Analyse Discriminante Linéaire - OVO

Analyse Discriminante Linéaire (LDA) - One-Versus-One (OVO)

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-One (OVO)** consiste à entraîner un modèle pour chaque paire de classes, ce qui est utile lorsque les classes sont bien séparées.

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.multiclass import OneVsOneClassifier
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_ovo = OneVsOneClassifier(LinearDiscriminantAnalysis(solver='lsqr', shrinkage=shrinkag
    lda_ovo.fit(X_train, y_train)

    y_train_pred = lda_ovo.predict(X_train)
    y_val_pred = lda_ovo.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 (OVO): {best_shrinkage}")

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

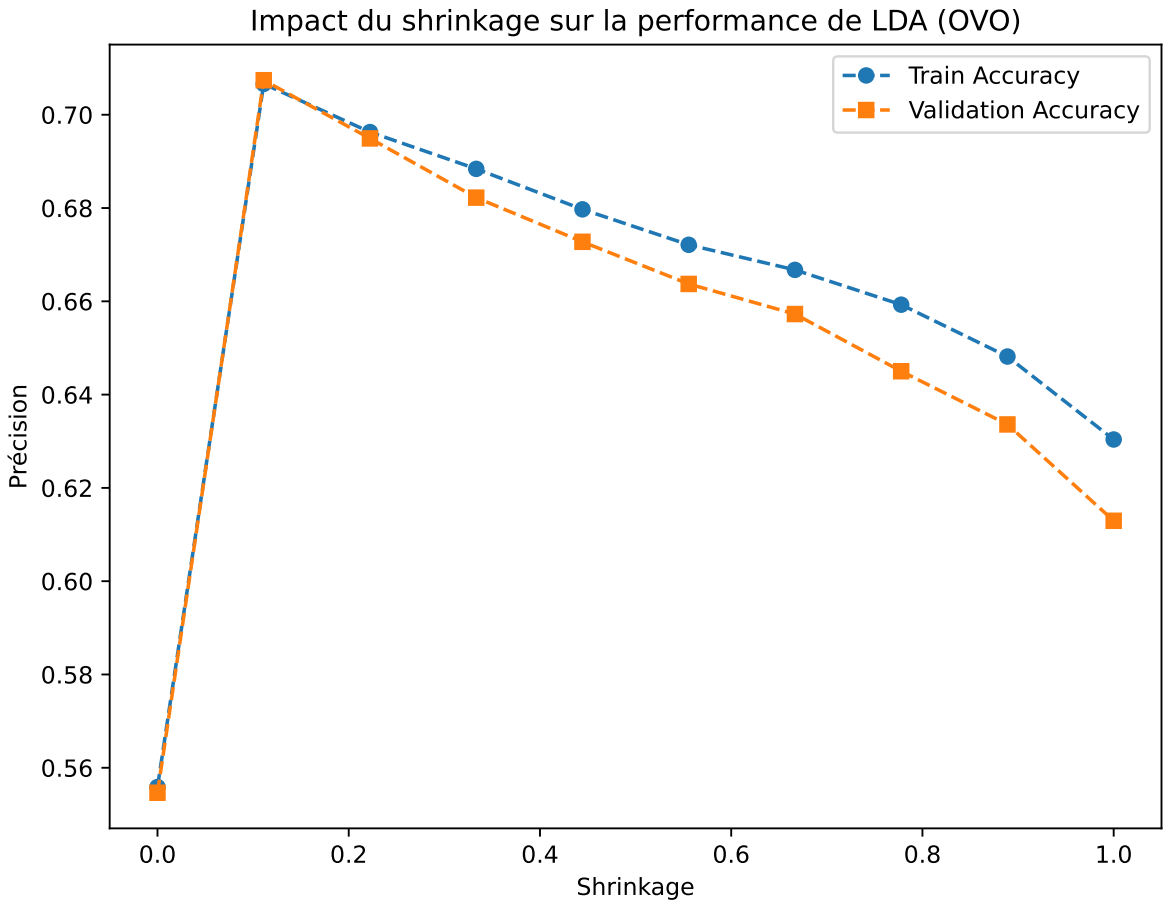
```
plt.plot(shrinkage_values, val_accuracies, marker='s', linestyle='dashed', label='Validation
plt.xlabel("Shrinkage")
plt.ylabel("Précision")
plt.title("Impact du shrinkage sur la performance de LDA (OVO)")
plt.legend()
plt.show()

# Modèle final avec le meilleur shrinkage
lda_ovo = OneVsOneClassifier(LinearDiscriminantAnalysis(solver='lsqr', shrinkage=best_shrinka
lda_ovo.fit(X_train, y_train)
y_test_pred = lda_ovo.predict(X_test)

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

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

```
Meilleur shrinkage LDA (OVO): 0.1111111111111111
```




```
Matrice de confusion (OVO) :
[[1085  501    2    0    3    3  114]
 [ 388 1777   62    0   10   14   10]
 [   0   23  229   13    0   16    0]
 [   0    0    4   12    0    5    0]
 [   2   59   12    0    1    0    0]
 [   0   39   89    1    0   15    0]
 [  30    0    0    0    0    0  129]]

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

           1       0.72      0.64      0.68      1708
           2       0.74      0.79      0.76      2261
           3       0.58      0.81      0.67       281
```

4	0.46	0.57	0.51	21
5	0.07	0.01	0.02	74
6	0.28	0.10	0.15	144
7	0.51	0.81	0.63	159
accuracy			0.70	4648
macro avg	0.48	0.53	0.49	4648
weighted avg	0.69	0.70	0.69	4648