

SVM - One-Versus-All (OVA)

SVM - One-Versus-All (OVA)

Théorie

Les **machines à vecteurs de support (SVM)** sont des modèles de classification supervisés qui cherchent à maximiser la marge de séparation entre les classes. Pour un problème **multiclasse**, l'approche **One-Versus-All (OVA)** entraîne un SVM pour chaque classe, la comparant à toutes les autres classes combinées.

Hyperparamètres

Nous allons tester un seul hyperparamètre pour réduire le temps d'entraînement : - **Paramètre de régularisation (C)** : contrôle la pénalisation des erreurs de classification (valeurs entre 0.1 et 1).

Exemple en Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
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('covertypes_train.csv')
val_data = pd.read_csv('covertypes_val.csv')
```

```

test_data = pd.read_csv('coverttype_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 du meilleur hyperparamètre (C seulement)
C_values = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1] # Entre 0.1 et 1
train_accuracies = []
val_accuracies = []

for C in C_values:
    model = OneVsRestClassifier(SVC(kernel='rbf', C=C))
    model.fit(X_train, y_train)

    y_train_pred = model.predict(X_train)
    y_val_pred = model.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 hyperparamètre
best_C = C_values[val_accuracies.index(max(val_accuracies))]
print(f"Meilleur hyperparamètre : C={best_C}")

# Affichage du graphique
plt.figure(figsize=(8, 6))
plt.plot(C_values, train_accuracies, marker='o', linestyle='dashed', label='Train Accuracy')
plt.plot(C_values, val_accuracies, marker='s', linestyle='dashed', label='Validation Accuracy')
plt.xlabel("Paramètre de régularisation (C)")

```

```

plt.ylabel("Précision")
plt.title("Impact de la régularisation sur la performance du SVM (OVA)")
plt.legend()
plt.show()

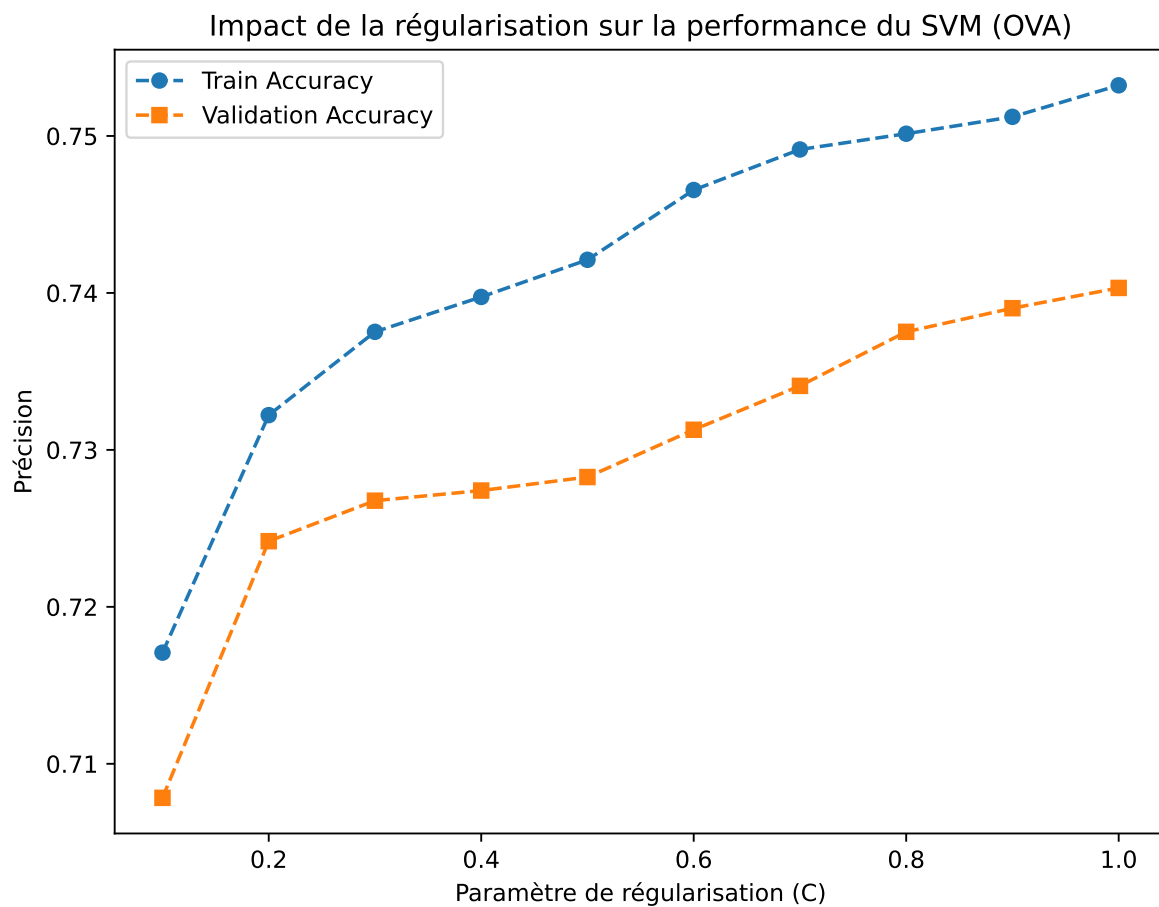
# Modèle final avec le meilleur hyperparamètre
final_model = OneVsRestClassifier(SVC(kernel='rbf', C=best_C))
final_model.fit(X_train, y_train)
y_test_pred = final_model.predict(X_test)

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

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

```

Meilleur hyperparamètre : C=1



Matrice de confusion :

```
[[1218 453 1 0 0 1 35]
 [ 368 1841 40 0 1 7 4]
 [ 0 23 250 0 0 8 0]
 [ 0 0 18 0 0 3 0]
 [ 4 59 9 0 2 0 0]
 [ 0 38 91 0 0 15 0]
 [ 57 0 0 0 0 0 102]]
```

Évaluation sur l'ensemble de test

	precision	recall	f1-score	support
1	0.74	0.71	0.73	1708
2	0.76	0.81	0.79	2261
3	0.61	0.89	0.72	281

4	0.00	0.00	0.00	21
5	0.67	0.03	0.05	74
6	0.44	0.10	0.17	144
7	0.72	0.64	0.68	159
accuracy			0.74	4648
macro avg	0.56	0.46	0.45	4648
weighted avg	0.73	0.74	0.72	4648

```

/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Und
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Und
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Und
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

```