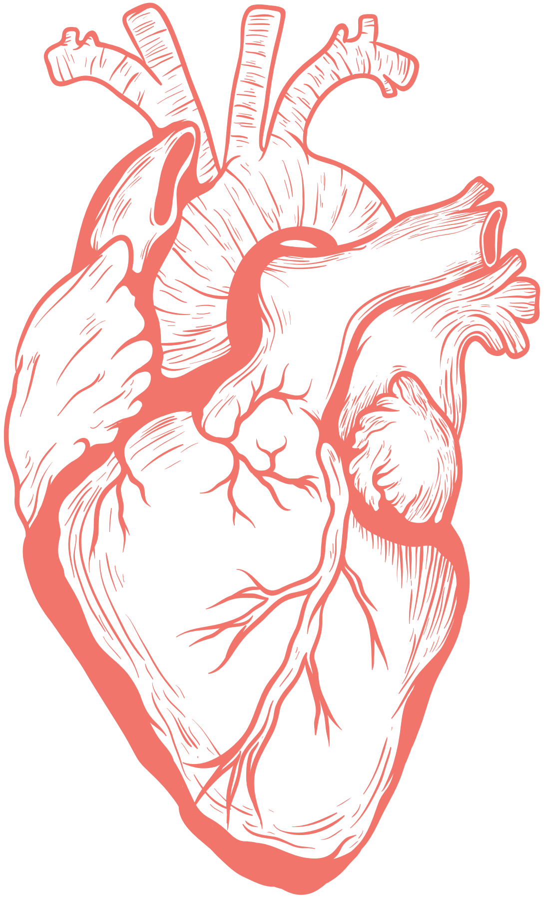


Application des Machines à Vecteurs de Support en cardiologie :
prévision intelligente des Cardiopathies

Othmane EL HAMDAOUI
Filière MP

Réalisé sous l'encadrement de:
M. Hassane SADDIKI

N° SCEI: 34098

CPGE Omar Ibn Al-Khattab (BEURD), Meknès

Sommaire

Introduction et problématique

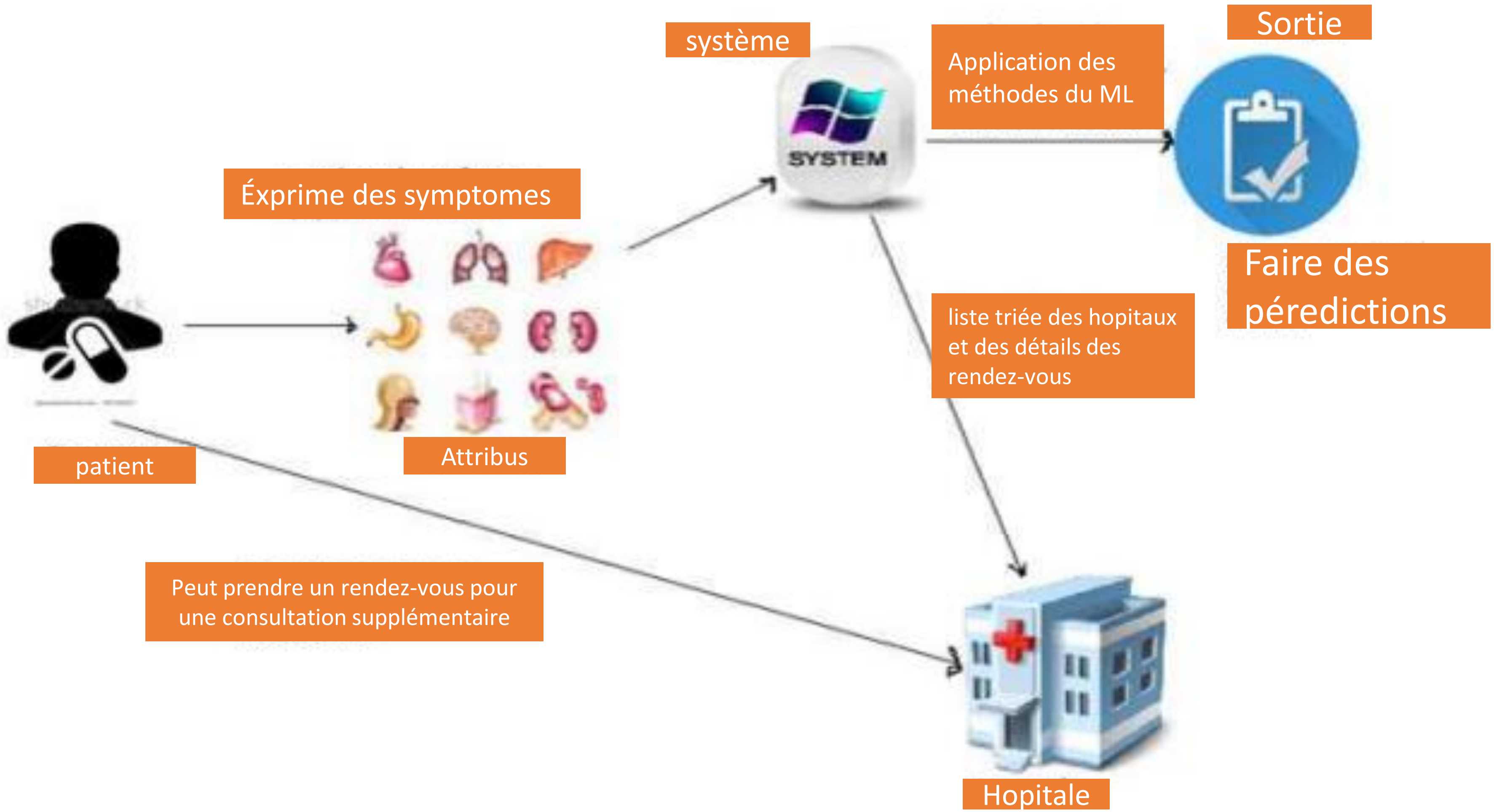


Fig 1: La médecine et l'intelligence artificielle

Introduction et problématique

Dans quelle mesure Le Machine Learning (SVM) peut aider à la prévention des cardiopathies?

