

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

- 1 Introduction
 - SMSN
 - Machine learning
 - La représentation des images
 - Extraction des caractéristiques
- 2 La théorie de la classification
 - Naïve bayes
 - SVM
- 3 Résultats
- 4 Conclusion et pistes d'amélioration
- 5 Annexe

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

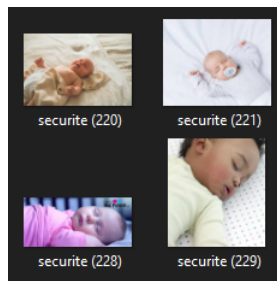


Figure – sécurité

⇒ **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 modèle de Machine Learning



La représentation des images

L'image numérique

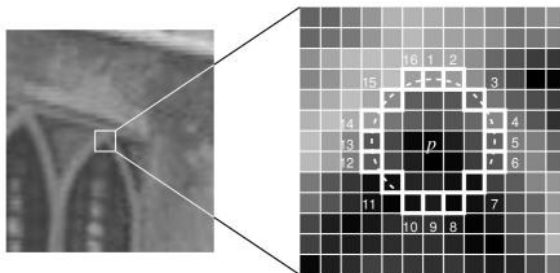


240	241	241	241	226	187	150	228	249	254	254	253	251
240	240	210	153	100	77	86	241	253	255	255	255	254
241	206	129	59	20	21	68	250	255	255	255	254	254
236	133	42	7	11	25	72	250	255	255	254	255	254
236	49	8	9	17	32	59	235	251	255	254	253	249
149	28	4	17	30	35	35	115	228	255	255	243	213
121	23	4	19	34	34	34	39	67	90	98	83	65
119	22	4	20	35	35	35	35	40	44	46	43	55
137	25	4	18	32	34	35	35	35	33	32	33	56
181	38	6	13	24	32	35	34	33	31	31	40	77
229	80	20	8	14	24	30	30	29	29	32	59	131
241	171	65	13	10	13	17	20	24	28	42	100	209
241	241	182	88	18	13	17	20	24	43	122	212	240
241	240	235	239	189	148	117	119	160	203	229	238	240
240	241	241	241	236	224	212	242	251	254	254	253	251
240	240	231	213	197	190	194	249	254	255	255	255	254
241	230	206	184	172	173	190	253	255	255	255	254	254
239	207	179	167	159	147	154	252	255	255	254	255	254
233	181	168	158	126	74	55	234	251	255	254	254	252
212	174	167	118	54	31	31	112	227	255	255	250	239
203	173	167	102	32	31	31	35	63	103	158	194	187
203	173	167	101	31	31	31	31	36	63	126	180	184
208	174	167	111	44	30	31	31	31	67	129	177	184
222	178	168	137	88	49	31	34	52	105	154	179	190
237	191	172	165	144	99	66	72	105	153	176	184	207
241	219	186	170	169	171	172	172	173	175	179	197	231
241	241	222	193	171	171	172	172	173	179	204	231	240
241	240	239	234	224	211	202	202	216	229	237	240	240
241	241	242	242	235	218	205	239	251	254	254	253	251
242	241	230	206	183	170	171	247	253	255	255	255	254
242	229	198	169	149	146	164	252	255	255	255	254	254
240	201	165	147	138	125	137	252	255	255	254	255	254
233	170	152	140	111	66	56	234	251	255	254	254	251
209	163	151	106	51	32	32	113	228	255	255	247	228
199	161	151	93	33	32	32	36	64	98	136	151	137
198	161	151	92	32	32	32	32	37	56	96	127	131
204	162	151	100	43	32	32	32	32	55	94	122	132
220	167	151	122	79	46	32	34	47	81	110	125	144
237	182	157	146	125	85	58	62	83	113	125	136	175
242	216	173	150	146	142	139	135	131	129	131	160	222
242	242	219	180	149	142	139	135	131	136	176	225	242
242	241	239	233	220	200	185	182	201	222	235	240	242

Extraction des caractéristiques

ORB

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

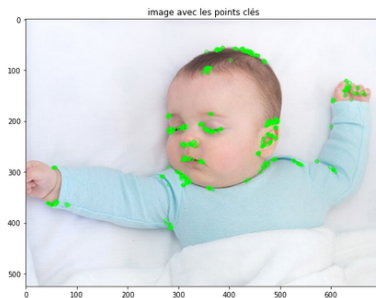
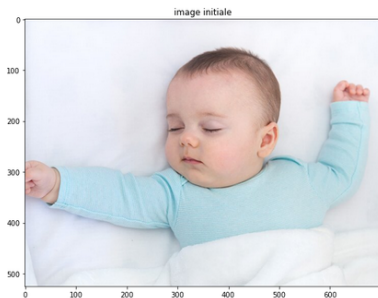


- ORB : compare la luminosité de p aux 16 pixels qui sont dans un petit cercle autour de p .

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

où :

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

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 naïve bayes** et **Bernoulli naïve bayes** .

