Santé prévention

Prévention du syndrome de la mort subite du nourrisson à l'aide du traitement d'image

ZHANI Reda

Numéro d'inscription:22965

Session 2022

Introduction

SMSN

- Le **SMSM** désigne la mort inattendue d'un bébé durant le sommeil .

- Chaque année, environ **2000 nourrissons** sont victimes du SMSN aux états unis.

- le risque du SMSN diminue de 40% quand le bébé est positionné sur le dos lors du sommeil

Introduction

- Base de données contenant deux classes :

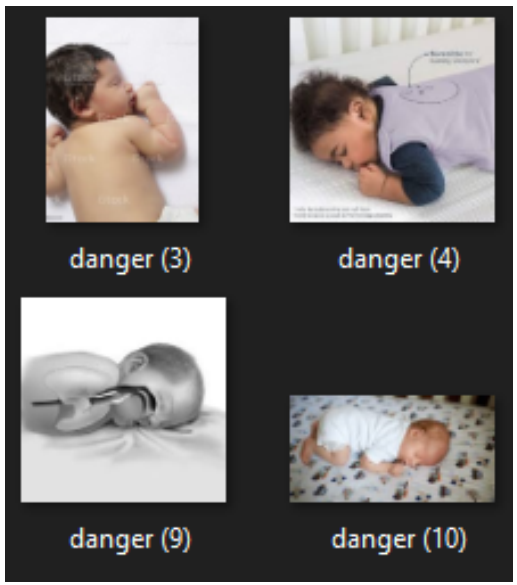


Figure – danger

Figure – sécurité

$\Longrightarrow$ **Problème de classification d'images.**

Introduction

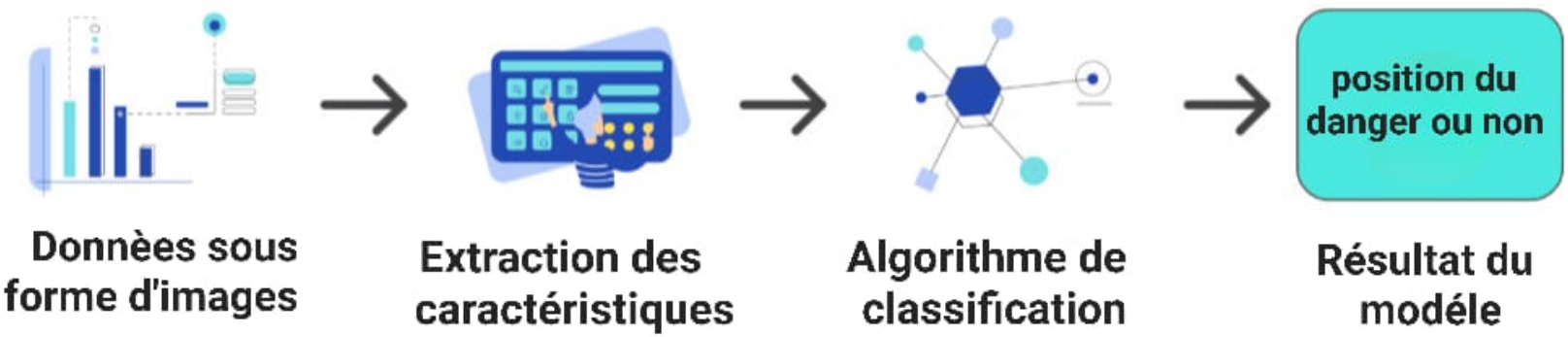
Machine learning

- **Le machine learning** est un champ d'étude de **l'intelligence artificielle** qui donne aux ordinateurs la capacité d'apprendre à partir de données.

- Un algorithme de machine learning améliore ses performances au fur et à mesure de son apprentissage et plus on le "nourrit" de données, plus il devient précis.

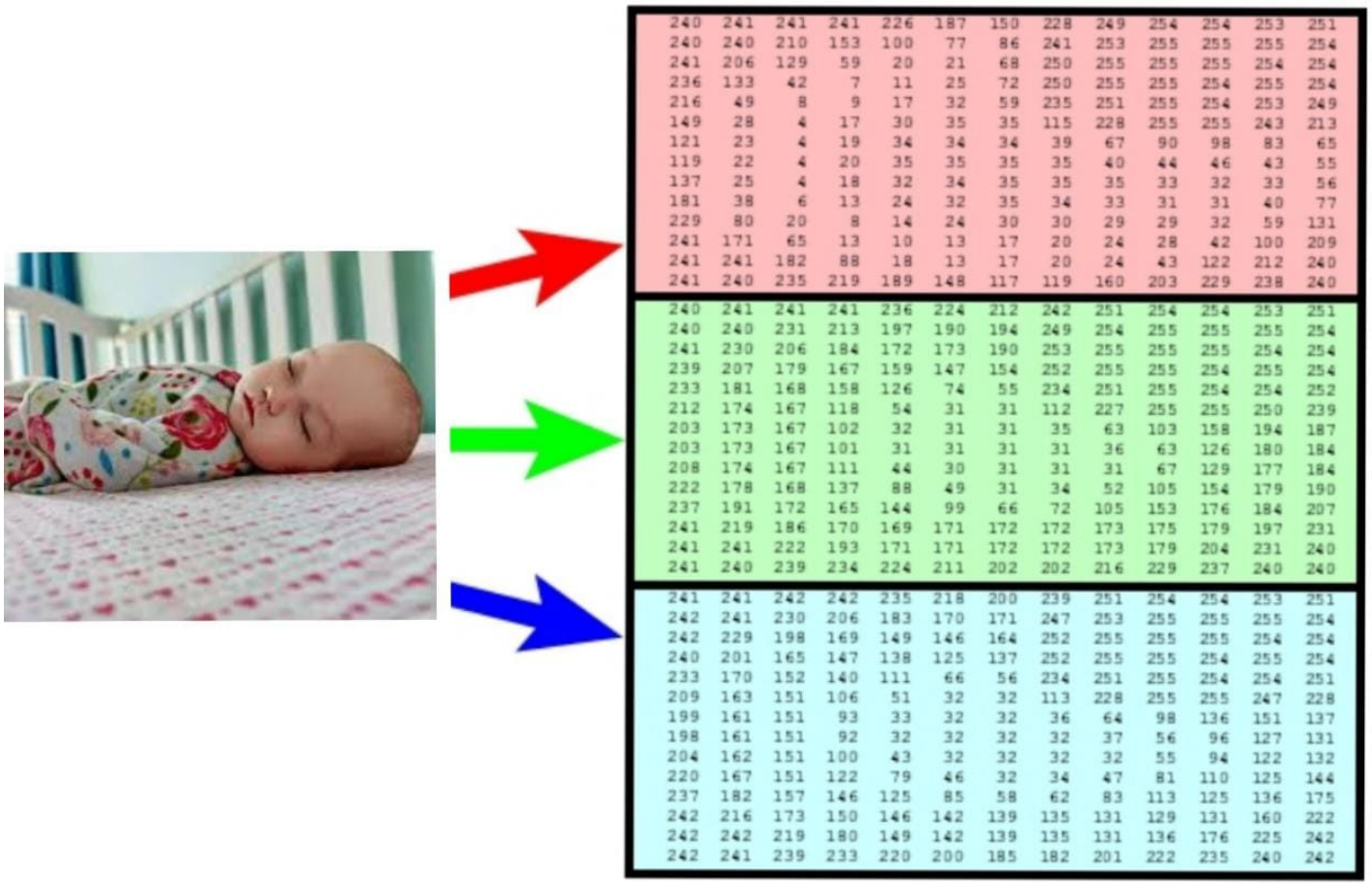
Introduction

Schéma du modéle de Machine Learning



La représentation des images

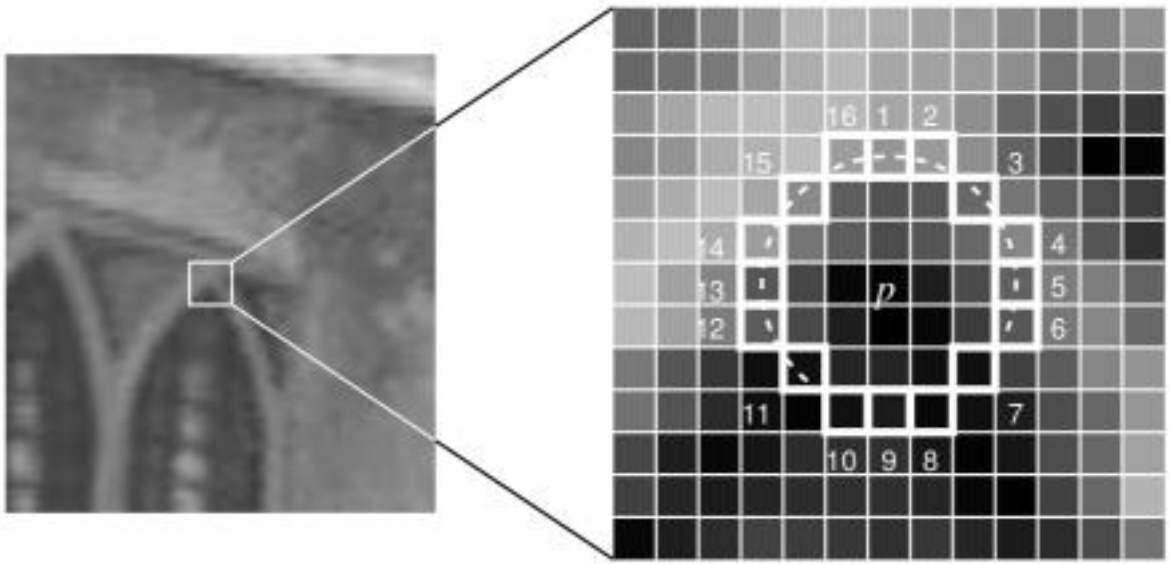
L'image numérique



Extraction des caractéristiques

ORB

- Les images peuvent se décrire localement à l'aide d'un descripteur pour effectuer l'indexation d'images.

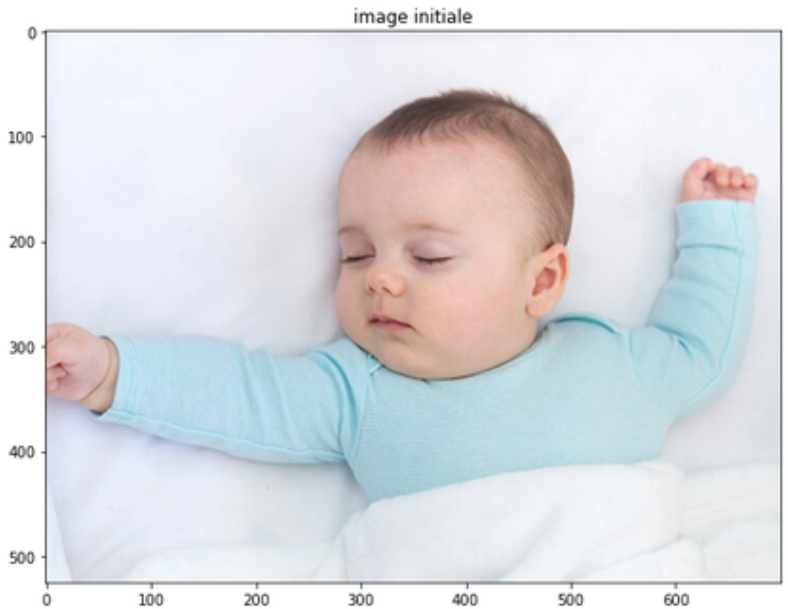


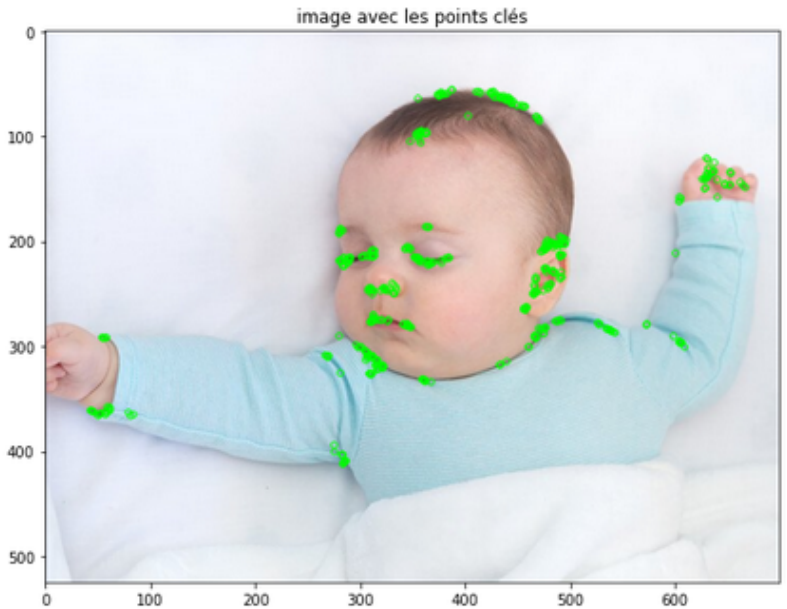
- ORB :compare la luminosité de p aux 16 pixels qui sont dans un petit cercle autour de p.

Extraction des caractéristiques

ORB :points clés

- Si plus de 8 pixels sont plus sombres ou plus clairs que p, il est sélectionné comme **point clé**.





Extraction des caractéristiques

ORB : création du descripteur

Pour un point clé p :

- On choisit N points $(p_1, p_2, ..., p_N)$ au voisinage de p.
- On construit la liste $L = [\tau(1), \tau(2), ..., \tau(N-1)]$

où :

$$\tau(i) = \begin{cases} 1 & si \quad I(p_i) \geq I(p_{i+1}) \\ 0 & sinon \end{cases}$$

Naïve bayes

- **la classification bayésienne naïve** : modèle simple et rapide qui se base sur le théorème de Bayes.

Théorème de Bayes

$$P(C_i/X) = \frac{P(C_i).P(X/C_i)}{P(X)}$$

Avec :

- X est un vecteur que l'on cherche sa classe
- C_i est l'hypothèse X appartient à la classe i

cet algorithme cherche alors $P(C_i/X)$: la probabilité de vérification de C_i après l'observation de X .

Naïve bayes

appellation

Pourquoi l'appellation "naïve" ?

Parce que pour utiliser cet algorithme nous supposons que les variables explicatives sont indépendantes.

