Machine Learning in Python

Unsupervised Learning - Clustering and Dimensionality Reduction

Cristian A. Marocico, A. Emin Tatar

Center for Information Technology
University of Groningen

Wednesday, July 9th 2025

Outline

Introduction to Unsupervised Learning

What is Unsupervised Learning? Definition

Introduction to Unsupervised Learning

What is Unsupervised Learning? Definition

In contrast to supervised learning, unsupervised learning does not use labeled data. Instead, it identifies patterns and structures in the data without predefined labels. This is particularly useful for clustering similar data points or reducing the dimensionality of the data.

Types of Unsupervised Learning

Types of Unsupervised Learning

Types of Unsupervised Learning

Types of Unsupervised Learning

Unsupervised learning can be broadly categorized into two main types:

- Dimensionality Reduction: Techniques that reduce the number of features while preserving the essential structure of the data.

Types of Unsupervised Learning

Types of Unsupervised Learning

Unsupervised learning can be broadly categorized into two main types:

- Dimensionality Reduction: Techniques that reduce the number of features while preserving the essential structure of the data.
- Clustering: Algorithms that group similar data points together based on their features.

Types of Unsupervised Learning

Types of Unsupervised Learning

Unsupervised learning can be broadly categorized into two main types:

- Dimensionality Reduction: Techniques that reduce the number of features while preserving the essential structure of the data.
- Clustering: Algorithms that group similar data points together based on their features.
- Anomaly Detection: Identifying rare items, events, or observations that raise suspicions by differing significantly from the majority of the data.

Dimensionality Reduction

Dimensionality Reduction

Dimensionality Reduction

Dimensionality Reduction

Dimensionality reduction techniques aim to reduce the number of features in a dataset while retaining its essential characteristics. Useful for visualizing high-dimensional data or improving the performance of machine learning models by reducing noise and computational complexity. Common dimensionality reduction techniques include:

- Principal Component Analysis (PCA): Projects the data onto a lower-dimensional space by maximizing the variance along the new axes (principal components).

Dimensionality Reduction

Dimensionality Reduction

Dimensionality reduction techniques aim to reduce the number of features in a dataset while retaining its essential characteristics. Useful for visualizing high-dimensional data or improving the performance of machine learning models by reducing noise and computational complexity. Common dimensionality reduction techniques include:

- Principal Component Analysis (PCA): Projects the data onto a lower-dimensional space by maximizing the variance along the new axes (principal components).
- Linear Discriminant Analysis (LDA): A supervised technique that finds a linear combination of features that separates two or more classes.

Dimensionality Reduction

Dimensionality Reduction

Dimensionality reduction techniques aim to reduce the number of features in a dataset while retaining its essential characteristics. Useful for visualizing high-dimensional data or improving the performance of machine learning models by reducing noise and computational complexity. Common dimensionality reduction techniques include:

- Principal Component Analysis (PCA): Projects the data onto a lower-dimensional space by maximizing the variance along the new axes (principal components).
- Linear Discriminant Analysis (LDA): A supervised technique that finds a linear combination of features that separates two or more classes.
- Autoencoders: Neural networks that learn to encode the input data into a lower-dimensional representation and then decode it back to the original space. They are particularly useful for learning complex, non-linear mappings.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction. It transforms the data into a new coordinate system where the greatest variance by any projection lies on the first coordinate (principal component), the second greatest variance on the second coordinate, and so on. This allows us to reduce the number of dimensions while retaining most of the information in the data.

Steps in PCA

Steps in PCA

Steps in PCA

Steps in PCA

The PCA process involves several key steps:

1. Standardization: Scale the data to have a mean of 0 and a standard deviation of 1 to ensure that all features contribute equally to the analysis.

Steps in PCA

Steps in PCA

The PCA process involves several key steps:

1. Standardization: Scale the data to have a mean of 0 and a standard deviation of 1 to ensure that all features contribute equally to the analysis.
2. Covariance Matrix Calculation: Compute the covariance matrix to understand how the features vary together.

Steps in PCA

Steps in PCA

The PCA process involves several key steps:

1. Standardization: Scale the data to have a mean of 0 and a standard deviation of 1 to ensure that all features contribute equally to the analysis.
2. Covariance Matrix Calculation: Compute the covariance matrix to understand how the features vary together.
3. Eigenvalue Decomposition: Calculate the eigenvalues and eigenvectors of the covariance matrix. The eigenvectors represent the directions of maximum variance, and the eigenvalues indicate the amount of variance in those directions.

Steps in PCA

Steps in PCA

The PCA process involves several key steps:

1. Standardization: Scale the data to have a mean of 0 and a standard deviation of 1 to ensure that all features contribute equally to the analysis.
2. Covariance Matrix Calculation: Compute the covariance matrix to understand how the features vary together.
3. Eigenvalue Decomposition: Calculate the eigenvalues and eigenvectors of the covariance matrix. The eigenvectors represent the directions of maximum variance, and the eigenvalues indicate the amount of variance in those directions.
4. Selecting Principal Components: Choose the top k eigenvectors (principal components) based on their corresponding eigenvalues, which capture the most variance in the data.

Steps in PCA

Steps in PCA

The PCA process involves several key steps:

1. Standardization: Scale the data to have a mean of 0 and a standard deviation of 1 to ensure that all features contribute equally to the analysis.
2. Covariance Matrix Calculation: Compute the covariance matrix to understand how the features vary together.
3. Eigenvalue Decomposition: Calculate the eigenvalues and eigenvectors of the covariance matrix. The eigenvectors represent the directions of maximum variance, and the eigenvalues indicate the amount of variance in those directions.
4. Selecting Principal Components: Choose the top k eigenvectors (principal components) based on their corresponding eigenvalues, which capture the most variance in the data.
5. Transforming the Data: Project the original data onto the selected principal components to obtain a lower-dimensional representation.

Applications of PCA

Applications of PCA

Applications of PCA

Applications of PCA

PCA is widely used in various fields for:

- Data Visualization: Reducing high-dimensional data to 2D or 3D for visualization purposes.

Applications of PCA

Applications of PCA

PCA is widely used in various fields for:

- Data Visualization: Reducing high-dimensional data to 2D or 3D for visualization purposes.
- Noise Reduction: Filtering out noise by retaining only the most significant components.

Applications of PCA

Applications of PCA

PCA is widely used in various fields for:

- Data Visualization: Reducing high-dimensional data to 2D or 3D for visualization purposes.
- Noise Reduction: Filtering out noise by retaining only the most significant components.
- Feature Extraction: Creating new features that capture the essential patterns in the data.

Applications of PCA

Applications of PCA

PCA is widely used in various fields for:

- Data Visualization: Reducing high-dimensional data to 2D or 3D for visualization purposes.
- Noise Reduction: Filtering out noise by retaining only the most significant components.
- Feature Extraction: Creating new features that capture the essential patterns in the data.
- Preprocessing for Machine Learning: Reducing dimensionality before applying machine learning algorithms to improve performance and reduce overfitting.

Limitations of PCA

Limitations of PCA

Limitations of PCA

Limitations of PCA

While PCA is a powerful technique, it has some limitations:

- Linearity: PCA assumes linear relationships between features, which may not capture complex patterns in the data.

Limitations of PCA

Limitations of PCA

While PCA is a powerful technique, it has some limitations:

- Linearity: PCA assumes linear relationships between features, which may not capture complex patterns in the data.
- Interpretability: The principal components may not have a clear interpretation in terms of the original features, making it difficult to understand the results.

Limitations of PCA

Limitations of PCA

While PCA is a powerful technique, it has some limitations:

- Linearity: PCA assumes linear relationships between features, which may not capture complex patterns in the data.
- Interpretability: The principal components may not have a clear interpretation in terms of the original features, making it difficult to understand the results.
- Sensitivity to Scaling: PCA is sensitive to the scale of the features, so standardization is crucial.

Limitations of PCA

Limitations of PCA

While PCA is a powerful technique, it has some limitations:

- Linearity: PCA assumes linear relationships between features, which may not capture complex patterns in the data.
- Interpretability: The principal components may not have a clear interpretation in terms of the original features, making it difficult to understand the results.
- Sensitivity to Scaling: PCA is sensitive to the scale of the features, so standardization is crucial.
- Loss of Information: Reducing dimensions may lead to loss of important information, especially if too many components are discarded.

Evaluating PCA Results

Evaluating PCA Results

Evaluating PCA Results

Evaluating PCA Results

To evaluate the effectiveness of PCA, we can use several methods:

- Explained Variance Ratio: This metric indicates the proportion of variance explained by each principal component. A higher explained variance ratio suggests that the component captures more information from the data.

Evaluating PCA Results

Evaluating PCA Results

To evaluate the effectiveness of PCA, we can use several methods:

- Explained Variance Ratio: This metric indicates the proportion of variance explained by each principal component. A higher explained variance ratio suggests that the component captures more information from the data.
- Scree Plot: A graphical representation of the eigenvalues associated with each principal component. It helps to visualize the amount of variance explained by each component and to determine the optimal number of components to retain.

Evaluating PCA Results

Evaluating PCA Results

To evaluate the effectiveness of PCA, we can use several methods:

- Explained Variance Ratio: This metric indicates the proportion of variance explained by each principal component. A higher explained variance ratio suggests that the component captures more information from the data.
- Scree Plot: A graphical representation of the eigenvalues associated with each principal component. It helps to visualize the amount of variance explained by each component and to determine the optimal number of components to retain.
- Reconstruction Error: After transforming the data using PCA, we can reconstruct the original data and measure the error (e.g., mean squared error) to assess how well PCA captures the original structure of the data.

Autoencoders

Autoencoders

Autoencoders

Autoencoders

Autoencoders are a type of neural network used for unsupervised learning, particularly for dimensionality reduction and feature learning. They consist of two main parts: an encoder that compresses the input data into a lower-dimensional representation (latent space) and a decoder that reconstructs the original data from this representation.

Structure of Autoencoders

Structure of Autoencoders

Structure of Autoencoders

Structure of Autoencoders

Autoencoders typically have the following structure:

- Encoder: Maps the input data to a lower-dimensional latent space using one or more hidden layers. The encoder learns to capture the most important features of the data.

Structure of Autoencoders

Structure of Autoencoders

Autoencoders typically have the following structure:

- Encoder: Maps the input data to a lower-dimensional latent space using one or more hidden layers. The encoder learns to capture the most important features of the data.
- Latent Space: The compressed representation of the input data, which contains the essential features learned by the encoder.

Structure of Autoencoders

Structure of Autoencoders

Autoencoders typically have the following structure:

- Encoder: Maps the input data to a lower-dimensional latent space using one or more hidden layers. The encoder learns to capture the most important features of the data.
- Latent Space: The compressed representation of the input data, which contains the essential features learned by the encoder.
- Decoder: Reconstructs the original data from the latent space representation using one or more hidden layers. The decoder learns to reverse the encoding process.

Structure of Autoencoders

Structure of Autoencoders

Autoencoders typically have the following structure:

- Encoder: Maps the input data to a lower-dimensional latent space using one or more hidden layers. The encoder learns to capture the most important features of the data.
- Latent Space: The compressed representation of the input data, which contains the essential features learned by the encoder.
- Decoder: Reconstructs the original data from the latent space representation using one or more hidden layers. The decoder learns to reverse the encoding process.
- Loss Function: The autoencoder is trained to minimize the difference between the original input and the reconstructed output, typically using mean squared error or binary cross-entropy as the loss function.

Applications of Autoencoders

Applications of Autoencoders

Applications of Autoencoders

Applications of Autoencoders

Autoencoders have various applications in unsupervised learning:

- Dimensionality Reduction: Reducing the number of features while retaining important information, similar to PCA but capable of capturing non-linear relationships.

Applications of Autoencoders

Applications of Autoencoders

Autoencoders have various applications in unsupervised learning:

- Dimensionality Reduction: Reducing the number of features while retaining important information, similar to PCA but capable of capturing non-linear relationships.
- Feature Learning: Learning meaningful representations of data that can be used for downstream tasks such as classification or clustering.

Applications of Autoencoders

Applications of Autoencoders

Autoencoders have various applications in unsupervised learning:

- Dimensionality Reduction: Reducing the number of features while retaining important information, similar to PCA but capable of capturing non-linear relationships.
- Feature Learning: Learning meaningful representations of data that can be used for downstream tasks such as classification or clustering.
- Anomaly Detection: Identifying anomalies by reconstructing input data and measuring the reconstruction error; high errors indicate potential anomalies.

Applications of Autoencoders

Applications of Autoencoders

Autoencoders have various applications in unsupervised learning:

- Dimensionality Reduction: Reducing the number of features while retaining important information, similar to PCA but capable of capturing non-linear relationships.
- Feature Learning: Learning meaningful representations of data that can be used for downstream tasks such as classification or clustering.
- Anomaly Detection: Identifying anomalies by reconstructing input data and measuring the reconstruction error; high errors indicate potential anomalies.
- Data Denoising: Removing noise from data by training the autoencoder to reconstruct clean data from noisy inputs.

Applications of Autoencoders

Applications of Autoencoders

Autoencoders have various applications in unsupervised learning:

- Dimensionality Reduction: Reducing the number of features while retaining important information, similar to PCA but capable of capturing non-linear relationships.
- Feature Learning: Learning meaningful representations of data that can be used for downstream tasks such as classification or clustering.
- Anomaly Detection: Identifying anomalies by reconstructing input data and measuring the reconstruction error; high errors indicate potential anomalies.
- Data Denoising: Removing noise from data by training the autoencoder to reconstruct clean data from noisy inputs.
- Generative Models: Variational autoencoders (VAEs) can generate new data samples by sampling from the latent space, making them useful for generative tasks.

Limitations of Autoencoders

Limitations of Autoencoders

Limitations of Autoencoders

Limitations of Autoencoders

While autoencoders are powerful tools, they have some limitations:

- Training Complexity: Training autoencoders can be complex and may require careful tuning of hyperparameters, such as the architecture, learning rate, and regularization techniques.

Limitations of Autoencoders

Limitations of Autoencoders

While autoencoders are powerful tools, they have some limitations:

- Training Complexity: Training autoencoders can be complex and may require careful tuning of hyperparameters, such as the architecture, learning rate, and regularization techniques.
- Overfitting: Autoencoders can overfit the training data, especially if the model is too complex or the dataset is small. Regularization techniques such as dropout or weight decay can help mitigate this.

Limitations of Autoencoders

Limitations of Autoencoders

While autoencoders are powerful tools, they have some limitations:

- Training Complexity: Training autoencoders can be complex and may require careful tuning of hyperparameters, such as the architecture, learning rate, and regularization techniques.
- Overfitting: Autoencoders can overfit the training data, especially if the model is too complex or the dataset is small. Regularization techniques such as dropout or weight decay can help mitigate this.
- Interpretability: The learned representations in the latent space may not have a clear interpretation, making it difficult to understand the features captured by the autoencoder.

Limitations of Autoencoders

Limitations of Autoencoders

While autoencoders are powerful tools, they have some limitations:

- Training Complexity: Training autoencoders can be complex and may require careful tuning of hyperparameters, such as the architecture, learning rate, and regularization techniques.
- Overfitting: Autoencoders can overfit the training data, especially if the model is too complex or the dataset is small. Regularization techniques such as dropout or weight decay can help mitigate this.
- Interpretability: The learned representations in the latent space may not have a clear interpretation, making it difficult to understand the features captured by the autoencoder.
- Non-Deterministic: The results can vary between runs due to the stochastic nature of training, especially if random initialization is used.

Clustering

Clustering — Definition

Clustering

Clustering Definition

Clustering is an unsupervised learning technique that groups similar data points together based on their features. The goal is to identify inherent structures in the data without prior knowledge of labels.

Types of Clustering Algorithms

Types of Clustering Algorithms

Definition

Types of Clustering Algorithms

Types of Clustering Algorithms

Definition

Bases on their approach, clustering algorithms can be categorized into several types:

- Partitioning Methods: These algorithms divide the data into k clusters, where k is a predefined number. The most common example is k-means clustering.

Types of Clustering Algorithms

Types of Clustering Algorithms

Definition

Bases on their approach, clustering algorithms can be categorized into several types:

- Partitioning Methods: These algorithms divide the data into k clusters, where k is a predefined number. The most common example is k-means clustering.
- Hierarchical Methods: These algorithms create a hierarchy of clusters, either by merging smaller clusters into larger ones (agglomerative) or by splitting larger clusters into smaller ones (divisive).

Types of Clustering Algorithms

Types of Clustering Algorithms

Definition

Bases on their approach, clustering algorithms can be categorized into several types:

- Partitioning Methods: These algorithms divide the data into k clusters, where k is a predefined number. The most common example is k-means clustering.
- Hierarchical Methods: These algorithms create a hierarchy of clusters, either by merging smaller clusters into larger ones (agglomerative) or by splitting larger clusters into smaller ones (divisive).
- Density-Based Methods: These algorithms group data points based on the density of data points in the feature space. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular density-based clustering algorithm.

Types of Clustering Algorithms

Types of Clustering Algorithms

Definition

Bases on their approach, clustering algorithms can be categorized into several types:

- Partitioning Methods: These algorithms divide the data into k clusters, where k is a predefined number. The most common example is k-means clustering.
- Hierarchical Methods: These algorithms create a hierarchy of clusters, either by merging smaller clusters into larger ones (agglomerative) or by splitting larger clusters into smaller ones (divisive).
- Density-Based Methods: These algorithms group data points based on the density of data points in the feature space. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular density-based clustering algorithm.
- Model-Based Methods: These algorithms assume that the data is generated from a mixture of underlying probability distributions. Gaussian Mixture Models (GMMs) are a common example, where each cluster is represented by a Gaussian distribution.

k-Means Clustering

k-Means Clustering

k-Means Clustering

k-Means Clustering

k-Means clustering is a widely used partitioning method that aims to divide the data into k clusters, where k is a user-defined parameter. The algorithm iteratively assigns data points to the nearest cluster centroid and updates the centroids based on the assigned points.

Steps in k-Means Clustering

Steps in k-Means Clustering

Steps in k-Means Clustering

Steps in k-Means Clustering

The k-means clustering algorithm follows these steps:

1. Initialization: Randomly select k data points as initial cluster centroids.

Steps in k-Means Clustering

Steps in k-Means Clustering

The k-means clustering algorithm follows these steps:

1. Initialization: Randomly select k data points as initial cluster centroids.
2. Assignment Step: Assign each data point to the nearest centroid based on a distance metric (e.g., Euclidean distance).

Steps in k-Means Clustering

Steps in k-Means Clustering

The k-means clustering algorithm follows these steps:

1. Initialization: Randomly select k data points as initial cluster centroids.
2. Assignment Step: Assign each data point to the nearest centroid based on a distance metric (e.g., Euclidean distance).
3. Update Step: Calculate the new centroids by taking the mean of all data points assigned to each cluster.

Steps in k-Means Clustering

Steps in k-Means Clustering

The k-means clustering algorithm follows these steps:

1. Initialization: Randomly select k data points as initial cluster centroids.
2. Assignment Step: Assign each data point to the nearest centroid based on a distance metric (e.g., Euclidean distance).
3. Update Step: Calculate the new centroids by taking the mean of all data points assigned to each cluster.
4. Convergence Check: Repeat the assignment and update steps until the centroids do not change significantly or a maximum number of iterations is reached.

Applications of k-Means Clustering

Applications of k-Means Clustering

Applications of k-Means Clustering

Applications of k-Means Clustering

k-Means clustering is widely used in various applications:

- Customer Segmentation: Grouping customers based on purchasing behavior to tailor marketing strategies.

Applications of k-Means Clustering

Applications of k-Means Clustering

k-Means clustering is widely used in various applications:

- Customer Segmentation: Grouping customers based on purchasing behavior to tailor marketing strategies.
- Image Compression: Reducing the number of colors in an image by clustering similar colors together.

Applications of k-Means Clustering

Applications of k-Means Clustering

k-Means clustering is widely used in various applications:

- Customer Segmentation: Grouping customers based on purchasing behavior to tailor marketing strategies.
- Image Compression: Reducing the number of colors in an image by clustering similar colors together.
- Document Clustering: Grouping similar documents based on their content for information retrieval or topic modeling.

Applications of k-Means Clustering

Applications of k-Means Clustering

k-Means clustering is widely used in various applications:

- Customer Segmentation: Grouping customers based on purchasing behavior to tailor marketing strategies.
- Image Compression: Reducing the number of colors in an image by clustering similar colors together.
- Document Clustering: Grouping similar documents based on their content for information retrieval or topic modeling.
- Anomaly Detection: Identifying outliers by detecting data points that do not belong to any cluster.

Limitations of k-Means Clustering

Limitations of k-Means Clustering

Limitations of k-Means Clustering

Limitations of k-Means Clustering

While k-means clustering is a powerful technique, it has some limitations:

- Choosing k: The number of clusters (k) must be specified beforehand, which can be challenging if the optimal number is unknown.

Limitations of k-Means Clustering

Limitations of k-Means Clustering

While k-means clustering is a powerful technique, it has some limitations:

- Choosing k: The number of clusters (k) must be specified beforehand, which can be challenging if the optimal number is unknown.
- Sensitivity to Initialization: The algorithm can converge to different solutions based on the initial centroids, leading to inconsistent results. Techniques like k-means++ initialization can help mitigate this issue.

Limitations of k-Means Clustering

Limitations of k-Means Clustering

While k-means clustering is a powerful technique, it has some limitations:

- Choosing k: The number of clusters (k) must be specified beforehand, which can be challenging if the optimal number is unknown.
- Sensitivity to Initialization: The algorithm can converge to different solutions based on the initial centroids, leading to inconsistent results. Techniques like k-means++ initialization can help mitigate this issue.
- Assumption of Spherical Clusters: k-means assumes that clusters are spherical and equally sized, which may not hold true for all datasets.

Limitations of k-Means Clustering

Limitations of k-Means Clustering

While k-means clustering is a powerful technique, it has some limitations:

- Choosing k: The number of clusters (k) must be specified beforehand, which can be challenging if the optimal number is unknown.
- Sensitivity to Initialization: The algorithm can converge to different solutions based on the initial centroids, leading to inconsistent results. Techniques like k-means++ initialization can help mitigate this issue.
- Assumption of Spherical Clusters: k-means assumes that clusters are spherical and equally sized, which may not hold true for all datasets.
- Sensitivity to Outliers: k-means is sensitive to outliers, which can skew the centroids and affect the clustering results. Preprocessing steps like outlier removal or using robust distance metrics can help address this issue.

Hierarchical Clustering

Hierarchical Clustering

Hierarchical Clustering

Hierarchical Clustering

Hierarchical clustering is a method that builds a hierarchy of clusters, either by merging smaller clusters into larger ones (agglomerative) or by splitting larger clusters into smaller ones (divisive).

Types of Hierarchical Clustering

Types of Hierarchical Clustering

Types of Hierarchical Clustering

Types of Hierarchical Clustering

Hierarchical clustering can be categorized into two main types:

- Agglomerative Clustering: Starts with each data point as its own cluster and iteratively merges the closest clusters until a single cluster remains or a stopping criterion is met.

Types of Hierarchical Clustering

Types of Hierarchical Clustering

Hierarchical clustering can be categorized into two main types:

- Agglomerative Clustering: Starts with each data point as its own cluster and iteratively merges the closest clusters until a single cluster remains or a stopping criterion is met.
- Divisive Clustering: Starts with all data points in a single cluster and recursively splits the clusters into smaller ones until each data point is its own cluster or a stopping criterion is met.

Steps in Agglomerative Clustering

Steps in Agglomerative Clustering

Steps in Agglomerative Clustering

Steps in Agglomerative Clustering

The agglomerative clustering algorithm follows these steps:

1. Initialization: Start with each data point as its own cluster.

Steps in Agglomerative Clustering

Steps in Agglomerative Clustering

The agglomerative clustering algorithm follows these steps:

1. Initialization: Start with each data point as its own cluster.
2. Distance Calculation: Calculate the distance between all pairs of clusters using a distance metric (e.g., Euclidean distance).

Steps in Agglomerative Clustering

Steps in Agglomerative Clustering

The agglomerative clustering algorithm follows these steps:

1. Initialization: Start with each data point as its own cluster.
2. Distance Calculation: Calculate the distance between all pairs of clusters using a distance metric (e.g., Euclidean distance).
3. Merge Clusters: Identify the two closest clusters and merge them into a single cluster.

Steps in Agglomerative Clustering

Steps in Agglomerative Clustering

The agglomerative clustering algorithm follows these steps:

1. Initialization: Start with each data point as its own cluster.
2. Distance Calculation: Calculate the distance between all pairs of clusters using a distance metric (e.g., Euclidean distance).
3. Merge Clusters: Identify the two closest clusters and merge them into a single cluster.
4. Update Distances: Recalculate the distances between the newly formed cluster and the remaining clusters.

Steps in Agglomerative Clustering

Steps in Agglomerative Clustering

The agglomerative clustering algorithm follows these steps:

1. Initialization: Start with each data point as its own cluster.
2. Distance Calculation: Calculate the distance between all pairs of clusters using a distance metric (e.g., Euclidean distance).
3. Merge Clusters: Identify the two closest clusters and merge them into a single cluster.
4. Update Distances: Recalculate the distances between the newly formed cluster and the remaining clusters.
5. Repeat: Repeat steps 3 and 4 until a stopping criterion is met, such as a predefined number of clusters or a distance threshold.

Applications of Hierarchical Clustering

Applications of Hierarchical Clustering

Applications of Hierarchical Clustering

Applications of Hierarchical Clustering

Hierarchical clustering is used in various applications:

- Gene Expression Analysis: Grouping genes with similar expression patterns to identify co-regulated genes or functional groups.

Applications of Hierarchical Clustering

Applications of Hierarchical Clustering

Hierarchical clustering is used in various applications:

- Gene Expression Analysis: Grouping genes with similar expression patterns to identify co-regulated genes or functional groups.
- Document Clustering: Organizing documents into a hierarchy based on their content for information retrieval or topic modeling.

Applications of Hierarchical Clustering

Applications of Hierarchical Clustering

Hierarchical clustering is used in various applications:

- Gene Expression Analysis: Grouping genes with similar expression patterns to identify co-regulated genes or functional groups.
- Document Clustering: Organizing documents into a hierarchy based on their content for information retrieval or topic modeling.
- Image Segmentation: Dividing an image into regions based on pixel similarity to facilitate object recognition or scene understanding.

Applications of Hierarchical Clustering

Applications of Hierarchical Clustering

Hierarchical clustering is used in various applications:

- Gene Expression Analysis: Grouping genes with similar expression patterns to identify co-regulated genes or functional groups.
- Document Clustering: Organizing documents into a hierarchy based on their content for information retrieval or topic modeling.
- Image Segmentation: Dividing an image into regions based on pixel similarity to facilitate object recognition or scene understanding.
- Market Basket Analysis: Identifying groups of products frequently purchased together to inform marketing strategies or product placement.

Limitations of Hierarchical Clustering

Limitations of Hierarchical Clustering

Limitations of Hierarchical Clustering

Limitations of Hierarchical Clustering

Hierarchical clustering has some limitations:

- Scalability: Hierarchical clustering can be computationally expensive, especially for large datasets, as it requires calculating distances between all pairs of data points.

Limitations of Hierarchical Clustering

Limitations of Hierarchical Clustering

Hierarchical clustering has some limitations:

- Scalability: Hierarchical clustering can be computationally expensive, especially for large datasets, as it requires calculating distances between all pairs of data points.
- Sensitivity to Noise: Hierarchical clustering can be sensitive to noise and outliers, which can affect the structure of the dendrogram and the resulting clusters.

Limitations of Hierarchical Clustering

Limitations of Hierarchical Clustering

Hierarchical clustering has some limitations:

- Scalability: Hierarchical clustering can be computationally expensive, especially for large datasets, as it requires calculating distances between all pairs of data points.
- Sensitivity to Noise: Hierarchical clustering can be sensitive to noise and outliers, which can affect the structure of the dendrogram and the resulting clusters.
- Choice of Distance Metric: The choice of distance metric can significantly impact the clustering results, and there is no one-size-fits-all solution.

Limitations of Hierarchical Clustering

Limitations of Hierarchical Clustering

Hierarchical clustering has some limitations:

- Scalability: Hierarchical clustering can be computationally expensive, especially for large datasets, as it requires calculating distances between all pairs of data points.
- Sensitivity to Noise: Hierarchical clustering can be sensitive to noise and outliers, which can affect the structure of the dendrogram and the resulting clusters.
- Choice of Distance Metric: The choice of distance metric can significantly impact the clustering results, and there is no one-size-fits-all solution.
- Interpretability: The dendrogram can be complex and difficult to interpret, especially for large datasets with many clusters.

Anomaly Detection

Anomaly Detection

Anomaly Detection

Anomaly Detection

Anomaly detection is the process of identifying rare items, events, or observations that raise suspicions by differing significantly from the majority of the data.

Types of Anomaly Detection Techniques

Types of Anomaly Detection Techniques

Types of Anomaly Detection Techniques

Types of Anomaly Detection Techniques

Anomaly detection techniques can be broadly categorized into three main types:

- Statistical Methods: These methods assume that normal data follows a specific statistical distribution (e.g., Gaussian) and identify anomalies as data points that deviate significantly from this distribution.

Types of Anomaly Detection Techniques

Types of Anomaly Detection Techniques

Anomaly detection techniques can be broadly categorized into three main types:

- Statistical Methods: These methods assume that normal data follows a specific statistical distribution (e.g., Gaussian) and identify anomalies as data points that deviate significantly from this distribution.
- Machine Learning Methods: These methods use machine learning algorithms to learn patterns in the data and identify anomalies based on deviations from these patterns. Examples include one-class SVMs, isolation forests, and autoencoders.

Types of Anomaly Detection Techniques

Types of Anomaly Detection Techniques

Anomaly detection techniques can be broadly categorized into three main types:

- Statistical Methods: These methods assume that normal data follows a specific statistical distribution (e.g., Gaussian) and identify anomalies as data points that deviate significantly from this distribution.
- Machine Learning Methods: These methods use machine learning algorithms to learn patterns in the data and identify anomalies based on deviations from these patterns. Examples include one-class SVMs, isolation forests, and autoencoders.
- Clustering-Based Methods: These methods use clustering algorithms to group similar data points together and identify anomalies as points that do not belong to any cluster or are far from the nearest cluster centroid.

Steps in Anomaly Detection

Steps in Anomaly Detection

Steps in Anomaly Detection

Steps in Anomaly Detection

The anomaly detection process typically involves the following steps:

1. Data Preprocessing: Clean and preprocess the data.

Steps in Anomaly Detection

Steps in Anomaly Detection

The anomaly detection process typically involves the following steps:

1. Data Preprocessing: Clean and preprocess the data.
2. Feature Selection/Extraction: Select or extract relevant features that are likely to contain information about anomalies.

Steps in Anomaly Detection

Steps in Anomaly Detection

The anomaly detection process typically involves the following steps:

1. Data Preprocessing: Clean and preprocess the data.
2. Feature Selection/Extraction: Select or extract relevant features that are likely to contain information about anomalies.
3. Model Training: Train an anomaly detection model using the selected features.

Steps in Anomaly Detection

Steps in Anomaly Detection

The anomaly detection process typically involves the following steps:

1. Data Preprocessing: Clean and preprocess the data.
2. Feature Selection/Extraction: Select or extract relevant features that are likely to contain information about anomalies.
3. Model Training: Train an anomaly detection model using the selected features.
4. Anomaly Scoring: Calculate an anomaly score for each data point based on the trained model.

Steps in Anomaly Detection

Steps in Anomaly Detection

The anomaly detection process typically involves the following steps:

1. Data Preprocessing: Clean and preprocess the data.
2. Feature Selection/Extraction: Select or extract relevant features that are likely to contain information about anomalies.
3. Model Training: Train an anomaly detection model using the selected features.
4. Anomaly Scoring: Calculate an anomaly score for each data point based on the trained model.
5. Thresholding: Set a threshold to classify data points as normal or anomalous based on their anomaly scores.

Steps in Anomaly Detection

Steps in Anomaly Detection

The anomaly detection process typically involves the following steps:

1. Data Preprocessing: Clean and preprocess the data.
2. Feature Selection/Extraction: Select or extract relevant features that are likely to contain information about anomalies.
3. Model Training: Train an anomaly detection model using the selected features.
4. Anomaly Scoring: Calculate an anomaly score for each data point based on the trained model.
5. Thresholding: Set a threshold to classify data points as normal or anomalous based on their anomaly scores.
6. Evaluation: Evaluate the performance of the anomaly detection model using metrics such as precision, recall, and F1-score, if labeled data is available.

Applications of Anomaly Detection

Applications of Anomaly Detection

Applications of Anomaly Detection

Applications of Anomaly Detection

Anomaly detection is widely used in various domains:

- Fraud Detection: Identifying fraudulent transactions in financial systems by detecting unusual patterns in transaction data.

Applications of Anomaly Detection

Applications of Anomaly Detection

Anomaly detection is widely used in various domains:

- Fraud Detection: Identifying fraudulent transactions in financial systems by detecting unusual patterns in transaction data.
- Network Security: Detecting intrusions or malicious activities in computer networks by identifying unusual traffic patterns or behaviors.

Applications of Anomaly Detection

Applications of Anomaly Detection

Anomaly detection is widely used in various domains:

- Fraud Detection: Identifying fraudulent transactions in financial systems by detecting unusual patterns in transaction data.
- Network Security: Detecting intrusions or malicious activities in computer networks by identifying unusual traffic patterns or behaviors.
- Fault Detection: Identifying equipment failures or malfunctions in industrial systems by monitoring sensor data and detecting deviations from normal operating conditions.

Applications of Anomaly Detection

Applications of Anomaly Detection

Anomaly detection is widely used in various domains:

- Fraud Detection: Identifying fraudulent transactions in financial systems by detecting unusual patterns in transaction data.
- Network Security: Detecting intrusions or malicious activities in computer networks by identifying unusual traffic patterns or behaviors.
- Fault Detection: Identifying equipment failures or malfunctions in industrial systems by monitoring sensor data and detecting deviations from normal operating conditions.
- Healthcare: Detecting anomalies in patient health records or medical imaging data to identify potential health issues or diseases.

Applications of Anomaly Detection

Applications of Anomaly Detection

Anomaly detection is widely used in various domains:

- Fraud Detection: Identifying fraudulent transactions in financial systems by detecting unusual patterns in transaction data.
- Network Security: Detecting intrusions or malicious activities in computer networks by identifying unusual traffic patterns or behaviors.
- Fault Detection: Identifying equipment failures or malfunctions in industrial systems by monitoring sensor data and detecting deviations from normal operating conditions.
- Healthcare: Detecting anomalies in patient health records or medical imaging data to identify potential health issues or diseases.
- Manufacturing: Monitoring production processes to detect anomalies that may indicate defects or quality issues in products.