
Université de Montpellier
Département de Mathématiques Appliquées

# TP3 : Support Vector Machine

*Apprentissage Statistique — HAX907X*

**Réalisé par :**
ATTOUMANI Ibrahim

STATISTIQUE
SCIENCE DES DONNÉES
UNIVERSITÉ DE MONTPELLIER

Année Universitaire 2025 – 2026

# Table des matières

# 1. Introduction

Les **SVM** (Support Vector Machines), introduits par Vapnik, sont des méthodes de classification très utilisées, en particulier pour la classification binaire. Elles reposent sur la recherche d'une règle de décision linéaire sous la forme d'un **hyperplan séparateur**. Pour traiter des problèmes plus complexes, cette recherche est effectuée non pas directement dans l'espace des données initiales, mais dans un **espace de caractéristiques** de grande dimension, obtenu grâce à une transformation non linéaire.

L'objectif de ce TP est d'appliquer les SVM sur des données réelles et simulées à l'aide de la librairie `scikit-learn` (qui s'appuie sur `libsvm`). Nous apprendrons également à ajuster les **hyperparamètres** et le **choix du noyau** afin de mieux contrôler la flexibilité du modèle.

# 2. Classification sur les donnée Iris

Dans cette section, nous allons mettre en œuvre un **SVM linéaire** afin de distinguer les classes **1 et 2** du jeu de données *iris*. Pour simplifier le problème, seules les **deux premières variables** seront utilisées. Nous conserverons la moitié des données pour l'**apprentissage** et l'autre moitié pour le **test**, afin d'évaluer la capacité du modèle à bien généraliser.

## 2.1. SVM linéaire : discrimination entre les classes 1 et 2 d'Iris

Ici nous allons sélectionner les classes 1 et 2 du dataset *iris*, puis conserver uniquement les deux premières variables (longueur et largeur des sépales) afin de simplifier la visualisation et la classification.

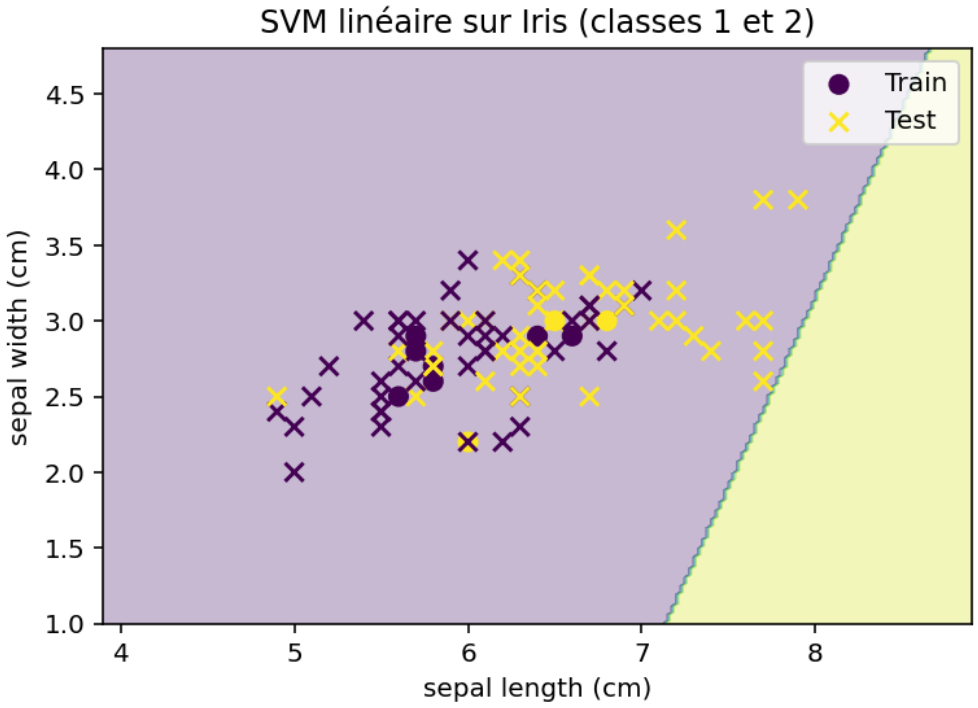


Figure 1: SVM linéaire avec 90% des données réservées au test

Ci-dessus, on a dédiées 90% des données pour le test, ce qui ne laisse qu'environ 10% pour l'entraînement (soit seulement 10 points). Avec si peu d'exemples, le SVM ne dispose pas d'assez d'informations pour bien apprendre. On a alors que la frontière de décision est très approximative et la précision sur le test chute à **0.48**, proche du hasard.

En utilisant 40% des données pour le test, nous obtenons une précision de 7%. Comparée à la **figure 1**, la frontière de décision sépare mieux les points, bien que quelques erreurs de prédiction subsistent. La **figure 2** ci-dessous illustre cette visualisation.

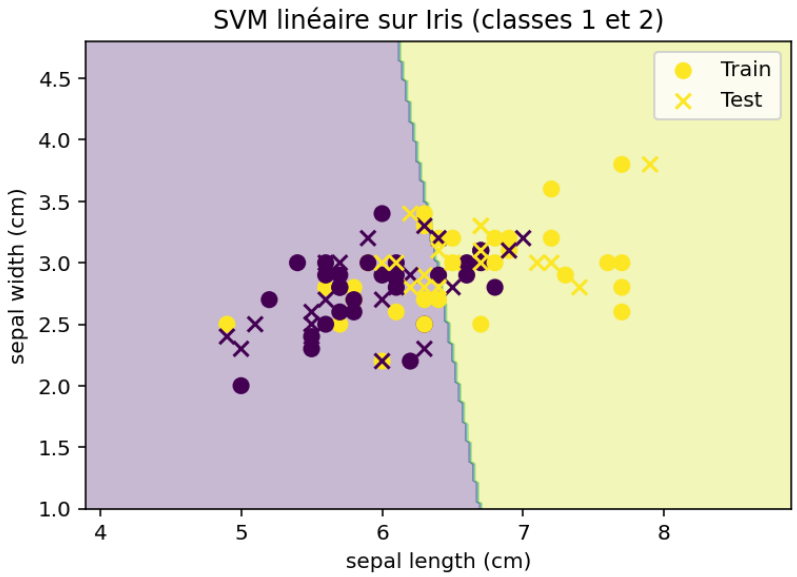


Figure 2: SVM linéaire avec 40% des données réservées au test

## 2.2. SVM polynomial : discrimination entre les classes 1 et 2 d'Iris

Dans cette section, nous entraînons un SVM avec noyau polynomial pour discriminer les classes 1 et 2 d'Iris. Ce noyau permet de modéliser des frontières de décision non linéaires, plus flexibles que celles du SVM linéaire. Nous comparerons ensuite ses performances et sa frontière de décision avec celles du SVM linéaire présenté précédemment.

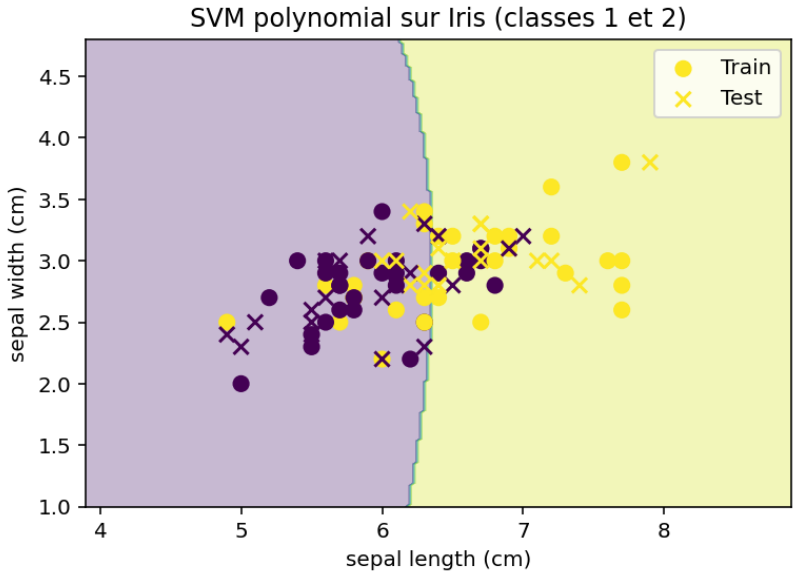


Figure 3: SVM polynomial avec 40% des données réservées au test

À présent, nous comparons la classification linéaire à la classification polynomiale. Cette

dernière ajuste mieux la frontière de décision, avec une précision de 0.75 sur le jeu de test. Comme pour la classification linéaire, 40% des données ont été utilisées pour l'évaluation.

