FOR SOFTWARE STUDENTS

Practical Lab Plan: Advanced Facial Recognition with HDBSCAN

Title: Advanced Facial Recognition Clustering using HDBSCAN

Objective:

1. Explore the application of **HDBSCAN** for clustering facial data in an unsupervised learning scenario.
2. Tackle real-world challenges like noisy datasets and unbalanced data distribution.
3. Visualize and evaluate clustering results for practical use cases.

Lab Prerequisites:

1. Basic knowledge of unsupervised learning and density-based clustering methods.
2. Familiarity with dimensionality reduction techniques such as PCA.
3. Proficiency in Python programming and related data analysis tools.

Required Tools:

- A Python development environment such as Jupyter Notebook or VS Code.
- Pre-installed libraries such as NumPy, Pandas, Scikit-learn, OpenCV, Matplotlib, and HDBSCAN.

Lab Setup:

Dataset:

Use a real-world dataset, such as:

- CelebA Dataset: Contains celebrity faces with multiple images per person.
- Labeled Faces in the Wild (LFW): A dataset of labeled facial images suitable for clustering tasks.

Lab Steps:

Step 1: Load and Explore the Dataset

- Load the dataset into the environment.
- Examine the shape and structure of the dataset to understand its features, including the number of images, dimensions, and labels (if available).

- Visualize a few images to familiarize students with the data format and potential challenges like varying image sizes or resolutions.

Step 2: Data Preprocessing

- Flatten the image data into a format suitable for machine learning models.
- Standardize the dataset to ensure uniform scaling, which is critical for clustering algorithms like HDBSCAN.

Step 3: Dimensionality Reduction

- Use PCA (Principal Component Analysis) to reduce the number of features while retaining essential variance in the data.
- Choose an appropriate number of components to optimize clustering performance without over-simplifying the dataset.

Step 4: Apply HDBSCAN Clustering

- Apply the HDBSCAN algorithm to the reduced data.
- Configure HDBSCAN parameters, such as minimum cluster size and distance metric, to achieve meaningful clusters.
- Extract and display the cluster labels assigned to each data point.

Step 5: Introduce Real-Life Challenges

1. **Noisy Data:**
    - Simulate noisy data by adding random variations to images and analyze the algorithm's performance on this altered dataset.
2. **Distance Metrics:**
    - Experiment with different distance metrics, such as Manhattan or Cosine, to observe their effects on clustering results.

Step 6: Visualize and Analyze Results

- Create scatter plots or other visual representations to display the clustering of the reduced data.
- Identify noise points, which HDBSCAN labels as outliers, and analyze their distribution.
- Evaluate clustering quality using metrics like silhouette scores or other relevant techniques.

Step 7: Real-Life Applications

1. **Test on New Data:**
    - Use HDBSCAN to assign a new image to one of the identified clusters.
    - Analyze how well the clustering algorithm generalizes to unseen data.
2. **Representative Images for Clusters:**

- Identify the most representative image in each cluster by finding the data point closest to the cluster center.

Expected Outcomes:

1. Hands-on experience in clustering facial data with unsupervised learning techniques.
2. Understanding the impact of dimensionality reduction and parameter tuning on clustering.
3. Insights into real-world challenges such as noise and imbalanced data.

Additional Challenges:

1. Work with a custom dataset by capturing images from a webcam or mobile camera.
2. Experiment with advanced feature extraction techniques like HOG or embeddings from pre-trained models.
3. Optimize HDBSCAN parameters for better performance in clustering tasks.

FOR NETWORKING STUDENTS

Lab: Building an AI-Based Anomaly Detection System for Network Security Using Unsupervised Learning

Objective:

1. **Identify anomalies** in network traffic using unsupervised learning techniques.
2. **Detect abnormal behaviors** in network traffic that could indicate security threats, such as Distributed Denial of Service (DDoS), network intrusions, or malware activity.
3. **Learn and apply unsupervised learning algorithms** like clustering (K-Means, DBSCAN) and dimensionality reduction (PCA, Autoencoders) to identify patterns in the network traffic data.

Skills Developed:

- Data preprocessing for network traffic data.
- Understanding unsupervised learning techniques for anomaly detection.
- Implementing clustering algorithms (K-Means, DBSCAN).
- Applying dimensionality reduction methods (PCA or Autoencoders) for anomaly detection.
- Evaluating anomaly detection models.
- Handling network traffic data and learning how to detect outliers.

Required Tools & Technologies:

- **Python Development Environment** (Jupyter Notebook or an IDE of your choice).
- **Libraries**:
  - Scikit-learn (for algorithms like DBSCAN, K-Means, PCA, etc.)
  - Pandas (for data manipulation)
  - Matplotlib/Seaborn (for data visualization)
  - NumPy (for numerical operations)
  - Isolation Forest or One-Class SVM (for anomaly detection).

Dataset:

- **KDD Cup 1999 Dataset** (or other network traffic datasets like UNSW-NB15 or CICIDS).
  - The KDD Cup 1999 dataset is commonly used for network anomaly detection tasks. Use only normal traffic data for this unsupervised learning task.
  - You may also use simulated **network traffic logs** or **synthetic data** that contains both normal and abnormal traffic.

Lab Steps:

Step 1: Data Preprocessing

1. **Loading and Exploring the Dataset:**
   - Load the dataset and inspect it to understand its structure (e.g., types of features such as duration, protocol type, bytes, packets, etc.).
   - Check for missing values or inconsistencies in the dataset.
   - Normalize or standardize the numerical features to bring them to the same scale.
2. **Feature Engineering:**
   - Select relevant features for anomaly detection, such as **Duration**, **Bytes**, **Packets**, and **Protocol Type**.
   - Convert categorical features (e.g., Protocol Type) into numerical representations using one-hot encoding or label encoding.
   - Normalize the features to ensure all of them are on the same scale, which is important for distance-based algorithms like K-Means or DBSCAN.

Step 2: Applying Unsupervised Learning Algorithms for Anomaly Detection

1. **K-Means Clustering:**
   - Use K-Means clustering to divide the data into different clusters. Anomalies are often points that don't fit well into any cluster.
   - After clustering, calculate the distance of each data point from its assigned cluster centroid.

   - Define a threshold to flag points with large distances from their centroids as anomalies.
2. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**
   - Use DBSCAN, a density-based clustering algorithm, to identify regions of high density in the data. Points that don't fit into any cluster (i.e., outliers) will be labeled as noise (anomalies).
   - Set appropriate parameters for `eps` (radius of neighborhood) and `min_samples` (minimum number of points in a neighborhood) to optimize the detection of anomalies.
3. **Principal Component Analysis (PCA) for Dimensionality Reduction:**
   - Apply PCA to reduce the dimensionality of the data. Anomalies tend to lie far from the main data points in the reduced-dimensional space.
   - Visualize the data after applying PCA to detect points that are far from the normal data cluster.
4. **Isolation Forest (Optional Alternative):**
   - Use the Isolation Forest algorithm, which is an unsupervised learning method designed for anomaly detection by isolating points that are different from others.
   - This method is efficient for detecting anomalies in high-dimensional datasets.

Step 3: Evaluation and Results

1. **Visualizing Anomalies:**
   - Use scatter plots or PCA to visualize the normal and anomalous data points. This helps in understanding how well the anomaly detection algorithm has performed.
   - If the dataset contains both normal and anomalous behavior (such as network attacks), compare the results with known anomalies in the dataset to see how accurately the model detects them.
2. **Evaluating Performance (if labeled data is available):**
   - If you have access to labeled data, calculate evaluation metrics like **precision**, **recall**, and **F1-score** to assess the performance of your anomaly detection model.
   - If labeled data is not available, you can compare detected anomalies with known attack signatures or incidents to verify the performance.

Step 4: Enhancing the Model (Optional)

- **Hyperparameter Tuning:**
  - Experiment with the parameters of DBSCAN (such as `eps` and `min_samples`) and K-Means (such as the number of clusters) to improve the anomaly detection results.
- **Advanced Feature Engineering:**

    - Add additional features that might help detect anomalies, such as network packet size, packet frequency, or statistical features like mean, median, and standard deviation of network traffic.
- **Model Comparison:**
    - Compare the performance of different anomaly detection methods (e.g., K-Means vs. DBSCAN vs. Isolation Forest) to determine which one works best for your dataset.

Expected Outcomes:

- **Anomaly Detection Insights:**
    - Students will learn how to identify and flag anomalies in network traffic data using unsupervised learning techniques.
    - They will gain hands-on experience applying clustering algorithms and dimensionality reduction to identify abnormal patterns in data.
- **Practical Understanding:**
    - Students will understand how unsupervised learning models can be used for detecting abnormal network behaviors, which is crucial for building security systems like Intrusion Detection Systems (IDS).

Assessment Criteria:

- **Report Submission:**
    - A comprehensive report describing the steps taken in the lab, from data preprocessing to model evaluation.
    - Include visualizations such as scatter plots, PCA plots, and anomaly detection results.
    - Discuss any challenges faced and how they were overcome.
- **Model Performance:**
    - Evaluate how well the unsupervised learning models identified anomalies in network traffic.
    - The ability to interpret results and verify their relevance to potential security threats.