Clustering | Different Methods and Applications

ALGORITHM CLUSTERING DATA SCIENCE INTERMEDIATE MACHINE LEARNING R STRUCTURED DATA UNSUPERVISED

When encountering an unsupervised learning problem initially, confusion may arise as you aren't seeking specific insights but rather identifying data structures. This process, known as clustering or cluster analysis, identifies similar groups within a dataset. It is one of the most popular clustering techniques in data science used by data scientists. Entities in each group are comparatively more similar to entities of that group than those of the other groups. In this article, we'll go over the types of clustering, different clustering techniques, and a comparison between two of the most commonly used methods of clustering techniques in machine learning.

Learning Objectives

- Learn about Clustering in machine learning, one of the most popular unsupervised classification techniques.
- Get to know k-means and hierarchical clustering and the difference between the two.

Table of contents

What Is Clustering in Machine Learning?

Clustering techniques in machine learning is the task of dividing the unlabeled data or data points into different clusters such that similar data points fall in the same cluster than those which differ from the

others. In simple words, the aim of the clustering process is to segregate groups with similar traits and assign them into clusters.

Let's understand this with an example. Suppose you are the head of a rental store and wish to understand the preferences of your customers to scale up your business. Is it possible for you to look at the details of each customer and devise a unique business strategy for each one of them? Definitely not. But, what you can do is cluster all of your customers into, say 10 groups based on their purchasing habits and use a separate strategy for customers in each of these 10 groups. And this is what we call clustering methods.

Now that we understand what clustering is. Let's take a look at its different types.

Types of Clustering Techniques in Machine Learning

Clustering broadly divides into two subgroups:

- **Hard Clustering:** Each input data point either fully belongs to a cluster or not. For instance, in the example above, every customer is assigned to one group out of the ten.
- **Soft Clustering:** Rather than assigning each input data point to a distinct cluster, it assigns a probability or likelihood of the data point being in those clusters. For example, in the given scenario, each customer receives a probability of being in any of the ten retail store clusters.

Different Types of Clustering Models

Since the task of **clustering methods** is subjective, the means that can be used for achieving this goal are plenty. Every methodology follows a different set of rules for defining the '*similarity*' among data points. Following are the models that are commonly used:

Connectivity Models

As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start by classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lack scalability for handling big datasets. Examples of these models are the hierarchical clustering algorithms and their variants.

Read this article about the [data Exploration]

Centroid Models

These clustering algorithms iterate, deriving similarity from the proximity of a data point to the centroid or cluster center. The k-Means clustering algorithm, a popular example, falls into this category. These models necessitate specifying the number of clusters beforehand, requiring prior knowledge of the dataset. They iteratively run to discover local optima.

Distribution Models

These clustering models are based on the notion of how probable it is that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is the Expectation-maximization algorithm which uses multivariate normal distributions.

Density Models

These models search the data space for areas of the varied density of data points in the data space. They isolate different dense regions and assign the data points within these regions to the same cluster. Popular examples of density models are DBSCAN and OPTICS. These models are particularly useful for identifying clusters of arbitrary shape and detecting outliers, as they can detect and separate points that are located in sparse regions of the data space, as well as points that belong to dense regions.

Clustering Algorithms

Now let's go over two of the most popular clustering algorithms in detail: k-means and Hierarchical.

K-means Clustering

K-means is an iterative clustering algorithm that aims to find local maxima in each iteration. This algorithm works in these 5 steps:

Step1:

Specify the desired number of clusters K: Let us choose k=2 for these 5 data points in 2-D space.

Step 2:

Randomly assign each data point to a cluster: Let’s assign three points in cluster 1, shown using red color, and two points in cluster 2, shown using grey color.

Step 3:

Compute cluster centroids: The centroid of data points in the red cluster is shown using the red cross, and those in the grey cluster using a grey cross.

Step 4:

Re-assign each point to the closest cluster centroid: Note that only the data point at the bottom is assigned to the red cluster, even though it’s closer to the centroid of the grey cluster. Thus, we assign that data point to the grey cluster.

Step 5:

Re-compute cluster centroids: Now, re-computing the centroids for both clusters.

Repeat steps 4 and 5 until no improvements are possible: Similarly, we'll repeat the 4th and 5th steps until we'll reach global optima, i.e., when there is no further switching of data points between two clusters for two successive repeats. It will mark the termination of the algorithm if not explicitly mentioned.

Hierarchical Clustering

Hierarchical clustering methods, as the name suggests, is an algorithm that builds a hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.

The results of hierarchical clustering can be shown using a dendrogram. The dendrogram can be interpreted as:

At the bottom, we start with 25 data points, each assigned to separate clusters. The two closest clusters are then merged till we have just one cluster at the top. The height in the dendrogram at which two clusters are merged represents the distance between two clusters in the data space.

The decision of the no. of clusters that can best depict different groups can be chosen by observing the dendrogram. The best choice of the no. of clusters is the no. of vertical lines in the dendrogram cut by a horizontal line that can transverse the maximum distance vertically without intersecting a cluster.

In the above example, the best choice of no. of clusters will be 4 as the red horizontal line in the dendrogram below covers the maximum vertical distance AB.

Important Points for Hierarchical Clustering

- This algorithm has been implemented above using a bottom-up approach. It is also possible to follow a top-down approach starting with all data points assigned in the same cluster and recursively performing splits till each data point is assigned a separate cluster.
- The decision to merge two clusters is taken on the basis of the closeness of these clusters. There are multiple metrics for deciding the closeness of two clusters:
  - Euclidean distance: |a – b|_2 = \sqrt{\sum (a_i – b_i)^2}
  - Squared Euclidean distance: (|a – b|_2^2 = \sum (a_i – b_i)^2)
  - Manhattan distance: (|a – b|_1 = \sum |a_i – b_i|)
  - Maximum distance (Chebyshev): (|a – b|_{\infty} = \max |a_i – b_i|)
  - Mahalanobis distance: (\sqrt{(a – b)^T S^{-1} (a – b)})
    {where (S) is the covariance matrix}

Difference Between k-means and Hierarchical Clustering

- Hierarchical clustering methods can’t handle big data well, but k-means can. This is because the time complexity of k-means is linear, i.e., O(n), while that of hierarchical is quadratic, i.e., O(n2).
- Since we start with a random choice of clusters, the results produced by running the algorithm multiple times might differ in k-means clustering. While in Hierarchical clustering, the results are reproducible.
- K-means is found to work well when the shape of the clusters is hyperspherical (like a circle in 2D or a sphere in 3D).

- K-means clustering requires prior knowledge of K, i.e., no. of clusters you want to divide your data into. But, you can stop at whatever number of clusters you find appropriate in hierarchical clustering by interpreting the dendrogram.

Improving Supervised Learning Algorithms With Clustering

Clustering is an unsupervised machine learning approach, but can it be used to improve the accuracy of supervised machine learning algorithms as well by clustering the data points into similar groups and using these cluster labels as independent variables in the supervised machine learning algorithm? Let's find out!

For demonstration purposes we would be using synthetically created data using the following code:

```
import pandas as pd import numpy as np np.random.seed(101) X = np.random.randn(3000, 100) y = np.random.choice([-1, 1], 3000, p=[0.45, 0.55]) # Create DataFrame with proper categorical conversion data = pd.DataFrame(X, columns=[f'feature_{i+1}' for i in range(100)]) data['Y'] = pd.Categorical(y) # Correct categorical conversion[1][5] data.to_csv('train.csv', index=False)
```

Let's apply random forest without clustering:

```
from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import roc_auc_score, confusion_matrix, precision_score, recall_score, f1_score # Load data data = pd.read_csv("train.csv") data['Y'] = data['Y'].astype('category') # Split into train and test sets train = data.iloc[:2000] test = data.iloc[2000:] # Fit Random Forest rf = RandomForestClassifier(random_state=42) rf.fit(train.drop('Y', axis=1), train['Y']) # Predict probabilities and classes probs = rf.predict_proba(test.drop('Y', axis=1))[:, 1] preds = rf.predict(test.drop('Y', axis=1)) # Evaluation metrics auc = roc_auc_score(test['Y'].cat.codes, probs) cm = confusion_matrix(test['Y'], preds) precision = precision_score(test['Y'], preds, pos_label=1) recall = recall_score(test['Y'], preds, pos_label=1) f1 = f1_score(test['Y'], preds, pos_label=1) print(classification_report(test['Y'], preds))
```

Output:

```
precision recall f1-score support

-1 0.41 0.22 0.29 437
1 0.55 0.75 0.64 563

accuracy 0.52 1000
macro avg 0.48 0.48 0.46 1000
weighted avg 0.49 0.52 0.48 1000
```

Now let's create five clusters based on values of independent variables using k-means and reapply random forest.

```
from sklearn.cluster import KMeans from sklearn.metrics import classification_report # Combine train and test for clustering combined = pd.concat([train, test], ignore_index=True) # Fit KMeans (excluding Y) kmeans = KMeans(n_clusters=5, random_state=42, n_init=10) combined['cluster'] = kmeans.fit_predict(combined.drop('Y', axis=1)) # Split again train_new = combined.iloc[:2000] test_new = combined.iloc[2000:] # Fit Random Forest with cluster feature rf_cluster = RandomForestClassifier(random_state=42) rf_cluster.fit(train_new.drop('Y', axis=1), train_new['Y']) # Predict probabilities and classes probs_cluster = rf_cluster.predict_proba(test_new.drop('Y', axis=1))[:, 1] preds_cluster = rf_cluster.predict(test_new.drop('Y', axis=1)) # Evaluation metrics auc_cluster = roc_auc_score(test_new['Y'].cat.codes, probs_cluster) cm_cluster = confusion_matrix(test_new['Y'],
```

```
preds_cluster) precision_cluster = precision_score(test_new['Y'], preds_cluster, pos_label=1) recall_cluster = recall_score(test_new['Y'], preds_cluster, pos_label=1) f1_cluster = f1_score(test_new['Y'], preds_cluster, pos_label=1) print(classification_report(test_new['Y'], preds_cluster))
```

Output:

```
precision recall f1-score support

-1 0.43 0.21 0.28 437
1 0.56 0.78 0.65 563

accuracy 0.53 1000
macro avg 0.49 0.50 0.47 1000
weighted avg 0.50 0.53 0.49 1000
```

In the above example, even though the final accuracy is poor, clustering has given our model a small boost. The increase in accuracy would be higher if the dataset is organic, and would have consistency in its data points.

This shows that clustering can indeed be helpful for supervised machine-learning tasks.

Applications of Clustering

Here are some applications of clustering:

- **Grouping genes or proteins in bioinformatics**: Clustering finds similar gene or protein expressions to uncover biological relationships.
- **Customer segmentation in marketing**: Clustering helps divide customers into distinct groups based on behavior and preferences.
- **Document or news article categorization**: It groups similar texts together to improve content organization and search efficiency.
- **Image compression by color quantization**: Clustering reduces the number of colors in an image to compress it without major quality loss.
- **Anomaly detection in network security**: It identifies unusual data patterns that could signal fraud or cyber threats.

Conclusion

In this article, we have discussed the various *clustering techniques in machine learning* and explored different ways of performing clustering. We came across the application of clustering for unsupervised learning in a large number of domains and also examined how *clustering techniques in machine learning* can improve the accuracy of a supervised machine learning algorithm.

Although clustering is easy to implement, you need to take care of some important aspects, like treating outliers in your data and making sure each cluster has a sufficient population. These aspects of clustering are dealt with in great detail in this article.

Hope you find this information on clustering machine learning insightful and valuable for your understanding of clustering in big data and its applications in cluster analysis!

Key Takeaways

- Clustering helps to identify patterns in data and is useful for exploratory data analysis, customer segmentation, anomaly detection, pattern recognition, and image segmentation.
- It is a powerful tool for understanding data and can help to reveal insights that may not be apparent through other methods of analysis.
- Its types include partition-based, hierarchical, density-based, and grid-based clustering.
- The choice of clustering algorithm and the number of clusters to use depend on the nature of the data and the specific problem at hand.

Note: *To learn more about clustering and other machine learning algorithms (both supervised and unsupervised) check out the following courses-*

- ***Applied Machine Learning Course***
- ***Certified AI & ML Blackbelt+ Program***

Frequently Asked Questions

Q1. What is clustering in machine learning?

A. Clustering in machine learning involves grouping similar data points together based on their features, allowing for pattern discovery without predefined labels.

Q2. What is clustering and its type?

A. Clustering is a method of unsupervised learning where data points are grouped based on similarity. Types include k-means, hierarchical, DBSCAN, and mean shift.

Q3. What is an example of clustering?

A. An example of clustering is customer segmentation, where a business groups customers based on purchasing behavior to tailor marketing strategies.

Q4. How does clustering work?

A. Clustering works by evaluating the distances or similarities between data points, then grouping them into clusters where intra-cluster similarity is maximized and inter-cluster similarity is minimized.

Article Url - https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

saurav kaushik

Saurav is a Data Science enthusiast, currently in the final year of his graduation at MAIT, New Delhi. He loves to use machine learning and analytics to solve complex data problems.