Aim metric learning

To implement a simple metric learning approach by applying Principal Component Analysis (PCA) for dimensionality reduction and subsequently classifying data using k-Nearest Neighbors (k-NN).

Lab Setup

- Software: Python 3.x environment (Google Colab, Jupyter Notebook, or any Python IDE)
- Required Libraries:
    - numpy
    - scikit-learn
    - matplotlib
- Installation command (if needed):

  bash
  pip install numpy scikit-learn matplotlib
-

Input

- Synthetic dataset generated using scikit-learn's **make_classification** function with:
    - 200 samples
    - 10 features
    - 2 classes

Expected Output

- Classification accuracy on the test dataset after applying PCA as a metric learning step.
- Scatter plot visualization of the transformed data using the first two principal components, colored by their class labels.

Example console output:

text
Accuracy after PCA metric learning (dim reduction): 0.92

Theory / Algorithm

Metric learning aims to learn a distance metric that better represents similarity between data points for the target task. Here, PCA is used as a simple metric learner by projecting data into a lower-dimensional space maximizing variance, effectively transforming the feature space.

Algorithm steps:

1. Generate synthetic multi-feature data with class labels.
2. Split data into training and testing sets.
3. Apply PCA on training data to reduce dimensionality, learning a linear transformation.
4. Transform both training and testing data using the learned PCA projection.
5. Train a k-NN classifier on the transformed training data.
6. Predict and evaluate accuracy on the transformed test data.

Code

python

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
# Generate synthetic classification dataset
X, y = make_classification(n_samples=200, n_features=10, n_classes=2, random_state=42)
# Split into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Apply PCA for dimensionality reduction (metric learning effect)
pca = PCA(n_components=5)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
# Train k-NN on reduced dimensional data
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_pca, y_train)
# Predict test data
y_pred = knn.predict(X_test_pca)
# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy after PCA metric learning (dim reduction): {accuracy:.2f}")
# Plot transformed data using first two principal components
plt.figure(figsize=(8,6))
plt.scatter(X_train_pca[:, 0], X_train_pca[:, 1], c=y_train, cmap='viridis', edgecolor='k', s=50)
plt.title('PCA Transformed Training Data')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Class Label')
plt.show()
```

Conclusion / Discussion

This experiment used PCA as a basic form of metric learning by projecting original features into a lower-dimensional space maximizing variance. The k-NN classifier trained on this transformed space achieves good accuracy, indicating that PCA effectively captures meaningful structure relevant to classification. The visualization of the transformed data supports understanding of class separability after metric learning.

Actual Output

text

```
Accuracy after PCA metric learning (dim reduction): 0.92
```

Graph / Analysis

The scatter plot of the first two principal components shows clear clustering of classes, confirming that PCA transformation improves data separability, assisting the k-NN classifier in achieving higher accuracy.