Theory: Introduction to clustering

27 minutes 0 / 5 problems solved

Clustering is a type of *unsupervised machine learning*. Who is that "supervisor"? It is not a professor who knows everything, but the so-called "class" labels (often denoted as y in the classification tasks). In other words, clustering is a classification problem where we don't know the classes in advance. A paradox it is! So, clustering can also be called "automatic classification".

Clustering is a task of dividing data into sets, where each cluster conventionally consists of similar elements. How can we measure similarity? How many groups can there be? How to distinguish one group from another? All of these questions are important in clustering.

Clustering can also represent tools for data preprocessing. For example, clustering approaches can reduce data's dimensionality or classify specific structures. Besides, the problem of anomaly identification can also use clustering approaches. Clustering can be a preparatory stage for regression or classification problems.

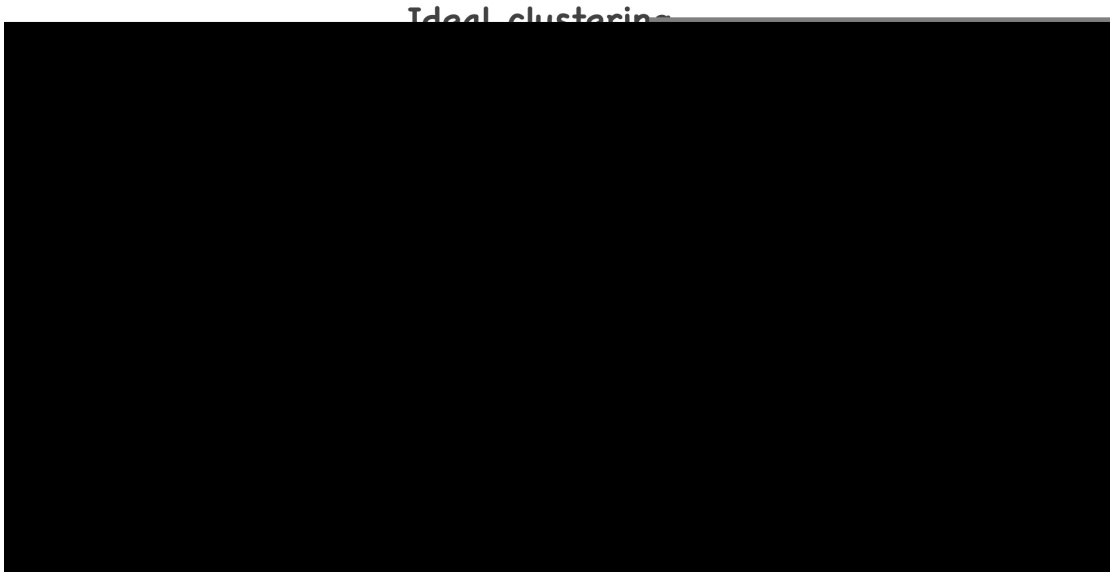
We classify and structure objects in the real world to obtain information, which is somewhat similar to clustering. Clustering is a powerful tool for many industries, from bioinformatics (looking for similar genomes) to marketing (finding clients with similar tastes). At the moment, clustering is used in image segmentation (computer vision), in the logistics for clustering locations, in the delivery of goods, in the advertising for recommendation systems. A lot of various applications. But first things first.

§1. Main Ideas

Clustering is a search for structures in the chaos; a non-trivial task. This is what Mendeleev once did, he could see the system in the chaos of chemical elements and organize them into a world-famous table.

We'll start with something a bit simpler than chemistry. Let's say you want to arrange your music on Spotify. You can order your music by genre. You can organize the music by artists. It is also convenient for someone to sort by years, everyone has their own ideas of order. As you can see from the example, there is no wrong approach.

Let's take a look at some of the following examples. What groups would you split this sample into?



Yes, it looks simple, but only at first glance. Look at the following examples and think about how you would divide the sample into clusters. How could you describe the rule by which the separation occurs? Could this rule be the same for the following examples?

