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.