TP 4 Notebook

December 30, 2019

image

Vous allez trouver la suite ici: Notebook Online

1 Partie 1 : Extraction de caractéristiques simples

1.0.1 Question 0

[27]:
```
from imageio import imread
import matplotlib.pyplot as plt
import numpy as np
#ouverture dune image couleur
im = imread('images/chat1.jpg')
"""Affchage en utilisant une figure de matplotlib"""
plt.imshow(im)
#Suppresssion des axes
plt.axis('off')
#affchage
plt.show ()
```

1.0.2 Question 1

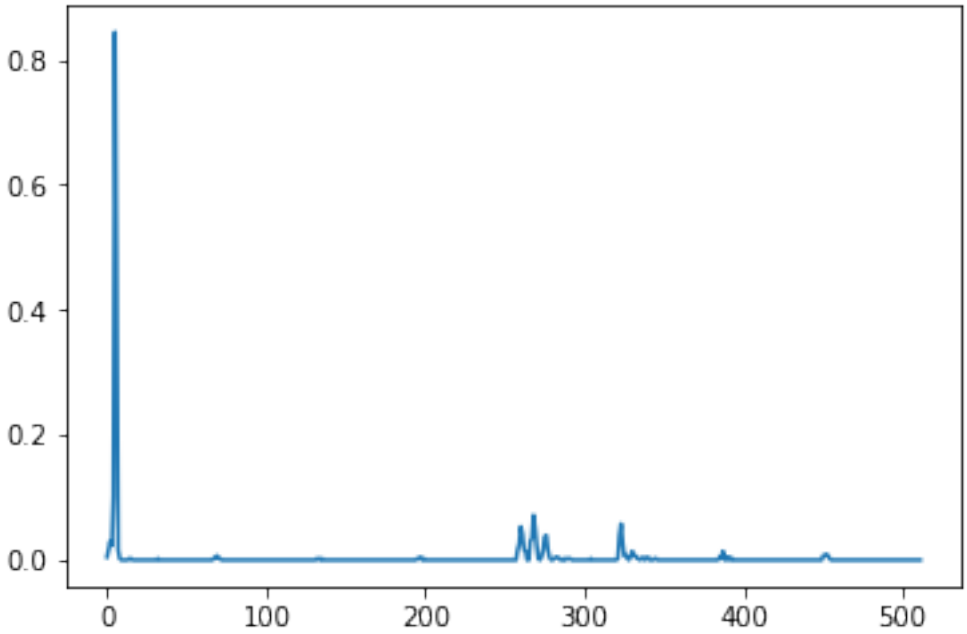
[2]:
```
from numpy import resize, shape
def applatir(image):
    image = resize(image, (32, 32))
    return image.flatten()
```

1.0.3 Question 2

[3]:
```
import cv2
import os
```

[4]:
```
def histogramme(image):
    resized = cv2.resize(image, (32, 32))
    bins=(8, 8, 8)
    hsv = cv2.cvtColor(resized, cv2.COLOR_BGR2HSV)
    hist = cv2.calcHist([hsv], [0,1,2], None, bins, [0, 180, 0, 256, 0, 265])
    cv2.normalize(hist, hist)
    return hist.flatten()
```

[5]:
```
HSVhist = histogramme(im)
plt.plot(HSVhist)
plt.show()
```



1.0.4 Question 3

[6]:
```
import os
path = 'images/'
images = ['images/{}'.format(i) for i in os.listdir(path)]
images
```

[6]:
```
['images/chien1.jpg',
 'images/chien.jpg',
 'images/chat2.jpg',
```

```
'images/chat1.jpg',
'images/chien2.jpg',
'images/chat.jpg',
'images/chat3.jpg']
```

[7]:
```
def images_en_vecteur_naif(images):
    x = []
    y = []
    for image in images:
        x.append(applatir(cv2.imread(image)))
        if 'chat' in image:
            y.append(1)
        elif 'chien' in image:
            y.append(0)
    return x, y
```

[8]:
```
# test de la fonction
X_naif, y_naif = images_en_vecteur_naif(images)
X_naif = np.array(X_naif)
y_naif = np.array(y_naif)
#y_naif = np.array(y_naif)
#y_naif = y_naif.reshape(y_naif.size, 1)
#transformation de la liste obtenu en matrice
```

1.0.5 Question 4

[9]:
```
def images_en_vecteur(images):
    x = []
    y = []
    for image in images:
        x.append(histogramme(cv2.imread(image)))
        if 'chat' in image:
            y.append(1)
        elif 'chien' in image:
            y.append(0)
    return x, y
```

[10]:
```
#test de la fonction
X_hist, y_hist = images_en_vecteur(images)
X_hist = np.array(X_hist)
y_hist = np.array(y_hist)
```

1.0.6 Question 5

[11]:
```
def images_en_vecteur_complet(images):
    x = []
    y = []
```

```
    for image in images:
        vect1 = applatir(cv2.imread(image))
        vect2 = histogramme(cv2.imread(image))
        x.append(np.concatenate((vect1, vect2), axis=0))
        if 'chat' in image:
            y.append(1)
        elif 'chien' in image:
            y.append(0)
    return x, y
```

[12]:
```
#test de la fonction
X_complet, y_complet = images_en_vecteur_complet(images)
X_complet = np.array(X_complet)
y_complet = np.array(y_complet)
```

