Kmeans Vs Spectral Clustering

Name: Rajarshi Chattopadhyay | Net Id: RXC170010

In [1]:
```
import numpy as np
from sklearn import datasets
from scipy.linalg import eigh
from scipy.spatial.distance import pdist
from scipy.spatial.distance import squareform
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
import math
```

In [2]:
```
# Generate a noisy, synthetic data set (data_points, label)
x, y = datasets.make_circles(n_samples=1500, factor=.5, noise=.05)
```

In [3]:
```
print(x) # 2D data points
```

```
[[ 0.87550855 -0.41175092]
 [-0.04662449  0.98527592]
 [ 0.63987212  0.83096838]
 ...
 [ 0.86606517  0.16920158]
 [ 0.75810365  0.68690089]
 [-0.41724887 -0.96049544]]
```

In [4]:
```
print(x.shape)
```

```
(1500, 2)
```

In [5]:
```
print(y) # class labels
```

```
[0 0 0 ... 0 0 0]
```

In [6]:
```
print(y.shape)
```

```
(1500,)
```

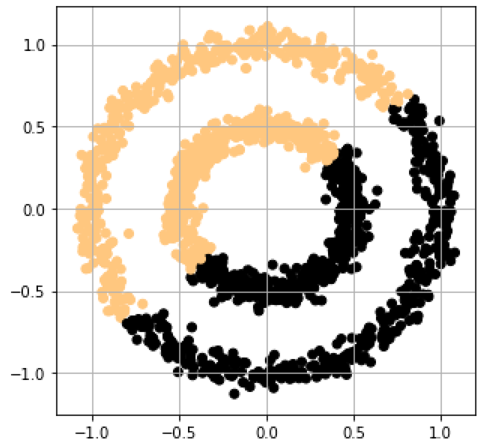
In [7]:
```
def similarity_matrix(x, gamma):
    p_dist = pdist(x)
    S = squareform(p_dist)
    A = np.exp(-gamma * S)
    return A
```

In [8]:
```
def laplacian_matrix(A):
    n = len(A)
    D = np.zeros((n, n))
    D = np.diag(np.sum(np.array(A),axis=1))
    for i in range(n):
        _sum = 0
        for j in range(n):
            _sum = _sum + A[i, j]
        D[i, i] = _sum
    L = D - A
    return L
```

In [9]:
```
def spectral_clustering(x, k, gamma):
    A = similarity_matrix(x, gamma)
    L = laplacian_matrix(A)
    eig_val, eig_vect = eigh(L, eigvals=(0,k-1))
    V = eig_vect
    kmeans = KMeans(n_clusters=k).fit(V)
    return kmeans.cluster_centers_, kmeans.labels_
```

In [10]:
```python
kmeans = KMeans(n_clusters=2).fit(x)
cluster_centers = kmeans.cluster_centers_
labels = kmeans.labels_
plt.figure(figsize=(5,5))
plt.grid()
plt.scatter(x[:,0],x[:,1], c=labels, cmap='copper')
```

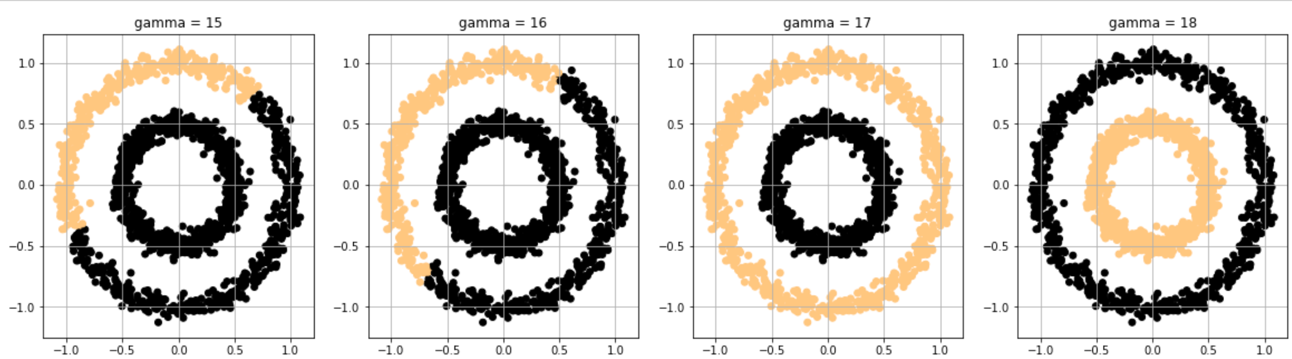
Out[10]: `<matplotlib.collections.PathCollection at 0x1a22d78860>`



Kmeans clustering generates a linear classifier that is unable to separate the inner and outer circles

We can tune the spectral clustering using 'gamma' to change the width of the RBF (non-linear circular) classifier.