Naïve bayes

types de Naïve bayes

- Les naïves bayes diffèrent dans la loi de probabilité choisie et parmi les plus connues on trouve : **Gaussien naive bayes** et **Bernoulli naive bayes** .

- on utilise gaussien naïve bayes qui suit la loi gaussienne centrée suivante :

Loi gaussienne centrée

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x}{\sigma})^2}$$

SVM

Principe

Principe du SVM

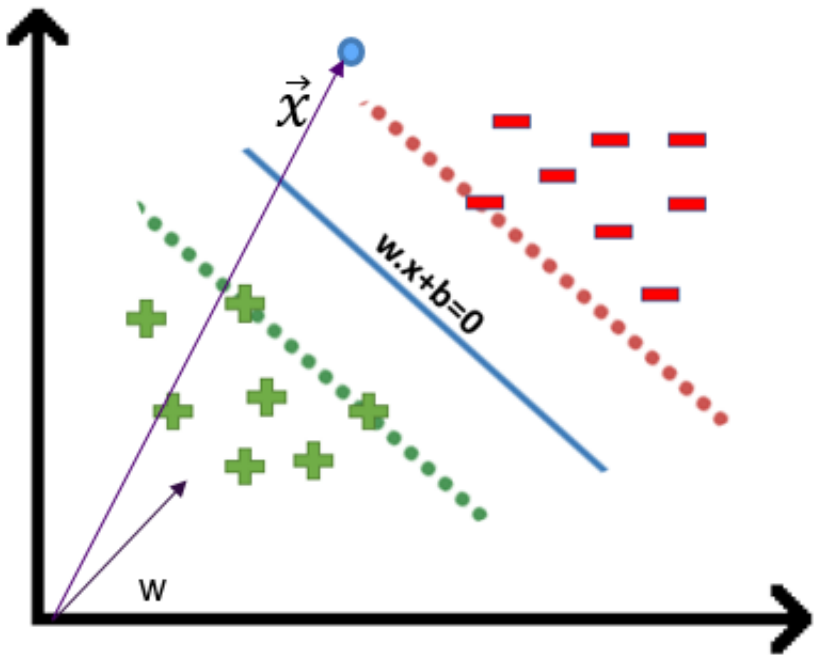
- Séparer les données en classes à l'aide d'une frontière, de telle façon que la distance entre les différents groupes de données et la frontière qui les sépare soit maximale.

- Cette distance est aussi appelée « marge » et les SVMs sont ainsi qualifiés de « séparateurs à vaste marge ».

SVM

Principe

- On suppose que le problème est linéairement séparable



Règle de la classification :

si $\vec{w}.\vec{x} + b \geq 0$ alors est un $-$, sinon c'est un $+$

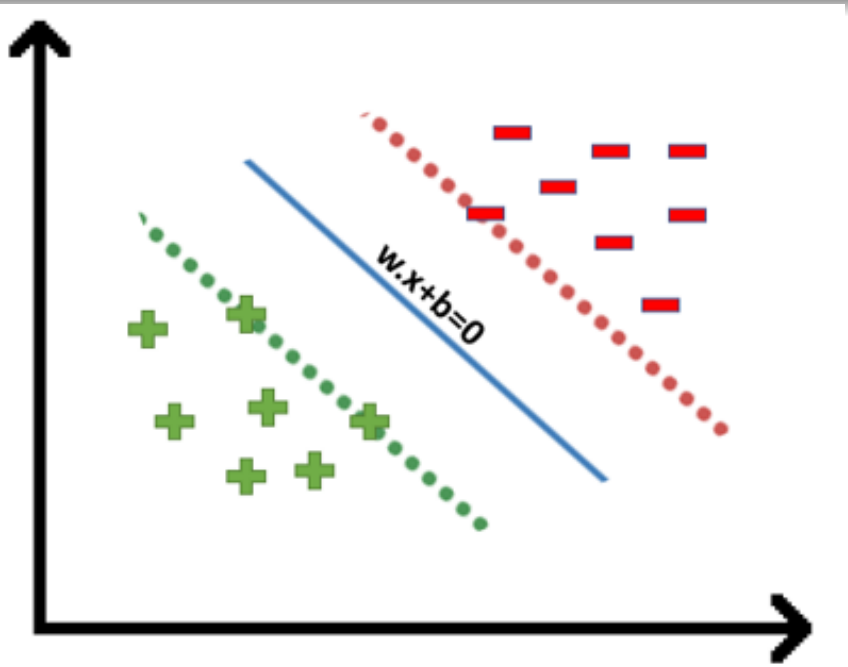
SVM

Principe

Comment déterminer w et b ?

SVM

Contraintes supplémentaires



$$\begin{cases} \vec{w}.\vec{x_-} + b \geq 1 \\ \vec{w}.\vec{x_+} + b \leq -1 \end{cases}$$

On pose $y_i = 1$ si $\vec{x_i} = \vec{x_-}$ et $y_i = -1$ sinon :

Nouvelle règle de la classification :

$$y_i.(\vec{w}.\vec{x_i} + b) - 1 \geq 0$$

SVM

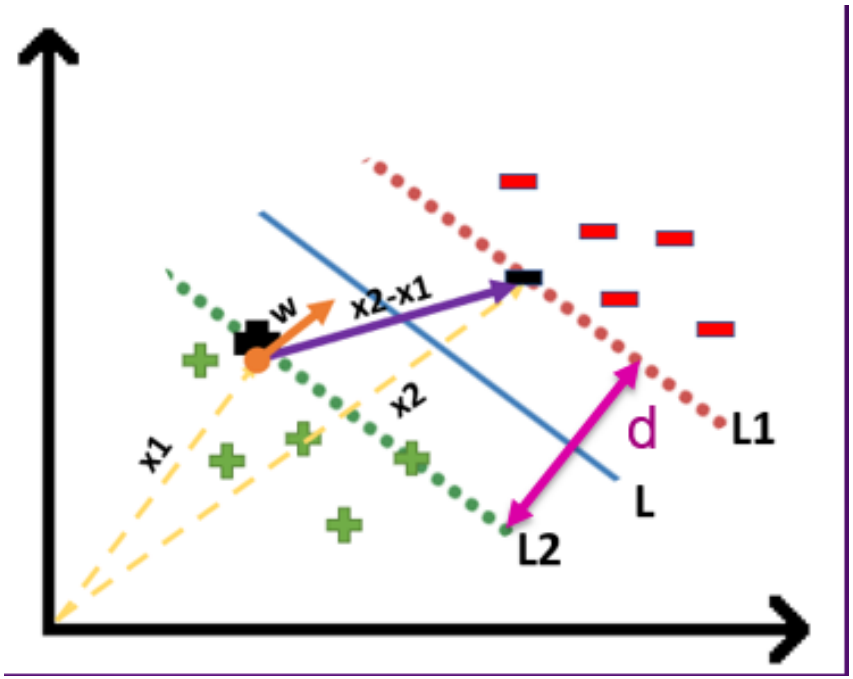
Contraintes supplémentaires

Contrainte supplémentaire :

- Sur les deux frontières : $y_i.(\vec{w}.\vec{x_i} + b) - 1 = 0 \quad (*)$

SVM

Contraintes supplémentaires



$$d = (\vec{x_2} - \vec{x_1}).\frac{\vec{w}}{||\vec{w}||}$$

SVM

Contraintes supplémentaires

- En utilisant la contrainte on trouve :

$$d = \frac{2}{||\vec{w}||}$$

- Pour avoir une bonne séparation il faut que d soit maximale càd on cherche w tq : $\frac{1}{2}||\vec{w}||^2$ soit minimale sous la contrainte :

$$y_i.(\vec{w}.\vec{x_i} + b) - 1 = 0 \quad (*)$$

SVM

Contraintes supplémentaires

- On utilise les multiplicateurs de Lagrange, en posant :

$$L = \frac{1}{2}||\vec{w}||^2 - \sum_i \alpha_i [y_i.(\vec{w}.\vec{x_i} + b) - 1]$$

On trouve :

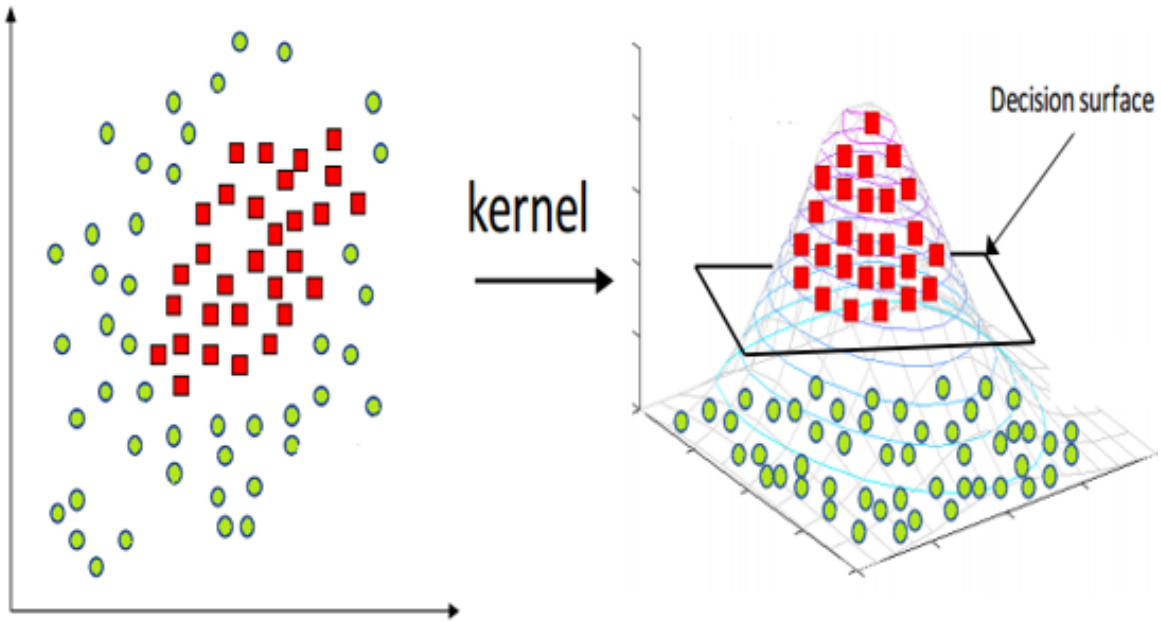
$$\begin{cases} \vec{w} = \sum_i \alpha_i y_i \vec{x_i} \\ \sum_i \alpha_i y_i = 0 \end{cases}$$

Résultat :

La minimisation ne dépend que du produit scalaire

SVM

Kernel-trick



- Si le problème n'est pas linéairement séparable il faut le transformer à un nouveau espace où il devient linéairement séparable à l'aide d'une bijection ϕ

SVM

Kernel-trick

Principe de Kernel-trick :

- Chercher et utiliser une fonction k tel que :

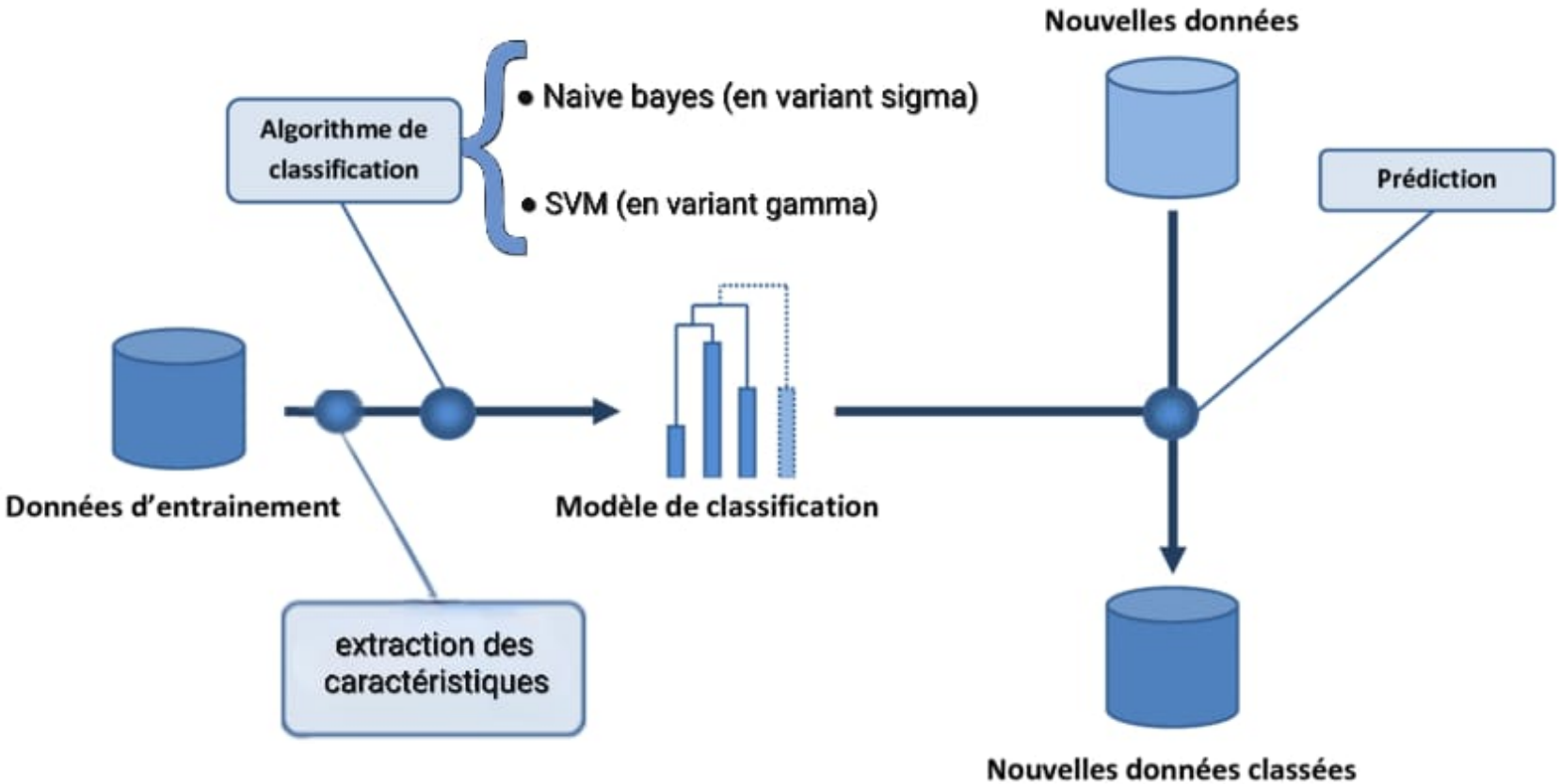
$$k(\vec{x}, \vec{y}) = \phi(\vec{x})\phi(\vec{y})$$

Exemple de K :

- Fonction de base radiale (rbf) :

$$k = e^{-\gamma \cdot ||\vec{x} - \vec{y}||^2}$$

Processus de l'algorithme



Résultats

Pourcentage des estimations correctes

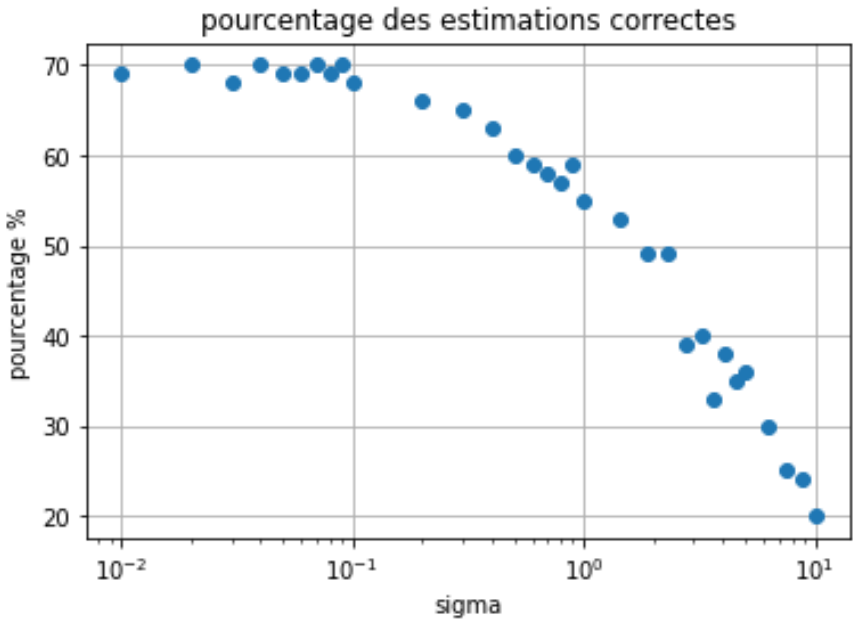


Figure – Résultat Naïve bayes

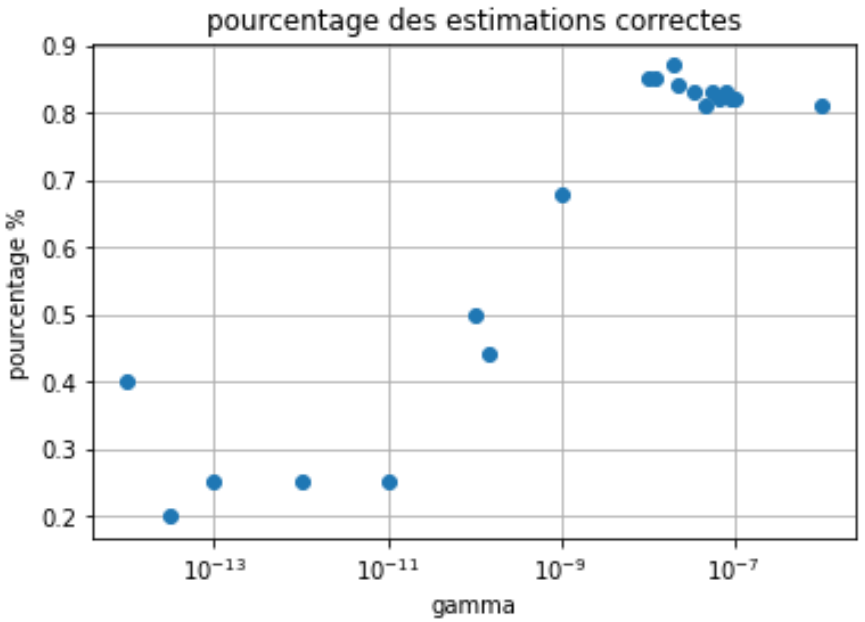


Figure – Résultat SVM

Conclusion

Naïve bayes

Points positifs

- Modéle simple
- Il a besoin d'une petite quantité de données d'entraînement.
- L'entrainement est rapide.

Point négatif

- S'il existe une grande corrélation entre les caractéristiques, il va donner une mauvaise performance.

Conclusion

SVM

Point positif

- SVM a une grande précision de prédiction.

Points négatifs

- Modéle complexe
- Il ne convient pas à des jeux de données plus volumineux, car le temps d'entraînement avec les SVM peut être long.

Conclusion

Choix et amélioration du l'algorithme

- Choisir SVM comme algorithme de classification.
- Augmenter sa précision en augmentant les données car plus qu'on le nourrit de données, plus il devient précis.

Pistes d'amélioration

Amélioration et futures actions

- Collection de plus de données.

- Evaluation de méthodes plus avancées (Les réseaux de neurones par exemple)

- Implémentation et intégration dans une carte électronique (type Arduino) avec camera.

Annexe : Extraction des caractéristiques

```
1   import cv2
2
3   import matplotlib.pyplot as plt
4   img = cv2.imread('donnees/entrainement\\pas_de_danger\\securite (252).jpg')
5   orb = cv2.ORB_create()
6
7   points_clés= orb.detect(img)
8   points_clés, descripteur = orb.compute(img, points_clés)
9
10  img=cv2.drawKeypoints(img,points_clés)
11  plt.title("image avec les points clés")
12  plt.imshow(img)
13
```

Annexe : collecte de données

```python
import os


def donnees(chemin):
    noms_classes = os.listdir(chemin)   #c'est une liste qui contient les deux classes
    chemins_images = []
    noms = []
    for nom in noms_classes :
        chemin_nom = os.path.join(chemin, nom)
        chemin_chaque_img = [os.path.join(chemin_nom, f) for f in os.listdir(chemin_nom)]
        chemins_images += chemin_chaque_img
        noms.extend([nom] * len(chemin_chaque_img))

    return chemins_images, noms


def donnees_entrainement():
    return donnees("donnees/entrainement")



def donnees_test():
    return donnees("donnees/test")


```

Annexe : extraction des caracteristiques

```python
import cv2
import numpy as np
from scipy.cluster.vq import kmeans, vq



def extraction_carac(images):
    n = len(images)
    descripteurs = []
    orb = cv2.ORB_create()

    for chemin_image in images:
        img = cv2.imread(chemin_image)
        caracteristiques = orb.detect(img)
        _, img_descripteurs = orb.compute(img, caracteristiques)
        descripteurs.append((chemin_image, img_descripteurs))

    concat_descripteurs = descripteurs[0][1]
    for chemin_image, descripteur in descripteurs[1:]:
        concat_descripteurs = np.vstack((concat_descripteurs, descripteur))
    concat_descripteurs = concat_descripteurs.astype(float)

    codebook, _ = kmeans(concat_descripteurs, 200, 1)

    img_carac = np.zeros((n, 200))
    for i in range(n):
        mots, distance = vq(descripteurs[i][1], codebook)
        for mot in mots:
            img_carac[i][mot] += 1

    return img_carac, codebook


```

Annexe : entrainement

```python
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC



def carac_img_entrainement():
    chemins_images, noms = donnees_entrainement()
    img_carac, codebook = extraction_carac(chemins_images)
    return img_carac, codebook, noms


def model_svm(gammaa):
    img_carac, codebook, noms = carac_img_entrainement()


    model = SVC(kernel='rbf',  gamma=gammaa)
    estimation = model.fit(img_carac, np.array(noms))
    return estimation,codebook




def model_nb(var):
    img_carac, codebook, noms = carac_img_entrainement()


    model = GaussianNB(var_smoothing=var)
    estimation = model.fit(img_carac, np.array(noms))
    return estimation,codebook

```

Annexe : evaluation des deux méthodes

```python
import cv2
import numpy as np


def validation_model(methode):
    if methode[0]=="svm":
        gammaa=methode[1]
        estimation,codebook=model_svm(gammaa)
    elif methode[0]=="nb":
        var=methode[1]
        estimation,codebook=model_nb(var)

    chemins_images, noms = donnees_test()
    img_carac, codebook = extraction_carac(chemins_images)

   n=len(noms)

    predictions = estimation.predict(img_carac)
    correct = 0
    for i in range(n):
        if predictions[i] == noms[i]:
            correct += 1

    pourcentage = correct / n
    return round(pourcentage, 2)
```

Annexe : pourcentage des estimations correctes pour NB

```
import numpy as np
import matplotlib.pyplot as plt


l0=np.linspace(10**(-2),10,40)
D0={}
for x in l0:
    D0[x]=validation_model(["nb",x])



X0,Y0=[],[]
for c in D0:
    X0+=[c]
    Y0+=[D0[c]*100]


plt.semilogx(X0,Y0,"o")
plt.ylabel('pourcentage %')
plt.xlabel("sigma")
plt.grid(True)
plt.title("pourcentage des estimations correctes")
plt.show()
print(max(sol0.values())*100,"%")

```

Annexe : pourcentage des estimations correctes pour SVM

```python
import numpy as np
import matplotlib.pyplot as plt


l=np.linspace(1e-13,1,30)
D={}
for x in l:
    D[x]=validation_model(["svm",x])



X,Y=[],[]
for c in D:
    X+=[c]
    Y+=[D[c]*100]

plt.semilogx(X,Y,"o")
plt.ylabel('pourcentage %')
plt.xlabel("gamma")
plt.grid(True)
plt.title("pourcentage des estimations correctes")
plt.show()
print(max(D.values())*100,"%")
```