## Affinity Propagation using Hierarchical Models in Unsupervised Learning

**Purpose:**

Affinity Propagation (AP) is an algorithm used for clustering data by identifying exemplars among data points. It is well-suited for tasks where the goal is to discover natural groupings in the data without predefined labels. When combined with hierarchical models, it can uncover multi-level cluster structures, providing a more detailed organization of the data.

**When it is Used:**

- **Large datasets**: AP is effective for clustering large datasets because it does not require specifying the number of clusters a priori.
- **Data with complex structures**: Hierarchical models help in understanding nested cluster structures, beneficial for data with inherent hierarchical relationships.
- **Exploratory Data Analysis (EDA)**: Useful for uncovering hidden patterns in data, identifying representative examples, and summarizing data distribution.

**Suitable Data Types:**

- **Numerical data**: Data points represented by numerical features (e.g., sensor readings, demographic data).
- **Text data**: Documents or sentences represented as vectors (e.g., TF-IDF vectors).
- **Image data**: Image features extracted using techniques like CNNs, represented as vectors.

**Examples:**

- **Customer segmentation**: Grouping customers based on purchasing behavior.
- **Document clustering**: Organizing documents into topics.
- **Image classification**: Grouping similar images based on visual content.

## Implementation

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.cluster import AffinityPropagation

from sklearn.decomposition import PCA


# Load the Iris dataset
```

```
iris = load_iris()
data = iris.data
target = iris.target

# Apply PCA to reduce the dimensionality of the data for visualization
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(data)

# Visualize the data before clustering
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=target, cmap='viridis',
edgecolor='k', s=50)
plt.title('Iris Data Before Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar()

# Apply Affinity Propagation
affinity_propagation = AffinityPropagation(random_state=0)
affinity_propagation.fit(data)
cluster_centers_indices = affinity_propagation.cluster_centers_indices_
labels = affinity_propagation.labels_

# Number of clusters
```

```
n_clusters = len(cluster_centers_indices)
print(f'Number of clusters: {n_clusters}')

# Visualize the data after clustering
plt.subplot(1, 2, 2)
colors = plt.cm.nipy_spectral(np.linspace(0, 1, n_clusters))
for i in range(n_clusters):
    cluster_data = reduced_data[labels == i]
    plt.scatter(cluster_data[:, 0], cluster_data[:, 1], color=colors[i], edgecolor='k', s=50, label=f'Cluster {i}')

# Highlight the cluster centers
cluster_centers = reduced_data[cluster_centers_indices]
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], s=300, c='black', marker='x', label='Centers')

plt.title('Iris Data After Affinity Propagation Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.show()
```

## Code Explanation

- `numpy`: A library for numerical operations in Python.
- `matplotlib.pyplot`: A plotting library for creating static, animated, and interactive visualizations in Python.
- `sklearn.datasets`: A module in scikit-learn that provides easy access to some standard datasets, including the Iris dataset.
- `sklearn.cluster.AffinityPropagation`: A module for performing Affinity Propagation clustering.
- `sklearn.decomposition.PCA`: A module for performing Principal Component Analysis (PCA) to reduce the dimensionality of data.

- `load_iris()`: Loads the Iris dataset into memory.
- `data`: Contains the feature values of the dataset.
- `target`: Contains the true class labels for each sample in the dataset.

- `PCA(n_components=2)`: Initializes PCA with the number of components set to 2, meaning we want to reduce our data to two dimensions for visualization.
- `fit_transform(data)`: Fits the PCA model on the data and applies the dimensionality reduction, resulting in `reduced_data` which has two dimensions.

- `plt.figure(figsize=(14, 6))`: Creates a new figure with a specified size (14x6 inches).
- `plt.subplot(1, 2, 1)`: Creates a subplot (1 row, 2 columns, 1st subplot).
- `plt.scatter(...)`: Creates a scatter plot of the `reduced_data`. The color of each point is determined by its true class label (`target`).

    - `reduced_data[:, 0]`: The first PCA component.
    - `reduced_data[:, 1]`: The second PCA component.
    - `c=target`: Colors points based on their true class labels.
    - `cmap='viridis'`: Uses the 'viridis' colormap for coloring the points.
    - `edgecolor='k'`: Sets the edge color of the points to black.
    - `s=50`: Sets the size of the points.

- `plt.title('Iris Data Before Clustering')`: Sets the title of the plot.
- `plt.xlabel('PCA Component 1')`: Labels the x-axis.
- `plt.ylabel('PCA Component 2')`: Labels the y-axis.
- `plt.colorbar()`: Adds a color bar to the plot.

- `AffinityPropagation(random_state=0)`: Initializes the Affinity Propagation model with a fixed random seed (for reproducibility).
- `fit(data)`: Fits the model to the data, performing the clustering.
- `cluster_centers_indices_`: The indices of the cluster centers identified by the algorithm.
- `labels_`: The labels of each point, indicating which cluster they belong to.

- `len(cluster_centers_indices)`: Calculates the number of clusters by counting the number of cluster centers.

- `print(...)`: Prints the number of clusters.

- `plt.subplot(1, 2, 2)`: Creates the second subplot.
- `plt.cm.nipy_spectral(np.linspace(0, 1, n_clusters))`: Generates a list of colors using the 'nipy_spectral' colormap, evenly spaced for the number of clusters.
- `for i in range(n_clusters)`: Iterates over each cluster.

    - `cluster_data = reduced_data[labels == i]`: Selects the data points belonging to the current cluster.
    - `plt.scatter(...)`: Creates a scatter plot for the current cluster's data points.
        - `color=colors[i]`: Sets the color for the current cluster.
        - `edgecolor='k'`: Sets the edge color of the points to black.
        - `s=50`: Sets the size of the points.
        - `label=f'Cluster {i}'`: Sets the label for the current cluster (for the legend).

- `cluster_centers = reduced_data[cluster_centers_indices]`: Selects the cluster centers from the reduced data.
- `plt.scatter(...)`: Creates a scatter plot for the cluster centers.

    - `s=300`: Sets the size of the cluster center markers to 300.
    - `c='black'`: Sets the color of the cluster center markers to black.
    - `marker='x'`: Sets the marker style to 'x'.
    - `label='Centers'`: Sets the label for the cluster centers (for the legend).
    - • `plt.title('Iris Data After Affinity Propagation Clustering')`: Sets the title of the plot.
    - • `plt.xlabel('PCA Component 1')`: Labels the x-axis.
    - • `plt.ylabel('PCA Component 2')`: Labels the y-axis.
    - • `plt.legend()`: Adds a legend to the plot, showing labels for clusters and centers.
    - • `plt.show()`: Displays the plot.