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

Loi gaussienne centrée

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

SVM

Principe

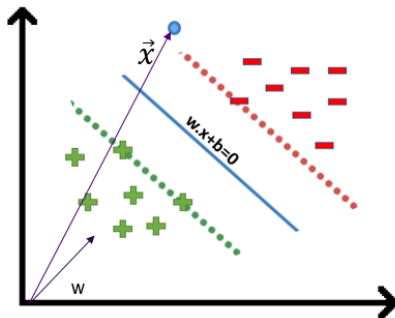
Principe du SVM

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

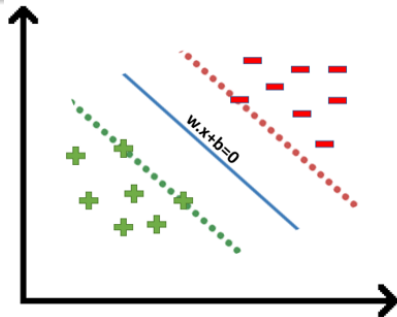
SVM

Principe

Comment déterminer w et b ?

SVM

Contraintes supplémentaires



$$\begin{cases} \vec{w} \cdot \vec{x}_- + b \geq 1 \\ \vec{w} \cdot \vec{x}_+ + b \leq -1 \end{cases}$$

On pose $y_i = 1$ si $\vec{x}_i = \vec{x}_-$ et $y_i = -1$ sinon :

Nouvelle règle de la classification :

$$y_i \cdot (\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0$$

SVM

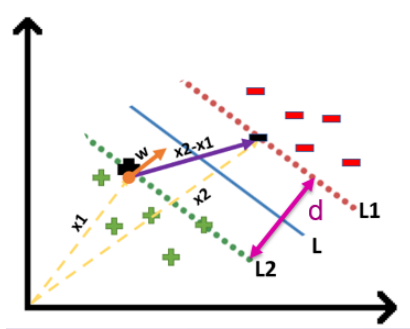
Contraintes supplémentaires

Contrainte supplémentaire :

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

SVM

Contraintes supplémentaires



$$d = (\vec{x}_2 - \vec{x}_1) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

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 c-à-d on cherche w tq : $\frac{1}{2}\|\vec{w}\|^2$ soit minimale sous la contrainte :

$$y_i \cdot (\vec{w} \cdot \vec{x}_i + b) - 1 = 0 \quad (*)$$

SVM

Contraintes supplémentaires

- On utilise les multiplicateurs de Lagrange, en posant :

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

On trouve :

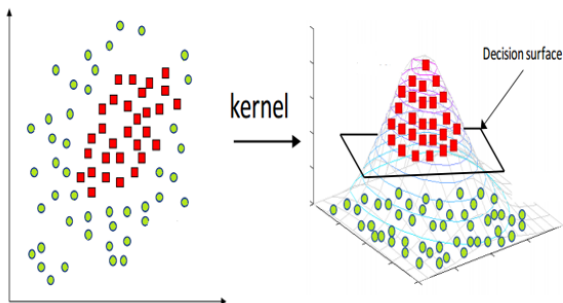
$$\begin{cases} \vec{w} = \sum_i \alpha_i y_i \vec{x}_i \\ \sum_i \alpha_i y_i = 0 \end{cases}$$

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 :

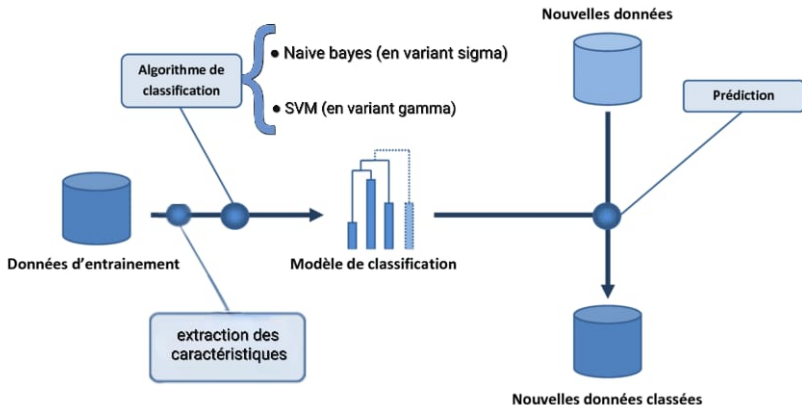
$$k(\vec{x}, \vec{y}) = \phi(\vec{x})\phi(\vec{y})$$

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

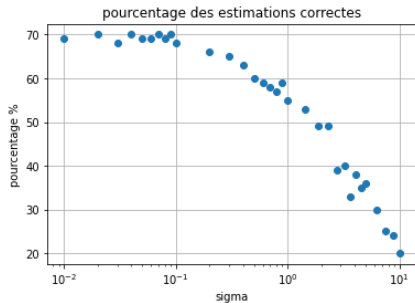


Figure – Résultat Naïve bayes

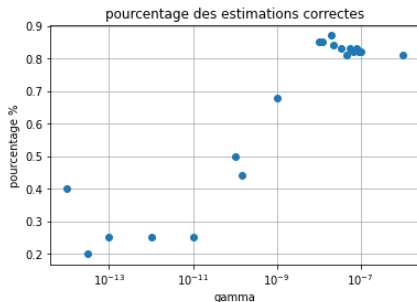


Figure – Résultat SVM

Conclusion

Naïve bayes

Points positifs

- Modèle simple
- Il a besoin d'une petite quantité de données d'entraînement.
- L'entraînement 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

- Modèle 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

```
1 import os
2
3
4 def donnees(chemin):
5     noms_classes = os.listdir(chemin) #c'est une liste qui contient les deux classes
6     chemins_images = []
7     noms = []
8     for nom in noms_classes :
9         chemin_nom = os.path.join(chemin, nom)
10        chemin_chaque_img = [os.path.join(chemin_nom, f) for f in os.listdir(chemin_nom)]
11        chemins_images += chemin_chaque_img
12        noms.extend([nom] * len(chemin_chaque_img))
13
14    return chemins_images, noms
15
16
17 def donnees_entrainement():
18     return donnees("donnees/entrainement")
19
20
21
22 def donnees_test():
23     return donnees("donnees/test")
24
25
```

Annexe : extraction des caracteristiques

```
1  import cv2
2  import numpy as np
3  from scipy.cluster.vq import kmeans, vq
4
5
6
7
8  def extraction_carac(images):
9      n = len(images)
10     descripteurs = []
11     orb = cv2.ORB_create()
12
13     for chemin_image in images:
14         img = cv2.imread(chemin_image)
15         caracteristiques = orb.detect(img)
16         _, img_descripteurs = orb.compute(img, caracteristiques)
17         descripteurs.append((chemin_image, img_descripteurs))
18
19     concat_descripteurs = descripteurs[0][1]
20     for chemin_image, descripteur in descripteurs[1:]:
21         concat_descripteurs = np.vstack((concat_descripteurs, descripteur))
22     concat_descripteurs = concat_descripteurs.astype(float)
23
24     codebook, _ = kmeans(concat_descripteurs, 200, 1)
25
26     img_carac = np.zeros((n, 200))
27     for i in range(n):
28         mots, distance = vq(descripteurs[i][1], codebook)
29         for mot in mots:
30             img_carac[i][mot] += 1
31
32     return img_carac, codebook
33
34
```


Annexe : entraînement

```
1  import numpy as np
2  from sklearn.naive_bayes import GaussianNB
3  from sklearn.svm import SVC
4
5
6
7  def carac_img_entrainement():
8      chemins_images, noms = donnees_entrainement()
9      img_carac, codebook = extraction_carac(chemins_images)
10     return img_carac, codebook, noms
11
12
13  def model_svm(gammaa):
14      img_carac, codebook, noms = carac_img_entrainement()
15
16
17      model = SVC(kernel='rbf', gamma=gammaa)
18      estimation = model.fit(img_carac, np.array(noms))
19      return estimation, codebook
20
21
22
23
24  def model_nb(var):
25      img_carac, codebook, noms = carac_img_entrainement()
26
27
28      model = GaussianNB(var_smoothing=var)
29      estimation = model.fit(img_carac, np.array(noms))
30      return estimation, codebook
31
```

Annexe : evaluation des deux méthodes

```
1 import cv2
2 import numpy as np
3
4
5 def validation_model(methode):
6     if methode[0]=="svm":
7         gammaa=methode[1]
8         estimation,codebook=model_svm(gammaa)
9     elif methode[0]=="nb":
10         var=methode[1]
11         estimation,codebook=model_nb(var)
12
13     chemins_images, noms = donnees_test()
14     img_carac, codebook = extraction_carac(chemins_images)
15
16     n=len(noms)
17
18     predictions = estimation.predict(img_carac)
19     correct = 0
20     for i in range(n):
21         if predictions[i] == noms[i]:
22             correct += 1
23
24     pourcentage = correct / n
25     return round(pourcentage, 2)
26
27
```

Annexe : pourcentage des estimations correctes pour NB

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4
5 l0=np.linspace(10**(-2),10,40)
6 D0={}
7 for x in l0:
8     D0[x]=validation_model(["nb",x])
9
10
11
12 X0,Y0=[],[]
13 for c in D0:
14     X0+= [c]
15     Y0+= [D0[c]*100]
16
17
18 plt.semilogx(X0,Y0,"o")
19 plt.ylabel('pourcentage %')
20 plt.xlabel("sigma")
21 plt.grid(True)
22 plt.title("pourcentage des estimations correctes")
23 plt.show()
24 print(max(sol0.values())*100,"%")
25
26
```

Annexe : pourcentage des estimations correctes pour SVM

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4
5 l=np.linspace(1e-13,1,30)
6 D={}
7 for x in l:
8     D[x]=validation_model(["svm",x])
9
10
11
12 X,Y=[],[]
13 for c in D:
14     X+=[c]
15     Y+=[D[c]*100]
16
17 plt.semilogx(X,Y,"o")
18 plt.ylabel('pourcentage %')
19 plt.xlabel("gamma")
20 plt.grid(True)
21 plt.title("pourcentage des estimations correctes")
22 plt.show()
23 print(max(D.values())*100,"%")
24
25
26
```