✏️

Visualization with Hierarchical Clustering and t-SNE

Created	@October 22, 2024 10:57 PM
Class	Unsupervised Learning with Python

Hierarchical Clustering for Data Visualization

Prerequisites

- Understanding of clustering concepts
- Sample dataset: Eurovision 2016 voting data

Key Details

- Focus: Hierarchical clustering as a visualization technique

- Application: Creating dendrograms for data interpretation
- Purpose: Communication of insights to non-technical audiences

Hierarchical Clustering Concept

Definition

A method that arranges samples into a hierarchy of nested clusters, represented as a tree-like structure.

Mathematical Foundation

For a set of samples S, the hierarchy H is defined as:

$H = \{C_1, C_2, ..., C_n\}$ where:

- Each C_i is a cluster
- If $C_i, C_j \in H$, then either:
    - $C_i \cap C_j = \emptyset$ (disjoint)
    - $C_i \subseteq C_j \text{ or } C_j \subseteq C_i$$ (nested)

Types of Hierarchical Clustering

1. Agglomerative Clustering (Bottom-up)

- Starts with n single-sample clusters
- Iteratively merges closest clusters
- Process continues until single cluster remains

Mathematical Process

For clusters A and B, at each step:

1. Find $(\hat{A}, \hat{B}) = \arg\min_{A,B} d(A, B)$
2. Merge: $C_{new} = A^* \cup B^*$
   Where

$d(A, B)$ is the distance between clusters

2. Divisive Clustering (Top-down)

- Starts with one cluster containing all samples
- Recursively splits clusters
- Continues until individual samples remain

Implementation

Basic Implementation

```
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Perform hierarchical clustering
hierarchical_cluster = linkage(
    X,  # Sample array
    method='complete'  # Linkage method
)

# Create dendrogram
plt.figure(figsize=(10, 7))
dendrogram(
    hierarchical_cluster,
    labels=country_names,  # Sample labels
    leaf_rotation=90
)
plt.title('Eurovision Voting Patterns')
plt.tight_layout()
plt.show()
```

Mathematical Distance Metrics

Common distance metrics between clusters A and B:

1. Single Linkage:

   $$d_{single}(A, B) = \min_{a \in A, b \in B} ||a - b||$$

2. Complete Linkage:

   $$d_{complete}(A, B) = \max_{a \in A, b \in B} ||a - b||$$

3. Average Linkage:

   $$d_{average}(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} ||a - b||$$

Dendrogram Interpretation

Structure

- Vertical axis: Distance/dissimilarity between clusters
- Horizontal axis: Samples/clusters
- Vertical lines: Clusters
- Horizontal lines: Merging points

Mathematical Representation

For a merge point at height h between clusters C_1 and C_2:

$$h = d(C_1, C_2)$$

Key Takeaways

- Hierarchical clustering creates interpretable visualizations
- Dendrograms show both cluster relationships and distances
- Useful for discovering natural groupings in data
- Effective for communicating patterns to non-technical audiences
- No need to specify number of clusters beforehand

Mathematical Concept Integration

Cophenetic Distance

The cophenetic distance $c(i, j)$ between samples i and j is:

$$c(i, j) = \text{height of lowest common ancestor in dendrogram}$$

Cophenetic correlation coefficient:

$$c = \frac{\sum_{i<j}(d_{ij}-\bar{d})(c_{ij}-\bar{c})}{\sqrt{\sum_{i<j}(d_{ij}-\bar{d})^2 \sum_{i<j}(c_{ij}-\bar{c})^2}}$$

Where:

- d_{ij} is the original distance
- c_{ij} is the cophenetic distance
- $\bar{d}$ and $\bar{c}$ are their respective means

This measures how faithfully the dendrogram represents the original distances between samples.

Cluster labels in Hierarchical Clustering

Introduction and Context

- Key purpose: Communication of data science insights, especially to non-technical audiences
- Part of unsupervised learning visualization techniques
- Paired with t-SNE (to be covered later) for 2D data mapping

Hierarchical Clustering Overview

Real-World Examples

- Biological classification system
  - Narrow groups: humans, apes, snakes, lizards

- Broader groups: mammals, reptiles
- Broadest groups: animals, plants

Key Concepts

- Arranges samples into nested clusters forming a hierarchy
- Can be applied to any type of data
- Two main types:
  1. Agglomerative Clustering (bottom-up approach)
  2. Divisive Clustering (top-down approach)

Mathematical Foundation

Distance Metrics

For two samples x_i and x_j in feature space, common distance metrics include:

1. Euclidean Distance:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

2. Manhattan Distance:

$$d(x_i, x_j) = \sum_{k=1}^{n} |x_{ik} - x_{jk}|$$

Linkage Criteria

For clusters A and B containing multiple points:

1. Single Linkage:

$$d(A, B) = \min_{a \in A, b \in B} d(a, b)$$

2. Complete Linkage:

$$d(A, B) = \max_{a \in A, b \in B} d(a, b)$$

3. Average Linkage:

$$d(A,B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b)$$

Implementation in Python

Basic Implementation

```
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Perform hierarchical clustering
hierarchical_cluster = linkage(samples, method='ward')

# Create dendrogram
plt.figure(figsize=(10, 7))
dendrogram(hierarchical_cluster, labels=sample_labels)
plt.show()
```

Example with Eurovision 2016 Dataset

```
# Assuming scores_array contains Eurovision voting data
# and country_names contains list of country names

from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Perform clustering
eurovision_clusters = linkage(scores_array, method='ward')

# Create visualization
plt.figure(figsize=(15, 10))
dendrogram(eurovision_clusters, labels=country_names)
plt.title('Eurovision 2016 Voting Patterns')
plt.xlabel('Countries')
```

```
plt.ylabel('Distance')
plt.show()
```

Dendrogram Interpretation

Structure

- Vertical axis: Distance or dissimilarity between clusters
- Horizontal axis: Samples or clusters
- Vertical lines: Clusters
- Horizontal lines: Merging of clusters

Reading the Dendrogram

1. Bottom level: Individual samples (one per cluster)
2. Moving upward: Progressive merging of closest clusters
3. Height of merge: Indicates dissimilarity between merged clusters
4. Top level: Single cluster containing all samples

Key Applications

- Geographic and cultural pattern detection
- Voting behavior analysis
- Biological taxonomy
- Market segmentation
- Document clustering

Advantages

- Visual representation of hierarchical relationships
- No need to specify number of clusters beforehand
- Reveals natural groupings in data

- Suitable for small to medium-sized datasets

Practical Example Findings (Eurovision Case)

- Clusters often correspond to:
    - Geographic proximity
    - Cultural ties
    - Political alliances
    - Language groups

This structure helps reveal underlying patterns in voting behavior without requiring prior assumptions about groupings.

t-SNE (t-Distributed Stochastic Neighbor Embedding) for Data Visualization

Introduction

- Purpose: Dimensionality reduction for visualization
- Full name: t-distributed stochastic neighbor embedding
- Primary use: Converting high-dimensional data to 2D/3D representations

Mathematical Foundation

Core Concept

t-SNE converts high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities:

1. Similarity of datapoint x_j to x_i in high-dimensional space:

   $$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

2. Joint probability in high-dimensional space:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

3. Student t-distribution in low-dimensional space:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

4. Cost function (Kullback-Leibler divergence):

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Implementation in Python

Basic Implementation

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Create and fit t-SNE
tsne = TSNE(n_components=2, learning_rate=100, random_state=4
2)
tsne_features = tsne.fit_transform(X)

# Visualize results
plt.figure(figsize=(10, 6))
plt.scatter(tsne_features[:, 0], tsne_features[:, 1], c=label
s)
plt.colorbar()
plt.title('t-SNE visualization')
plt.show()
```

Example with Iris Dataset

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
```

```
# Create t-SNE object
tsne = TSNE(learning_rate=100)

# Fit and transform the data
tsne_features = tsne.fit_transform(iris_samples)

# Create visualization
plt.figure(figsize=(8, 6))
plt.scatter(tsne_features[:, 0], tsne_features[:, 1], c=speci
es_labels)
plt.title('Iris Dataset t-SNE Visualization')
plt.show()
```

Key Characteristics

Advantages

- Preserves local structure of the data
- Can reveal clusters and patterns
- Works well with non-linear relationships
- Effective for high-dimensional data visualization

Limitations

1. No Transform Method
    - Only has fit_transform
    - Cannot extend map to new samples
    - Must recompute for new data
2. Learning Rate Sensitivity
    - Requires experimentation (typically 50-200)
    - Poor choice results in clustered points
    - Dataset-dependent parameter

3. Axis Interpretation
    - Axes have no inherent meaning
    - Different orientations for same data
    - Maintains relative positions between clusters

Practical Applications

Case Study: Iris Dataset

- 4D to 2D reduction
- Unsupervised learning (no species information used)
- Reveals:
    - Natural separation of species
    - Close proximity of versicolor and virginica
    - Potential 2-cluster structure

Case Study: Piedmont Wine Dataset

- Multiple runs produce different orientations
- Preserves relative positions of wine varieties
- Demonstrates consistency in cluster relationships

Best Practices

1. Data Preparation:
    - Scale features before applying t-SNE
    - Remove irrelevant features
    - Handle missing values
2. Parameter Selection:
    - Try multiple learning rates
    - Check for point crowding

   - Validate cluster separation

3. Interpretation:
   - Focus on relative positions
   - Don't interpret axis values
   - Consider multiple runs

Applications

- Cluster analysis
- Pattern recognition
- Feature visualization
- Dimensionality reduction
- Data exploration

This technique serves as a powerful tool for initial data exploration and pattern discovery in high-dimensional datasets.