Clustering exercises of FMLCV course

Eva Cernadas
CITIUS: Centro de Tecnoloxías Intelixentes da USC
Universidade de Santiago de Compostela

19 de enero de 2023

During the master course FMLCV (Fundamentals of machine learning for computer vision), students will do different exercises in order to practice the practical contents of unsupervised machine learning models.

The objective is to test the clustering `kmeans` algorithm to the purpose of image segmentation. In the class exercice, apply the `kmeans` algorithm to segment some example images. You can use the `Matlab`[1] software, the computer vision library `OpenCV` [2] (with interface for `C++` and `Python`) or the `KMeans` function of the Python module `sklearn.cluster`[3].

1. Programs in Matlab

For `Matlab`, the basic code is:

```
clear all;
name = input('Image: ', 's')
rgb = imread(name);
figure(1); imshow(rgb);
[nrow, ncol, nf]=size(rgb);
aux=reshape(rgb, nrow*ncol, nf);
k=input('Number of clusters:');
out=kmeans(double(aux), k);
out2=reshape(out, nrow, ncol,1);
figure(2); imshow(mat2gray(out2))
```

which apply the `kmeans` algorithm to the input RGB image using as variables the R, G and B colors and using the number of clusters k.

[1] https://es.mathworks.com/products/matlab.html
[2] https://opencv.org/
[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

2. Programs in Python

We use the `KMeans` function of `sklearn.cluster` object to segment the image. Then, we need to install the `numpy` and matplotlib.pyplot modules to manage the images as `array` objects and to visualize images respectively. We need also to install the `python` interface to `Opencv` library in order to read the images. The following program segments an image into two clusters in order to divide the input image into foreground and background:

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
import cv2
# load image
img = cv2.imread("Image42.jpg", cv2.IMREAD_COLOR)
print('Image Dimensions :', img.shape)
# show image
plt.imshow(img); plt.show() # as RGB Format
# Flatten Each channel of the Image
pixels  = img.reshape((-1,3))
print('New shape: ',pixels.shape)
nclusters = 2
km = KMeans(n_clusters=nclusters)
km.fit(pixels)
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=30,
       n_clusters=nclusters, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
centers = km.cluster_centers_
labels=km.labels_
centers = np.array(centers,dtype='uint8')
labels = np.array(labels,dtype='uint8')
print('Center of clusters: ',centers)
print('Labels shape: ',labels.shape, ' Value: ', labels)
segimg = np.zeros((1, img.shape[0]*img.shape[1]),dtype='uint8')
print('Segmented image shape: ', segimg.shape)
# Iterate over the image
for ix in range(segimg.shape[1]):
    segimg[0, ix] = labels[ix]*255
segimg = segimg.reshape((img.shape[0],img.shape[1]))
plt.imshow(segimg); plt.gray(); plt.show()
```

The use of clustering to segment an image can be statistically evaluated considering that each pixel is a pattern. In this case, we can use all the metrics studied in the model selection and evaluation subject (kappa, accuracy, precision, recall, etc). The next code is an example:

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import *
import cv2
# load images
img = cv2.imread("images/images/Image10.jpg", cv2.IMREAD_COLOR) # original
binimg = cv2.imread("images/mask/Image10.png",cv2.IMREAD_GRAYSCALE)
print('Image Dimensions :', img.shape)
# show images
plt.imshow(img); plt.show() # as RGB Format
plt.imshow(binimg); plt.gray(); plt.show() # as grey Format
# Flatten Each channel of the Image
pixels  = img.reshape((-1,3))
print('New shape: ',pixels.shape)
nclusters = 2
km = KMeans(n_clusters=nclusters)
km.fit(pixels)
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=30,
       n_clusters=nclusters, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
centers = km.cluster_centers_
labels=km.labels_
centers = np.array(centers,dtype='uint8')
labels = np.array(labels,dtype='uint8')
print('Center of clusters: ',centers)
print('Labels shape: ',labels.shape, ' Value: ', labels)
segimg = np.zeros((1, img.shape[0]*img.shape[1]),dtype='uint8')
print('Segmented image shape: ', segimg.shape)
# Iterate over the image
for ix in range(segimg.shape[1]):
    segimg[0, ix] = labels[ix]*255
segimg = segimg.reshape((img.shape[0],img.shape[1]))
plt.imshow(segimg); plt.gray(); plt.show()
# evaluation
binimg[binimg[:,:]==255]=1 # desired ouput for each pixel
y=binimg.flatten()
print('y size: ', y.shape)
z=labels
print('z size: ', z.shape)
kappa=cohen_kappa_score(y,z);acc=accuracy_score(y,z)
print('kappa=%.1f%% accuracy=%.1f%%'%(100*kappa,100*acc))
cf=confusion_matrix(y,z)
```

```
print('confusion matrix:'); print(cf)
pre=precision_score(y,z)
re=recall_score(y,z)
f1=f1_score(y,z)
print('precision=%.1f%% recall=%.1f%% f1=%.1f%%'%(100*pre,100*re,100*f1))
```

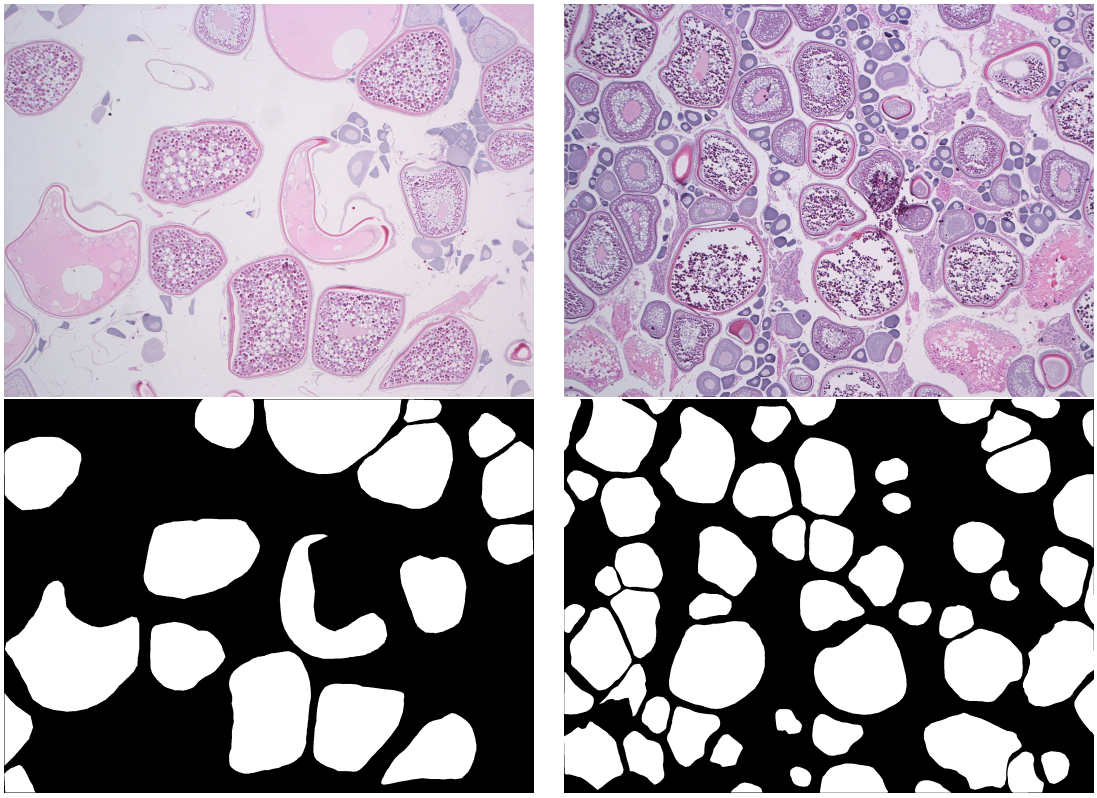

Figura 1: Histological images of fish ovaries of European Hake: original images (upper row) and true annotated images provided by the experts (lower row).

The main problem with the current implementation is that the `Kmeans` algorithm uses a random initialization for the initial clusters. This fact can do the algorithm converge to the optimal solution or to other solutions not optimal from the segmentation point of view (for example, the background is considered as foreground). We can solve this limitation providing initial seeds for the `kmeans` function.

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import *
import cv2
```

```
# load images
img = cv2.imread("images/images/Image10.jpg", cv2.IMREAD_COLOR) # original
binimg = cv2.imread("images/mask/Image10.png",cv2.IMREAD_GRAYSCALE)
print('Image Dimensions :', img.shape)
# show images
plt.imshow(img); plt.show() # as RGB Format
plt.imshow(binimg); plt.gray(); plt.show() # as grey Format
# Flatten Each channel of the Image
pixels  = img.reshape((-1,3))
print('New shape: ',pixels.shape)
nclusters = 2
km = KMeans(n_clusters=nclusters)
km.fit(pixels)
# cluster initialization
icluster=np.array([[255,255,255], [255, 0 ,0]], dtype=np.uint8)
KMeans(algorithm='auto', copy_x=True, init=icluster, max_iter=30,
       n_clusters=nclusters, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
centers = km.cluster_centers_
labels=km.labels_
centers = np.array(centers,dtype='uint8')
labels = np.array(labels,dtype='uint8')
print('Center of clusters: ',centers)
print('Labels shape: ',labels.shape, ' Value: ', labels)
segimg = np.zeros((1, img.shape[0]*img.shape[1]),dtype='uint8')
print('Segmented image shape: ', segimg.shape)
# Iterate over the image
for ix in range(segimg.shape[1]):
    segimg[0, ix] = labels[ix]*255

segimg = segimg.reshape((img.shape[0],img.shape[1]))
plt.imshow(segimg); plt.gray(); plt.show()
# evaluation
binimg[binimg[:,:]==255]=1 # desired ouput for each pixel
y=binimg.flatten()
print('y size: ', y.shape)
z=labels
print('z size: ', z.shape)
kappa=cohen_kappa_score(y,z);acc=accuracy_score(y,z)
print('kappa=%.1f%% accuracy=%.1f%%'%(100*kappa,100*acc))
cf=confusion_matrix(y,z)
print('confusion matrix:'); print(cf)
pre=precision_score(y,z)
```

```
re=recall_score(y,z)
f1=f1_score(y,z)
print('precision=%.1f%% recall=%.1f%% f1=%.1f%%'%(100*pre,100*re,100*f1))
```

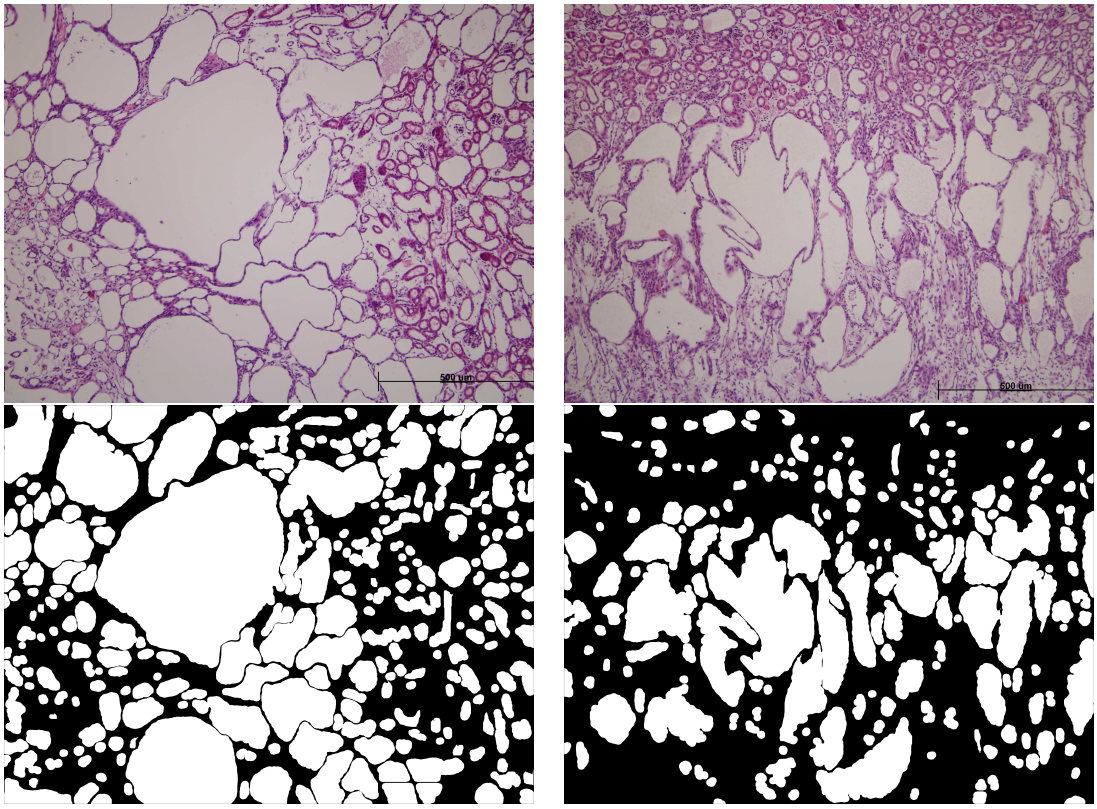

Figura 2: Histological images of kidney mouse: original images (upper row) and true annotated images provided by the experts (lower row).

3. Exercises to do by the students

In the lab work, we will use the `kmeans` algorithm to segment some biological and biomedical images. The segmentation objective depends on the problem in hand. The example images can be downloaded from the subject webpage `images.zip`. They are the following problems:

1. **Histological images of fish ovary:** our objective is recognized the matured cells in the image in order to measure and count the cells of each development state. The example images and their true segmentation provided by the experts can be

see in figure 1. In the CiTIUS webpage more information, images and the software STERapp[4]

2. **Histological images of kidney mouse:** our objective is recognized the cyst in the image in order to measure and count. The example images and their true segmentation provided by the experts can be see in figure 2. In the CiTIUS webpage more information, images and software CystAnalyser[5]

3. **Immunohistochemical images of mouth tissue:** our objective is to detect and count the number of nuclui in the image. The nucleus can be in blue color (not stained) in brown color (stained). Figure 3 show some examples.

4. **Histological images of liver using oilred protocol:** our objective is to quantify the percentage area of orange/red spots in the image. See example images in figure 4.

5. **Histological images of liver:** our objective is to quantify the percentage area of fat (white spots in the image). See example images in figure 5.

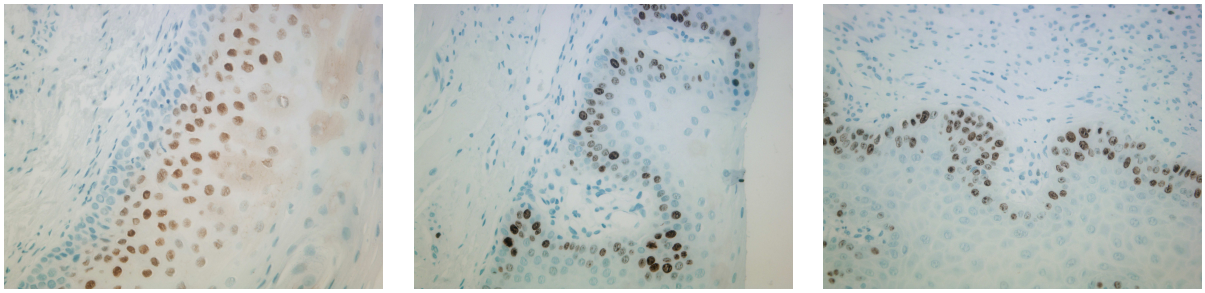

Figura 3: Immunohistochemical images of mouth tissue.

The lab work for the students is:

1. Observe visually the behaviour of the `kmeans` algorithm for the different types of images mentioned above. You can change the number of clusters used and the features used to do the clustering (for example, pixel grey level, pixel RGB color, Lab color space, etc).

2. Evaluated statistically at pixel level the performance of the clustering for the histological images of fish gonads and kidney.

3. Analyse visually the changes in the resultant segmentation when the initial clusters are provided to the `kmeans` algorithm. Calculate the classification metrics for some image.

[4]https://citius.usc.es/transferencia/software/sterapp
[5]https://citius.usc.es/transferencia/software/cystanalyser

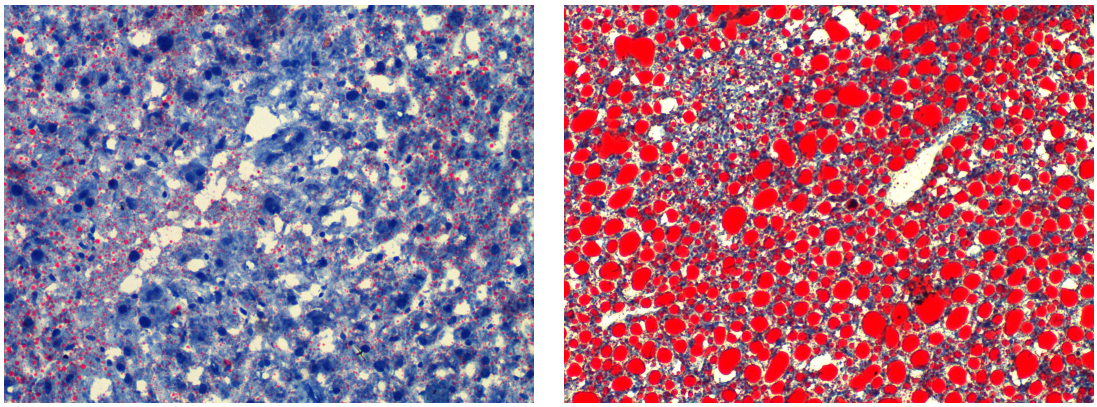

Figura 4: Original oilred histological images of liver mouse.

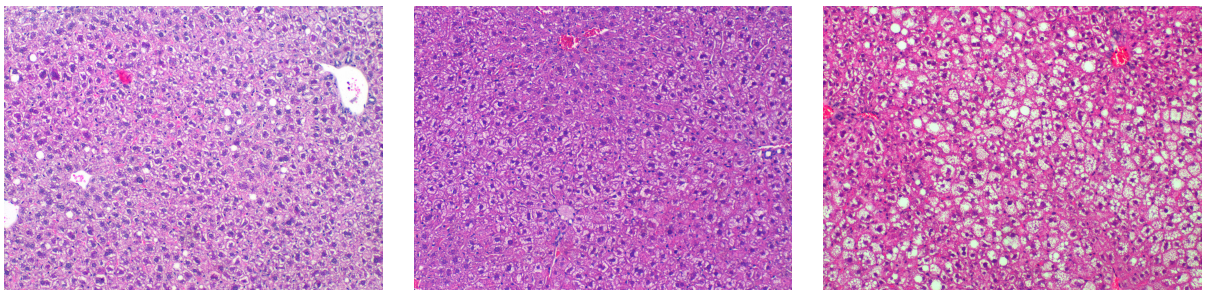

Figura 5: Original oilred histological images of liver mouse.

4. **Optional** Check other changes that you can imagine (for example, pre-processing the original image with filter or calculate texture features) or implement the metrics to do a region evaluation of the segmentation process.