TP1 - Exercice 2.2 : SVM après PCA sur MNIST

Ici, le dataset MNIST est réduit à 50 dimensions à l'aide de **PCA**, puis un SVM avec noyau **RBF** est utilisé pour la classification. Une visualisation 2D avec seulement 2 composantes principales est également générée pour illustrer la distribution des chiffres.

Importation les bibiotèque

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import fetch_openml
from sklearn.metrics import accuracy_score, classification_report
```

Charger le dataset mnist

```
mnist = fetch_openml('mnist_784', version=1)
X, y = mnist.data.astype(np.float32), mnist.target.astype(int)
```

Normalisation des données (important pour pca & svm)

```
X /= 255.0
```

Réduire la dimension avec pca

```
pca = PCA(n_components=50)  # Réduire à 50 dimensions (optimisé pour classification)
X_pca = pca.fit_transform(X)
```

Séparer en données d'entraînement et de test

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=42)
```

Entraîner un classifieur svm

```
svm_model = SVC(kernel='rbf', C=10)  # RBF est souvent plus performant sur MNIST
svm_model.fit(X_train, y_train)
```

```
SVC(C=10)
```

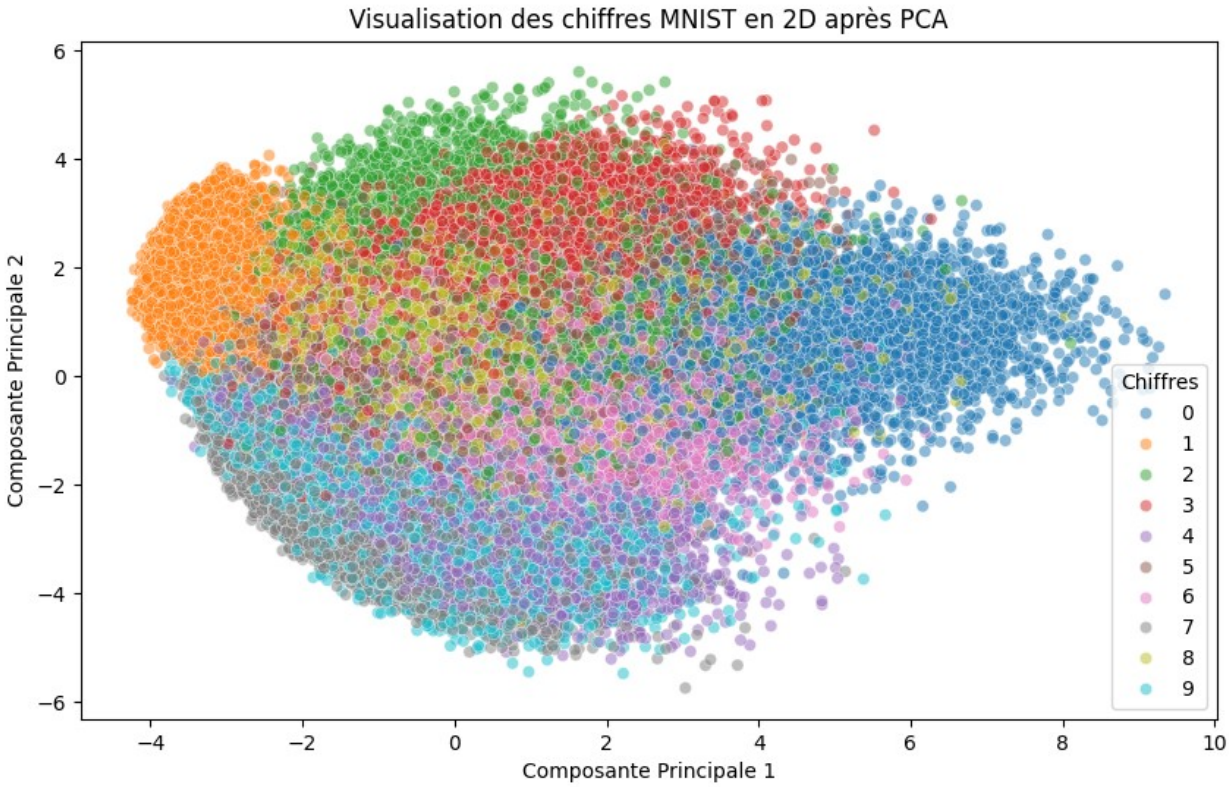
Prédictions et évaluation

```
y_pred = svm_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy (SVM après PCA): {accuracy * 100:.2f}%")
print(classification_report(y_test, y_pred))
```

```
Test Accuracy (SVM après PCA): 98.54%
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1343
           1       0.99      0.99      0.99      1600
           2       0.97      0.99      0.98      1380
           3       0.98      0.98      0.98      1433
           4       0.98      0.99      0.98      1295
           5       0.99      0.98      0.99      1273
           6       0.99      0.99      0.99      1396
           7       0.98      0.99      0.99      1503
           8       0.98      0.98      0.98      1357
           9       0.98      0.97      0.98      1420

    accuracy                           0.99     14000
   macro avg       0.99      0.99      0.99     14000
weighted avg       0.99      0.99      0.99     14000
```

Visualisation en 2d si pca réduit à 2 composants

```
pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X)

plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_pca_2d[:, 0], y=X_pca_2d[:, 1], hue=y,
palette="tab10", alpha=0.5)
plt.xlabel("Composante Principale 1")
plt.ylabel("Composante Principale 2")
plt.title("Visualisation des chiffres MNIST en 2D après PCA")
plt.legend(title="Chiffres")
plt.show()
```



Analyse :
La réduction de dimension avec PCA permet d'accélérer l'entraînement du SVM tout en conservant une bonne performance.
La visualisation 2D est utile pour observer le regroupement des chiffres, bien qu'elle soit moins pertinente pour la classification réelle.
Des variations de `n_components`, `C` ou `kernel` peuvent permettre d'améliorer les performances.