SY32 – Analyse et synthèse d'images
TD 03 : Classification de visages

Pour manipuler des images, on utilisera la bibliothèque `scikit-image`. Les images sont représentées par des tableaux multidimensionnels comme dans `numpy`. Le tableau ci-dessous donne quelques commandes de base pour manipuler les images. On utilisera la commande `from skimage import` *module*.

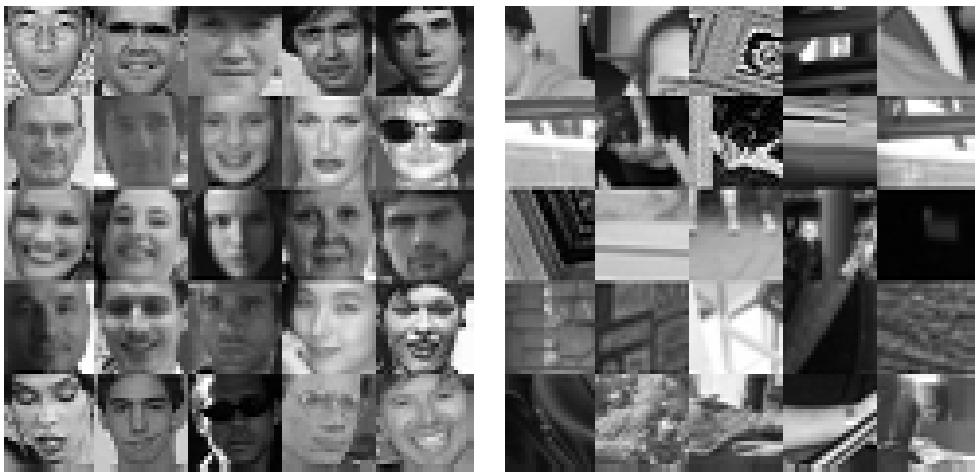
modules	fonctions	description
io	imread	charge une image depuis un fichier
io	imshow	affiche une image
util	img_as_float	convertit les valeurs des pixels en flottants à valeur dans $[0, 1]$.
feature	hog	calcule les caractéristiques HOG

Classification de visages

L'archive `imageface.zip` contient 4 000 images de visages et 17 256 images de fond réparties en un ensemble d'apprentissage (dossier `train`) et un ensemble de test (dossier `test`) de la manière suivante :

	# positifs	# négatifs
apprentissage	3 000	12 000
test	1 000	5 256

La Figure 1 montre un sous-échantillon de ces images. Toutes les images sont de taille 24×24.



(a) Visages (b) Non visages

FIGURE 1 – Exemples d'images.

1. La commande ci-dessous permet de lire l'image `imageface/train/pos/00001.png`, de convertir les valeurs des pixels à valeurs entières $\{0, 1, \ldots, 255\}$ en valeurs flottantes $[0, 1]$ et d'afficher l'image.

```
from skimage import io, util
i = 1
I = io.imread('imageface/train/pos/%05d.png'%i)
I = util.img_as_float(I)
io.imshow(I)
```

2. On souhaite utiliser directement les pixels comme représentation des images. La commande `I.flatten()` permet d'aplatir l'image qui est un tableau de taille 24×24 en un vecteur de longueur $576 = 24 \times 24$. Charger toutes les images de l'ensemble d'apprentissage sous la forme d'un tableau `X_train` comportant $15000 = 3000 + 12000$ lignes, une pour chaque image et 576 colonnes.

3. Construire le vecteur `y_train` associé aux images, on utilisera la valeur $+1$ pour les visages et -1 pour les non visages.

4. Utiliser la librairie Scikit-learn pour apprendre un classifieur Adaboost utilisant 10 itérations.

```
from sklearn.ensemble import AdaBoostClassifier
clf = AdaBoostClassifier(n_estimators=10)
clf.fit(X_train, y_train)
```

5. Calculer le taux d'erreur du classifieur sur les images de test.

6. Reprendre l'apprentissage avec 50 itérations.

7. Essayer d'autres classifieurs de la librairies :

- k-plus proches voisins : `https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification`
- Arbre de décision : `https://scikit-learn.org/stable/modules/tree.html#classification`
- Forêt aléatoire : `https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees`
- SVM : `https://scikit-learn.org/stable/modules/svm.html#classification`

8. Reprendre l'exercice en utilisant les caractéristiques HOG pour représenter les images.

```
from skimage.feature import hog
I = io.imread('imageface/train/pos/00001.png')
X = hog(I)
```