Machine-Learning-9 Assignment

Segmenting the picture of a raccoon face in regions

This example uses `spectral_clustering` on a graph created from voxel-to-voxel difference on an image to break this image into multiple partly-homogeneous regions.

This procedure (spectral clustering on an image) is an efficient approximate solution for finding normalized graph cuts.

There are two options to assign labels:

- with 'kmeans' spectral clustering will cluster samples in the embedding space using a kmeans algorithm
- whereas 'discrete' will iteratively search for the closest partition space to the embedding space.

In [8]:

```
%matplotlib inline
```

Compress racoon grey scale image into 5 clusters using Sklearn Spectral Clustering Api

In [30]:

```
import time
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
from sklearn.feature_extraction import image
from sklearn.cluster import spectral_clustering

try:
    from scipy.misc import face
    face = face(gray=True)
except ImportError:
    face = sp.face(gray=True)

face = sp.misc.imresize(face, 0.10) / 255.

graph = image.img_to_graph(face)

beta = 5
eps = 1e-6
graph.data = np.exp(-beta * graph.data / graph.data.std()) + eps

print(type(graph))
print(graph.shape)
```

```
<class 'scipy.sparse.coo.coo_matrix'>
(7752, 7752)
```

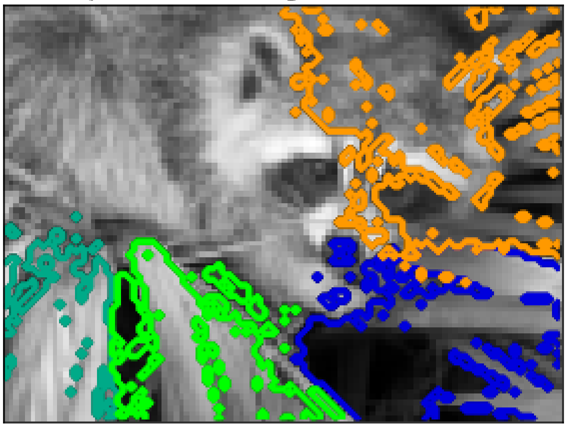
Visualize the resulting regions

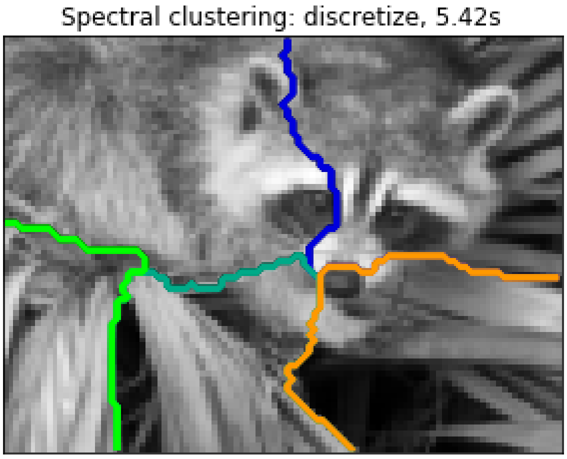
In [31]:

```
import time
N_REGIONS = 5
for assign_labels in ('kmeans', 'discretize'):
    t0 = time.time()
    labels = spectral_clustering(graph, n_clusters=N_REGIONS,
                                 assign_labels=assign_labels, random_state=1)
    t1 = time.time()
    print(labels.shape)
    labels = labels.reshape(face.shape)
    print(labels.shape)
    print(np.unique(labels))
    plt.figure(figsize=(5, 5))
    plt.imshow(face, cmap=plt.cm.gray)
    for l in range(N_REGIONS):
        plt.contour(labels == l, contours=1,
                    colors=[plt.cm.nipy_spectral(l / float(N_REGIONS))])
    plt.xticks(())
    plt.yticks(())
    title = 'Spectral clustering: %s, %.2fs' % (assign_labels, (t1 - t0))
    print(title)
    plt.title(title)
plt.show()
```

```
(7752,)
(76, 102)
[0 1 2 3 4]
Spectral clustering: kmeans, 5.43s
(7752,)
(76, 102)
[0 1 2 3 4]
Spectral clustering: discretize, 5.42s
```

Spectral clustering: kmeans, 5.43s





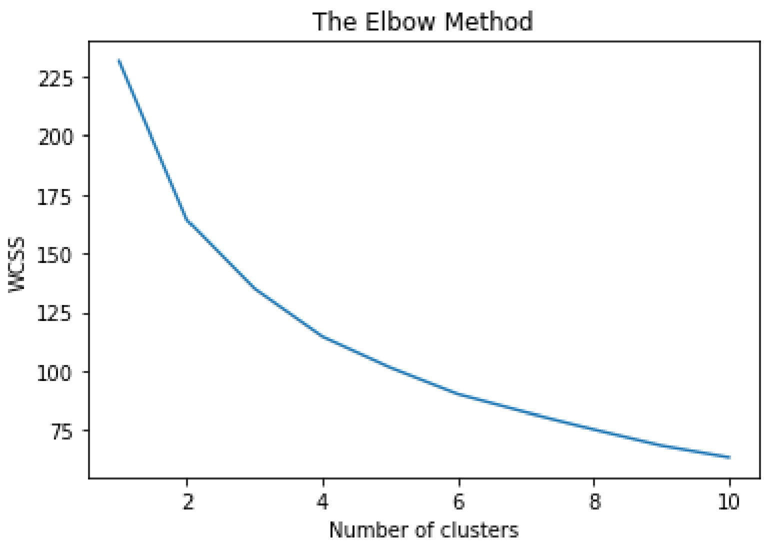
Clustering using K Means Algorithm - Kmeans sklearn Api

In [44]:

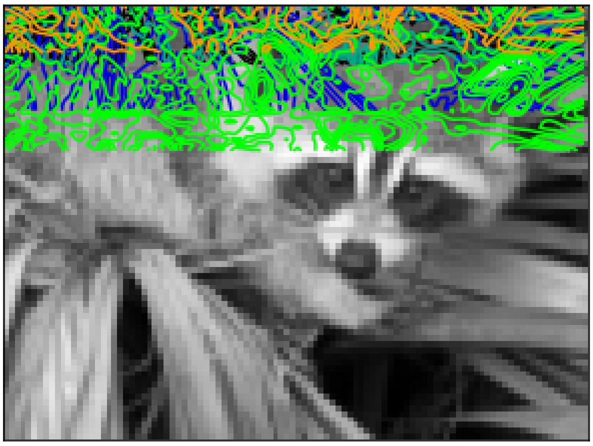
```
from sklearn.cluster import KMeans


n_clusters = 5
np.random.seed(0)
X = face
wcss=[]
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()


#face = sp.misc.imresize(face, 0.10) / 255.
X =face
k_means = KMeans(n_clusters=5,random_state=0,init='k-means++')
k_means.fit(X)
labels = k_means.labels_
print("labels generated :\n",labels)
N_REGIONS =5
#labels = labels.reshape(face.shape)
#print(labels.reshape(-1,102))
#plt.figure(figsize=(5, 5))
plt.imshow(face, cmap=plt.cm.gray)
for l in range(N_REGIONS):
    plt.contour(X[labels == l], contours=1,colors=[plt.cm.nipy_spectral(l / float(N_REGIONS
plt.xticks(())
plt.yticks(())
plt.show()
```




```
labels generated :
 [3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4
 4 4 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1]
```



Spectaral Clustering using similarity distance matrix, Degree diagonal matrix, Laplacian matrix

In [14]:

```
import numpy as np
from scipy import linalg as LA
var = 1.5
k = 5
def RbfKernel(data1, data2, sigma):
    delta =np.matrix(abs(np.subtract(data1, data2)))
    squaredEuclidean = (np.square(delta).sum(axis=1))
    result = np.exp(-(squaredEuclidean)/(2*sigma**2))
    return result

def buildSimmilarityMatrix(dataIn):
    nData = dataIn.shape[0]
    result = np.matrix(np.full((nData,nData), 0, dtype=np.float))
    for i in range(0,nData):
        for j in range(0, nData):
            weight = RbfKernel(dataIn[i, :], dataIn[j, :], var)
            result[i,j] = weight
    return result

def buildDegreeMatrix(similarityMatrix):
    diag = np.array(similarityMatrix.sum(axis=1)).ravel()
    result = np.diag(diag)
    return result

def unnormalizedLaplacian(simMatrix, degMatrix):
    result = degMatrix - simMatrix
    return result

def transformToSpectral(laplacian):
    global k
    e_vals, e_vecs = LA.eig(np.matrix(laplacian))
    ind = e_vals.real.argsort()[:k]
    result = np.ndarray(shape=(laplacian.shape[0],0))
    for i in range(1, ind.shape[0]):
        cor_e_vec = np.transpose(np.matrix(e_vecs[:,np.asscalar(ind[i])]))
        result = np.concatenate((result, cor_e_vec), axis=1)
    return result
```

In [15]:

```
simMat = buildSimmilarityMatrix(face)
degMat = buildDegreeMatrix(simMat)
lapMat = unnormalizedLaplacian(simMat, degMat)
transformedData = transformToSpectral(lapMat)
```

In [17]:

```
transformedData.shape
```

Out[17]:

```
(76, 4)
```

In [18]:

```
lapMat.shape
```

Out[18]:

```
(76, 76)
```

In [21]:

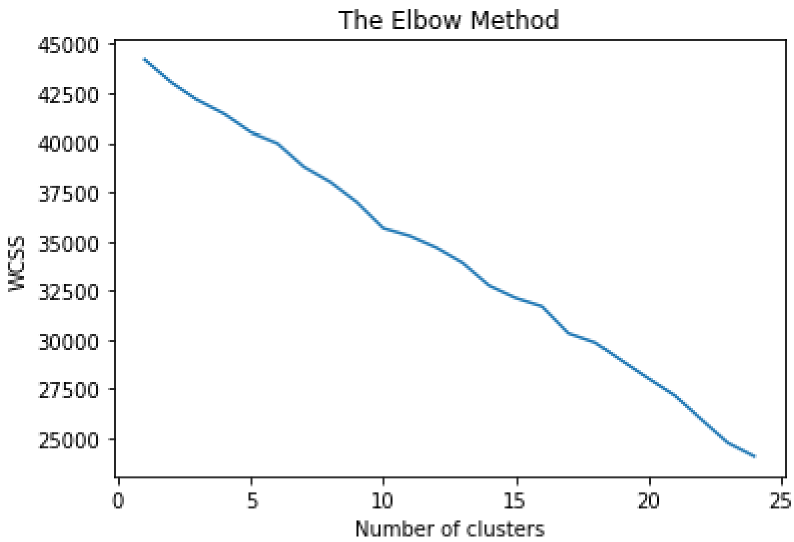
```
simMat.shape
```

Out[21]:

```
(76, 76)
```

In [24]:

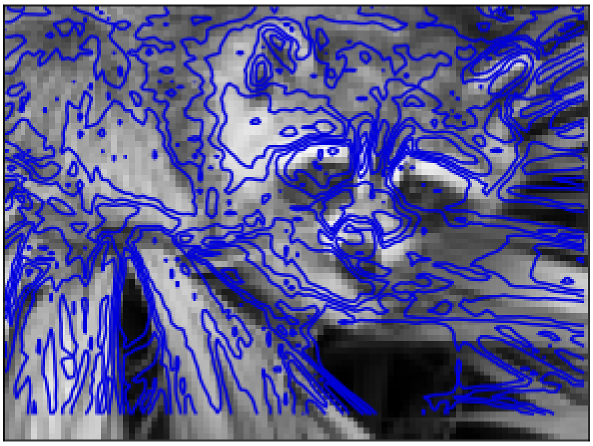
```
wcss=[]
for i in range(1,25):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 0)
    kmeans.fit(lapMat)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 25), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [27]:

```
k_means = KMeans(n_clusters=5,random_state=0,init='k-means++')
k_means.fit(lapMat)
labels = k_means.labels_
print("labels generated :\n",labels)
N_REGIONS =5
#labels = labels.reshape(face.shape)
#print(labels.reshape(-1,102))
#plt.figure(figsize=(5, 5))
plt.imshow(face, cmap=plt.cm.gray)
for l in range(N_REGIONS):
    plt.contour(X[labels == l], contours=1,colors=[plt.cm.nipy_spectral(l / float(N_REGIONS
plt.xticks(())
plt.yticks(())
plt.show()
```

```
labels generated :
 [1 1 2 1 0 1 1 1 3 1 1 1 1 1 1 1 1 1 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1]
```



In [ ]: