Distance-Based Classification with Experiment Tracking

This project applies distance-based classification to a dataset using Python and various tools, including OpenCV, Scikit-Learn, Docker, GitHub Actions, and Weights & Biases (WandB). The main objective is to detect faces, extract features, perform clustering, and track experiments in a fully automated and containerized workflow.

Project Overview

This project implements distance-based classification using K-Means clustering on face images. The workflow includes:

- **Face Detection:** Using OpenCV’s Haar cascades.
- **Feature Extraction:** Extracting hue and saturation from detected faces.
- **Clustering:** Applying K-Means clustering to group faces based on their color features.
- **Experiment Tracking:** Logging results to WandB.
- **Automation & Containerization:** Running experiments using Docker and GitHub Actions.

Setup & Installation

Prerequisites

Ensure you have the following installed:

- Python 3.11
- Git
- Docker
- Weights & Biases (WandB)
- Jupyter Notebook (for Kaggle experiments)

Installation Steps

1. **Clone the Repository**

```
git clone https://github.com/Mohil-Ahuja/distance_classification.git
cd distance_classification
```

Results & Findings

The project successfully detected faces, extracted features, and clustered them using K-Means. Here are the key findings:

- Detected **X faces** in `plaksha_Faculty.jpg`.
- K-Means successfully grouped faces into **2 clusters**.
- The template image was classified into **Cluster X**.
- All results were logged to Weights & Biases.

Report:

1. What are common distance metrics used in distance-based classification algorithms?

- **Euclidean Distance:** Measures the straight-line distance between two points in an n-dimensional space. It is commonly used in K-Nearest Neighbors (KNN) and clustering algorithms like K-Means.
- **Manhattan Distance:** Also known as taxicab distance, it calculates the sum of absolute differences between points along coordinate axes. Used in applications where movement is restricted to grid-based systems.
- **Minkowski Distance:** A generalization of both Euclidean and Manhattan distances, defined by a parameter p. When p=2, it behaves like Euclidean distance; when p=1, it behaves like Manhattan distance.
- **Cosine Similarity:** Measures the cosine of the angle between two non-zero vectors. Used in text classification and document similarity tasks.

2. What are some real-world applications of distance-based classification algorithms?

- **Face Recognition:** Identifying individuals based on facial features extracted from images.
- **Image Retrieval:** Searching for similar images based on visual content rather than metadata.
- **Fraud Detection:** Identifying anomalous transactions based on their similarity to previous patterns.
- **Medical Diagnosis:** Classifying diseases by comparing patient symptoms or medical images to known cases.
- **Document Classification:** Organizing texts into categories by analyzing word distributions and semantic similarities.

3. Explain various distance metrics.

- **Euclidean Distance:** Used when the magnitude of differences between data points matters, such as in continuous numerical data.
- **Manhattan Distance:** Suitable for scenarios with grid-based movement, such as robotics and city navigation.
- **Minkowski Distance:** A versatile metric that can adapt to different data distributions based on its parameter.
- **Cosine Similarity:** Effective for text and high-dimensional sparse data where direction is more important than magnitude.

4. What is the role of cross-validation in model performance?

- Cross-validation is a resampling technique used to evaluate model performance by partitioning the data into training and validation sets multiple times.
- **k-Fold Cross-Validation:** Splits data into k subsets and trains the model k times, using each subset as a validation set once.
- **Leave-One-Out Cross-Validation (LOOCV):** Uses one data point as a validation set and the rest for training, repeated for every data point.
- **Stratified Cross-Validation:** Ensures that class distributions are maintained across training and validation sets.
- Helps in detecting overfitting, fine-tuning hyperparameters, and improving generalization.

5. Explain variance and bias in terms of KNN.

- **Variance:** The sensitivity of the model to small fluctuations in the training set. Low values of k (e.g., k=1) lead to high variance, as the decision boundary changes drastically with different training data.
- **Bias:** The error introduced by approximating a real-world problem with a simplified model. High values of k (e.g., k=10+) lead to high bias, as the decision boundary becomes too smooth and fails to capture finer patterns.