

Signaux, Sons et Images pour l'Informaticien: Classification d'images

Diane Lingrand

Polytech SI3

2017 - 2018

- Différencier des classes d'images
- Méthodes :
 - Descripteurs des images
 - par ex : SIFT
 - classification par k-mean
 - en 2 étapes

Un exemple

Différencier les sapins et les pères Noël :

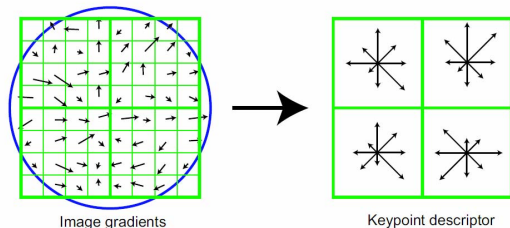


sapin0 sapin1 sapin2 sapin3 sapin4 sapin5 sapin6 sapin7 sapin8 sapin9



père 0 père 1 père 2 père 3 père 4 père 5 père 6 père 7 père 8 père 9

Descripteur SIFT (Rappel)



- vecteur de 128 entiers
- 4 étapes :
 - détection des points d'intérêts
 - orientation des gradients dans le voisinage (16x16 pixel en 4x4 blocs)
 - histogramme des orientations (quantifié sur 8 valeurs) par blocs de 4x4 pixels
 - $8 \times 4 \times 4 = 128$
 - normalisation

- Code : depuis la page de démonstration de l'auteur, David Lowe :
<http://www.cs.ubc.ca/~lowe/keypoints/> :
<http://www.cs.ubc.ca/~lowe/keypoints/siftDemoV4.zip>
- Utilisation :
 - les images en entrée sont au format PGM
 - entrée : stdin ; sortie : stdout
 - `src/siftDemoV4/sift < image.pgm > image.sift`



Pour le TP : Descripteur SIFT en python / OpenCV

- OpenCV (Open Source Computer Vision Library) : opencv.org
- Installation
 - pip3 install opencv-contrib-python

- Utilisation

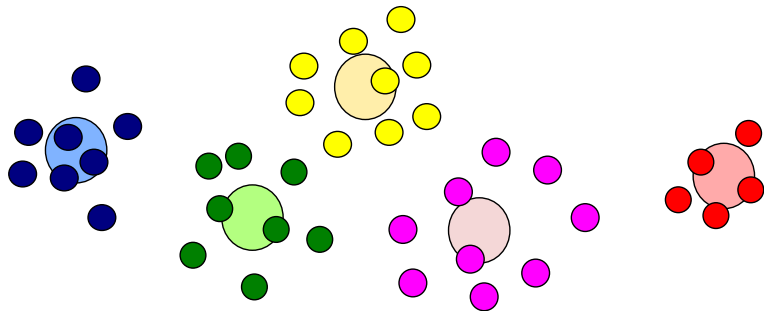
```
import cv2;  
image = cv2.imread('pereNoel.jpg')  
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)  
sift = cv2.xfeatures2d.SIFT_create()  
keypoints, descriptors = sift.detectAndCompute(gray,None)  
print('nb. of keypoints: ',len(keypoints))  
image = cv2.drawKeypoints(gray,keypoints,image,\  
    flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)  
cv2.imwrite('sift_keypoints.jpg',image)
```



Pour le TP : classification par K-means, le principe (rappel)

Partitionnement d'un ensemble de données x_i en K ensembles E_1, E_2, \dots, E_k en minimisant la somme des distances des données aux barycentres des ensembles :

$$\min \sum_{i=1}^k \sum_{x \in E_i} \|x - b_i\|^2 \text{ avec } b_i \text{ barycentre de } E_i$$



On reprend le code python utilisé précédemment :

```
from sklearn.cluster import KMeans  
monKmean = KMeans(n_clusters=nombre,  
random_state=0).fit(mesDonness)
```


- réduire la représentation des données
- par exemple, perenoel10.jpg : image 275x183, 188 SIFT = 24064 paramètres
- un premier k-mean : on classe tous les descripteurs de toutes les images en k_1 ensembles
- pour chaque image :
 - on regarde dans quelle classe sont les différents descripteurs
 - on forme ainsi un nouveau descripteur appelé *bow*
- on cherche ensuite à classer les données représentées par *bow*

Un exemple : $k_1 = 30$; $k_2 = 2$



class 0 class 0 class 0 class 0 class 0 class 0 class 0 class 0 class 0 class 1



class 1 class 1 class 1 class 0 class 1 class 1 class 1 class 1 class 1 class 1

Vrais faux positifs négatifs

vrais positifs (VP) : données positives calculées comme positives

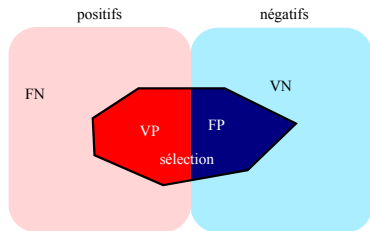
vrais négatifs (VN) : données négatives calculées comme négatives

faux positifs (FP) : données négatives calculées comme positives

faux négatifs (FN) : données positives calculées comme négatives

Matrice de confusion :

		classes estimées	
		sapin	père Noël
classes réelles	sapin	9	1
	père Noël	1	9



Vrais faux positifs négatifs

vrais positifs (VP) : données positives calculées comme positives

vrais négatifs (VN) : données négatives calculées comme négatives

faux positifs (FP) : données négatives calculées comme positives

faux négatifs (FN) : données positives calculées comme négatives

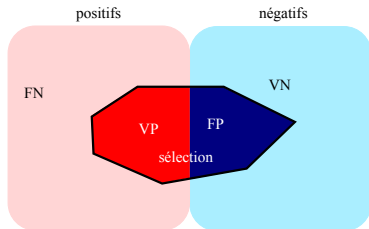
sensibilité (taux de VP) : $\frac{VP}{VP+FN}$

spécificité (taux de VN) : $\frac{VN}{VN+FP}$

précision : $\frac{VP}{VP+FP}$

rappel : = sensibilité

F-mesure : $2 \frac{\text{précision} \cdot \text{rappel}}{\text{précision} + \text{rappel}}$



- Installation de SIFT
- Installations alternatives en Java : plugin sift de ImageJ, weka ou orange pour le k-mean (ou toute autre version)
- Choisir un problème :
 - au moins 2 classes d'images
- Trouver les images et les stocker dans un même répertoire avec des noms explicites.
- Adapter le code pour les sons `recocons2018.py` au problème des images. Pour cela, il faudra mettre en forme les descripteurs SIFT pour le kmeans.
- Tester la méthode de classification par k-means via des *bow*. Trouver des valeurs de k correspondant à votre problème.
- Utiliser d'autres images et tester votre classification (matrice de confusion)

Ce sont celles qui n'ont pas été utilisées auparavant. Pour une nouvelle image, il faudra :

- la convertir en pgm
- extraire les différents descripteurs SIFT
- pour chaque descripteur
 - calculer sa distance avec chaque barycentre issu de la première classification (avec k_1)
 - lui associer la classe correspondant à la distance la plus petite
- construire le *bow*
- calculer la distance entre ce *bow* et chaque barycentre issu de la seconde classification (avec k_2)
- lui associer la classe correspondant à la distance la plus petite

- k-mean : initialisation
- construction du bow par k-mean : plus proche voisin
- classification
- image en niveaux de gris : perte de l'information couleur
- ajouter des informations de couleur / texture

Une approche classique

