	1400	1671

PCA

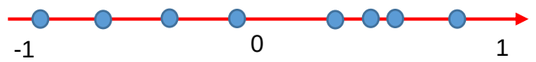
Dataset 2	
Index	**Z1**
0	0.54
1	0.21
2	0.32
3	0.11
4	0.04
5	-0.23

This has two main uses:

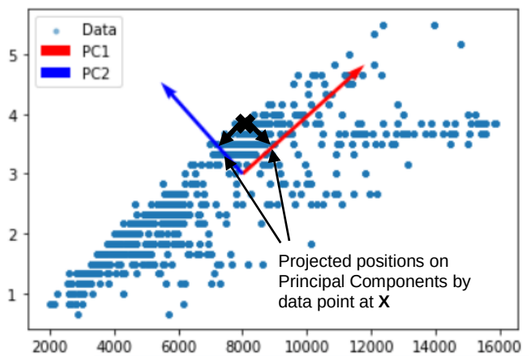
- Data compression – As mentioned earlier, PCA maintains as much variation between data as possible while reducing the number of dimensions. This is useful for storing data where it would otherwise use too much memory, and as an input for machine learning models, as they need to consider fewer input variables and will be faster to train.
- Visualisation – Since the data will contain fewer dimensions but still considers all original variables, PCA can be useful to reduce data to 3, 2 or 1D such that the data can be visualised.

PCA works by calculating the axis (Principal Component) in which the most variation in the data is present, followed by the axis in which the second highest amount of variation is present and so on until there are as many principal components as dimensions in the original dataset, which will each contain a diminishing amount of the original variation.

Once these axes are calculated, so is the position of each data point on the axes, and this is the dimensionless value for that Principal Component.



As mentioned earlier, there will be as many Principal Components as columns (dimensions) in the original data, and once the data positions on each axis are calculated, you can choose how many Principal Components you want to save. The more components saved, the more variation in the data will be captured, but the more memory storing it will use.



Index	Z1	Z2	Z3	Z4
0	0.54	-0.13	-0.48	0.86
1	0.21	0.07	0.83	-0.86
2	0.32	-0.65	0.28	0.96
3	0.11	-0.95	0.70	0.10
4	0.04	0.58	-0.66	-0.84
5	-0.23	0.26	-0.29	-0.22

% of original variation expressed:

- 68%
- 95%
- 87%
- 100%

Implementing Unsupervised Learning

Source code can be found at: P:\ENGINEERING\FPO\Digitalisation\1 FPO Projects\IIX Data Analytics\02-General presentations\Knowledge folder - preparation\Data Science Learning Framework\Machine Learning Posters\5 - Unsupervised Learning

Implementing K-Means

K-Means is a relatively simple model to implement, and to do so we'll use the example of clustering timeseries data into airlines based on their latitude longitude coordinates. To get an understanding of our data, we can plot it.

It looks like there are four clusters (probably one in Europe, one in South America and two in Asia), so we can tell K-Means to create four clusters.

To initialize K-Means, the cluster centroids' (i.e. the "centre" of a group) locations should be assigned to random data points. Here two have been assigned to one cluster, but that's not a problem.

After centroid assignment, the following should be performed iteratively (recommend 50 iterations):

1. Data points should be assigned to the closest cluster centroid, i.e. the centroid with the lowest Euclidian separation:

$$separation = \left\| x^{(i)} - \mu_k \right\|$$

where $x^{(i)}$ is the i^{th} data point's coordinates and μ_k the k^{th} centroid's.

2. The cluster centroids should be reassigned as the centroid of all the data points assigned to them, using the equation

$$\mu_k = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

μ_k - Centroid of k^{th} cluster
n – Number of data points in the k^{th} cluster
$x^{(i)}$ - i^{th} Data point of the k^{th} cluster

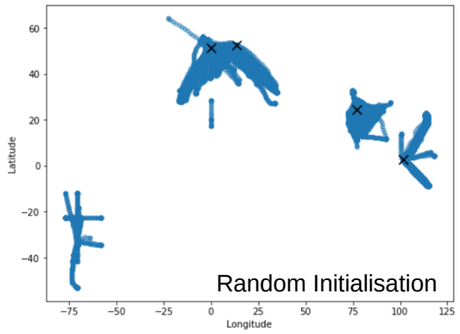
After this has been performed for a number of iterations (You can see this process in the gif in the same folder location as the source code), the centroids will settle on their own clusters.

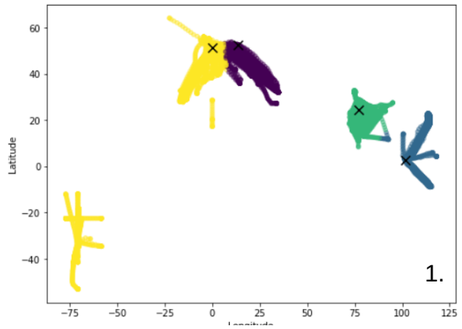
Finally, the objective of K-Means is to minimise the average distance between data point and assigned centroid, so the cost function is:

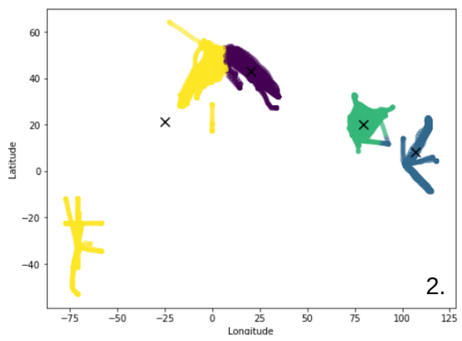
$$J = \frac{1}{m}\sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

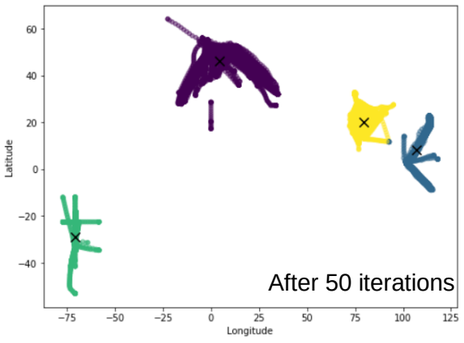
J - Cost
$\mu_{c^{(i)}}$ - Centroid of cluster assigned to i^{th} data point
m – Number of data points
$x^{(i)}$ - i^{th} data point

K-Means can converge to local minima, so it can be useful to run multiple models and pick the one with lowest cost.





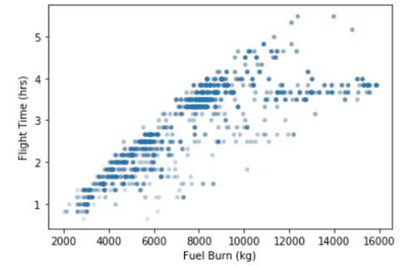




Implementing PCA

For PCA, we'll calculate the Principal Components for fuel burn and flight time data, and combine them into one variable. It can be seen they have a strong correlation, so their variation should be captured well by one dimension.

fuel_burn_total2	flight_time
8582.0	4.0
7130.0	3.0
8981.0	4.0
8237.0	4.0
10560.0	3.0



The first step is to normalise both columns, which can be done using the same equation used in linear and logistic regression:

$$X_{i,j} = \frac{X_{i,j} - \overline{X_j}}{\max X_j - \min X_j}$$

X – Input data
i – Row index
j – Column index

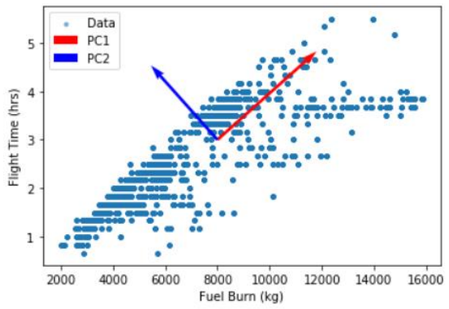
This will return all values scaled to $-0.5 < x < 0.5$. The next step is to calculate the Covariance Matrix Σ, an $n \times n$ matrix where n is the original number of dimensions, that represents the variation in each dimension.

Summation symbol

Covariance Matrix $\Sigma = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)})(x^{(i)})^T$

m – Number of data points
$x^{(i)}$ - i^{th} row

The eigenvector matrix of the covariance matrix can then be calculated (using a simple function such as `np.linalg.eig`), where each column represents a Principal Component vector. The variance captured is highest for the Principal Component in the first column, and diminishes from there, so the first k columns should be selected as your Principal Components, where k is the number of dimensions you wish to end up with. This will give you a matrix of Principal Components U. For our data, the two principal components are plotted over the data on the right.



To calculate the dimensionless values for each data point:

$$Z^{(i)} = U^T (x^{(i)})^T$$

$Z^{(i)}$ – Dimensionless values for i^{th} row
U – Principal Components
$x^{(i)}$ - i^{th} row

Z1	Z2
0.038077	0.073005
-0.236860	-0.097571
0.182495	0.036528
-0.366555	0.018211
0.538520	-0.188855

The original data point values can be approximated in reverse from the dimensionless values, using the equation

$$x_{approx}^{(i)} = U(Z^{(i)})^T$$

Which can be used to assess how many dimensions you need to keep to represent a certain percentage of the variation, by using the mean squared error. For 90% retained accuracy, you'd need $Error \leq 0.1$. 71% of the variation in our data is captured by one column.

$$Error = \frac{1}{m}\frac{\sum_{i=1}^{m} \left\| x^{(i)} - x_{approx}^{(i)} \right\|}{\sum_{i=1}^{m} \|x^{(i)}\|}$$