CLUSTERING

Emong Paul

Musema Hamzah

Obudra Ceaser

Muni University, P.O Box 725 Arua

November 3, 2023

Definition 1.1

Clustering (or cluster analysis) is a category of unsupervised learning techniques that allows us to discover hidden structures or information in data. It allows us to find groups of similar objects that are more related to each other than to objects in other groups.

Examples of business-oriented applications of clustering

1. Grouping of documents, music, and movies by different topics.
2. Finding customers that share similar interests based on common purchase behaviors as a basis for recommendation engines.

- The k-means algorithm belongs to the category of prototype-based clustering.
- Prototype-based clustering means that each cluster is represented by a prototype.
- A prototype can either be the centroid (average) of similar points with continuous features, or the medoid (the most representative or the point that minimizes the distance to all other points that belong to a particular cluster) in the case of categorical features.

- While k-means is very good at identifying clusters with a spherical shape, one of the drawbacks of this clustering algorithm is that we have to specify the number of clusters, k , a priori.
- An inappropriate choice for k can result in poor clustering performance.
- Later in this presentation, we will discuss the elbow method and silhouette plots, which are useful techniques to evaluate the quality of a clustering to help us determine the optimal number of clusters, k.

- k-means clustering can be applied to data in higher dimensions.
- Given the dataset, our goal is to group the examples based on their feature similarities.
- This can be achieved using the k-means algorithm, as summarized by the following four steps:

1. Randomly pick k centroids from the examples as initial cluster centers.
2. Assign each example to the nearest centroid, $\mu^{(j)},\ j \in \{1, \cdots,\ \mathrm{k}\}$.
3. Move the centroids to the center of the examples that were assigned to it.
4. Repeat steps 2 and 3 until the cluster assignments do not change or a user- defined tolerance or maximum number of iterations is reached.

Question: *How do we measure similarity between objects ?*

- We can define similarity as the opposite of distance.
- A commonly used distance for clustering examples with continuous features is the squared Euclidean distance between two points, x and y, in m-dimensional space:

$$d(x,y)^2 = \sum_{j=1}^{m}(x_j - y_j)^2 = ||x - y||_2^2 \tag{1}$$

- From equation (1), the index j refers to the jth dimension (feature column) of the example inputs, x and y.

- Based on this Euclidean distance metric, we can describe the k-means algorithm as a simple optimization problem.
- It's an iterative approach for minimizing the within-cluster sum of squared errors (SSE), which is sometimes also called cluster inertia:

$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)} ||x^{(i)} - \mu^{(j)}||_2^2 \qquad (2)$$

- Here, $\mu^{(j)}$ is the representative point (centroid) for cluster j.

- $w^{(i,j)} = 1$ if the example, $x^{(i)}$, is in cluster j, or 0 otherwise.
- Now that you have learned how the simple k-means algorithm works,
- let's apply it to our example dataset using the KMeans class from scikit-learn's cluster module (see Android-ML-task.ipynb).

A smarter way of placing the initial cluster centroids using k-means++

- So far, we have discussed the classic k-means algorithm, which uses a random seed to place the initial centroids.
- This can sometimes result in bad clusterings or slow convergence if the initial centroids are chosen poorly.
- To address this issue, we do the following:

1. Run the k-means algorithm multiple times on a dataset and choose the best performing model in terms of the SSE.
2. Another strategy is to place the initial centroids far away from each other via the k-means++ algorithm, which leads to better and more consistent results than the classic k-means.

The initialization in k-means++ can be summarized as follows:

1. Initialize an empty set, M, to store the k centroids being selected.
2. Randomly choose the first centroid, $\mu^{(j)}$, from the input examples and assign it to M.
3. For each example, $x^{(i)}$, that is not in M, find the minimum squared distance, $d(x^{(i)}, M)^2$, to any of the centroids in M.
4. To randomly select the next centroid, $\mu^{(p)}$, use a weighted probability distribution equal to $\frac{d(\mu^{(p)}, M)^2}{\sum_i d(x^{(i)}, M)^2}$.

5. Repeat steps 2 and 3 until k centroids are chosen.
6. Proceed with the classic k-means algorithm.

- To use k-means++ with scikit-learn's KMeans object, we just need to set the init parameter to 'k-means++'.
- In fact, 'k-means++' is the default argument to the init parameter, which is strongly recommended in practice (see Android-ML-task.ipynb).

Using the elbow method to find the optimal number of clusters

- To quantify the quality of clustering, we need to use intrinsic metrics such as the within-cluster SSE (distortion) to compare the performance of different k-means clusterings.
- Conveniently, we don't need to compute the within-cluster SSE explicitly when we are using scikit-learn, as it is already accessible via the inertia attribute after fitting a KMeans model.

- Based on the within-cluster SSE, we can use a graphical tool, the so-called elbow method, to estimate the optimal number of clusters, k, for a given task.
- We can say that if k increases, the distortion will decrease. . This is because the examples will be closer to the centroids they are assigned to.

- The idea behind the elbow method is to identify the value of k where the distortion begins to increase most rapidly.
- This will become clearer if we plot the distortion for different values of k (see Android-ML-task.ipynb).

Quantifying the quality of clustering via silhouette plots

- Another intrinsic metric to evaluate the quality of a clustering is silhouette analysis, which can also be applied to clustering algorithms other than k-means, which we will discuss later in this presentation.
- Silhouette analysis can be used as a graphical tool to plot a measure of how tightly grouped the examples in the clusters are.
- To calculate the silhouette coefficient of a single example in our dataset, we can apply the following three steps:

1. Calculate the cluster cohesion, $a^{(i)}$, as the average distance between an example, $x^{(i)}$, and all other points in the same cluster.
2. Calculate the cluster separation, $b^{(i)}$, from the next closest cluster as the average distance between the example, $x^{(i)}$, and all examples in the nearest cluster.
3. Calculate the silhouette, $s^{(i)}$, as the difference between cluster cohesion and separation divided by the greater of the two, as shown in the following equation:

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max[b^{(i)}, a^{(i)}]}. \tag{3}$$

- The silhouette coefficient is bounded in the range –1 to 1.
- Based on equation (3), we can see that the silhouette coefficient is 0 if the cluster separation and cohesion are equal $(b^{(i)} = a^{(i)})$.
- Furthermore, we get close to an ideal silhouette coefficient of 1 if $b^{(i)} \gg a^{(i)}$.

- Here, $b^{(i)}$ quantifies how dissimilar an example is from other clusters, and $a^{(i)}$ tells us how similar it is to other examples in its own cluster.
- The silhouette coefficient is available as silhouette-samples from scikit-learn's metric module, and optionally, the silhouette-scores function can be imported for convenience.
- The silhouette-scores function calculates the average silhouette coefficient across all examples.

Organizing clusters as a hierarchical tree

- Let us take a look at an alternative approach to prototype-based clustering: hierarchical clustering.
- Hierarchial clustering is used to group the data into hierarchy or tree-like structure.
- For example, in a machine learning problem of organizing academic staff/teaching staff of a university,

- within each department, the academic staff/teaching staff can be grouped according to their position such as Professor, Associate Professor, Senior Lecturer, Lecturer, Assistant Lecturer, and Teaching Assistant.
- This creates a hierarchial structure of the academic staff/teaching staff data and eases visualization and analysis.
- Similarly, there may be a data set which has an underlying hierarchy structure, that can use the hierarchial clustering to achieve that.

Advantages of hierarchial clustering

- It allows us to plot dendrograms (visualizations of a binary hierarchical clustering), which can help with the interpretation of the results by creating meaningful taxonomies.
- We do not need to specify the number of clusters upfront.

Approaches of hierarchial clustering

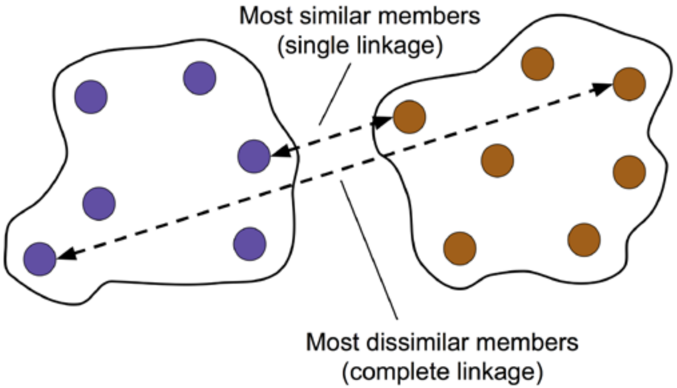
- The two main approaches to hierarchical clustering are agglomerative and divisive hierarchical clustering.
- In divisive hierarchical clustering, we start with one cluster that encompasses the complete dataset, and we iteratively split the cluster into smaller clusters until each cluster only contains one example.

- In this presentation, we will focus on agglomerative clustering, which takes the opposite approach.
- In agglomerative clustering, we start with each example as an individual cluster and merge the closest pairs of clusters until only one cluster remains.

Grouping clusters in bottom-up fashion

- The two standard algorithms for agglomerative hierarchical clustering are **single linkage** and **complete linkage**.
- Using single linkage, we compute the distances between the most similar members for each pair of clusters and merge the two clusters for which the distance between the most similar members is the smallest.

- The complete linkage approach is similar to single linkage but, instead of comparing the most similar members in each pair of clusters, we compare the most dissimilar members to perform the merge.
- This is shown in the following figure:


Most similar members
(single linkage)
Most dissimilar members
(complete linkage)

- In this presentation, we will focus on agglomerative clustering using the complete linkage approach.
- Hierarchical complete linkage clustering is an iterative procedure that can be summarized by the following steps:

1. Compute the distance matrix of all examples.
2. Represent each data point as a singleton cluster.
3. Merge the two closest clusters based on the distance between the most dissimilar (distant) members.
4. Update the similarity matrix.
5. Repeat steps 2-4 until one single cluster remains.

- Next, we will discuss how to compute the distance matrix (step 1).
- But first,let's generate a random data sample to work with.
- The rows represent different observations (IDs 0-4), and the columns are the different features (X, Y, Z) of those examples (see Android-ML-task.ipynb).

Locating regions of high density via DBSCAN

- This type of clustering algorithm does not make assumptions about spherical clusters like k-means, nor does it partition the dataset into hierarchies that require a manual cut-off point.
- As its name implies, density-based clustering assigns cluster labels based on dense regions of points.
- In DBSCAN, the notion of density is defined as the number of points within a specified radius, ϵ.
- In the DBSCAN algorithm, a special label is assigned to each example (data point) using the following criteria:

(a) A point is considered a core point if at least a specified number (MinPts) of neighboring points fall within the specified radius, ϵ.

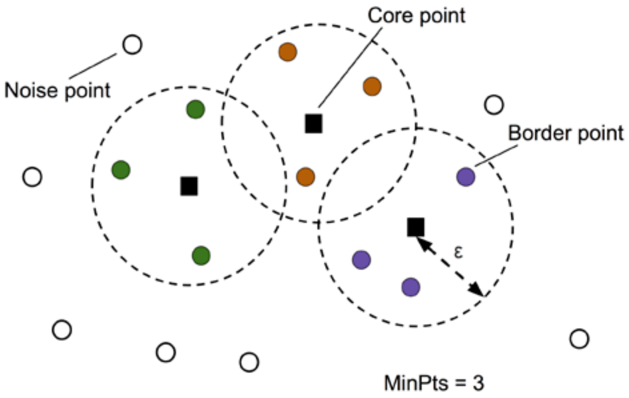
(b) A border point is a point that has fewer neighbors than MinPts within ϵ, but lies within the ϵ radius of a core point.

(c) All other points that are neither core nor border points are considered noise points.

After labeling the points as core, border, or noise, the DBSCAN algorithm can be summarized in two simple steps:

1. Form a separate cluster for each core point or connected group of core points. (Core points are connected if they are no farther away than ϵ).
2. Assign each border point to the cluster of its corresponding core point.

- To get a better understanding of what the result of DBSCAN can look like, before jumping to the implementation,
- let's summarize what we have just learned about core points, border points, and noise points in the following figure:


Core point
Noise point
Border point
ε
MinPts = 3

Advantages of DBSCAN algorithm

1. DBSCAN does not assume that the clusters have a spherical shape as in k-means.
2. DBSCAN is different from k-means and hierarchical clustering in that it doesn't necessarily assign each point to a cluster but is capable of removing noise points.

- For a more illustrative example, let's create a new dataset of half-moon-shaped structures to compare k-means clustering, hierarchical clustering, and DBSCAN (see Android-ML-task.ipynb).

Disadvantages of DBSCAN algorithm

1. With an increasing number of features in our dataset (assuming a fixed number of training examples), the negative effect of the curse of dimensionality increases. This is especially a problem if we are using the Euclidean distance metric. However, the problem of the curse of dimensionality is not unique to DBSCAN: it also affects other clustering algorithms that use the Euclidean distance metric, for example, k-means and hierarchical clustering algorithms.

2. In addition, we have two hyperparameters in DBSCAN (MinPts and ϵ) that need to be optimized to yield good clustering results.
3. Finding a good combination of MinPts and ϵ can be problematic if the density differences in the dataset are relatively large.

- In this presentation, we have discussed three different clustering algorithms that can help us with the discovery of hidden structures or information in data.
- We started this presentation with a prototype-based approach, k-means, which clusters examples into spherical shapes based on a specified number of cluster centroids.
- Since clustering is an unsupervised method, we do not enjoy the luxury of ground truth labels to evaluate the performance of a model.

- Thus, we used intrinsic performance metrics, such as the elbow method or silhouette analysis, as an attempt to quantify the quality of clustering.
- We then looked at a different approach to clustering: agglomerative hierarchical clustering.
- Hierarchical clustering does not require specifying the number of clusters upfront, and the result can be visualized in a dendrogram representation, which can help with the interpretation of the results.

- The last clustering algorithm that we covered in this presentation was DBSCAN, an algorithm that groups points based on local densities and is capable of handling outliers and identifying non-globular shapes.

Rashka, S., & Mirdzhalili, V. (2020). Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. *Birmingham, Mumbai. Packt.*

Thanks a lot for your attention!