Current topic:

Introduction to clustering

Topic depends on:

× Typical ML pipeline

Table of contents:

As you can see from the pictures, the structure itself is the main issue of clustering. It can be very diverse, and it may be quite hard to describe it. Moreover, most of the data is multidimensional and is difficult to visualize, so it complicates the issue because you cannot initially estimate which approach will be the most effective. Like any other machine learning problem, the data can be fuzzy. It can significantly change the model's performance and predictions. Also, the boundary between clusters can be implicit.
The initial number of classes is unknown, so you do not have an exact formulation of the problem. How to navigate through this uncertainty? We will tell you about this further.

§2. Similarity Metric

We used the term "similar objects" above. What is this "similarity"? How to compare two objects and understand that they are similar? How similar are Spotify and Apple Music? If they are similar, are the differences between Spotify and Apple Music, similar to the differences between Samsung's and iPhone's latest models?

In common practice, one takes a distance between objects as a similarity criterion. How to calculate the distance between Spotify and Apple Music? For example, let's represent all features (income, number of users, number of songs, etc.) as vectors and calculate the distance between these two vectors.

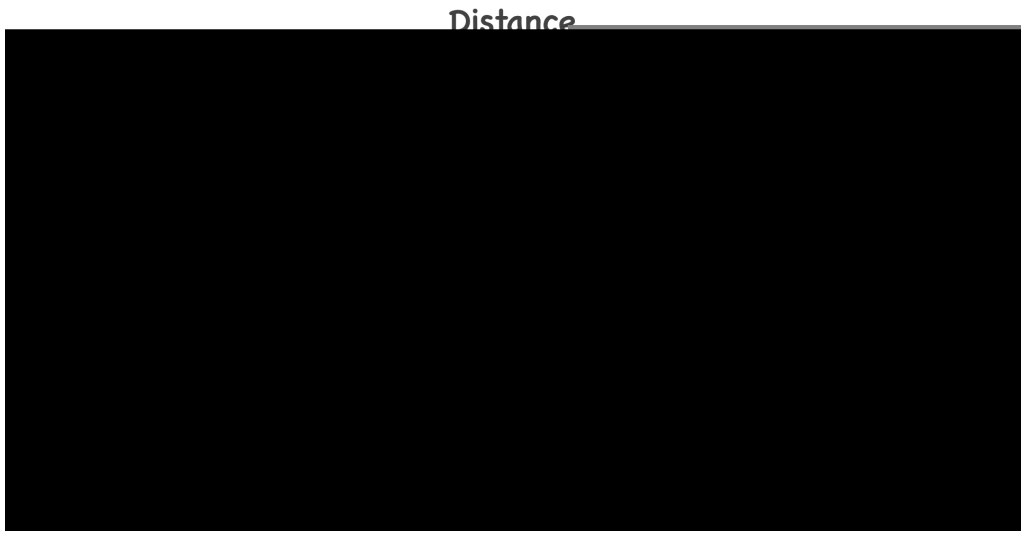
Let us recall without going into details that the distance between two vectors, which represent a set of features, can be very diverse. The most important distances are the *Euclidean* distance*, the Manhattan* distance, and the *Minkowski* distance*.*

For two vectors $\vec{x}$ and $\vec{y}$:

$$D_{Manhattan} = \sum |\left(x_i - y_i\right)|$$
$$D_{Euclidean} = \sqrt{\sum \left(x_i - y_i\right)^2}$$
$$D_{Minkowski} = (\sum |x_i - y_i|^p)^{\frac{1}{p}}$$

As you can see, the Euclidean distance and the Manhattan distance are the special cases of the Minkowski distance.

To understand the differences, take a look at the following picture:



Once we have chosen a specific metric, we can calculate the distance matrix, representing a measurement of the distance between objects. We should remember that the model's performance is highly dependent on the choice of distance, and the distance, in turn, strongly depends on the data and the task.

§3. Hierarchical Clustering

What model is the best for your task? Clustering models are based on very different theories and ideas, they are in abundance. However, we can roughly distinguish three big groups: algorithms associated with hierarchy (hierarchical algorithms), algorithms for optimizing distances (partitional algorithms), and algorithms related to density analysis (density-based clustering).

Hierarchical clustering is a convenient tool if you know that your data contains nested structures, for example, the directories in which movies are stored on Netflix. In turn, hierarchical algorithms are divided into two types. The agglomerative algorithms start with each element as a unique cluster and then join them into larger clusters, while the division algorithms begin with the entire sample and divide it into smaller clusters.

Let's start with division algorithms. It is an intuitive graph-based algorithm; we connect all points with edges, like graphs. We will exclude the greater edges until we are satisfied with the result. The disadvantage of this algorithm is immediately evident from its main logic — it is necessary to find a criterion to stop the replacement of graph edges. It also makes the algorithm very convenient for fine-tuning. Besides, you can configure more complex edge removal rules. But when there is a large amount of fuzzy data, this algorithm becomes very sensitive and imprecise.

On the other hand, we have agglomerative algorithms. It's like rain that consists of smaller drops, but after falling, each drop becomes a part of the puddle. This algorithm's idea is based on a similar principle: we consider each element as a separate cluster and then merge similar clusters into one. This approach is inherently the opposite of the division. It has the same disadvantages and advantages — it allows you to choose a stopping criterion to adjust the required degree of detail, but is very sensitive to noise. Besides, these algorithms are very fast, which is very important with big data.

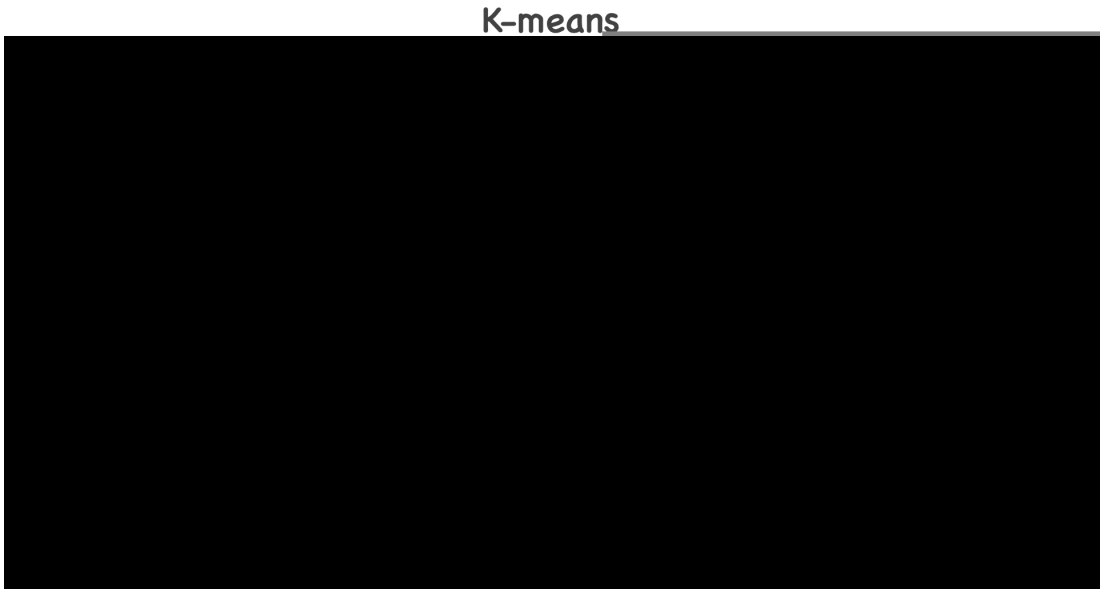
The picture below shows the different stages of hierarchical clustering and how the two approaches, agglomerative and division, mirror each other.

§4. Partitional Algorithms

Partitional algorithms usually determine all clusters at once. That means you need to set the number of clusters in advance. Usually, these algorithms optimize a distance metric between clusters.

The most important algorithm here is the K-means. The main idea is to find the center of the clusters and minimize each point's distance in this cluster to that center. You can think of it as hobby clubs (clusters) with a chairperson (the center). The algorithm works iteratively. At the first iteration, it randomly selects k points. It then calculates the mass center for each cluster, determining which points will be the next iteration center. We will look at the algorithm in more detail in another topic.

A significant disadvantage immediately catches the eye: you need to know in advance the number and the structure of the algorithms that should be centered. It also rarely converges to the global minimum, so it rarely finds the most optimal solution. Like many algorithms, it is very sensitive to noise. However, this algorithm is high-speed, and it is easy to understand, so the results are easy to interpret.

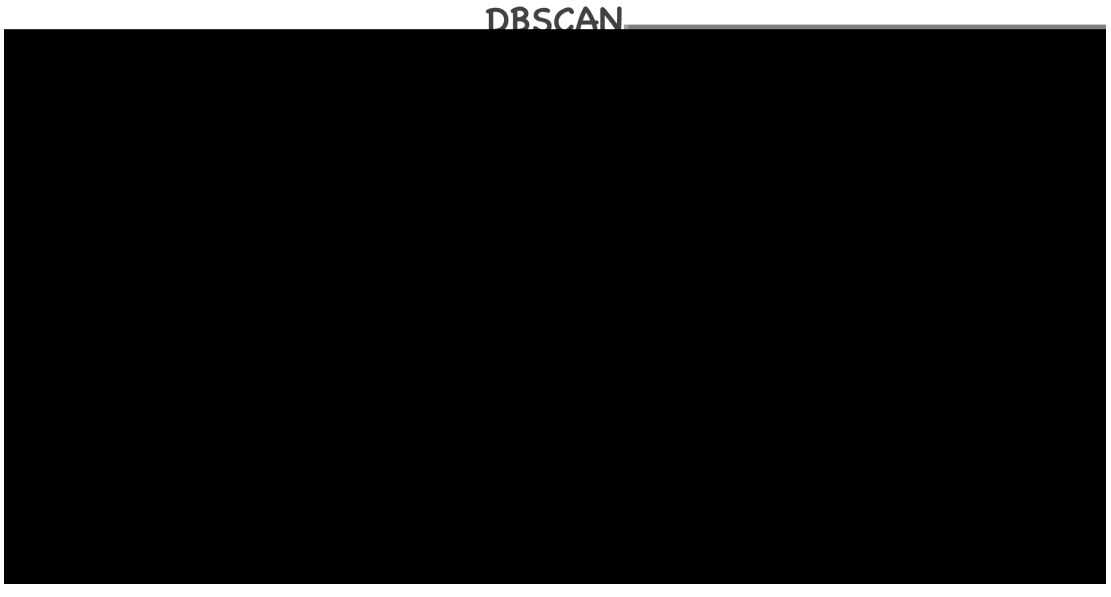


Similarly, with classification tasks, you can predict the probability — it's called **c-means** or **fuzzy** clustering. In this way, c-means can overcome one of the disadvantages of the k-means method — the random selection of cluster centers.

§5. Density-Based Clustering

Another approach is to look at the **density** of the samples. This is the method that works best with noisy data. It groups the points that stand together (they are called neighbors), and everything else is considered as noise. The threshold allows you to make the boundaries of the cluster more flexible.

One of the most successful implementations of this idea is called **DBSCAN**. It stands for **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise. One of the essential advantages of this algorithm is the resistance to noisy data and finding non-trivial clusters. In addition, as the name of the algorithm suggests, it is not necessary to know the exact number of clusters in advance.

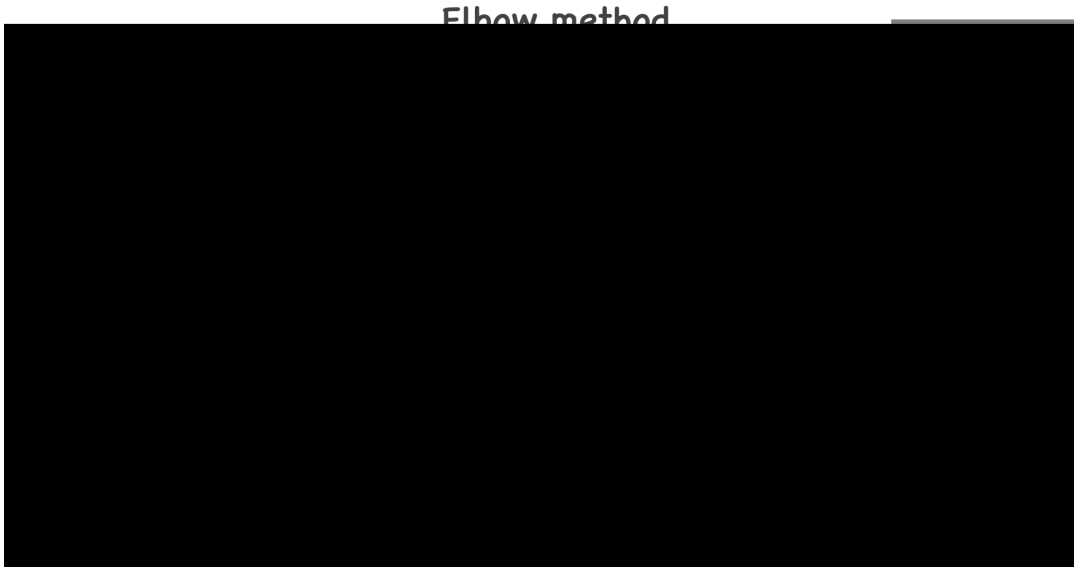


§6. Cluster Evaluation

Unfortunately, in contrast to classification, with clustering, we do not have the correct "labels" to check our models for reliability. Besides, as we discussed above, the issue is somewhat ambiguous — you often need to choose the method for calculating the distance and the number of clusters. Still, some approaches can help evaluate the model performance. The **elbow** method and the **silhouette** analysis are the two most popular.

Let's start with the **elbow** method. The idea behind the elbow method is to cluster with one method, using different assumptions about the number of clusters, and then calculate the sum of squared errors for each number. If

you draw a sum graph of the squared errors and the number of clusters, you should get a diagram that looks like a forearm, and the optimal number of clusters is located at the "elbow" joint. Why so? Because we want to choose a small value of clusters that has still a small sum of squared errors.



The *silhouette* analysis is a tool for determining the interface between the cluster boundaries or for measuring the cluster density. For finding the silhouette coefficient, we should calculate the D_1 distance — the average distance between a particular point and all other data points in the cluster, and D_2 — the minimum average distance from this point to all outside clusters.

$$Silhouette = \frac{D_2 - D_1}{\max(D_1, D_2)}$$

Depending on the value, the following conclusions can be drawn:

- -1: the result assigned to wrong clusters
- 0: the result indicates overlapping clusters
- 1: the result means the point is exactly within the same cluster to which it belongs.

§7. Summary

Clustering is an unsupervised machine learning task that splits data into clusters. It is also one of the most ambiguous problems in machine learning since most of the parameters are unknown and therefore must be narrowed down on our own. Different metrics of distance, the Euclidean distance, the Manhattan distance, and the Minkowski distance*,* can determine the similarity of two vectors.

Let's recall the main models, their pros and cons:

- Hierarchical algorithms (*agglomerative algorithms, division algorithms*) are based on the sequential joining or dividing points into clusters. Very handy for finding nested structures. They are quick to implement and can provide varying degrees of detail. But they are susceptible to noise, do not work well with complex-shaped clusters.
- Partitional algorithms (*K-means, C-means*) are based on a sequential search for cluster centers. They are very convenient for finding

spherical clusters of the same density. Fast to implement. They are susceptible to noise and to clusters of different density.
- Density-based clustering (*DBSCAN*) is based on the definition of a point cloud at a given density. Resistant to noise and clusters of different density. However, they can incorrectly interpret the data that contain no clusters.

To evaluate the performance of clustering, you can use the following quality metrics:

- The Elbow method — a method of finding the optimal number of clusters using the sum of squared errors.
- The Silhouette analysis — cluster density estimation method based on average distances.

Despite all the complexities, clustering is a potent tool that can be used to preprocess data for regression and classification. Also, keep in mind that the specifics of your task at hand can provide hints for fine-tuning clustering parameters. So let's practice it!

Report a typo

8 users liked this theory. **0** didn't like it. **What about you?**

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion