Introduction aux méthodes de classification d'images

Travaux pratiques

TS341

Michaël Clément
michael.clement@enseirb-matmeca.fr

1 Mise en place de l'environnement de développement

Sur les machines de l'ENSEIRB, ajouter le dossier `~/.local/bin` à votre variable d'environnement `$PATH` (ajouter éventuellement cette ligne à votre fichier `.bashrc`) :

```
export PATH=$PATH:~/.local/bin
```

Ensuite, installer manuellement `pip` dans votre dossier personnel en exécutant les commandes suivantes dans un terminal (bien taper `python3` et ne pas oublier l'option `--user`) :

```
wget https://bootstrap.pypa.io/get-pip.py
python3 get-pip.py --user
```

Nous allons maintenant créer un environnement virtuel local pour y installer les librairies Python. Commencer par installer `virtualenv` localement avec :

```
pip install --user virtualenv
```

Ensuite, créer l'environnement virtuel `tp-image` :

```
.local/bin/virtualenv tp-image
```

Activer l'environnement virtuel :

```
source tp-image/bin/activate
```

Maintenant que l'environnement virtuel est activé (si vous êtes sur votre machine personnelle, toute la partie avec `virtualenv` n'est pas obligatoire), il est possible d'installer les librairies Python nécessaires (cela peut prendre quelques minutes) :

```
pip install numpy matplotlib scikit-image scikit-learn jupyter
```

Vous pouvez désormais programmer en Python 3 et utiliser les librairies installées :

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2 * np.pi, num=100)
y = np.sin(4 * x) * np.exp(-x / 4)
plt.plot(x, y)
plt.show()
```

Vous pouvez également utiliser Jupyter Notebook qui est un environnement de développement Python en interactif au sein de votre navigateur web. Le serveur peut être lancé avec la commande `jupyter notebook`.

Si vous êtes vraiment débutants en Python, vous pouvez vous référer au site suivant pour en apprendre plus au sujet du langage et de ses librairies scientifiques :

https://www.scipy-lectures.org

2 Manipulation d'un jeu de données

La librairie scikit-learn propose un certain nombre de fonctions permettant de charger différents jeux de données standards en *machine learning*. Pour commencer, nous allons travailler avec le jeu de données *iris*. Il est composé de 150 échantillons répartis en 3 classes correspondant à différentes variétés de fleurs. Chaque échantillon est décrit par 4 caractéristiques : longueur et largeur des pétales et des sépales.

Pour charger le jeu de données, vous pouvez utiliser le code suivant :

```
import sklearn.datasets
iris = sklearn.datasets.load_iris()
X, y = iris.data, iris.target
print(X.shape, y.shape)
```

Les variables `X` et `y` contiennent respectivement les vecteurs de caractéristiques et les labels des classes pour l'ensemble du jeu de données.

Comme vu en cours, il est important de diviser ce jeu de données en un ensemble d'apprentissage (*train set*) et un ensemble de test (*test set*). La librairie scikit-learn propose plusieurs méthodes utilitaires pour réaliser automatiquement ce type de traitement.

Par exemple :

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

→ Familiarisez-vous avec les méthodes de validation croisée disponible dans scikit-learn à partir de la documentation en ligne :

https://scikit-learn.org/stable/modules/cross_validation.html

→ Expliquez le principe des validations croisées *Leave-One-Out*, *K-Fold* et *Stratified-K-Fold*.

3 Méthode des k plus proches voisins

Dans scikit-learn, tous les classifieurs suivent généralement le même principe, avec un appel à la méthode `fit` pour entraîner le modèle, et appel à la méthode `predict` pour générer des prédictions à partir de ce modèle :

```
clf = Classifier([hyperparameters])
clf.fit(X_train, y_train)   # pour entrainer le modele
preds = clf.predict(X_test) # pour obtenir des predictions
```

La classification par la méthodes des k plus proches voisins est disponible avec la classe :

`sklearn.neighbors.KNeighborsClassifier`

→ Familiarisez-vous avec la documentation de cette classe.

→ Écrire le code permettant d'effectuer une classification avec une séparation apprentissage/test de 50% et 3 voisins. Quel taux de reconnaissance obtenez-vous ? Quels sont les scores de précision et de rappel pour chacune des trois classes ? Voir les méthodes du module `sklearn.metrics` pour calculer ces évaluations.

→ Faire une figure qui montre l'évolution du taux de reconnaissance pour un nombre de voisins allant de 1 à 9. Appliquer la classification sur tout le jeu de données à partir d'une validation croisée *Stratified-K-Fold* avec $K = 3$ (utiliser pour cela la méthode `cross_val_predict`). Afficher les courbes pour la méthode par vote majoritaire et par pondération de la distance.

4 Classifieur bayésien naïf

Pour ce classifieur, nous allons utiliser le jeu de données *wine* :

```
wine = sklearn.datasets.load_wine()
```

Ce jeu de données est similaire au précédent, avec environ 178 échantillons correspondant à 3 types de vins, mais ici chaque échantillon est décrit par 14 caractéristiques (quantités des différents composants chimiques). Nous travaillons donc ici dans un espace de plus grande dimension.

La classifieur bayésien naïf faisant l'hypothèse que les classes suivent une loi normale peut être utilisé avec la classe suivante :

`sklearn.naive_bayes.GaussianNB`

Il fait l'hypothèse que les caractéristiques sont indépendantes entre elles (matrice de covariance diagonale), et ramène le problème à la combinaison de plusieurs lois normales unidimensionnelles.

→ Écrire le code permettant de classifier le jeu de données avec une validation croisée de type *Leave-One-Out*. Quel est le taux de reconnaissance ?

→ Comparer avec la méthode des k plus proches voisins.

5 Machines à vecteurs de support

Dans scikit-learn, il existe plusieurs implémentations du classifieur SVM. Nous allons utiliser la classe suivante, qui permet de changer facilement le type de noyau utilisé :

`sklearn.svm.SVC`

Pour ce classifieur, nous allons utiliser un jeu de données plus conséquent, correspondant à des images de visages (400 images de taille 64x64 provenant de 40 personnes différentes). Le jeu de données peut être chargé avec :

```
faces = sklearn.dataset.fetch_olivetti_faces()
```

Vous pouvez également visualiser des images du jeu de données, par exemple avec :

```
from skimage.io import imshow
im = faces.data[0].reshape(64, 64)
imshow(im)
```

→ Comme pour les sections précédentes, mettre en place un protocole de validation croisée permettant de classifier les images de ce jeu de données à partir de *SVM*s.

→ Expérimenter avec différents hyperparamètres (valeur de C, noyau linéaire ou gaussien). Comparer les résultats avec les autres classifieurs, notamment lorsque l'on ajoute du bruit sur les images (par exemple du bruit gaussien)

→ Bien que nous soyons dans un espace de grande dimension (4096 valeurs de pixels), expliquer pourquoi ce jeu de données peut être considéré comme facile.

6 Bags of features

Dans cette partie, nous proposons de mettre en place de méthode de classification par *bags of features* à partir de caractéristiques *HOG* calculés de manière dense sur les images.

Le jeu de données utilisé est composé de 2686 images en couleurs réparties en 8 classes selon le type de scènes qu'elles représentent. Vous pouvez télécharger une archive contenant le jeu de données ainsi qu'un squelette de code à l'adresse suivante :

`https://www.labri.fr/perso/mclement/teaching/ts341/scenes.tar.gz`

Le fichier scenes.py contient quelques fonctions permettant de charger le jeu de données et de calculer les caractéristiques HOG de manière dense sur une image.

→ L'objectif est de compléter le fichier afin qu'il réalise les étapes suivantes :

1. Calcul des caractéristiques HOG pour toutes les images[1] ;
2. Séparation en deux ensembles : 75% apprentissage et 25% test ;
3. Construction du vocabulaire de mots visuels par k-means à partir de l'ensemble d'apprentissage ;
4. Pour chaque image, calcul de son histogramme de mots visuels ;
5. Classification des images à partir de ces histogrammes. Vous pouvez notamment utiliser les différents classifieurs vus pendant la séance et comparer les résultats obtenus.

[1] Pour gagner du temps, il est aussi possible de télécharger les caractéristiques pré-calculées à l'adresse suivante : `https://www.labri.fr/perso/mclement/teaching/ts341/hog.pickle` et de les charger avec la fonction `load_hog_features` du fichier `scenes.py`