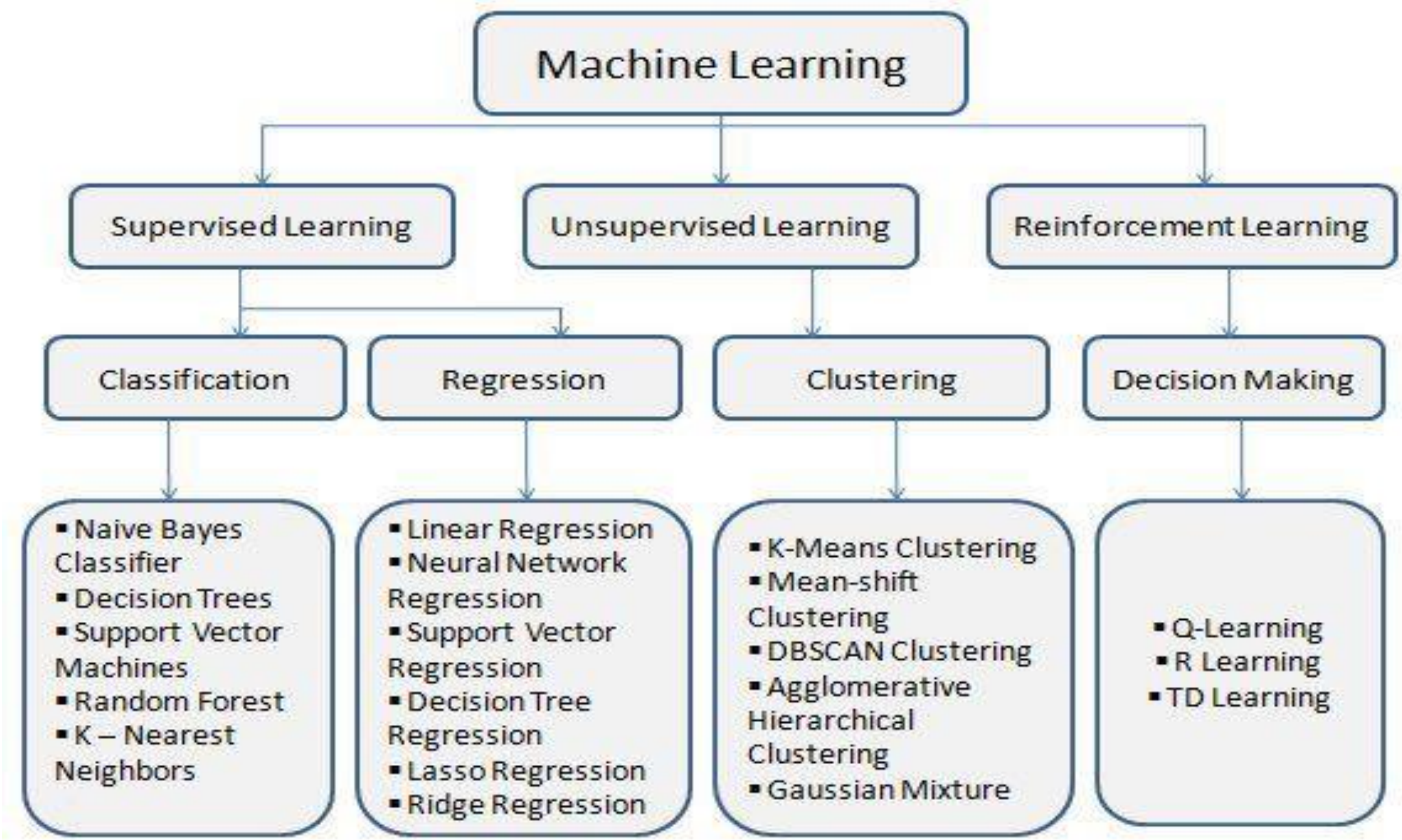
L’apprentissage automatique

L’apprentissage automatique, également connu sous le nom d’apprentissage machine et en Anglais Machine Learning, est une forme d’intelligence artificielle (IA) qui permet à un système d’apprendre à partir des données et non à l’aide d’une programmation explicite.

Machine Learning (ML) **!=** Programmation classique

Les algorithmes d’apprentissage peuvent se catégoriser selon le type d’apprentissage qu’ils emploient...

L'apprentissage automatique



Qu'est-ce que sont les Machines à Vecteurs de Support?

C'est une méthode de l'apprentissage automatique supervisée. Elles utilisent les hyperplans pour résoudre des problèmes de la regréssion et de la classification.

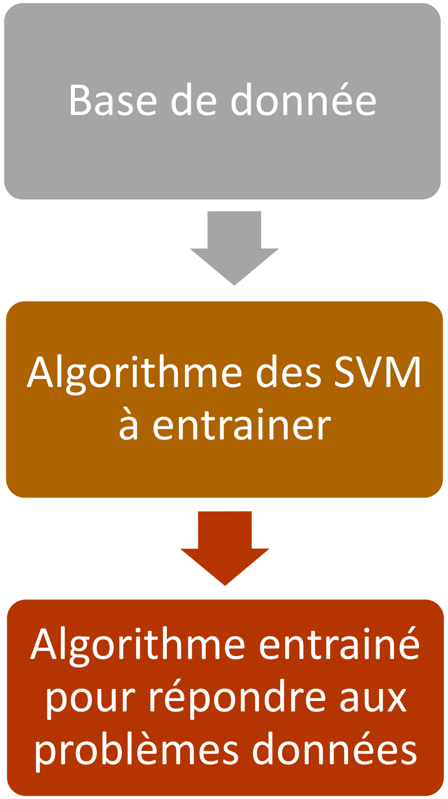
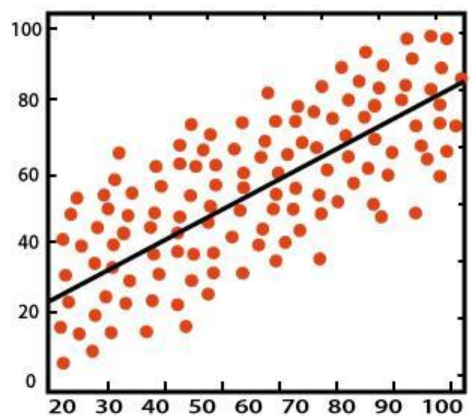


Fig 2: Processus générale de la résolution d'un problème avec SVMs

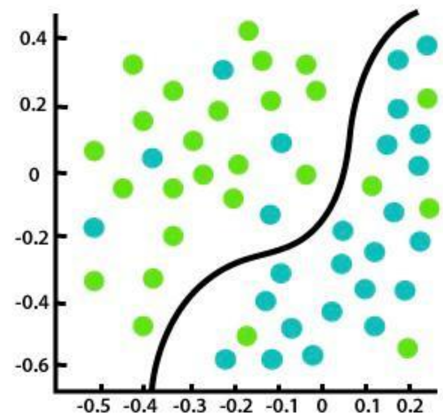
Les cas d’utilisation des SVMs

Regréssion



<u>Exemples:</u>
+ La prediction de la meteo

Classification



<u>Exemples:</u>
+ Détection de courier indésirable
+ Domaine médical

Fig 3: Les cas d’utilisation des SVMs

La classification:

Les données sont linéairement séparables

Les données ne sont pas linéairement séparables

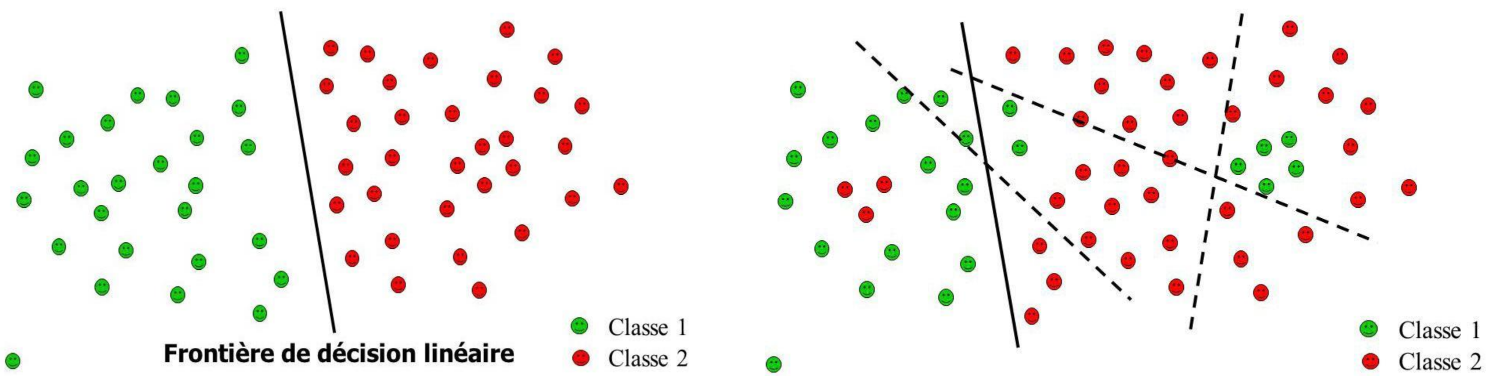


Fig 4: SVMs pour la classification

Cas des données linéairement séparables

Construction de l’hyperplan de la séparation:

$$h(x) = w^T x + w_0$$

$x = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T$ Le vecteur d’entrée

$w = (\mathbf{w}_1, \ldots, \mathbf{w}_N)^T$ Un vecteur de poids

w_0 : Bias

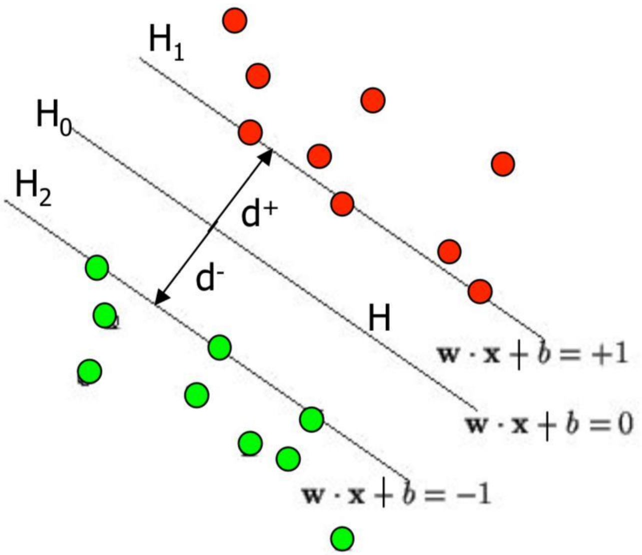


Fig 5: Séparation par hyperplans

N est la dimesion du vecteur d’entrée

Cas des données linéairement séparables

La frontière de décision est:

$$h(x) = w^T x + w_0 = 0$$

Appelée l'hyperplan séparateur

Cas des données linéairement séparables

Problème: Il éxiste plusieurs hyperplans séparateurs

Fig 6: Plusieurs hyperplans séparateurs

Comment choisir le plus optimale?

Cas des données linéairement séparables

On doit choisir alors celui qui a la marge maximale

La marge: c'est la distance entre l'hyperplan et les échantiollons les plus proches (vecteurs de supports)

L'hyperplan qui maximise la marge:

$$\arg\max_{w,w_0} \min_k \{||x - x_k|| : x \in \mathbf{R}^n, w^T x + w_0 = 0\}$$

Il suffit de trouver w et w0 pour determiner h(x)=0

Cas des données linéairement séparables

La distance minimale d'un échantillon x_k à l'hyperplan est donnée par sa projection orthogonale sur le vecteur de poids w

$$\frac{y_k\left(w^T x_k + w_0\right)}{||w||}$$

L'hyperplan qui maximise la marge devient:

$$\arg\max_{w,w_0}\{\frac{1}{||w||}\min_x |w^T x + w_0|\}$$

Cas des données linéairement séparables

Pour des arguments de simplification on prend:

$$w^T x^+_{marge} + w_0 = 1$$

$$w^T x^-_{marge} + w_0 = -1$$

$$\forall k \in [|1, p|]$$

$$w^T x_k + w_0 \geq 1;\ y_k = 1$$

$$w^T x_k + w_0 \leq 1;\ y_k = -1$$

$$y_k(w^T x_k + w_0) \geq 1$$

Alors la marge vaut:

$$\frac{1}{||w||}$$

Cas des données linéairement séparables

La formulation primale

Un problème d'optimisation quadratique:

Il faut donc déterminer w et w_0 qui minimisent:

$$\frac{1}{2}||w||^2$$

Sous contraintes:

$$\forall k \in [|1, p|]$$

$$y_k(w^T x_k + w_0) \geq 1$$

P: est la dimension de l'ensemble d'apprentissage

Cas des données linéairement séparables

La formulation dual

Transformation du problème d'optimisation

Méthode des multiplicateurs de Lagrange

$$L(w, w_0, \alpha) = \frac{1}{2}||w||^2 - \sum_{k=1}^{p} \alpha_k \{y_k(w^T x + w_0) - 1\}$$

Le Lagragien doit être minimisé par rapport a w et w_0 et maximisé par rapport à α

Problème dual:

$$\tilde{L}(\alpha) = \sum_{k=1}^{p} \alpha_k - \frac{1}{2} \sum_{i,j} \alpha_j \alpha_i y_i y_j x_i^T x_j$$

Sous contrainte: $\alpha_k \geq 0 \text{ et } \sum_{k=1}^{p} \alpha_k y_k = 0$

Cas des données linéairement séparables

Solution du problème d'optimisation

$$\sum_{k=1}^{p} \alpha_k y_k x_k = w^*$$

$$\sum_{k=1}^{p} \alpha_k y_k = 0$$

$$w_0^* = y_s^* - \sum_{i=1}^{m} \alpha_i^* y_i (x_i \cdot x_s)$$

Propriété1 : seuls les ai correspondant aux points les plus proches sont non-nuls. On parle des points de support (exemples critiques).

Propriété 2 : seuls interviennent les produits scalaires entre les observations y dans le problème d'optimisation.

Cas des données linéairement séparables

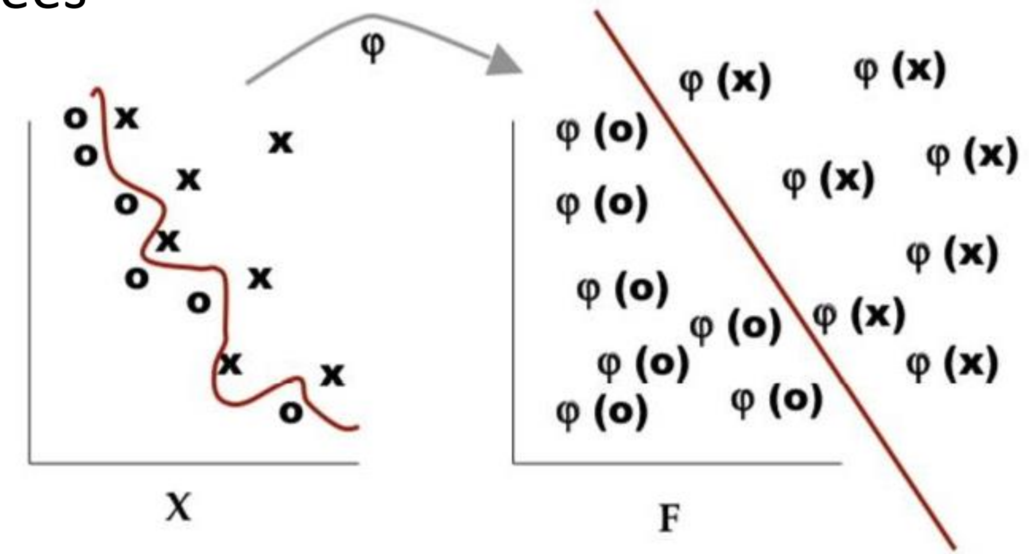
D'ou l'hyperplan optimal est:

$$h(x) = \sum_{k=1}^{p} \alpha_k^* y_k (x \cdot x_k) + w_0$$

Qu'en est-il du contraire?

Cas des données non séparables linéairement

Ce qui est le cas pour la majorité écrasante des bases de données fournies



On projettent les données dans un espace de grande dimension où les données seront linéairement séparables.

Cas des données non séparables linéairement

Principe des méthodes à noyaux:

Appliquer aux vecteurs d'entrée une transformation non-lineaire $\phi(x_i)$

Cette transformation serve de transporter les données d'un espace où ils sont linéairement inséparables à un espace nommé **espace de redescription** où ils seront linéairement séparables .

Dans cet espace on cherche l'hyperplan:

$$h(x) = w^T \phi(x) + w_0$$

Cas des données non séparables linéairement

En utilisant la même procédure que dans le cas sans transformation:

Problème d'optimisation devient:

$$\tilde{L}(\alpha) = \sum_{k=1}^{p} \alpha_k - \frac{1}{2} \sum_{i,j} \alpha_j \alpha_i y_i y_j \phi(x_i)^T \phi(x_j)$$

Sous contraintes: $\alpha_k \geq 0$ et $\sum_{k=1}^{p} \alpha_k y_k = 0$

Cas des données non séparables linéairement

Pour résoudre ce problème on utilise l'astuce de noyau qui conciste à utiliser une fonction noyau qui vérifie:

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

Elle doit étre symétrique et semi-définie positive.

Cas des données non séparables linéairement

D’où l’éxpression de l’hyperplan séparateur en fonction noyau est:

$$h(x) = \sum_{k=1}^{p} \alpha_k^* y_k K(x_k, x) + w_0$$

Cas des données non séparables linéairement

Quelques fonctions noyaux usuelles:

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$$

Le noyau Gaussien (RBF)

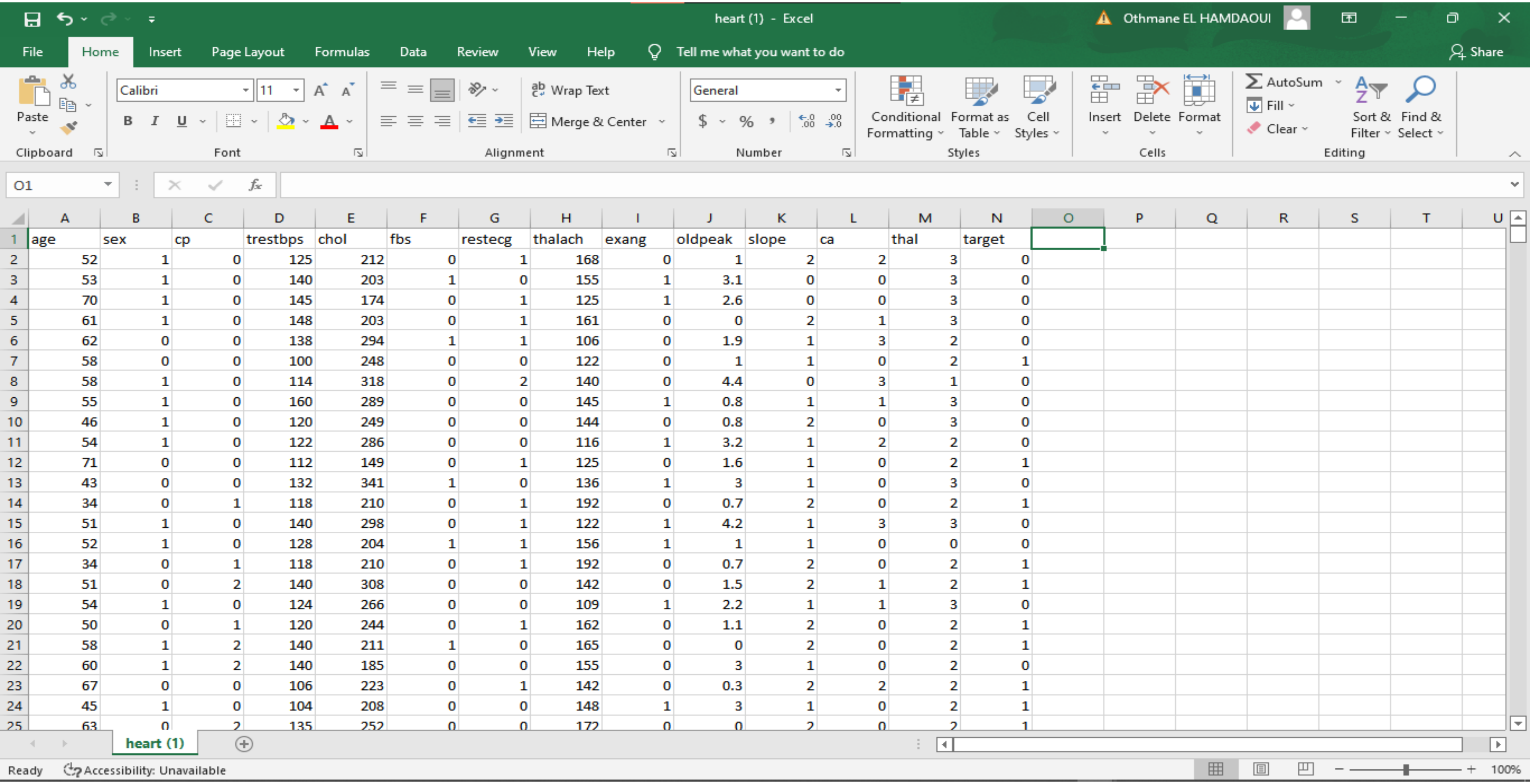
$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathbf{x}_i^T \cdot \mathbf{x}_j$$

Le noyau linéaire

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + c\right)^d$$

Le noyau polynomiale

Simulation informatique: application à la Médecine



age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	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
34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
51	0	2	140	308	0	0	142	0	1.5	2	1	2	1
54	1	0	124	266	0	0	109	1	2.2	1	1	3	0
50	0	1	120	244	0	1	162	0	1.1	2	0	2	1
58	1	2	140	211	1	0	165	0	0	2	0	2	1
60	1	2	140	185	0	0	155	0	3	1	0	2	0
67	0	0	106	223	0	1	142	0	0.3	2	2	2	1
45	1	0	104	208	0	0	148	1	3	1	0	2	1
63	0	2	135	252	0	0	172	0	0	2	0	2	1

Fig 7: La base de données utilisée

Simulation informatique: application à la Médecine

Base de données:

<u>Paramétre :</u>
Age: age
Sex: sexe
Cp: douleur thoracique
Trestbps: pression arterielle
Chol: cholesterol sérique
Fbs: glycémie à jeun
Restecg: résultats électrocardiographiques au repos
Thalach: fréquence cardiaque maximale atteinte
Exang: angine d’effort
Oldpeak, slope: valeurs prises de l’électrograme
Ca: le nombres des vaisseaux cardiaques majeurs
Thal: résultat de stresse au thalium
Target: malade ou non

Simulation informatique: application à la Médecine

```
In [22]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
```

Fig 8: Création du modèle

```
In [24]: from sklearn.svm import SVC


         svm_clf = SVC(kernel='rbf', gamma=0.1, C=1.0)
         svm_clf.fit(X_train, y_train)

         print_score(svm_clf, X_train, y_train, X_test, y_test, train=True)
         print_score(svm_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 95.40%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.969419    0.941026  0.953975    0.955222      0.954490
recall       0.932353    0.973475  0.953975    0.952914      0.953975
f1-score     0.950525    0.956975  0.953975    0.953750      0.953916
support    340.000000  377.000000  0.953975  717.000000    717.000000
_______________________________________________
Confusion Matrix:
 [[317  23]
 [ 10 367]]

Test Result:
================================================
Accuracy Score: 90.26%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.944828    0.865031  0.902597    0.904929      0.906225
recall       0.861635    0.946309  0.902597    0.903972      0.902597
f1-score     0.901316    0.903846  0.902597    0.902581      0.902540
support    159.000000  149.000000  0.902597  308.000000    308.000000
_______________________________________________
Confusion Matrix:
 [[137  22]
 [  8 141]]
```

Fig 9: Construction du modèle de la classification

```
In [27]: from sklearn.model_selection import GridSearchCV

svm_clf = SVC(kernel='rbf', gamma=0.1, C=1.0)

params = {"C":(0.1, 0.5, 1, 2, 5, 10, 20),
          "gamma":(0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1),
          "kernel":('linear', 'poly', 'rbf')}

svm_cv = GridSearchCV(svm_clf, params, n_jobs=-1, cv=5, verbose=1, scoring="accuracy")
svm_cv.fit(X_train, y_train)
best_params = svm_cv.best_params_
print(f"Best params: {best_params}")

svm_clf = SVC(**best_params)
svm_clf.fit(X_train, y_train)

print_score(svm_clf, X_train, y_train, X_test, y_test, train=True)
print_score(svm_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Fitting 5 folds for each of 147 candidates, totalling 735 fits
Best params: {'C': 2, 'gamma': 0.5, 'kernel': 'rbf'}
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
               0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0       1.0        1.0           1.0
recall       1.0    1.0       1.0        1.0           1.0
f1-score     1.0    1.0       1.0        1.0           1.0
support    340.0  377.0       1.0      717.0         717.0
_______________________________________________
Confusion Matrix:
 [[340   0]
 [  0 377]]

Test Result:
================================================
Accuracy Score: 98.05%
```

Fig 10: Modèle après tunning

Conclusion

Les SVMs sont alors des algorithmes robustes pour répondre au problèmes de classification vu leur simplicité dans la programmation qui n'utilise que des bibliothèques du language informatique Python et donnent pourtant des résultats de classification precise .

Annexe

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import numpy as np
        import hvplot.pandas
        from scipy import stats

        %matplotlib inline
        sns.set_style("whitegrid")
        plt.style.use("fivethirtyeight")
```

```
In [6]: data = pd.read_csv("C:/Users/othma/desktop/heart.csv")
        data.head()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Fig 11: Importation de la base de données

Annexe

```
In [23]: from sklearn.model_selection import train_test_split

         X = dataset.drop('target', axis=1)
         y = dataset.target

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Fig 12: Division de la base de données en deux sous bases, une pour l’entrainement et l’autre pour le test de l’entrainement

Merci pour votre attention