Analyse Discriminante Linéaire - OVA

Analyse Discriminante Linéaire (LDA) - One-Versus-All (OVA)

Théorie

L'**Analyse Discriminante Linéaire (LDA)** est une technique de classification qui cherche à trouver une combinaison linéaire de caractéristiques maximisant la séparation entre les classes.

L'approche **One-Versus-All (OVA)** consiste à entraîner un modèle pour chaque classe, en distinguant chaque classe des autres combinées.

Hyperparamètres

Nous allons tester les hyperparamètres suivants : - **Régularisation (`shrinkage`)** : contrôle la variance de la covariance estimée (valeurs entre `0` et `1`). - **Standardisation des données** : normalisation des features avant l'entraînement.

Exemple en Python

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Chargement des ensembles de données
```

```
train_data = pd.read_csv('covertype_train.csv')
val_data = pd.read_csv('covertype_val.csv')
test_data = pd.read_csv('covertype_test.csv')

# Préparation des données
X_train = train_data.drop('Cover_Type', axis=1)
y_train = train_data['Cover_Type']

X_val = val_data.drop('Cover_Type', axis=1)
y_val = val_data['Cover_Type']

X_test = test_data.drop('Cover_Type', axis=1)
y_test = test_data['Cover_Type']

# Standardisation des données
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

# Recherche des meilleurs hyperparamètres
shrinkage_values = np.linspace(0, 1, 10)
train_accuracies = []
val_accuracies = []

for shrinkage in shrinkage_values:
    lda_ova = OneVsRestClassifier(LinearDiscriminantAnalysis(solver='lsqr', shrinkage=shrinka
    lda_ova.fit(X_train, y_train)

    y_train_pred = lda_ova.predict(X_train)
    y_val_pred = lda_ova.predict(X_val)

    train_accuracies.append(accuracy_score(y_train, y_train_pred))
    val_accuracies.append(accuracy_score(y_val, y_val_pred))

# Sélection du meilleur shrinkage
best_shrinkage = shrinkage_values[val_accuracies.index(max(val_accuracies))]
print(f"Meilleur shrinkage LDA (OVA): {best_shrinkage}")

# Affichage du graphique
plt.figure(figsize=(8, 6))
plt.plot(shrinkage_values, train_accuracies, marker='o', linestyle='dashed', label='Train Ac
```

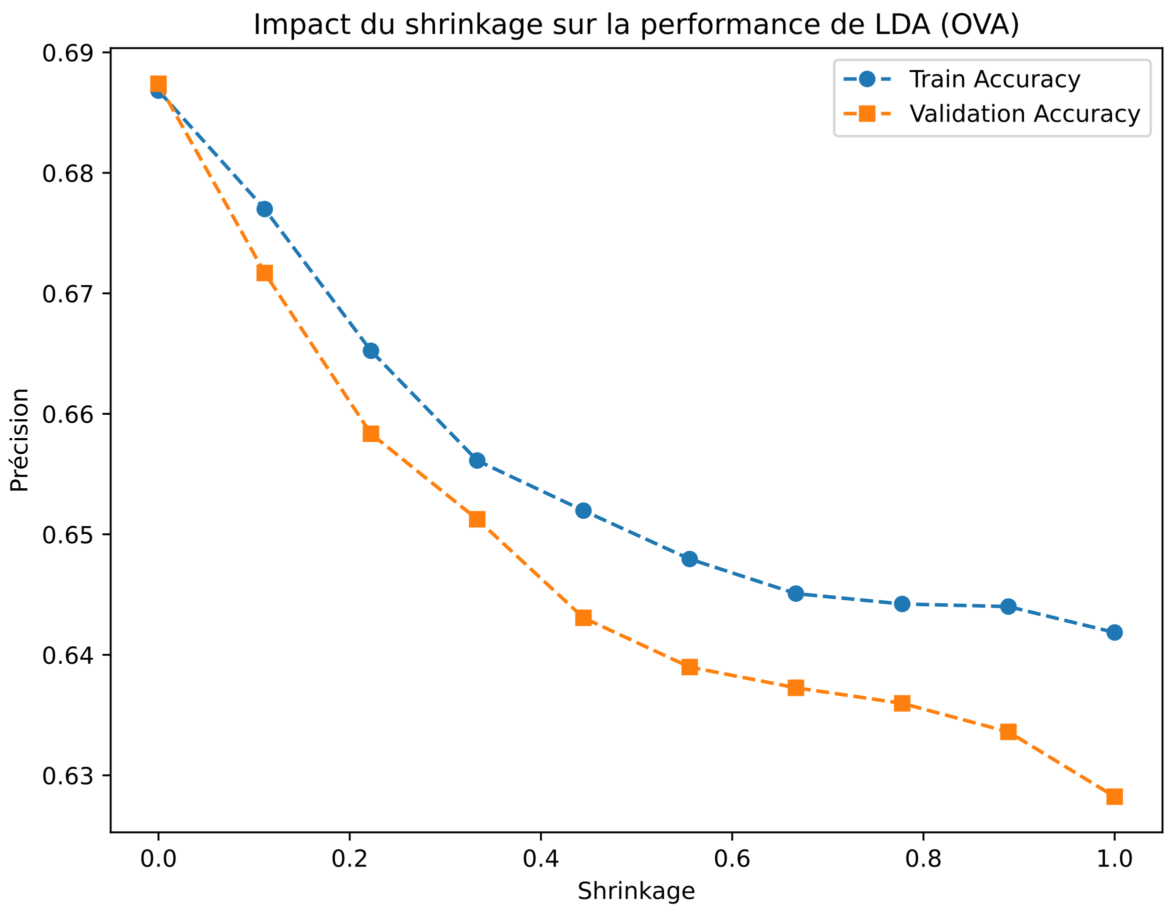
```
plt.plot(shrinkage_values, val_accuracies, marker='s', linestyle='dashed', label='Validation
plt.xlabel("Shrinkage")
plt.ylabel("Précision")
plt.title("Impact du shrinkage sur la performance de LDA (OVA)")
plt.legend()
plt.show()

# Modèle final avec le meilleur shrinkage
lda_ova = OneVsRestClassifier(LinearDiscriminantAnalysis(solver='lsqr', shrinkage=best_shrin
lda_ova.fit(X_train, y_train)
y_test_pred = lda_ova.predict(X_test)

# Affichage de la matrice de confusion
conf_matrix = confusion_matrix(y_test, y_test_pred)
print("\nMatrice de confusion (OVA) :")
print(conf_matrix)

print("\nÉvaluation sur l'ensemble de test")
print(classification_report(y_test, y_test_pred))
```

```
Meilleur shrinkage LDA (OVA): 0.0
```




```
Matrice de confusion (OVA) :
[[1035  478    2    0    6    6  181]
 [ 396 1710   47   12   12   73   11]
 [   0   16  211   17    1   36    0]
 [   0    0    5   12    0    4    0]
 [   2   57   12    0    1    2    0]
 [   0   24   69    2    4   45    0]
 [  24    0    0    0    0    0  135]]
```

```
Évaluation sur l'ensemble de test
              precision    recall  f1-score   support

           1       0.71      0.61      0.65      1708
           2       0.75      0.76      0.75      2261
           3       0.61      0.75      0.67       281
```

4	0.28	0.57	0.38	21
5	0.04	0.01	0.02	74
6	0.27	0.31	0.29	144
7	0.41	0.85	0.56	159
accuracy			0.68	4648
macro avg	0.44	0.55	0.47	4648
weighted avg	0.69	0.68	0.68	4648