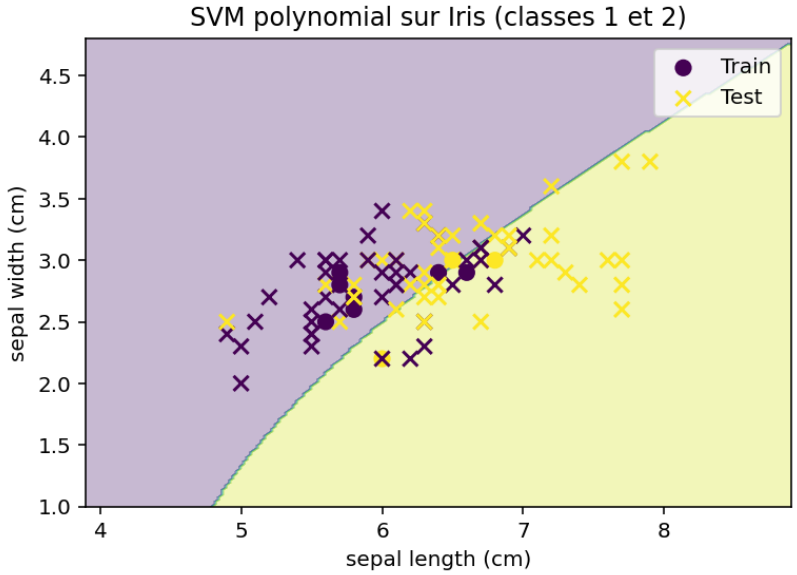


Figure 4: SVM polynomial avec 90% des données réservées au test

Nous avons testé deux modèles SVM (linéaire et polynomial) sur les classes 1 et 2 de l'iris, en utilisant les caractéristiques *sepal length* et *sepal width*.

Avec une grande proportion de données en test (90%), les deux modèles donnent de faibles scores, mais le SVM polynomial s'adapte mieux grâce à une frontière de décision non linéaire.

Avec une répartition plus équilibrée (60% train / 40% test), les performances s'améliorent. Le SVM polynomial reste légèrement meilleur, car il capture mieux la séparation non linéaire entre les classes.

## 2.3. SVM GUI: données simulées déséquilibré

Cette application svm_gui.py(https://scikit-learn.org/1.2/auto_examples/applications/svm_gui.html) permet en temps réel d'évaluer l'impact du noyaux et du paramètre de régularisation $C$.

Dans ce qui suit, nous allons générer un jeu de données très déséquilibré avec beaucoup plus de points dans une classe que dans l'autre. Ensuite, nous utiliserons un noyau linéaire tout en réduisant le paramètre $C$ et observerons le résultat.

Lorsque l'on diminue la valeur du paramètre $C = 0,001$, le SVM devient plus tolérant aux erreurs. La frontière de décision se simplifie et ne cherche plus à bien séparer toutes les classes.

Dans un jeu de données très déséquilibré (90% vs 10%), cela entraîne un fort biais vers la classe majoritaire. La classe minoritaire est alors mal reconnue, voire ignorée. La figure ci-dessous illustre la frontière de décision d'un problème déséquilibré:

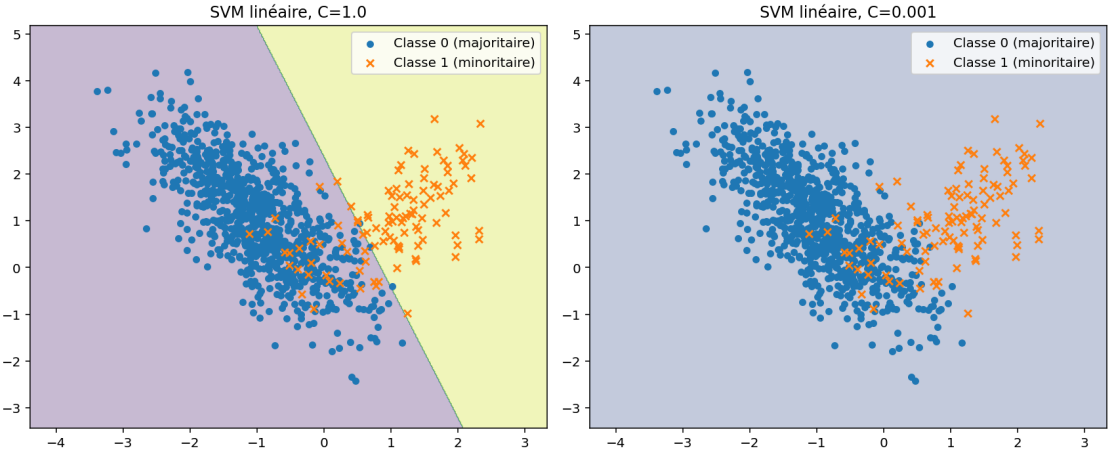


Figure 5: SVM linéaire avec $C = 0.001$ sur données déséquilibrées.

Pour résoudre le problème de données déséquilibrées, nous pouvons utiliser l'option `class_weight="balanced"` dans le SVM. Cette approche permet d'ajouter une correction par pondération, en donnant davantage de poids à la classe minoritaire afin d'éviter qu'elle soit ignorée par le modèle.

L'illustration de ce phénomène, avec et sans correction par pondération, est présentée en annexe.

# 3. Classification de visage

# Annexe

Ce code génère un jeu de données binaire fortement déséquilibré (90,% vs 10,%). Deux modèles SVM linéaires sont ensuite entraînés : l'un sans pondération et l'autre avec l'option `class_weight="balanced"`. Enfin, la frontière de décision de chaque modèle est tracée afin de comparer l'effet de la pondération sur la classe minoritaire.

```
# Générer un jeu de données binaire fortement déséquilibré (90% vs 10%)
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.svm import SVC
from collections import Counter
import numpy as np

# paramètres
n_samples = 1000
n_features = 2
weights = [0.9, 0.1]
class_sep = 1.0
random_state = 0

X, y = make_classification(n_samples=n_samples,
                           n_features=n_features,
                           n_informative=2,
                           n_redundant=0,
                           n_clusters_per_class=1,
                           weights=weights,
                           flip_y=0.01,
                           class_sep=class_sep,
                           random_state=random_state)

print("Répartition des classes :", Counter(y))

# Deux modèles : non pondéré vs pondéré
models = {
    "non pondéré": SVC(kernel="linear", C=1.0, class_weight=None),
    "pondéré": SVC(kernel="linear", C=1.0, class_weight="balanced")
}

# Entraînement
for model in models.values():
    model.fit(X, y)

# Tracé des points
plt.figure(figsize=(7, 6))
```

```
plt.scatter(X[y == 0, 0], X[y == 0, 1], s=20, c="lightblue", label="Classe 0 (majoritair
plt.scatter(X[y == 1, 0], X[y == 1, 1], s=30, c="brown", marker="x", label="Classe 1 (mi

# Grille pour la frontière
xx, yy = np.meshgrid(
    np.linspace(X[:, 0].min()-1, X[:, 0].max()+1, 500),
    np.linspace(X[:, 1].min()-1, X[:, 1].max()+1, 500)
)

# Frontières de décision
colors = {"non pondéré": "black", "pondéré": "red"}

for name, model in models.items():
    Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contour(xx, yy, Z, levels=[0], colors=colors[name], linewidths=2, label=name)

# Légende personnalisée
handles = [
    plt.Line2D([0], [0], color="black", label="non pondéré"),
    plt.Line2D([0], [0], color="red", label="pondéré")
]
plt.legend(handles=handles + plt.gca().get_legend_handles_labels()[0], loc="best")

plt.title("SVM linéaire : comparaison non pondéré vs pondéré")
plt.show()
```

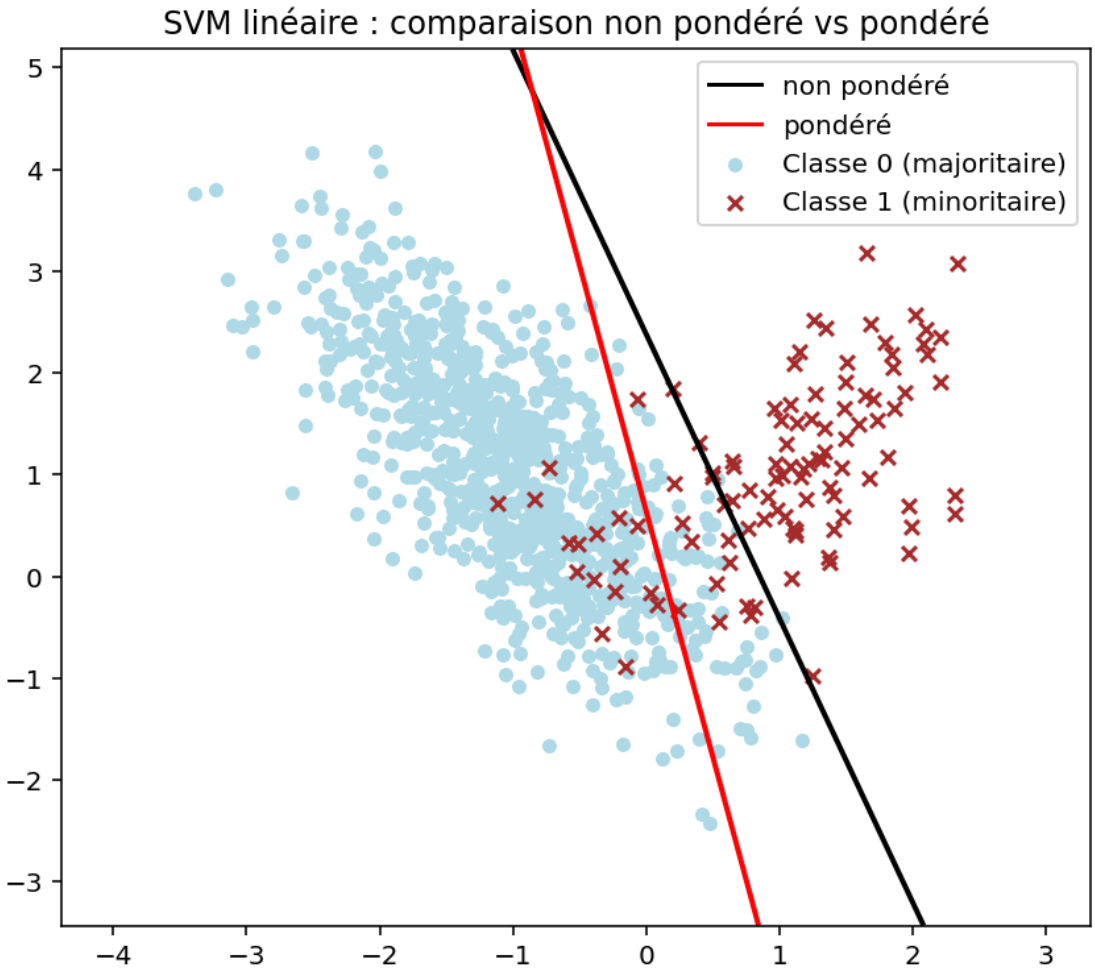


Figure 6: Frontière de décision d'un SVM linéaire avec et sans correction par pondération

# References

[1] scikit-learn developers. *Support Vector Machines — scikit-learn documentation*. http://scikit-learn.org/stable/modules/svm.html. Consulté le 24 septembre 2025.

[2] Wikipedia. *Support vector machine*. http://en.wikipedia.org/wiki/Support_vector_machine. Consulté le 24 septembre 2025.

[3] Wikipédia. *Machine à vecteurs de support*. http://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support. Consulté le 24 septembre 2025.