Prédiction des Tremblements Physiologiques dans la Chirurgie Robotique par les Machines à Vecteurs de Support

Nizar Chtia
MK250M

Sommaire

- Motivation
- Problématique
- Machines à Vecteurs de Support
- Problème d’optimisation
- Implémentation
- Méthode des noyaux
- Résolution
- Visualisation
- Conclusion
- Annexe

1 MOTIVATION

Motivation

Figure 1 - Détection de tumeurs

Figure 2 - Assistance médicale robotique

Motivation

Figure 3 – Imagerie médicale

Figure 4 – Prédiction des maladies

2 PROBLEMATIQUE

Problématique

- Comment implémenter un algorithme des Machines à Vecteurs de Support (SVM) pour améliorer la précision d'un robot chirurgical?

Objectifs

- Comprendre le fonctionnement d'un algorithme SVM
- Trouver les paramètres optimales pour une haute précision du système robotique

3

SVM

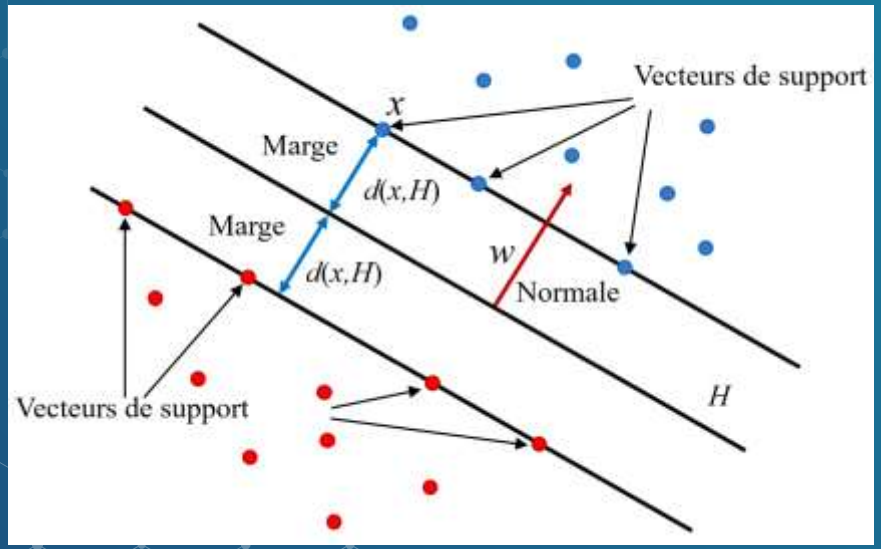
Définition et Fonctionnement

SVM

- **Définition:** Les SVM sont des modèles de Machine Learning supervisés pour résoudre les problèmes de classification ou de régression.

- **Fonctionnement:** Ramener le problème de classification/régression à un hyperplan qui sépare les classes des données, dont la frontière est la plus éloignée possible des points de données (Marge maximale).

SVM



Equation de l'hyperplan:

$$(H):\ w.x + b = 0$$

Fonction d'hypothèse:

$$\mathrm{h}(x_i) = \begin{cases} +1\ \ si:\ \ w.x + b \geq 0 \\ -1\ si: w.x + b < 0 \end{cases}$$

SVM *Critères de comparaison*

Critère 1

On considère l'hyperplan: (H): $w.x + b = 0$

et la base de données : $D = \{(x_i, y_i) | x_i \in R^n, y_i \in \{1, -1\}\}$

On obtient: $$\mathrm{h}(x_i) = \begin{cases} w.x + b = 0 \ \ si\ x \in H \\ \begin{cases} w.x + b < 0 \\ w.x + b > 0 \end{cases} si\ x\ n'appartient\ pas\ \text{à}\ H \end{cases}$$

Pour trouver le point le plus proche à l'hyperplan, on calcule pour chaque point: $\beta = |w.x + b|$

Pour chaque hyperplan on obtient: $B = min_{i=1\ldots m}|w.x + b|$

Pour s hyperplans, l'optimal est: $H = max_{i=1\ldots S}\{h_i | B_i\}$

SVM *Critères de comparaison*

Problème: Puisqu'on calcule la valeur absolue de $w.x + b$, on peut trouver la même valeur pour un hyperplan correct et un autre incorrect.

SVM *Critères de comparaison*

Critère 2

Pour résoudre ce problème, on définit: $f(x) = y(w.x + b)$

On obtient: $\begin{cases} f(x) \geq 0 \ \ si\ x\ est\ correctement\ classé \\ f(x) < 0 \ \ si\ x\ n'est\ pas\ correctement\ classé \end{cases}$

On calcule pour chaque point: $F = \ min_{i=1\ldots m} y_i(w.x + b)$

Pour s hyperplans, l'optimal est: $H = max_{i=1\ldots s}\{h_i | F_i\}$

SVM *Critères de comparaison*

Problème: Invariance d’échelle,
Deux vecteurs qui ont le même vecteur unitaire auront de différentes valeurs de F même s’ils représentent le même Hyperplan.

SVM *Critères de comparaison*

Critère 3

Pour résoudre ce problème, on définit: $\gamma = y\left(\frac{w}{\|w\|}.x + \frac{b}{\|w\|}\right)$

On calcule pour chaque point **la marge géométrique**:

$$\boldsymbol{M = min_{i=1\ldots m} y_i \left(\frac{w}{\|w\|}.x + \frac{b}{\|w\|}\right)}$$

Pour s hyperplans, l'optimal est celui avec la marge M la plus grande.

4

Problème d'optimisation

Trouver les valeurs w et b pour l'hyperplan optimal

Problème d'optimisation

$$max_{w,b}\ M$$
$$contrainte:\ \gamma_i \geq M, \qquad i = 1 \dots m$$

Or: $M = \frac{F}{\|w\|}$

$$max_{w,b}\ M$$
$$contrainte:\ f_i \geq F, \qquad i = 1 \dots m$$

On pose: $F = 1$

$$max_{w,b}\ \frac{1}{\|w\|}$$
$$contrainte:\ f_i \geq 1, \qquad i = 1 \dots m$$

Problème d'optimisation

$$min_{w,b}\|w\|$$
$$contrainte:\ f_i \geq 1, \qquad i = 1 \ldots m$$

Pour simplifier les dérivées:

Optimisation Quadratique

$$\boldsymbol{min_{w,b}\frac{1}{2}\|w\|^2}$$
$$\boldsymbol{contrainte:\ f_i \geq 1, \qquad i = 1 \ldots m}$$

5 Implémentation

Implémentation

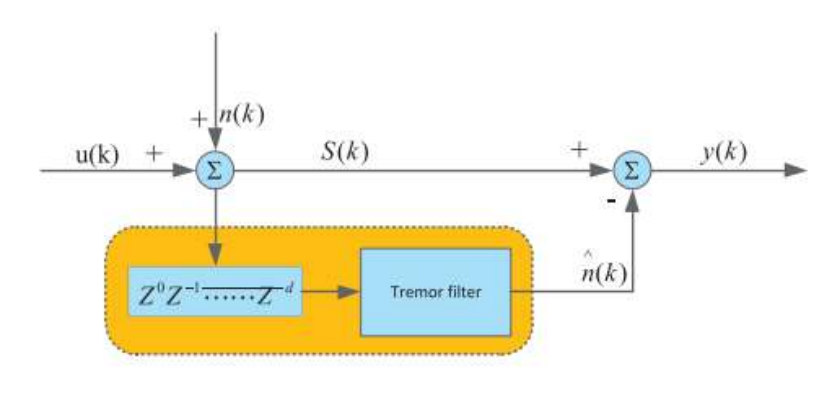


Figure 6 – Architecture des SVM

$u(k)$: Signal de mouvement désiré

$n(k)$: signal du tremblement réel

$\hat{n}(k)$: signal du tremblement estimé

$S(k) = u(k) + n(k)$

Implémentation

On définit, les données d'entrée:

$$(S_k, n_k)_k^N \ , \ k = 1 \dots N$$

Alors, la sortie est sous la forme:

$$y(k) = S(k) - \hat{n}(k) = u(k) + n(k) - \hat{n}(k)$$

Implémentation

La fonction du **signal désiré** est:

$$\hat{n}(k) = \omega^T \phi(S) + b$$

ω:Vecteur des paramètres du poids
b: Le biais
ϕ:Application linéaire qui transporte les données vers un espace de plus grande dimension

$$\textbf{On a: } \forall \varepsilon > 0,$$
$$\textbf{un hyperplan } \hat{n}(k) = \omega^T \phi(S) + b \textbf{ vérifie :}$$
$$|n(k) - \hat{n}(k)| \leq \varepsilon$$

Implémentation

Résoudre ce problème, revient à résoudre le problème d'**Optimisation Quadratique** déjà trouvé:

$$\min \quad \frac{1}{2}\omega^T\omega + \frac{1}{2}Cm_k\sum_{i=1}^{N}\xi_i^2$$
$$\text{subject to} \quad n_i = \omega^T\phi(S) + b + \xi_i$$

$\boldsymbol{\xi}$:Variable d'écart, elle permet de satisfaire à la contrainte même si quelques points n'y répondent pas

$\mathbf{C}$:Paramètre de régularisation, il détermine le taux d''influence

$\boldsymbol{m_k}$:Poids, exprime le contribution de chaque exemple

Implémentation

Théorème: Pour trouver le minimum d'une fonction ***f*** sous une contrainte ***g***, il faut résoudre l'égalité suivante:

$$\nabla \mathbf{f}(\mathbf{x}) - \boldsymbol{\alpha} \nabla \mathbf{g}(\mathbf{x}) = \mathbf{0}$$

Pour résoudre le problème d'Optimisation Quadratique, on introduit le **Lagrangien**:

$$L(\omega, b, \xi, \lambda) = \frac{1}{2}\omega^T\omega + \frac{1}{2}Cm_k\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\lambda_i\left[\omega^T\phi(S_i) + b + \xi_i - n_i\right]$$

Implémentation

On calcule des dérivées partielles:

$$(\partial L/\partial \omega) = 0,\ (\partial L/\partial b) = 0,\ (\partial L/\partial \xi_i) = 0$$
$$(\partial L/\partial \lambda_i) = 0$$

On obtient les équations suivantes:

$$\begin{cases} \omega = \sum_{i=1}^{N} \lambda_i \phi(S_i) \\ \sum_{i=1}^{N} \lambda_i = 0 \\ \lambda_i = C m_k \xi_i, \quad i = 1, 2, \ldots, N \\ \omega^T \phi(S_i) + b + \xi_i - n_i = 0. \end{cases}$$

Implémentation

On résout le système afin d'éliminer ω et ξ_i :

$$\sum_{i=1}^{N} \lambda_i \phi(S_i)^T \phi(S_j) + b + \frac{\lambda_i}{Cm_k} = n_i, \quad i, j = 1, 2, \ldots, N.$$

On réécrit les équations précédentes sous forme matricielle:

$$\begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & Q + (Cm_k)^{-1} E \end{bmatrix} \begin{bmatrix} b \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix}$$

avec:

$$\begin{cases} \vec{1} = [1, 1, \ldots, 1]^T \\ Q = \phi(S_i)^T \phi(S_j) \\ \lambda = [\lambda_1, \lambda_2, \ldots, \lambda_N]^T \\ n = [n_1, n_2, \ldots, n_N]^T. \end{cases}$$

et E: la matrice identité

Implémentation

On en déduit que:

$$\phi - \begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & Q + (Cm_k)^{-1}E \end{bmatrix}$$

Alors on obtient:

$$\begin{bmatrix} b \\ \lambda \end{bmatrix} = (Q)^{-1}\phi^T \begin{bmatrix} 0 \\ n \end{bmatrix}$$

$$Q_{i,j} = \phi(S_i)^T \phi(S_j)$$

6 Méthode des Noyaux

Méthode des Noyaux

Problème: La méthode SVM qu'on a adopté tout au long de l'étude est incapable de classer des données linéairement inséparables.

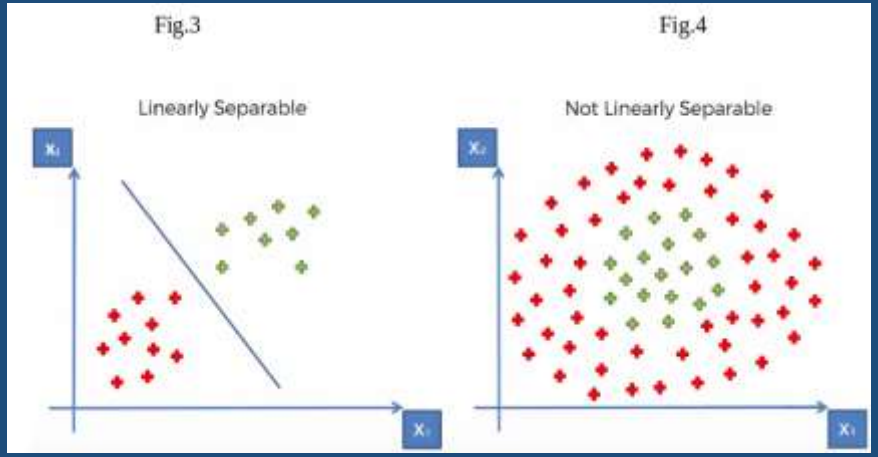


Figure 7 – Séparation linéaire et non linéaire

Méthode des Noyaux

Pour dépasser cet obstacle, on introduit la méthode des **Noyaux**.

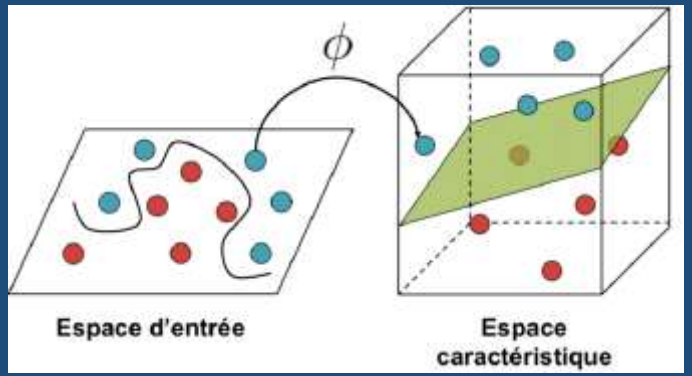


Figure 8 - Transformation de noyau (Kernel)

Elle consiste à utiliser une **fonction noyau** qui transformera notre espace en un espace de plus grande dimension ou il existe un hyperplan qui sépare nos données.

Méthode des Noyaux

On va utiliser une fonction noyau Hybride composée de la fonction noyau radiale (RBF) et polynomiale.

$$K(S,S_j)_{\text{Hybrid}} = K(S,S_j)_{\text{RBF}} + K(S,S_j)_{\text{Poly}}$$
$$= \alpha \exp\left(-\frac{|S-S_j|^2}{\sigma^2}\right) + (1-\alpha)\left[S\cdot S_j + 1\right]^q$$

Il en résulte:

$$\phi = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & K(S_1,S_1)+(Cm_k)^{-1} & \cdots & K(S_1,S_N) \\ \cdots & \cdots & \cdots & \cdots \\ 1 & K(S_N,S_1) & \cdots & K(S_N,S_N)+(Cm_k)^{-1} \end{bmatrix}$$

$$Q_{i,j} = \phi(S_i)^T \phi(S_j) = K(S_i,S_j), \quad i,j = 1,2,\ldots,N$$

7 Résolution

Résolution

Alors la sortie de la méthode des SVM est sous la forme:

$$n(S) = \sum_{i=1}^{N} \lambda_i K(S, S_i) + b.$$

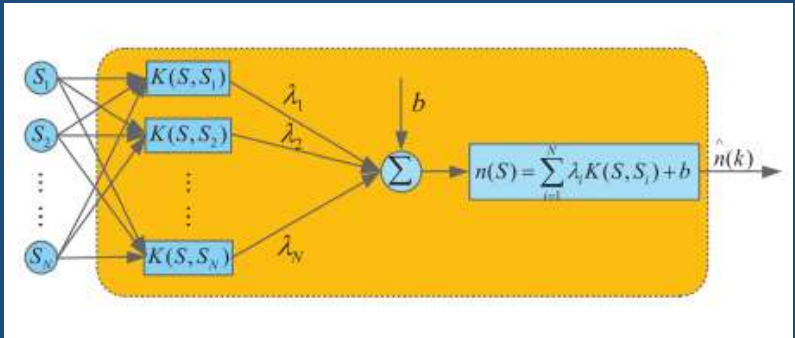
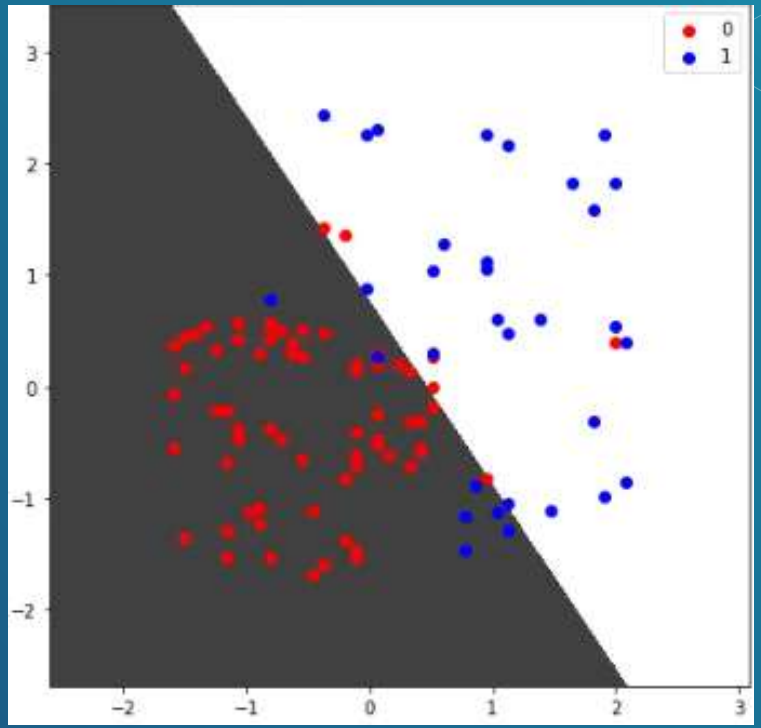


Figure 9 – Model SVM

8 Visualisation

Visualisation



Coefficient de précision: 0,88

Figure 10 – Visualisation de la prédiction en utilisant le fonction noyau Linéaire

Visualisation

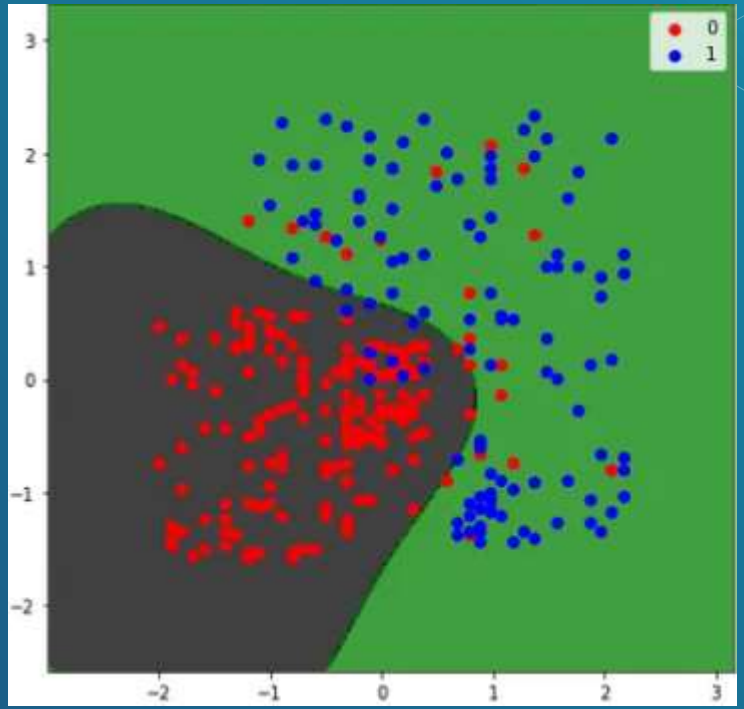
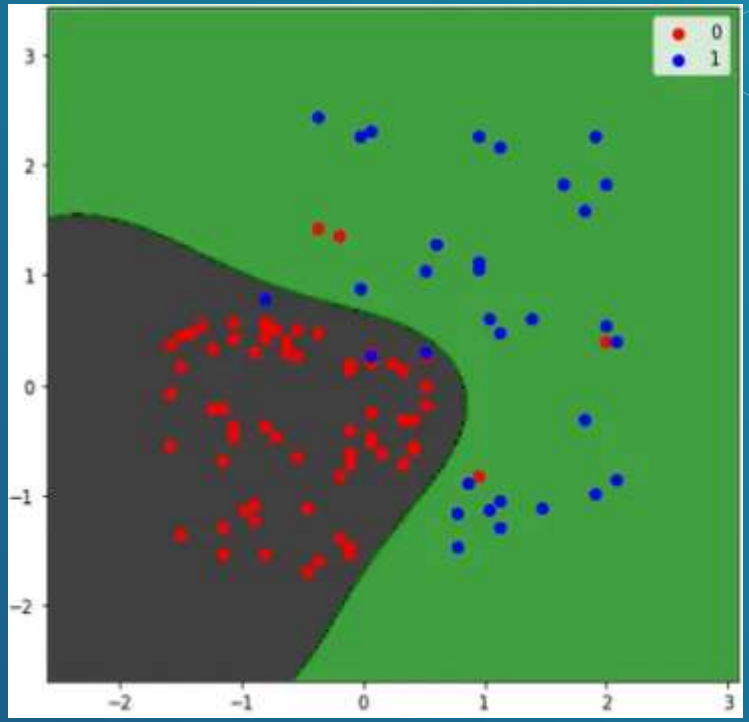


Figure 10 – Visualisation de la fonction noyau radiale

Visualisation



Coefficient de précision: 0,93

Figure 11 – Visualisation de la prédiction en utilisant la fonction de noyau radiale

9 Conclusion

Conclusion

- Meilleure précision avec la fonction noyau radiale (RBF)
- Meilleure précision du mouvement effectué par le robot
- SVM donnent même résultat chaque fois

MERCI POUR VOTRE ATTENTION

10 Annexe

Annexe

```
# importing SVM module
from sklearn.svm import SVC

# kernel to be set radial bf
classifier1 = SVC(kernel='rbf')

# traininf the model
classifier1.fit(X_train, y_train)

# testing the model
y_pred = classifier1.predict(X_test)

# importing accuracy score
from sklearn.metrics import accuracy_score

# printing the accuracy of the model
print(accuracy_score(y_test, y_pred))
```

Annexe

```
# plotting the fgiure
plt.figure(figsize = (7,7))

# assigning the input values
X_set, y_set = X_train, y_train

# ploting the linear graph
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier1.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('black', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

# ploting scattered graph for the values
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'blue'))(i), label = j)

# labeling the graph
plt.title('Correct VS Incorrect')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```