2 Partie 2 : Sélection d'attributs / Réduction de dimension

2.0.1 Question 0

Déja sur les parties de teste

[13]:
```
print(X_naif)
print("Its shape is :",np.shape(X_naif))
```

```
[[140 149 162 ... 236 250 226]
 [ 51 113  89 ... 105  79  50]
 [127 184 229 ... 170 227  83]
 ...
 [206 207 235 ... 237 251 228]
 [246 248 248 ... 236 232 237]
 [ 41  39  39 ...  53  57  54]]
Its shape is : (7, 1024)
```

[14]:
```
X_naif[0]
```

[14]:
```
array([140, 149, 162, ..., 236, 250, 226], dtype=uint8)
```

[15]:
```
print(y_naif)
print("Its shape is :",np.shape(y_naif))
```

```
[0 0 1 1 0 1 1]
Its shape is : (7,)
```

[16]:
```
print(X_hist)
print("Its shape is :",np.shape(X_hist))
```

```
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
```

```
 [0.00544638 0.         0.0490174  ... 0.         0.         0.        ]
 ...
 [0.08579136 0.03431654 0.01143885 ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.01274118 0.08918826 0.12104121 ... 0.         0.         0.        ]]
Its shape is : (7, 512)
```

[17]:
```
print(y_hist)
print("Its shape is :",np.shape(y_hist))
```

```
[0 0 1 1 0 1 1]
Its shape is : (7,)
```

[18]:
```
print(X_complet)
print("Its shape is :",np.shape(X_complet))
```

```
[[140. 149. 162. ...   0.   0.   0.]
 [ 51. 113.  89. ...   0.   0.   0.]
 [127. 184. 229. ...   0.   0.   0.]
 ...
 [206. 207. 235. ...   0.   0.   0.]
 [246. 248. 248. ...   0.   0.   0.]
 [ 41.  39.  39. ...   0.   0.   0.]]
Its shape is : (7, 1536)
```

[19]:
```
print(y_complet)
print("Its shape is :",np.shape(y_complet))
```

```
[0 0 1 1 0 1 1]
Its shape is : (7,)
```

2.0.2 Question 1

[ ]:
```

```

[20]:
```
from sklearn.decomposition import PCA
```

[21]:
```
# X_naif
# PCA à 20%
pca20 = PCA(n_components=int(X_naif.shape[0]*0.8))
X_naif_ACP20 = pca20.fit_transform(X_naif)
# PCA à 40%
pca40 = PCA(n_components=int(X_naif.shape[0]*0.6))
X_naif_ACP40 = pca40.fit_transform(X_naif)
# PCA à 60%
pca60 = PCA(n_components=int(X_naif.shape[0]*0.4))
X_naif_ACP60 = pca60.fit_transform(X_naif)
```

[22]:
```
# X_hist
# PCA à 20%
pca20_hist = PCA(n_components=int(X_hist.shape[0]*0.8))
X_hist_ACP20 = pca20_hist.fit_transform(X_hist)
# PCA à 40%
pca40_hist = PCA(n_components=int(X_hist.shape[0]*0.6))
X_hist_ACP40 = pca40_hist.fit_transform(X_hist)
# PCA à 60%
pca60_hist = PCA(n_components=int(X_hist.shape[0]*0.4))
X_hist_ACP60 = pca60_hist.fit_transform(X_hist)
```

[23]:
```
# X_complet
# PCA à 20%
pca20_hist = PCA(n_components=int(X_complet.shape[0]*0.8))
X_complet_ACP20 = pca20_hist.fit_transform(X_complet)
# PCA à 40%
pca40_hist = PCA(n_components=int(X_complet.shape[0]*0.6))
X_complet_ACP40 = pca40_hist.fit_transform(X_complet)
# PCA à 60%
pca60_hist = PCA(n_components=int(X_complet.shape[0]*0.4))
X_complet_ACP60 = pca60_hist.fit_transform(X_complet)
```

3 Partie 3 : Classification avec des machines à vecteurs de support

[24]:
```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import cross_validate
from sklearn import linear_model
```

[25]:
```
lasso = linear_model.Lasso()
cv_results = cross_validate(lasso, X_complet, y_complet, cv=3)
cv_results
```

```
/home/nbuser/anaconda3_420/lib/python3.5/site-
packages/sklearn/linear_model/coordinate_descent.py:491: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Fitting data with very small alpha may cause precision problems.
  ConvergenceWarning)
/home/nbuser/anaconda3_420/lib/python3.5/site-
packages/sklearn/linear_model/coordinate_descent.py:491: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Fitting data with very small alpha may cause precision problems.
  ConvergenceWarning)
/home/nbuser/anaconda3_420/lib/python3.5/site-
packages/sklearn/utils/deprecation.py:122: FutureWarning: You are accessing a
training score ('train_score'), which will not be available by default any more
```

```
in 0.21. If you need training scores, please set return_train_score=True
  warnings.warn(*warn_args, **warn_kwargs)
```

[25]:
```
{'fit_time': array([0.32001901, 0.04164314, 0.06860423]),
 'score_time': array([0.00971746, 0.00038648, 0.0006032 ]),
 'test_score': array([-2.33516706, -1.39038317,  0.        ]),
 'train_score': array([0.98763648, 0.9946662 , 0.98252724])}
```

[26]:
```
linear_svc = svm.SVC(kernel='linear')
print(linear_svc.fit(X_complet, y_complet))
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```