Hyperiondev

TASK

Unsupervised Learning - K-means Clustering

Visit our website

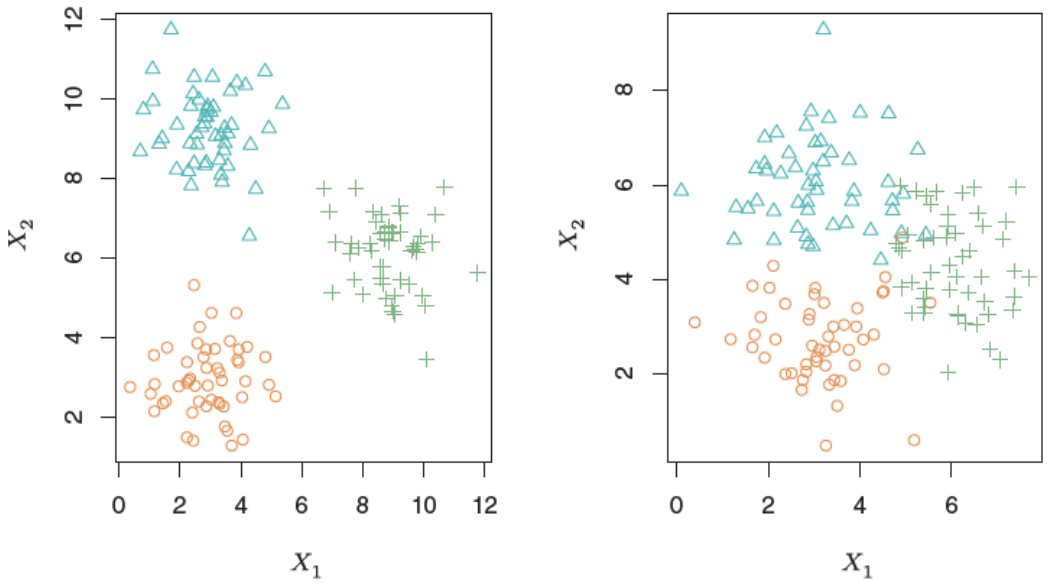
Introduction

WELCOME TO THE UNSUPERVISED LEARNING - K-MEANS CLUSTERING TASK!

In supervised learning, we work with datasets that contain both input and output variables. However, there is another fascinating class of problems called unsupervised learning, where we only have access to input variables. In this task, we will explore clustering, which is a type of unsupervised learning method. Clustering is really useful because it allows us to discover patterns, structures, and relationships within data, even when we don't have explicit labels or targets.

.

INTRODUCTION TO CLUSTERING



The graphs above show two datasets that are good candidates for applying clustering. The data on the left exhibits a very clear grouping that a clustering algorithm could readily identify for us. The data on the right has groups with more overlap and will be harder to identify, but still suited to a clustering approach.

If a clustering approach seems suitable, you can use cluster analysis to ascertain, on the basis of your input variables x1, . . . , xn, whether or not the observations fall into relatively distinct groups by asserting that observations within a group are similar to each other, while observations in different groups are different from each other.

Note that, as usual, we use examples that can be visualised on a two-dimensional plane. In practice, datasets can contain many more than two variables with more complicated relationships among them.

K-MEANS CLUSTERING

K-means clustering is the most well-known clustering algorithm. It is a simple and elegant approach for partitioning a dataset into K distinct clusters. To perform K-means clustering, we first specify the desired number of clusters, K, and then assign each observation to exactly one of the K clusters.

The principle behind K-means clustering is that a good clustering is one for which the within-cluster variation is as small as possible. The within-cluster variation — measured as inertia or within-cluster sum-of-squares — is a measure of the amount by which the observations within a cluster differ from each other. The within-cluster variation should be low and the variation between different clusters should be high.

Feature space

The similarity between observations is determined by the distance between their points in space. If observation #1 in a dataset of flowers has a height of 40 cm, and a flower width of 2 cm, that data point has the coordinates [40, 2] in the *feature space* of the dataset. If observation #2 has the coordinates [43, 2], and observation #3 has the values [10, 0.5], we can say that observation #2 is 'closer' to #1 than #3 is. A clustering algorithm will place #1 and #2 in the same cluster based on that closeness. There are a number of different distance metrics that are used in algorithms to decide how similar instances are. The most common one is called Euclidean distance. In two dimensions, the Euclidean distance between the points: (x_i, y_i) and (x_j, y_j) is $\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$

To compute the mean (or average) of a number of observations, you divide the sum of those observations by the number of observations. A multi-dimensional mean is equally straightforward. To compute the mean of a certain number of (x, y) points, you compute the mean of all x values and the mean of all the y values.

The K-means algorithm

The K-means algorithm follows the following steps:

1. Select a number of clusters, K.

2. Select random points from the data as starting values and initialise the mean of each cluster (e.g. using the `sample()` function in Python).
3. For *n* number of iterations:
    a. Assign each point to the cluster whose mean (or "centroid") is the nearest.
    b. Re-compute the means for each cluster based on its current members.
4. Repeat 3 until convergence is reached.

Convergence here means that the mean values of each cluster no longer or barely change between iterations, i.e. the value has 'stabilised'. When the K-means algorithm converges, it may have reached a local optimum. This means that the algorithm found the best values given the initialisation, but that there exists a clustering of the data with an even lower within-cluster variation. To avoid this, run the algorithm multiple times and select the most optimum solution.

The following figure shows 6 iterations of the K-means algorithm. At the start of iteration 1, the centres are based on random points and are therefore close to the global average, producing a very poor clustering of the points. Subsequent iterations drag the cluster centres out. By the 6th iteration, a sensible clustering has been found.

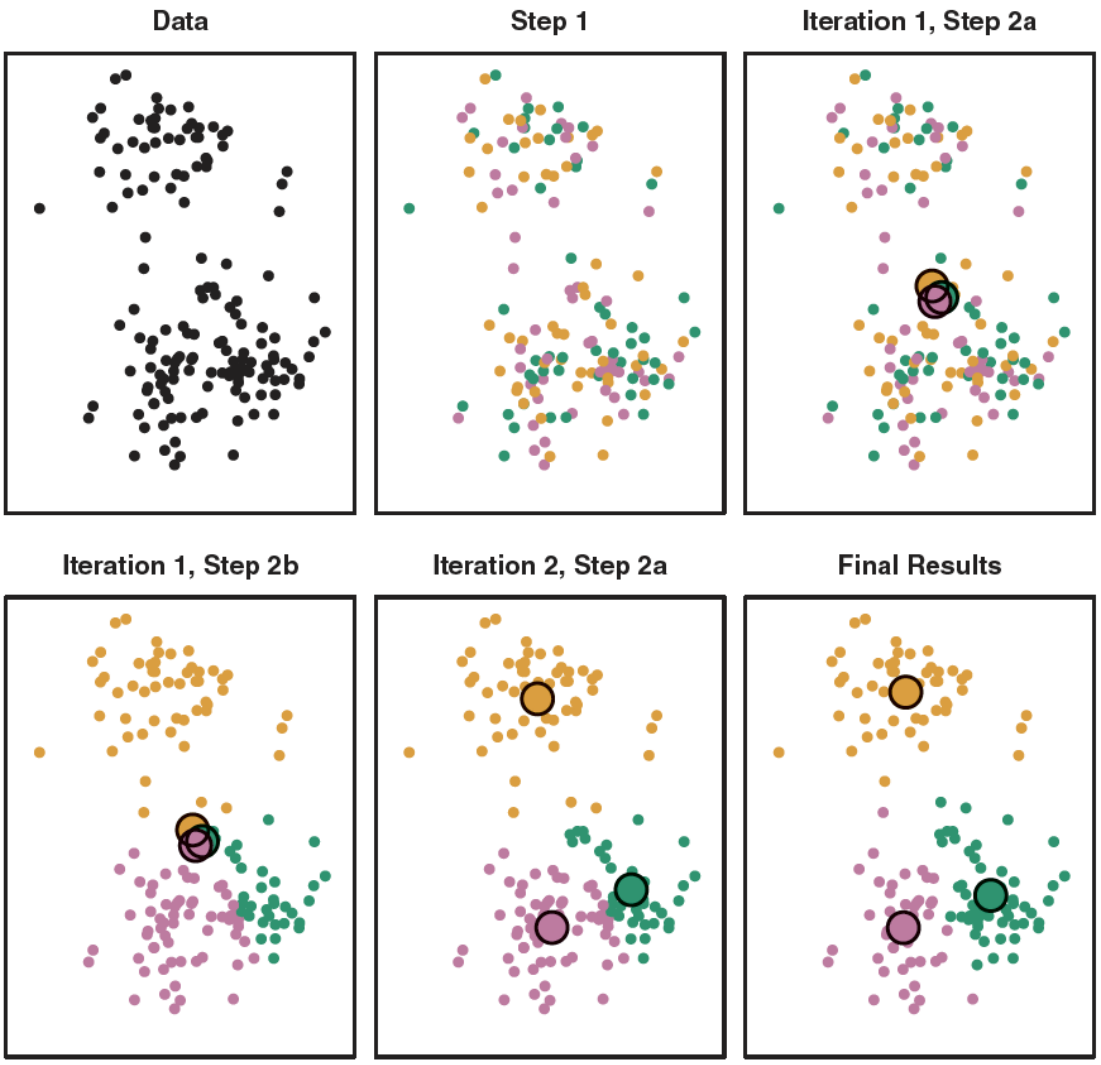


Image source: Lin, 2015

Choosing K

An important consideration when applying the K-means algorithm is that you need to choose K before running the analysis. The value of K will greatly affect the outcome and the accuracy of the clusters. The following plots show different outcomes of the algorithm depending on the value chosen for K.

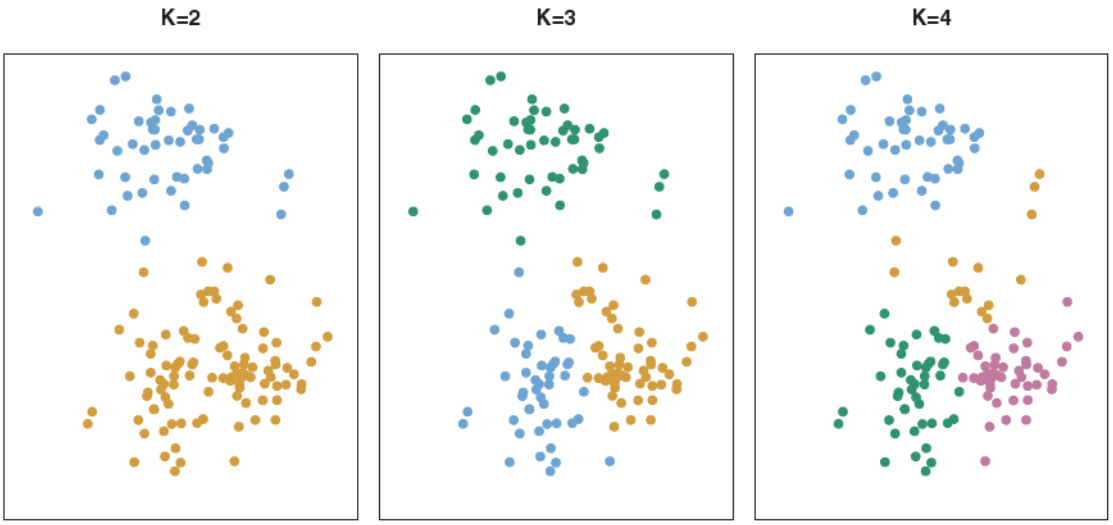


Image source: Lin, 2015

Validating clusters

It is possible to find clusters in any data, but it is important to determine if these clusters actually represent underlying subgroups in the data or are merely groupings with similar noise.

This is a very hard question to answer. There exists a number of techniques for assigning a significance value to a cluster in order to assess whether there is more evidence for the cluster than one would expect due to chance. However, there has been no consensus on a single best approach. The silhouette coefficient or score is an example of an evaluation metric which indicates how similar samples within a cluster are, compared to other clusters. A higher silhouette score relates to a model with better-defined clusters.

The silhouette score

The silhouette score is a measure of how well-defined the clusters are. The **sklearn documentation** defines two metrics, **a** and **b**.

a is defined as the mean *intra*-cluster distance (the average of distances within a specific cluster), and **b** is defined as the mean *nearest*-cluster distance (the distance from a specific point to the nearest cluster). The computation for the silhouette score is (b - a) / max(a, b). This gives a value between -1 and 1, where -1 is the worst score and 1 is the best score.

If the silhouette score is close to -1, this means that there are many samples belonging to the wrong cluster. This is because **b** is smaller than **a**. Practically

speaking, it means that, for many points within a specific cluster, there is actually another cluster that is closer to that point than most of the rest of the cluster. This means that the point is not in the right cluster.

If the silhouette score is close to 0, this denotes overlapping clusters. Visualise two clusters that are basically next to each other: for about half the points there, the other cluster will be closer than the rest of its own cluster. **a** and **b** will be about the same at that stage, as the distances will be very similar.

If the silhouette score is close to 1, this means that the clusters are well-defined. This is because, in all cases, **b** will be much larger than **a**, as the nearest cluster will be so far away that it will always produce a positive number.

Extra resource

Check out **this blog**, written by one of HyperionDev's former Data Science graduates, which provides a detailed guide on performing **K-means clustering from scratch**.

Instructions

Read the **Kmeans.ipynb** Jupyter notebook in this task's folder before attempting the compulsory task.

Compulsory Task 1

Follow these steps:

- Open **Kmeans_task.ipynb**
- Load the **Country-data.csv** dataset.
- Drop any non-numeric columns from the dataset.
- Plot nine different scatter plots with different combinations of variables against GDPP. For example, GDPP vs health.
    - Note which of these plots looks the most promising for separating into clusters.
- Normalise the dataset using `MinMaxScaler` from sklearn.
- Find the optimal number of clusters using the elbow and silhouette score method.
- Fit the scaled dataset to the optimal number of clusters. Report back on the silhouette score of the model.
- Visualise the clusters for the following two groups:
    - Child mortality vs GDPP
    - Inflation vs GDPP
- Label the groups of countries in the plots you created based on child mortality, GDPP, and inflation. You may use **terms** such as: least developed, developing and developed, or low, low-middle, upper-middle and high income. Alternatively, simply rank them from highest to lowest. Justify the labels you assign to each group.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

REFERENCES

Lin, P. (2015). Ch10: Unsupervised Learning. Retrieved 28 August 2020, from **http://datasciencehc.github.io/Study-ISLR/Ch10_Unsupervised%20Learning/Ch10.html#(6)**