cpge-paradise.com

Prévention des Accidents Cardiaques dans le sport grâce au Machine Learning

Hamama Mohammed Reda

Épreuve de TIPE

Session 2024

cpge-paradise.com

Introduction

Présentation du problème

cpge-paradise.com

Plan de l'exposé

1 Introduction et problématique

2 Principe des MVS

3 Cas des données linéairement séparables

4 Cas de données non séparables linéairement

5 Implémentation sur python et résultats

6 Analyse des résultats et conclusion

cpge-paradise.com

Problématique

De quelle manière peut-on exploiter les machines à vecteurs de support afin de contribuer à la prédiction des accidents cardiaques ?

cpge-paradise.com

Modèle

Machine Learning → Supervisé → Classification → SVM

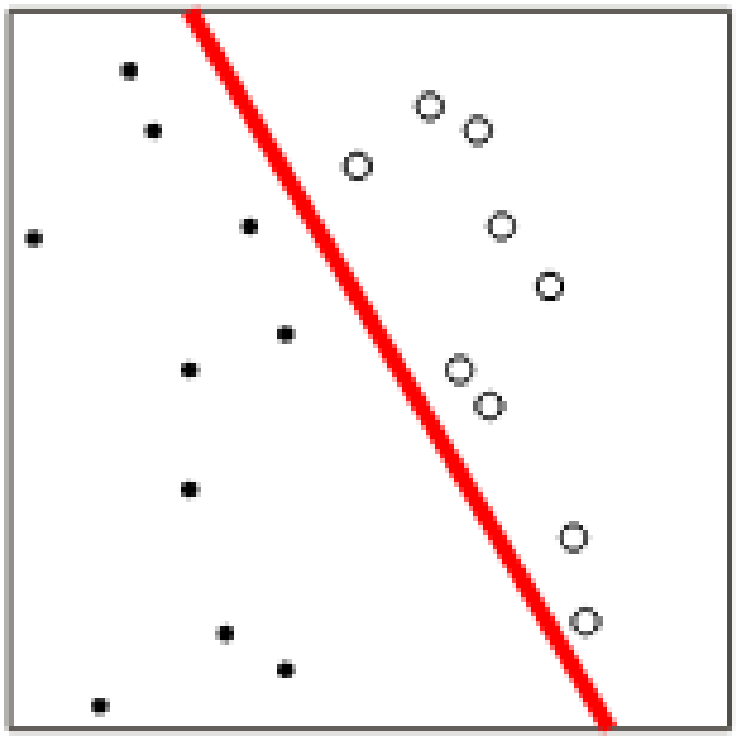
- On se donne un ensemble d'apprentissage $\{(x_i, y_i)\}_{i=1,\dots,n}$ où $y_i \in \{-1, +1\}$ et $x_i \in \mathbb{R}^d$.
- Objectif : chercher une fonction $f : \mathbb{R}^d \to \mathbb{R}$ qui permet de prédire si un nouvel exemple appartient à la classe -1 ou à la classe $+1$.
- Le principe est de séparer l'espace en deux parties $\{x \in \mathbb{R}^d \mid f(x) > 0\} \cup \{x \in \mathbb{R}^d \mid f(x) < 0\}$.

cpge-paradise.com

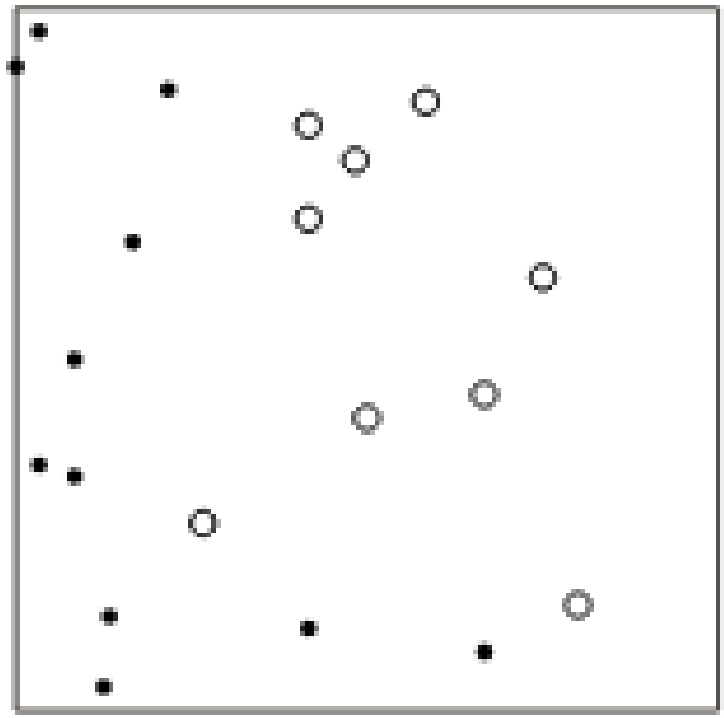
Séparabilité linéaire

Soient X_0 et X_1 deux ensembles de points dans un espace euclidien n-dimensionnel.
Alors X_0 et X_1 sont **linéairement séparables** s'il existe $n+1$ nombres réels $w_1, w_2, ..., w_n, k$ tels que chaque point $x \in X_0$ satisfait $\sum_{i=1}^{n} w_i x_i > k$ et chaque point $x \in X_1$ satisfait $\sum_{i=1}^{n} w_i x_i < k$.

cpge-paradise.com



(a) Les données sont séparables linéairement



(b) Les données ne sont pas séparables linéairement

cpge-paradise.com

Construction de l'hyperplan

L'existence d'un hyperplan séparateur

Soient A et B deux ensembles non vides, convexes et disjoints. Alors, il existe $w \neq 0$ et b tels que :

- $w^\top \cdot x + b \leq 0 \quad \forall x \in A$
- $w^\top \cdot x + b \geq 0 \quad \forall x \in B$

L'hyperplan $w^\top \cdot x + b = 0$ est appelé l'hyperplan séparateur pour A et B.

$\rightarrow$ démonstration : voir annexe 1

cpge-paradise.com

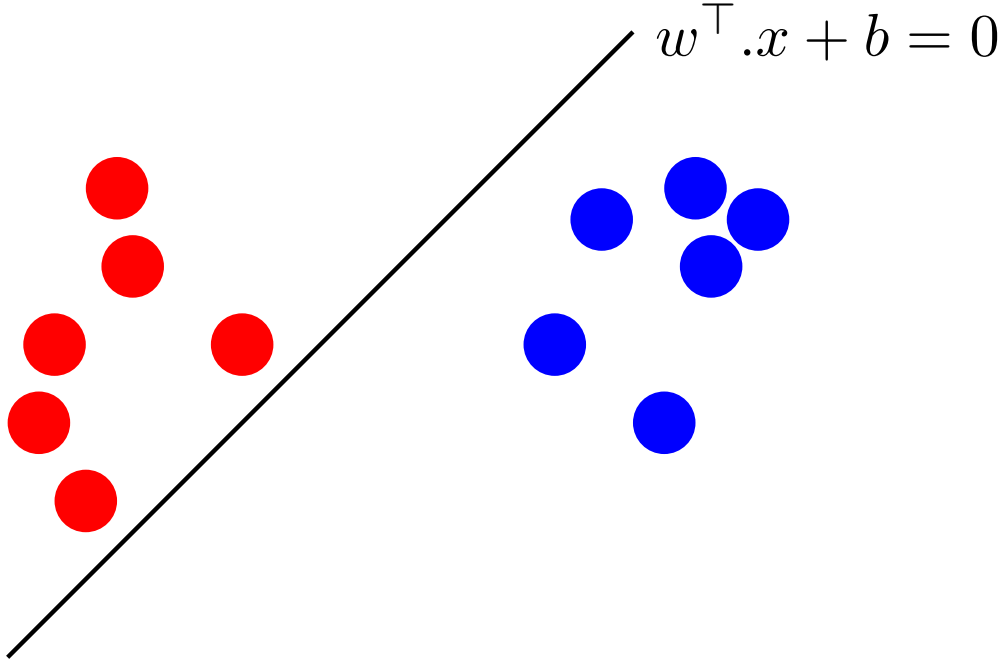
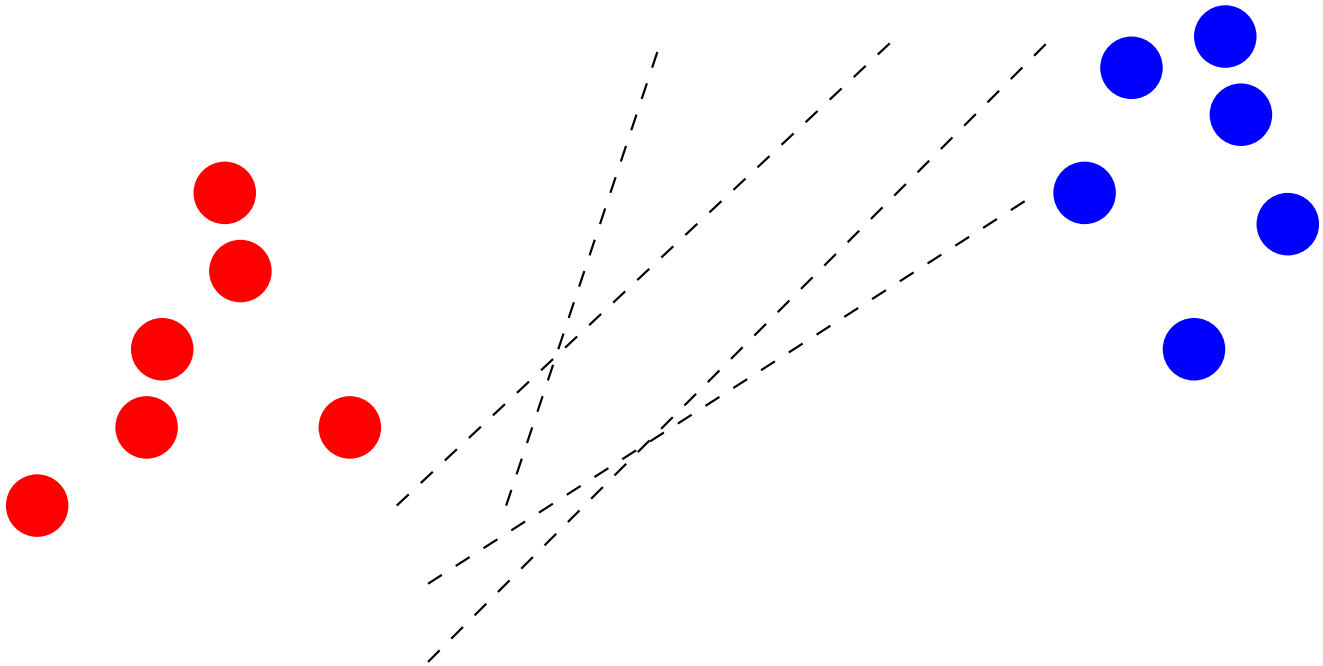


Figure – L'hyperplan qui assure la séparation entre les deux ensembles .

cpge-paradise.com

Recherche de solution

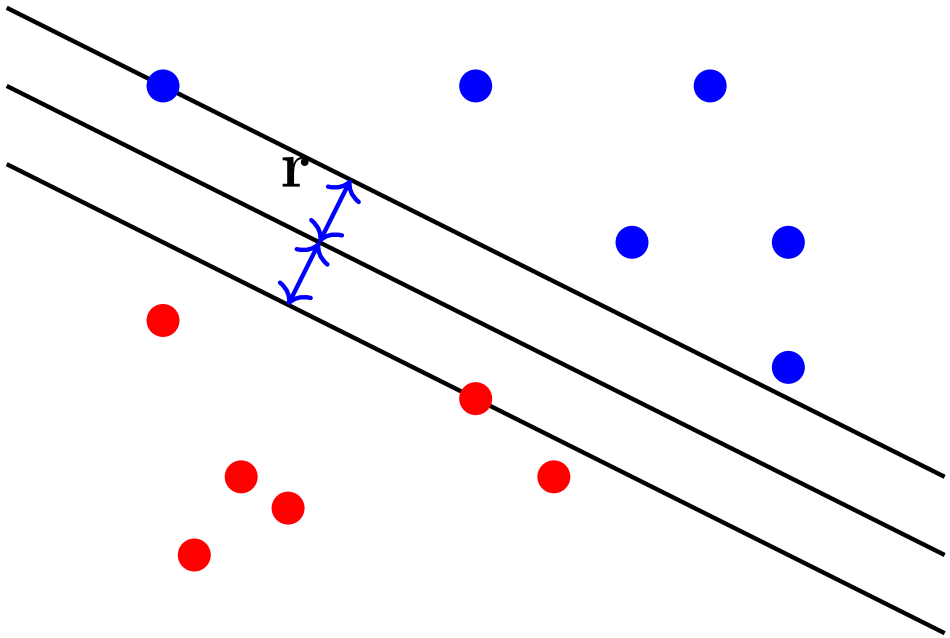
PROBLEME : Face à la présence de multiples lignes droites pouvant séparer les deux ensembles, comment procéder pour sélectionner celle qui convient le mieux ?



cpge-paradise.com

La marge

- La **marge** est définie comme la plus petite distance,entre la surface de décision et les points d'entraînement de l'ensemble de données.
- Il faut choisir l'hyperplan qui maximise la marge !



cpge-paradise.com

Formulation primale du problème

$$\begin{aligned} &\max \quad r \\ &\text{s.c} \quad y_i.(\langle w, x_i \rangle + b) \geq r \quad \forall i = 1, \ldots, N \end{aligned}$$

Expression de la marge

On pose : (condition de normalisation)

$$|\mathbf{w}^T \cdot \mathbf{x}_s + b| = 1$$

Alors : $r = \|\frac{1}{\mathbf{w}}\|$

$\rightarrow$ démonstration : voir annexe 2

cpge-paradise.com

Problème primal

$$\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|^2$$
$$\text{s.c} \quad y_i.(\langle w, x_i \rangle + b) \geq 1 \quad \forall i = 1, \ldots, N$$

Il est toutefois mieux de passer à la formulation duale de ce problème

cpge-paradise.com

Formulation standard d'un probl'eme d'optimisation

- La forme standard d'un problème d'optimisation sur $x \in \mathbb{R}^n$

$$\begin{array}{ll} \min & f_0(x) \\ \text{sous contraintes} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

où toutes les fonctions sont à valeurs dans $\mathbb{R}$.

- La valeur optimale du problème :

$$p^* = \inf_{x \in \mathbb{R}^n} \{ f_0(x) \mid f_i(x) \leq 0,\ i = 1, \ldots, m,\ h_i(x) = 0,\ i = 1, \ldots, p \}$$

cpge-paradise.com

Dualité Lagrangienne

Le Lagrangien

On définit le Lagrangien : $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ par

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

Les scalaires λ_i et ν_i sont appelés multiplicateurs de Lagrange.

cpge-paradise.com

Les conditions de Karush-Kuhn-Tucker (KKT) sont un ensemble de conditions d'optimalité pour les problèmes d'optimisation avec contraintes.

- **Conditions de faisabilité primale :**
  $\forall i \; y_i(w^T x_i + b) \geq 1$.
- **Conditions de faisabilité duales :**
  $\forall i \; \alpha_i \geq 0$.
- **Conditions de complémentarité :**
  $\forall i \; \alpha_i(y_i(w^T x_i + b) - 1) = 0$
- **Conditions de stationnarité :** $\sum_{i=1}^{n} \alpha_i y_i = 0$

cpge-paradise.com

Résolution par la méthode de Lagrange :

Pour résoudre le problème mentionné précédemment, on applique la méthode de Lagrange :

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(\langle w, x_i \rangle + b) - 1]$$

- L doit être minimisée par rapport aux variables primales et maximisée par rapport aux variables duales.

$$\frac{\partial L(w, b, \alpha)}{\partial w} = 0 \quad \frac{\partial L(w, b, \alpha)}{\partial b} = 0$$

- Un rapide calcul mène aux relations suivantes :

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

cpge-paradise.com

Formulation duale du problème :

- Le problème d'optimisation dual s'exprime :

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

s.c.

$$\sum_{i=1}^{N} y_i \alpha_i = 0 \quad \alpha_i \geq 0, \quad \forall i = 1, \ldots, N$$

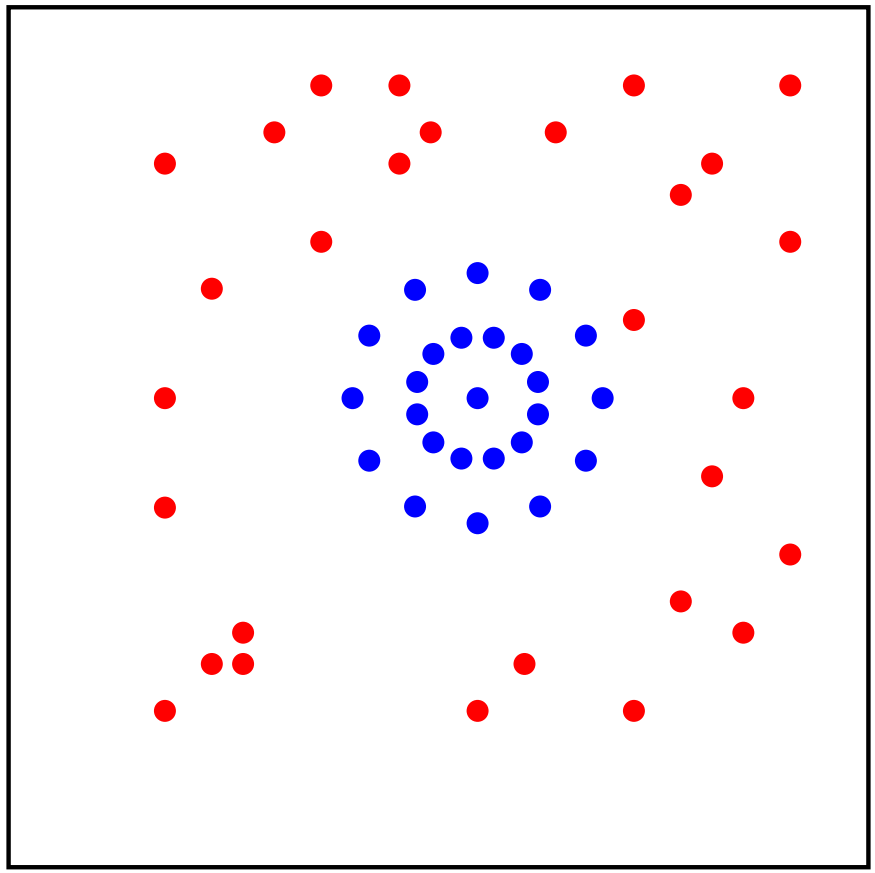
- d'où

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i x_i^T x + b$$

On obtient b par la condition de normalisation sur les vecteurs de support.

cpge-paradise.com

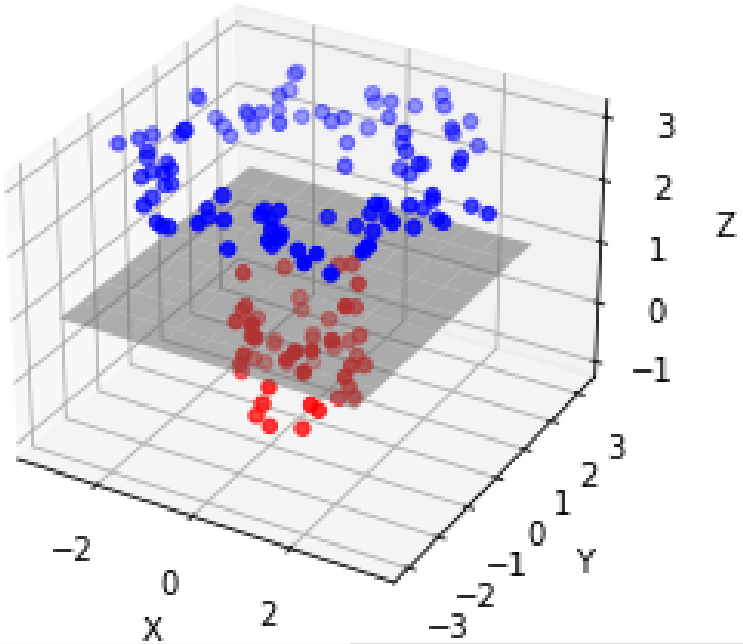
En réalité, les données sont distribuées de manière irrégulière, ce qui rend difficile la séparation linéaire des différentes classes.



cpge-paradise.com

Astuce du noyau

Principe : transposer les données dans un autre espace (en général de plus grande dimension) dans lequel elles sont linéairement séparables et ensuite appliquer l'algorithme SVM sur les données transposées.



cpge-paradise.com

Principe de la méthode à noyaux :

- On applique une transformation (non nécesairement linéaire) aux données :

$$\begin{array}{ccc} \boldsymbol{x} & \mapsto & \phi(\boldsymbol{x}) \\ \text{dimension d} & & \text{dimension D} \end{array}$$

$$\boxed{\begin{array}{c} \phi : E \to F \\ (x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2) \end{array}}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mapsto \quad \phi(\boldsymbol{x}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}$$

cpge-paradise.com

- L'hyperplan séparateur devient :

$$H(x) = \boldsymbol{w}^T \cdot \phi(x) + b = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)^T \phi(x) + b$$

- Le problème d'optimisation dual devient :

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle \\ \text{s.c.} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & \forall\, i = 1, \ldots, N \end{aligned}$$

Si $D \gg d$, la transformation ne semble pas utile!!.

cpge-paradise.com

L'idée est donc de remplacer ce calcul par une fonction noyau de la forme :

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

Pour réaliser cela, on utilise le théorème (admis) de Mercer, qui montre qu'une fonction noyau K continue, symétrique et définie positive peut s'exprimer comme un produit scalaire dans un espace de Hilbert (potentiellement de grande dimension).

exemple : $\phi(x)^T \phi(z) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \cdot \begin{bmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{bmatrix}$

$$= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1z_1x_2z_2$$
$$= (x_1z_1 + x_2z_2)^2 = (x.z)^2$$

cpge-paradise.com

- Il est alors possible d'obtenir l'équation de l'hyperplan séparateur sans calculer $\phi(x)$ de manière explicite.
  Il suffit de trouver une fonction K qui vérifie :

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

  La complexité de l'apprentissage dépendra alors N et non pas de D.

- Le noyau polynomial :

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^d$$

- Le noyau gaussien :

$$K(x_i, x_j) = \exp\left(-\frac{\|x - y\|^2}{2\gamma^2}\right)$$

cpge-paradise.com

Le problème d'optimisation dual devient alors :

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\ \text{s.c.} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \qquad \forall\, i = 1, \ldots, N \end{aligned}$$

Le calcul se fait dans l'espace d'origine, ceci est beaucoup moins coûteux qu'un produit scalaire en grande dimension.
La transformation n'a pas besoin d'être connue explicitement, seule la fonction noyau intervient dans les calculs.

cpge-paradise.com

Base de données

age	sex	doul_tor	pa	chol	glyc_jeun	restecg	ryth_max	ex_ang	segm_st	pente_st	vaisseaux	thal	target
52	1	0	125	212	0	1	168	0	1	2	2	3	0
53	1	0	140	203	1	0	155	1	3,1	0	0	3	0
70	1	0	145	174	0	1	125	1	2,6	0	0	3	0
61	1	0	148	203	0	1	161	0	0	2	1	3	0
62	0	0	138	294	1	1	106	0	1,9	1	3	2	0
58	0	0	100	248	0	0	122	0	1	1	0	2	1
58	1	0	114	318	0	2	140	0	4,4	0	3	1	0
55	1	0	160	289	0	0	145	1	0,8	1	1	3	0
46	1	0	120	249	0	0	144	0	0,8	2	0	3	0
54	1	0	122	286	0	0	116	1	3,2	1	2	2	0
71	0	0	112	149	0	1	125	0	1,6	1	0	2	1
43	0	0	132	341	1	0	136	1	3	1	0	3	0
34	0	1	118	210	0	1	192	0	0,7	2	0	2	1
51	1	0	140	298	0	1	122	1	4,2	1	3	3	0
52	1	0	128	204	1	1	156	1	1	1	0	0	0

Figure – Base de données des informations médicales de plusieurs patients

cpge-paradise.com

Algorithm 1: Algorithme de mise à jour des α_i

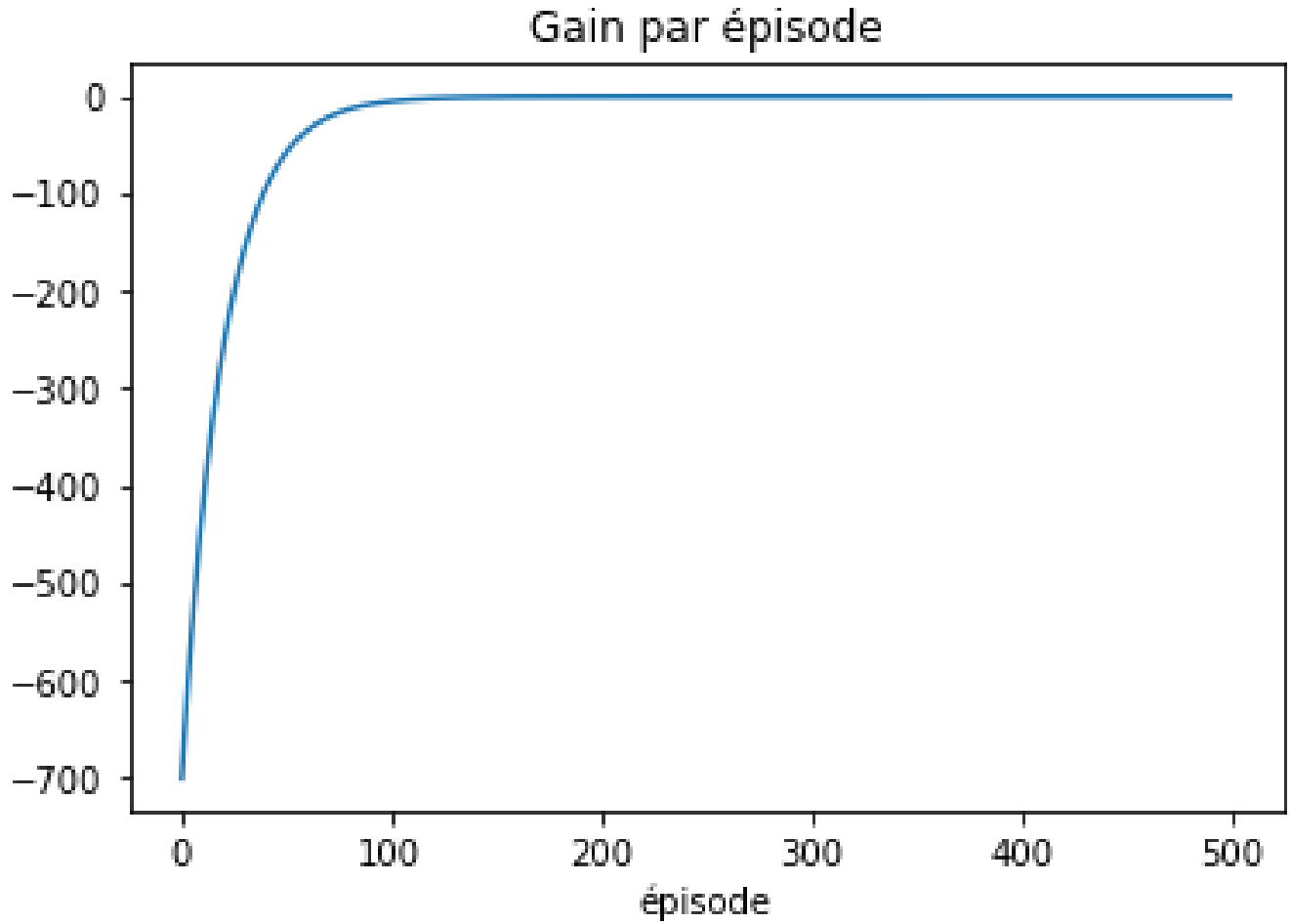
Données: - Ensemble d'entraînement $\{(x_i, y_i)\}_{i=1}^{N}$
- Taux d'apprentissage η
- Nombre d'itérations T

Résultat: Vecteur α maximisant la fonction $L(\alpha)$

Initialiser $\alpha_i \leftarrow 0$ pour tout $i = 1, \ldots, N$;
for $t = 1$ **to** T **do**
  for $i = 1$ **to** N **do**
    Calculer le gradient :
    $\frac{\partial L(\alpha)}{\partial \alpha_i} = 1 - \sum_{j=1}^{N} \alpha_j y_i y_j K(x_i, x_j)$;
    Mettre à jour $\alpha_i \leftarrow \alpha_i + \eta \frac{\partial L(\alpha)}{\partial \alpha_i}$;
  Retourner α ;

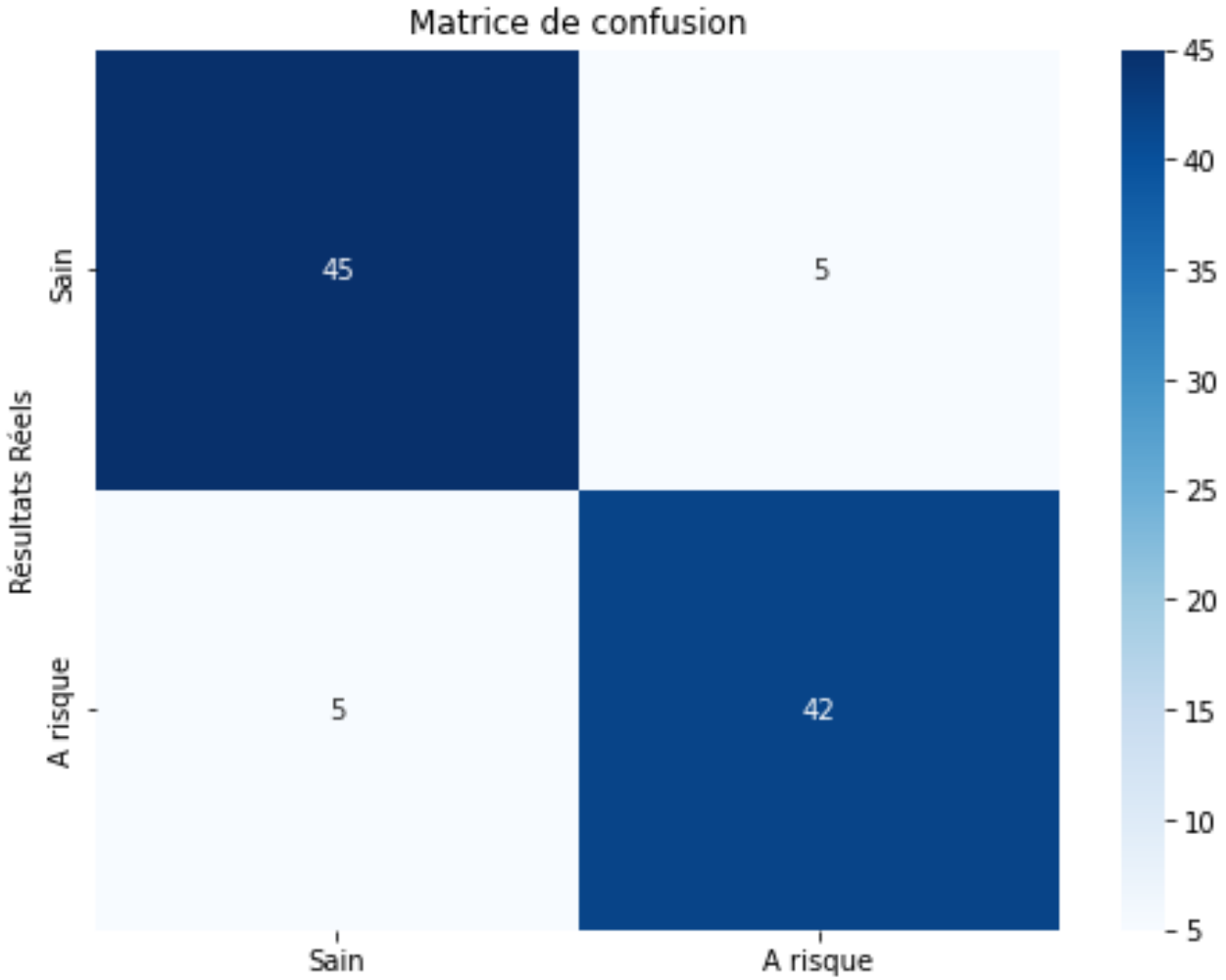
cpge-paradise.com

Convergence de l'algorithme



cpge-paradise.com

Performance de l'algorithme



cpge-paradise.com

Performance de l'algorithme

Précision du modèle développé : 89.7%
Précision du modèle généré par la bibliothèque prédéfinie scikit-learn :

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```
# Afficher la précision
print(f"Précision du modèle SVM: {accuracy * 100:.2f}%")
```

```
In [7]: runfile('C:/Users/Lenovo/Desktop/untitled2.py', wdir='C:/Users/Lenovo/
Desktop')
Précision du modèle SVM: 97.78%
```

cpge-paradise.com

Conclusion

- Initiative possible : introduire une marge souple qui tolère les mauvais classements.
- Les MVS offrent une approche prometteuse pour renforcer les initiatives de prévention pour sportifs,mais malgré cela il est important de noter que la médecine reste un domaine complexe et que des facteurs externes doivent être pris en compte.

cpge-paradise.com

Annexe 1 : démonstration de l'existence de l'hyperplan

supposons qu'il existe $c \in C$, $d \in D$ tel que : dist(C,D) = $\|u - v\|_2$
on pose :

$$w = d - c$$

$$b = \frac{\|d\|_2^2 - \|c\|_2^2}{2}$$

et $f(x) = w^T \cdot x + b = (d-c)^T \cdot x - \frac{1}{2}(\|d\|^2 - \|c\|^2)$
Il suffit de montrer que $\forall (u,v) \in C \times D$ on a
$f(u) < 0 \text{et} f(v) > 0$
on raisonne par absurde :
supposons qu'il existe $u \in D$ tel que :
$f(u) = (d-c)^T \cdot \left(u - \frac{1}{2}(d+c)\right) = (d-c)^T \cdot (u-d) + \frac{1}{2}\|d-c\|_2^2 < 0$

cpge-paradise.com

Annexe 1 : démonstration de l'existence de l'hyperplan

alors $(d-c)^T \cdot (u-d) < 0$
par conséquent $\frac{d}{dt}[\|d+t(u-d)-c\|^2]_{t=0} = (d-c)^T \cdot (u-d) < 0$
donc pour un certain $t \in]0,1]$:

$$\|d+t(u-d)-c\|_2 < \|d-c\|_2$$

donc le point $d+t(u-d)-$ c est plus proche à c que d, ce qui est absurde, ainsi :

$$\forall v \in D, f(v) > 0$$

de même on montre que :

$$\forall u \in C, f(u) < 0$$

cpge-paradise.com

Annexe 2 : démonstration de l'expression de la marge

Montrons que w est orthogonal à l'hyperplan :

Soient x_a et x_b deux vecteurs de l'hyperplan alors :

$\langle w, x_a \rangle + b = 0 \quad$ et $\quad \langle w, x_b \rangle + b = 0$

et par suite : $\langle w, x_a - x_b \rangle = 0$

donc w est orthogonal à tout vecteur de l'hyperplan.

cpge-paradise.com

Annexe 2 : démonstration de l'expression de la marge

Montrons que $r = \frac{1}{\|w\|}$:

On considère l'hyperplan d'équation $\langle w, x\rangle + b = 0$ et x_a un vecteur (de support) tel que $\langle w, x_a\rangle + b = 1$

On note x'_a la projection orthogonale de x_a sur l'hyperplan.

Or w est orthogonal à l'hyperplan, ainsi $x_a = x'_a + r \cdot \frac{w}{\|w\|}$

On a par définition $\langle w, x'_a\rangle + b = 0$

On injecte cette expression dans l'équation précédente

$\langle w, x_a - r \cdot \frac{w}{\|w\|}\rangle + b = 0$

D'où $\langle w, x_a\rangle + b - r \cdot \frac{\langle w,w\rangle}{\|w\|} = 0$

D'après les hypothèses $\langle w, x_a\rangle + b = 1$

Finalement $r = \frac{1}{\|w\|}$

cpge-paradise.com

Annexe 3 : codes python

```python
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

class SVMDualProblem:
    def __init__(self, C=0, kernel='rbf', sigma=2, degree=2):
        self.C = C
        if kernel == 'poly':
            self.kernel = self._polynomial_kernel
            self.c = 1
            self.degree = degree
        else:
            self.kernel = self._rbf_kernel
            self.sigma = sigma

        self.X = None
        self.y = None
        self.alpha = None
        self.b = 0
        self.ones = None

    def _rbf_kernel(self, X1, X2):
        return np.exp(-(1 / self.sigma ** 2) * np.linalg.norm(X1[:, np.newaxis] - X2[np.newaxis, :], axis=2) ** 2)

    def _polynomial_kernel(self, X1, X2):
        return (self.c + X1.dot(X2.T)) ** self.degree
```

cpge-paradise.com

Annexe 3 : codes python

```
def fit(self, X, y, lr=1e-3, epochs=500):
    self.X = X
    self.y = y
    self.alpha = np.random.random(X.shape[0])
    self.b = 0
    self.ones = np.ones(X.shape[0])
    y_iy_jk_ij = np.outer(y, y) * self.kernel(X, X)

    losses = []
    for epoch in range(epochs):
        gradient = self.ones - y_iy_jk_ij.dot(self.alpha)
        self.alpha = self.alpha + lr * gradient
        self.alpha[self.alpha > self.C] = self.C
        self.alpha[self.alpha < 0] = 0
        loss = np.sum(self.alpha) - 0.5 * np.sum(np.outer(self.alpha, self.alpha) * y_iy_jk_ij)
        losses.append(loss)
        if epoch % 100 == 0:
            print(f"Epoch {epoch}: Loss = {loss}")

    index = np.where((self.alpha > 0) & (self.alpha < self.C))[0]
    b_i = y[index] - (self.alpha * y).dot(self.kernel(X, X[index]))
    self.b = np.mean(b_i)

    plt.plot(losses)
    plt.title("Loss per Epoch")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.show()

def _decision_function(self, X):
    return (self.alpha * self.y).dot(self.kernel(self.X, X)) + self.b
```

cpge-paradise.com

Annexe 3 : codes python

```
    def predict(self, X):
        return np.sign(self._decision_function(X))

    def score(self, X, y):
        y_hat = self.predict(X)
        return np.mean(y == y_hat)

def load_data(file_path, delimiter):
    data = pd.read_csv(file_path, delimiter=delimiter, header=None, na_values='?')
    data = data.dropna()
    data.columns = ["age", "sex", "cp", "restbp", "chol", "fbs", "restecg", "thalach", "exang",
                    "oldpeak", "slope", "ca", "thal", "hd"]
    le = preprocessing.LabelEncoder()
    y = le.fit_transform(data["hd"])
    y = np.where(y > 0, 1, -1)  # Converting the target to binary (-1, 1)
    X = data.drop(columns=["hd"])
    return X.values, y

def plot_confusion_matrix(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Disease', 'Disease'], yticklabels=['No Disease', 'Disease'])
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.title('Confusion Matrix')
    plt.show()

if __name__ == "__main__":
    file_path = "/Users/Lenovo/Downloads/processed.cleveland.data"
    X, y = load_data(file_path, delimiter=',')

    # Split data into training and testing sets with stratification
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

cpge-paradise.com

Annexe 3 : codes python

```python
if __name__ == "__main__":
    file_path = "/Users/Lenovo/Downloads/processed.cleveland.data"
    X, y = load_data(file_path, delimiter=',')

    # Split data into training and testing sets with stratification
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

    svm = SVMDualProblem(C=1.0, kernel='rbf', sigma=0.5)
    svm.fit(X_train, y_train, lr=1e-3, epochs=500)

    y_train_pred = svm.predict(X_train)
    y_test_pred = svm.predict(X_test)

    print("Training accuracy:", accuracy_score(y_train, y_train_pred))
    print("Testing accuracy:", accuracy_score(y_test, y_test_pred))

    print("\nClassification Report (Training):")
    print(classification_report(y_train, y_train_pred))

    print("\nClassification Report (Testing):")
    print(classification_report(y_test, y_test_pred))

    print("\nConfusion Matrix (Testing):")
    plot_confusion_matrix(y_test, y_test_pred)
```

cpge-paradise.com

Annexe 4 :Théorème de Mercer

Soit K une fonction continue, symétrique et définie positive sur un domaine compact $X \subset \mathbb{R}^n$. Alors, il existe une transformation $\phi : X \to H$, où H est un espace de Hilbert, telle que $K(x, y) = \langle \phi(x), \phi(y) \rangle$
pour tous $x, y \in X$.
Définie positive : Pour toute fonction g non nulle intégrable sur X, l'intégrale suivante est positive

$$\iint_{X \times X} K(x, y) g(x) g(y) \, dx \, dy \geq 0$$