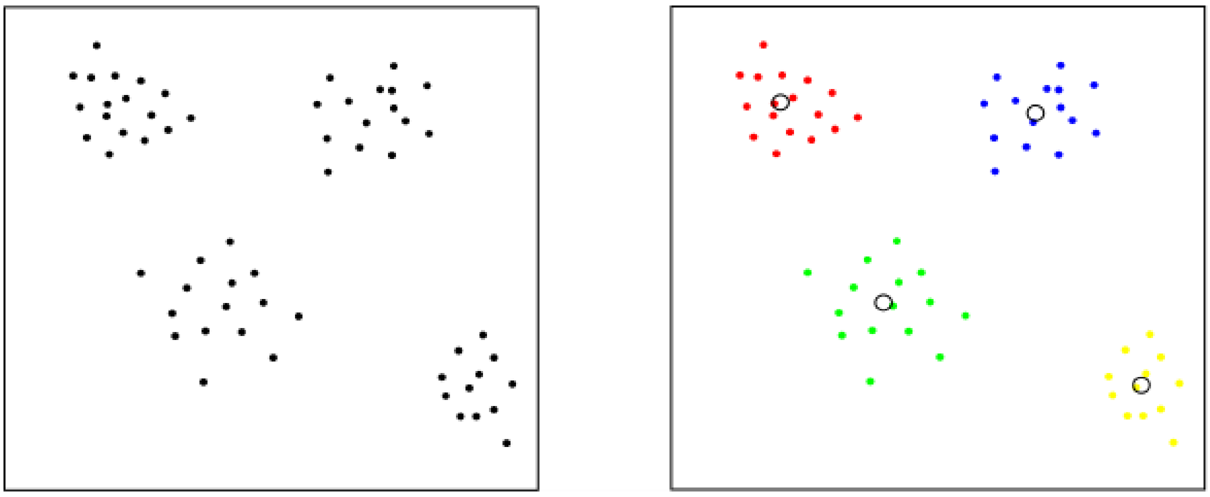
Clustering

Introduction

Let’s suppose we give a child different object to group. How does a child make a group? The child may group over the colour, over the shape, over the hardness or softness of the objects etc. The basic idea here is that the child tries to find out similarities and dissimilarities between different objects and then tries to make a group of similar objects. This is called **clustering**, the method of identifying similar instances and keeping them together. In Other words, clustering identifies homogeneous subgroups among the observations.

Clustering is an unsupervised approach which finds a structure/pattern in a collection of unlabelled data. A cluster is a collection of objects which are “similar” amongst themselves and are “dissimilar” to the objects belonging to a different cluster. For example:



In the figure above, we can easily identify 4 different clusters. The clustering criteria here is distance. Whichever points are near to each other are kept in the same cluster and the faraway points belong to a different cluster.

Goal of Clustering

The goal of clustering is to determine the intrinsic groups in unlabelled data. The question is: what constitutes a good cluster? It can be shown that there is no absolute “best” criterion for cluster validation. Consequently, it is the user who must supply the criterion for validating the cluster. For example, we might be interested in finding representatives of homogeneous instances for finding the “natural clusters” and identifying their unknown properties (like “natural” data types), for finding appropriate groupings or in finding unusual (which are different from all other data) data objects (outlier detection).

Applications

The scikit-learn book describes the various applications of clustering as follows:

- **For customer segmentation:** You can cluster your customers based on their purchases, their activity on your website, and so on. This is useful to understand who your customers are and what they need, so you can adapt your products and marketing campaigns to each segment. For example, this can be useful in recommender systems to suggest content that other users in the same cluster enjoyed.
- **For data analysis:** When analysing a new dataset, it is often useful to first discover clusters of similar instances, as it is often easier to analyse clusters separately.
- **As a dimensionality reduction technique:** Once a dataset has been clustered, it is usually possible to measure each instance's affinity with each cluster (affinity is any measure of how well an instance fits into a cluster). Each instance's feature vector x can then be replaced with the vector of its cluster affinities. If there are k clusters, then this vector is k dimensional. This is typically much lower dimensional than the original feature vector, but it can preserve enough information for further processing.
- **For anomaly detection (also called outlier detection):** Any instance that has a low affinity to all the clusters is likely to be an anomaly. For example, if you have clustered the users of your website based on their behaviour, you can detect users with unusual behaviour, such as an unusual number of requests per second, and so on. Anomaly detection is particularly useful in detecting defects in manufacturing, or for fraud detection.
- **For semi-supervised learning:** If you only have a few labels, you could perform clustering and propagate the labels to all the instances in the same cluster. This can greatly increase the number of labels available for a subsequent supervised learning algorithm, and thus improve its performance.
- **For search engines:** For example, some search engines let you search for images that are similar to a reference image. To build such a system, you would first apply a clustering algorithm to all the images in your database: similar images would end up in the same cluster. Then when a user provides a reference image, all you need to do is to find this image's cluster using the trained clustering model, and you can then simply return all the images from this cluster.
- **To segment an image:** By clustering pixels according to their color, then replacing each pixel's color with the mean color of its cluster, it is possible to reduce the number of different colors in the image considerably. This technique is used in many objects detection and tracking systems, as it makes it easier to detect the contour of each object.

Main Requirements

The primary requirements that should be met by a clustering algorithm are:

- It should be scalable
- It should be able to deal with attributes of different types;
- It should be able to discover arbitrary shape clusters;
- It should have an inbuilt ability to deal with noise and outliers;

- The clusters should not vary with the order of input records;
- It should be able to handle data of high dimensions.
- It should be easy to interpret and use.

K-Means Clustering

K-Means is a clustering approach in which the data is grouped into K distinct non-overlapping clusters based on their distances from the K centres. The value of **K** needs to be specified first and then the algorithm assigns the points to exactly one cluster.

1. Partitional clustering approach.
2. Each cluster is associated with a centroid (centre point).
3. Each point is assigned to the cluster with the closest centroid.
4. Number of clusters, K, must be specified
5. The objective is to minimize the sum of distances of the points to their respective centroid.

Theory

The theory discussed above can be mathematically expressed as:

- Let C1, C2, Ck be the K clusters
- Then we can write: C1UC2UC3U…U Ck = {1,2, 3…n} i.e., each datapoint has been assigned to a cluster.
- Also,

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

  This means that the clusters are non-overlapping.

- The idea behind the K-Means clustering approach is that the within-cluster variation amongst the point should be minimum. The within-cluster variance is denoted by: W(Ck). Hence, according to the statement above, we need to minimize this variance for all the clusters. Mathematically it can be written as:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}.$$

- The next step is to define the criterion for measuring the within-cluster variance. Generally, the criterion is the Euclidean distance between two data points.

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

- The above formula says that we are calculating the distances between all the point in a cluster, then we are repeating it for all the K clusters (That’s why two summation signs) and then we are dividing it by the number of observations in the clusters (Ck is the number of observations in the Kth cluster) to calculate the average.

So, ultimately our goal is to minimize:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}.$$

The following algorithm steps are used to solve this problem.

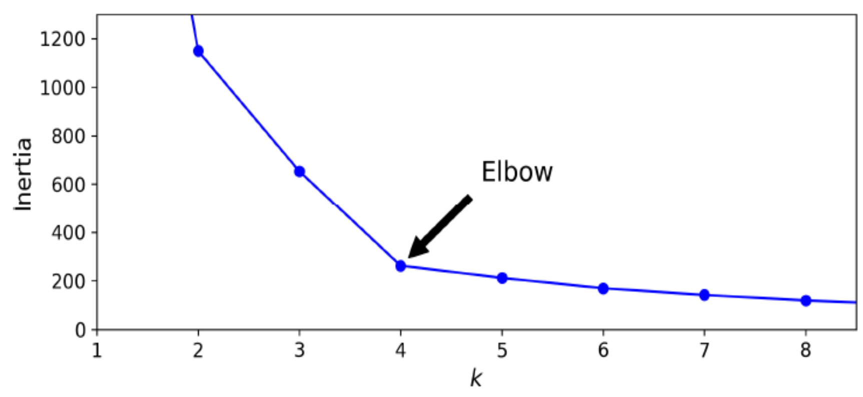
Algorithm Steps:

1. Randomly assign K centres.
2. Calculate the distance of all the points from all the K centres and allocate the points to cluster based on the shortest distance. The model’s *inertia* is the mean squared distance between each instance and its closest centroid. The goal is to have a model with the lowest intertia.
3. Once all the points are assigned to clusters, recompute the centroids.
4. Repeat the steps 2 and 3 until the locations of the centroids stop changing and the cluster allocation of the points becomes constant.

As we saw earlier, we need to provide the value of K beforehand. But the question is how to get a good value of K. An optimum value of K is obtained using the Elbow Method.

The Elbow-Method

This method is based on the relationship between the within-cluster sum of squared distances (WCSS Or Inertia) and the number of clusters. It is observed that first with an increase in the number of clusters WCSS decreases steeply and then after a certain number of clusters the drop in WCSS is not that prominent. The point after which the graph between WCSS and the number of clusters becomes comparatively smother is termed as the elbow and the number of clusters at that point are the optimum number of clusters as even after increasing the clusters after that point the variation is not decreasing by much i.e., we have accounted for almost all the dissimilarity in the data. An elbow-curve looks like:



Dealing with Initialization

- Do multiple runs and select the clustering with the smallest error.
- Select original set of points by methods other than random. E.g., pick the most distant (from each other) points as cluster centres (K-means++ algorithm).

Challenges and improvements in K-Means:

1. We need to specify the number of clusters beforehand.
2. It is required to run the algorithm multiple times to avoid a sub-optimal solution
3. K-Means does not behave very well when the clusters have varying sizes, different densities, or non-spherical shapes.
4. The clusters sometimes vary based on the initial choice of the centroids. An important improvement to the K-Means algorithm, called **K-Means++**, was proposed in a *2006 paper by David Arthur and Sergei Vassilvitskii*. They introduced a smarter initialization step that tends to select centroids that are distant from one another, and this makes the K-Means algorithm much less likely to converge to a suboptimal solution.
5. Another important improvement to the K-Means algorithm was proposed in a *2003 paper by Charles Elkan*. It considerably accelerates the algorithm by avoiding many unnecessary distance calculations: this is achieved by exploiting the *triangle inequality* (i.e., the straight line is always the shortest; in a triangle with sides a, b and c=> **a+b>c**) and by keeping track of lower and upper bounds for distances between instances and centroids.

Hierarchical clustering

One main disadvantage of K-Means is that it needs us to pre-enter the number of clusters (K). Hierarchical clustering is an alternative approach which does not need us to give the value of K beforehand and also, it creates a beautiful tree-based structure for visualization.

Here, we are going to discuss the bottom-up (or Agglomerative) approach of cluster building. We start by defining any sort of similarity between the datapoints. Generally, we consider the Euclidean distance. The points which are closer to each are more similar than the points which re farther away. The Algorithms starts with considering all points as separate clusters and then grouping pints together to form clusters.

The Algorithm:

1. Begin with n observations and a measure (such as Euclidean distance) of all the n(n−1)/2 pairwise dissimilarities (or the Euclidean distances generally). Treat each observation as its own cluster. Initially, we have n clusters.
2. Compare all the distances and put the two closest points/clusters in the same cluster. The dissimilarity (or the Euclidean distances) between these two clusters indicates the height in the dendrogram at which the fusion line should be placed.
3. Compute the new pairwise inter-cluster dissimilarities (or the Euclidean distances) among the remaining clusters.
4. Repeat steps 2 and 3 till we have only one cluster left.

The idea behind linkage clustering, or hierarchical clustering, is to put things that are close together into the same cluster. Linkage clustering is usually based on distances only.

Linkage Methods

Since clusters are sets of points, there are many different kinds of linkage methods:

- Single Linkage: cluster distance = smallest pairwise distance
- Complete Linkage: cluster distance = largest pairwise distance
- Average Linkage: cluster distance = average pairwise distance
- Centroid Linkage: cluster distance= distance between the centroids of the clusters
- Ward’s Linkage: cluster criteria= Minimize the variance in the cluster

Single Linkage: Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.

- cluster distance is the smallest distance between any point in cluster 1 and any point in cluster 2
- highly sensitive to outliers when forming flat clusters
- works well for low-noise data with an unusual structure

Complete Linkage: Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the largest of these dissimilarities.

- cluster distance is the largest distance between any point in cluster 1 and any point in cluster 2
- less sensitive to outliers than single linkage

Average Linkage: Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the average of these dissimilarities.

- cluster distance is the average distance of all pairs of points in clusters 1 and 2

Centroid Linkage: The dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable inversions.

- cluster distance is the distance of the centroids of both clusters

Ward linkage: Wikipedia says *Ward's minimum variance criterion minimizes the total within-cluster variance. To implement this method, at each step find the pair of clusters that leads to minimum increase in total within-cluster variance after merging.*

- based on minimizing a variance criterion before and after merging

DBSCAN (Density Based Spatial Clustering of Applications with Noise)

It is an unsupervised machine learning algorithm. This algorithm defines clusters as continuous regions of high density.

Some definitions first:

Epsilon: This is also called eps. This is the distance till which we look for the neighbouring points.

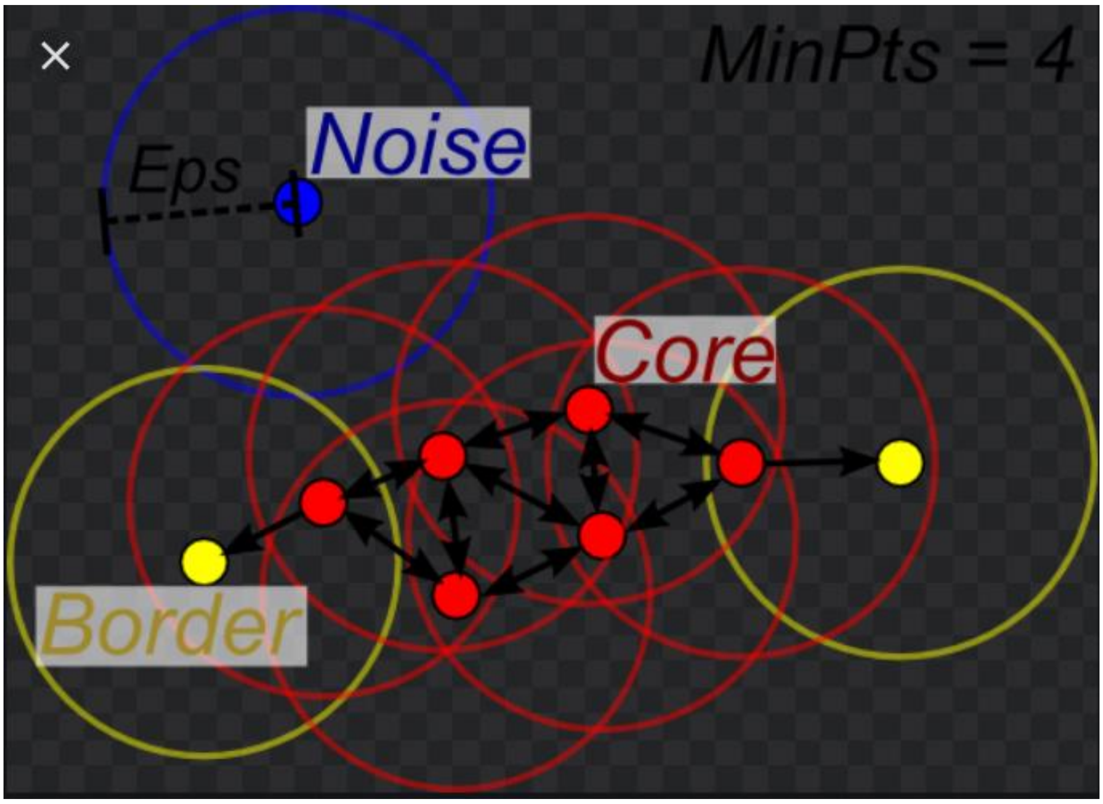
Min_points: The minimum number of points specified by the user.

Core Points: If the number of points inside the *eps radius* of a point is greater than or equal to the *min_points* then it's called a core point.

Border Points: If the number of points inside the *eps radius* of a point is less than the *min_points* and it lies within the *eps radius* region of a core point, it's called a border point.

Noise: A point which is neither a core nor a border point is a noise point.

Let's say if the eps=1 and min_points =4



Algorithm Steps:

1. The algorithm starts with a random point in the dataset which has not been visited yet and its neighbouring points are identified based on the eps value.
2. If the point contains greater than or equal points than the min_pts, then the cluster formation starts and this point becomes a *core point*, else it's considered as noise. The thing to note here is that a point initially classified as noise can later become a border point if it's in the eps radius of a core point.
3. If the point is a core point, then all its neighbours become a part of the cluster. If the points in the neighbourhood turn out to be core points, then their neighbours are also part of the cluster.
4. Repeat the steps above until all points are classified into different clusters or noises.

This algorithm works well if all the clusters are dense enough, and they are well separated by low-density regions.

In short, DBSCAN is a very simple yet powerful algorithm, capable of identifying any number of clusters, of any shape, it is robust to outliers, and it has just two hyper parameters (eps and min_samples). However, if the density varies significantly across the clusters, it can be impossible for it to capture all the clusters properly. Moreover, its computational complexity is roughly O (m log m), making it pretty close to linear with regards to the number of instances.

Evaluation of Clustering

Cluster Validity

The validation of clusters created is a troublesome task. The problem here is:” clusters are in the eyes of the beholder”

A good cluster will have:

- High inter-class similarity, and
- Low intraclass similarity

Performance Measure

Silhouette is a measure of how a clustering algorithm has performed. After computing the silhouette coefficient of each point in the dataset, plot it to get a visual representation of how well the dataset is clustered into k clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like the number of clusters visually. This measure has a range of [-1, 1]. There are two aspects to measure it.

- **Cohesion:** How closely the objects in the same cluster are related to each other. It is the within-cluster sum of squared distances. It is the same metric that we used to calculate for the K-Means algorithm. $WCSS=\sum\sum(x-mi)2$
- **Separation:** How different the objects in different clusters are and how distinct a well-separated cluster is from other clusters. It is the between cluster sum of squared distances. $BSS=\sum Ci(m-mi)2$

Where C is the size of the individual cluster and m is the centroid of all the data points.

Note: BSS+WSS is always a constant.

The silhouette can be calculated as:

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

$s(x)$ = [-1, +1]: -1=bad, 0=indifferent, 1=good

Where a(x) is the average distance of x from all the other points in the same cluster and b(x) is the average distance of x from all the other points in the other clusters.

Important Points:
The Silhouette coefficient of +1 indicates that the sample is far away from the neighboring clusters.
The Silhouette coefficient **of 0** indicates that the sample is on or very close to the decision boundary between two neighboring clusters.
Silhouette coefficient **<0** indicates that those samples might have been assigned to the wrong cluster or are outliers.

Overall, a higher silhouette score means that the inter cluster similarity is less and the intra cluster dissimilarity is more which is the measure of good clustering.