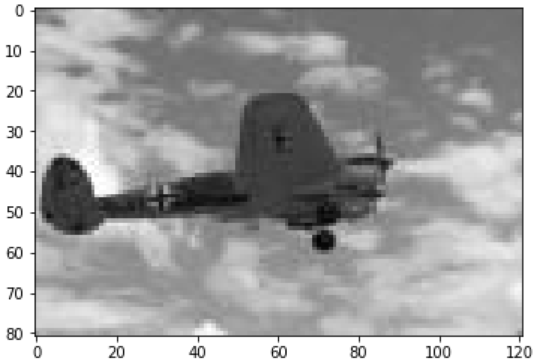
In [11]:
```python
gamma = [15, 16, 17, 18]
n = len(gamma)
plt.figure(figsize=(5*n,5))
i=1
for g in gamma:
    cluster_centers, labels = spectral_clustering(x,2,g)
    plt.subplot(1,n,i)
    plt.gca().set_title("gamma = " + str(g))
    plt.grid()
    plt.scatter(x[:,0],x[:,1], c=labels, cmap='copper')
    i=i+1
```



- **This puts a stamp on the fact that K-means clustering produces linear decision boundaries. As a significance, Kmeans clustering can form clusters of data points enclosed in cellular spaces enclosed by linear boundaries described by the Voronoi diagram.**
- **In a dataset like above, it is impossible to separate out the data points, particularly with just k=2 (2 clusters).**
- **On the other hand, spectral clustering with an optimal width of the circular (non-linear) decision boundary is able to deal with the problem and outperforms Kmeans clustering.**

In [12]:
```
import cv2
image = cv2.imread('data/seg.jpg', 0)
plt.figure()
plt.imshow(image, cmap='gray', vmin=0, vmax=255)
```

Out[12]: <matplotlib.image.AxesImage at 0x1a2e9b9828>



In [13]:
```
print(image.shape)
```

```
(81, 121)
```

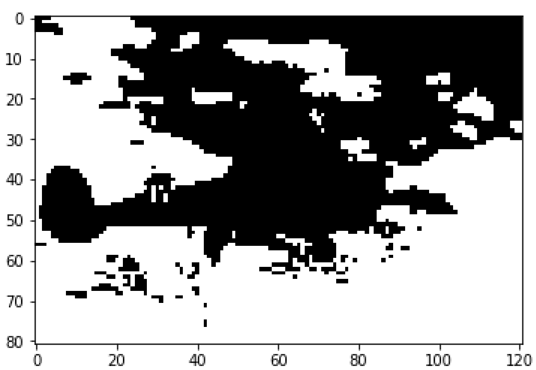
In [14]:
```
height, width = image.shape
image1d = image.reshape(height*width, 1)
```

In [15]:
```
print(image1d.shape)
```

```
(9801, 1)
```

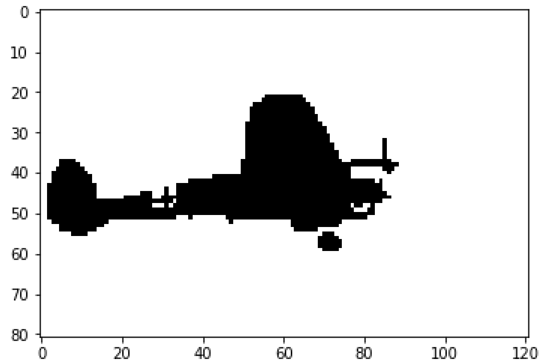
In [16]:
```
clf = KMeans(n_clusters=2)
clf.fit(image1d)
index = np.copy(clf.labels_)
plt.figure()
plt.imshow(index.reshape(height, width), cmap='gray', vmin=0, vmax=1)
```

Out[16]: <matplotlib.image.AxesImage at 0x1a2eaf3668>



In [18]:
```
c, index = spectral_clustering(image1d,2,0.1)
plt.figure()
plt.imshow(index.reshape(height, width), cmap='gray', vmin=0, vmax=1)
```

Out[18]: <matplotlib.image.AxesImage at 0x1a239b6048>



The spectral clustering gives us a better image segmentation than Kmeans

Thus, we can conclude and justify that *Kmeans* classifies based on the compactness of the data points, whereas *Spectral clustering* uses connectivity.
As we use a tunable (width controlled) Gaussian kernel to decide on the simlarity of the data points, we can control the size of the neighborhood.
As a result we can obtain near to ideal segmentation of data points using Spectral Clustering

In [ ]: