BRSU

Neural Networks
Assignment 8

Bastian Lang

November 28, 2015

1 Exercise 5.10

1.1 Note

This exercise does not make much sense if the other parameters are not given. If I would have chosen a very big radius, then the vertical separation would not have had any impact at all...

1.2 output

Figure 1.1 till 1.14 show clustered data with variation of vertical separation.
One can observe that the clusters "mix", i.e. parts of the lower figure become part of clusters of the upper figure and vice versa.

This also effects the mean and the variance, but mostly in y-direction. The other figures show all means for the different clusters. I had to drop the variance figures because of technical problems and no time left while building the pdf...

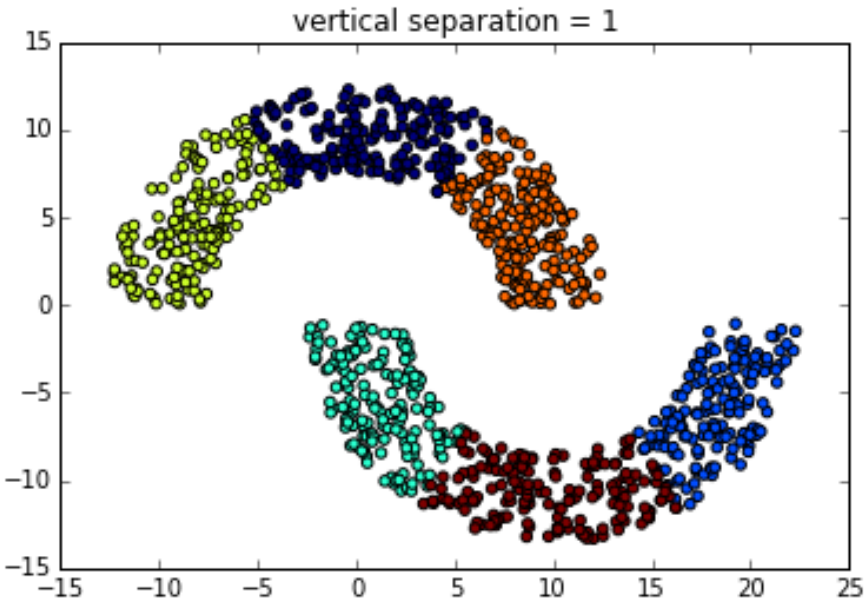

Figure 1.1: Clustered samples from figure with vertical separation of 1

1.3 code

```
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 28 14:24:39 2015
```

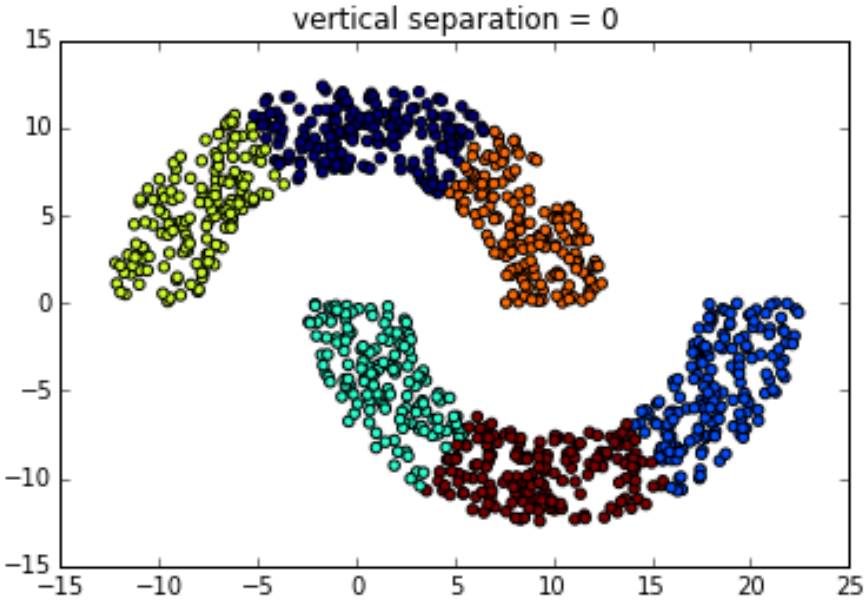

Figure 1.2: Clustered samples from figure with vertical separation of 0

```
@author: bastian
"""

import random
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans


RADIUS = 10
WIDTH = 5
VERTICAL_SEPARATION = 5

def sample_upper_halfmoon(radius, width):
    distance = random.random() * width + radius - 0.5 * width
    angle = random.random() * 180
    angle = np.radians(angle)
    x = np.cos(angle) * distance
    y = np.sin(angle) * distance
    return x,y

def sample_lower_halfmoon(radius, width, vertical_separation):
    distance = random.random() * width + radius - 0.5 * width
```

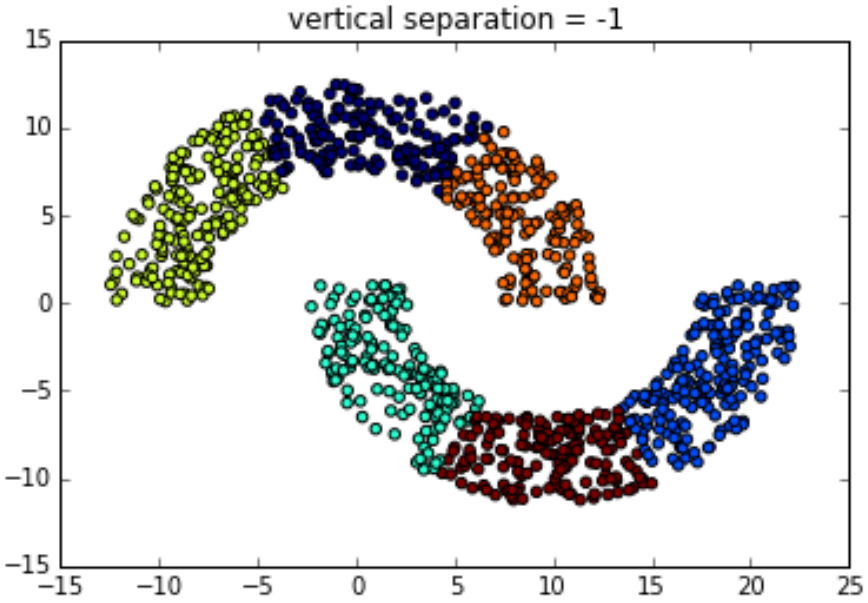

Figure 1.3: Clustered samples from figure with vertical separation of -1

```
    angle = random.random() * 180 + 180
    angle = np.radians(angle)
    x = np.cos(angle) * distance + radius
    y = np.sin(angle) * distance - vertical_separation
    return x,y

def sample_figure(radius, width, vertical_separation):
    index = random.randint(0,1)
    if(index == 0):
        return sample_upper_halfmoon(radius, width)
    else:
        return sample_lower_halfmoon(radius, width, vertical_separation)

def sample_and_cluster(radius, width, vertical_separation):
    X = []
    Y = []
    points = []
    for i in range (1000):
        x,y = sample_figure(radius, width, vertical_separation)
        X.append(x)
        Y.append(y)
        points.append([x,y])
```

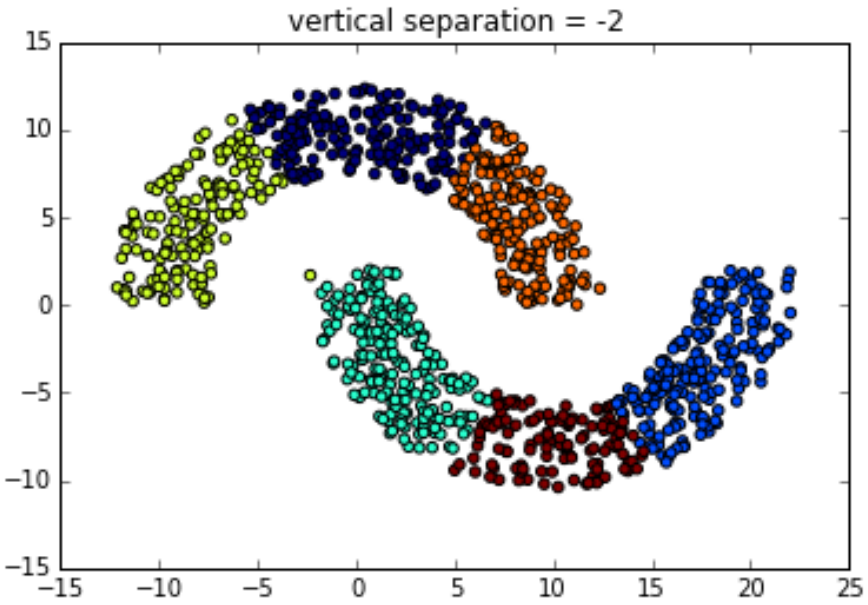

Figure 1.4: Clustered samples from figure with vertical separation of -2

```
kmeans = KMeans(n_clusters = 6)
kmeans.fit(points)
labels = kmeans.labels_
fig1 = plt.figure()
plt.title('vertical separation = '+ str(vertical_separation))
ax1 = fig1.add_subplot(111)
ax1.scatter(X, Y, c=labels.astype(np.float))

c1=[]
c2=[]
c3=[]
c4=[]
c5=[]
c6=[]
for i in range(len(labels)):
    cluster = labels[i]
    point = points[i]
    if(cluster == 0):
        c1.append(point)
    elif(cluster == 1):
        c2.append(point)
    elif(cluster == 2):
        c3.append(point)
```

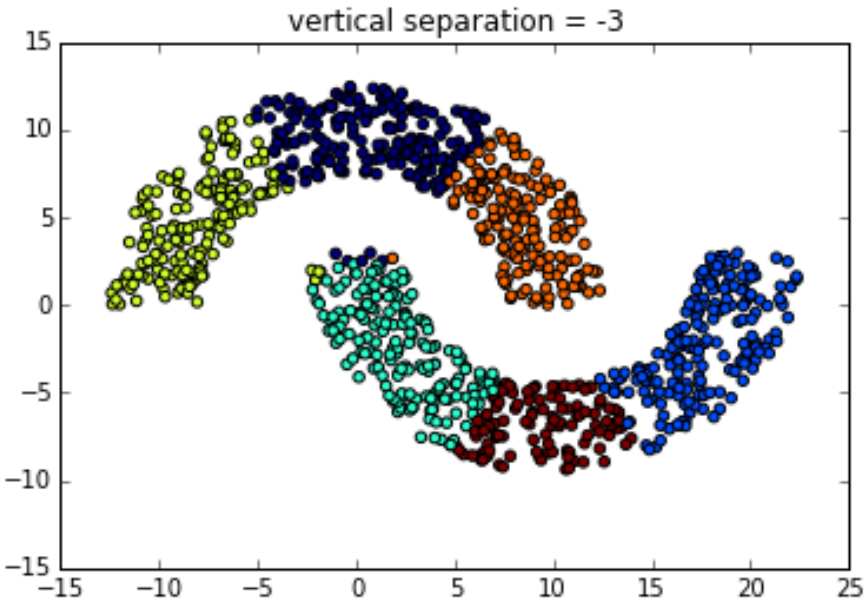

Figure 1.5: Clustered samples from figure with vertical separation of -3

```
        elif(cluster == 3):
            c4.append(point)
        elif(cluster == 4):
            c5.append(point)
        elif(cluster == 5):
            c6.append(point)
    return c1,c2,c3,c4,c5,c6

def get_clusters(data, labels):
    c1=[]
    c2=[]
    c3=[]
    c4=[]
    c5=[]
    c6=[]
    for i in range(len(labels)):
        cluster = labels[i]
        point = data[i]
        if(cluster == 0):
            c1.append(point)
        elif(cluster == 1):
            c2.append(point)
        elif(cluster == 2):
```

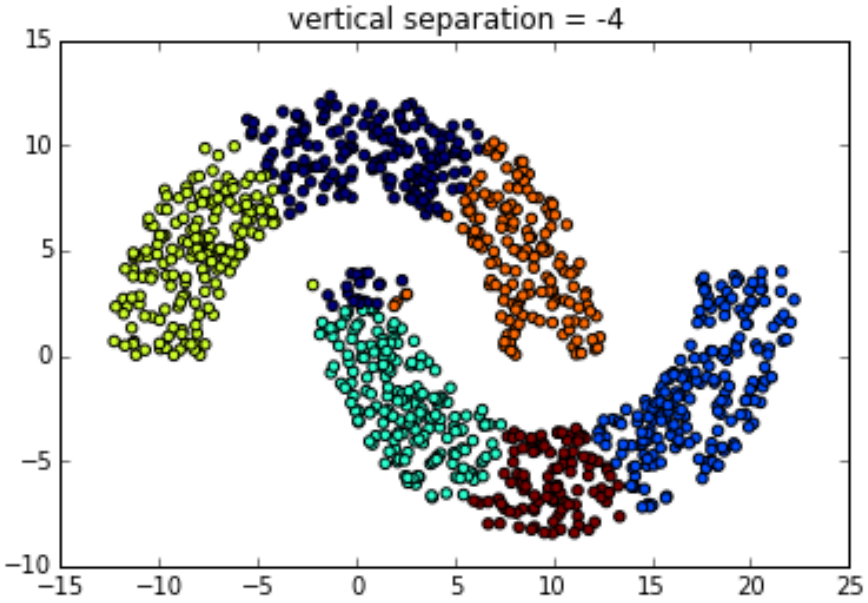

Figure 1.6: Clustered samples from figure with vertical separation of -4

```
            c3.append(point)
        elif(cluster == 3):
            c4.append(point)
        elif(cluster == 4):
            c5.append(point)
        elif(cluster == 5):
            c6.append(point)
    return c1,c2,c3,c4,c5,c6

def plot_data(data, title):
    fig = plt.figure()
    plt.title(title)
    ax = fig.add_subplot(111)
    x,y = get_x_y_values(data)
    ax.scatter(x, y)

def vary_vertical_separation():
    c1_means = []
    c2_means = []
    c3_means = []
    c4_means = []
    c5_means = []
    c6_means = []
```

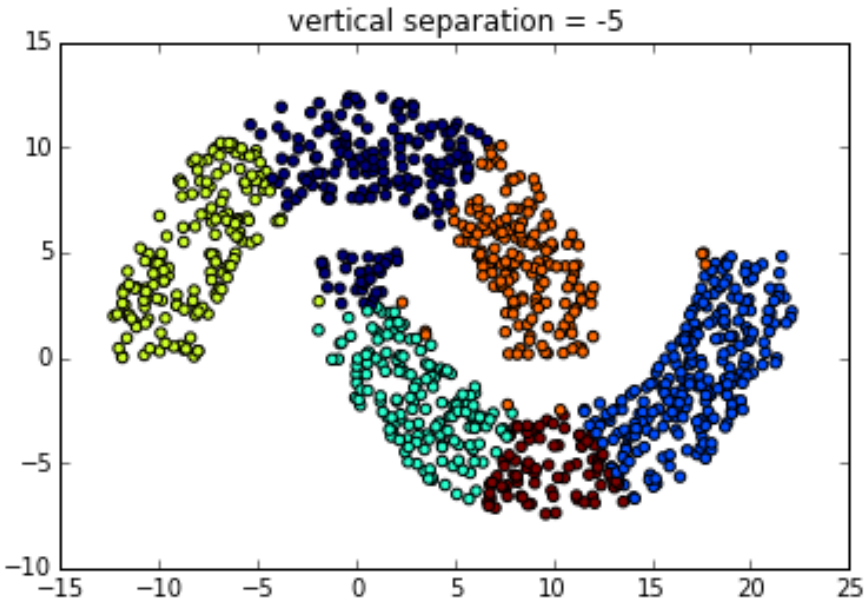

Figure 1.7: Clustered samples from figure with vertical separation of -5

```
c1_variances = []
c2_variances = []
c3_variances = []
c4_variances = []
c5_variances = []
c6_variances = []
for i in range(-6,2):
    c1,c2,c3,c4,c5,c6 = sample_and_cluster(RADIUS,WIDTH,i)
    mean_1 = compute_mean(c1)
    c1_means.append(mean_1)
    c1_variances.append(compute_variance(c1, mean_1))
    mean_2 = compute_mean(c2)
    c2_means.append(mean_2)
    c2_variances.append(compute_variance(c2, mean_2))
    mean_3 = compute_mean(c3)
    c3_means.append(mean_3)
    c3_variances.append(compute_variance(c3, mean_3))
    mean_4 = compute_mean(c4)
    c4_means.append(mean_4)
    c4_variances.append(compute_variance(c4, mean_4))
    mean_5 = compute_mean(c5)
    c5_means.append(mean_5)
    c5_variances.append(compute_variance(c5, mean_5))
```

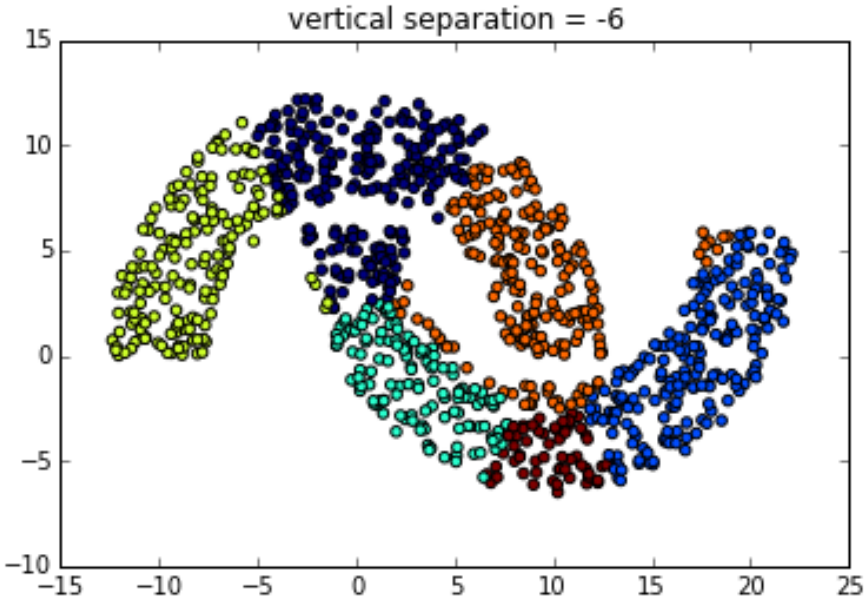

Figure 1.8: Clustered samples from figure with vertical separation of -6

```
    mean_6 = compute_mean(c6)
    c6_means.append(mean_6)
    c6_variances.append(compute_variance(c6, mean_6))

fig = plt.figure()
plt.title('means of cluster 1')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c1_means)
ax.scatter(x, y)

fig = plt.figure()
plt.title('means of cluster 2')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c2_means)
ax.scatter(x, y)

fig = plt.figure()
plt.title('means of cluster 3')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c3_means)
ax.scatter(x, y)

fig = plt.figure()
```

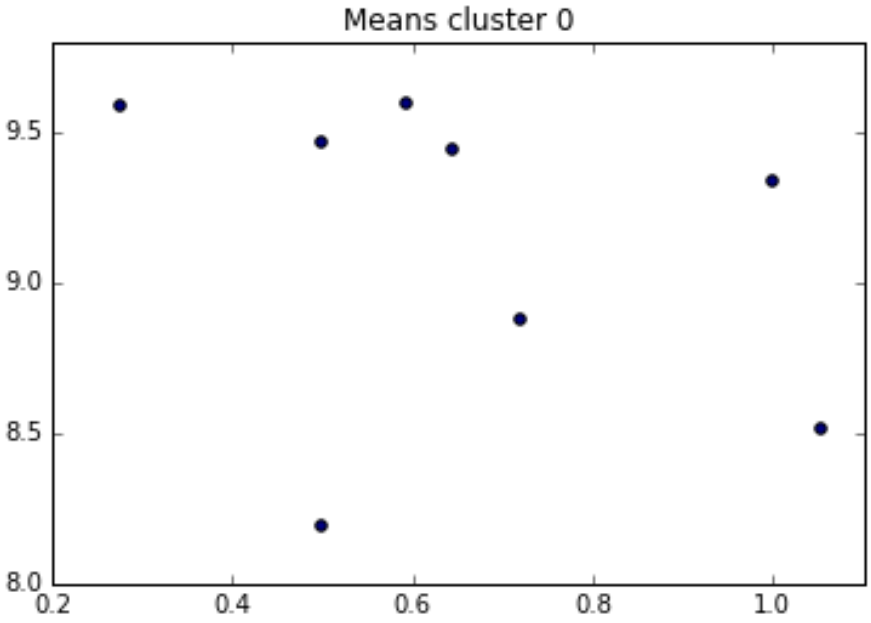

Figure 1.9: Means of cluster 1

```
plt.title('means of cluster 4')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c4_means)
ax.scatter(x, y)

fig = plt.figure()
plt.title('means of cluster 5')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c5_means)
ax.scatter(x, y)

fig = plt.figure()
plt.title('means of cluster 6')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c6_means)
ax.scatter(x, y)

fig = plt.figure()
plt.title('variances of cluster 1')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c1_variances)
ax.scatter(x, y)
```

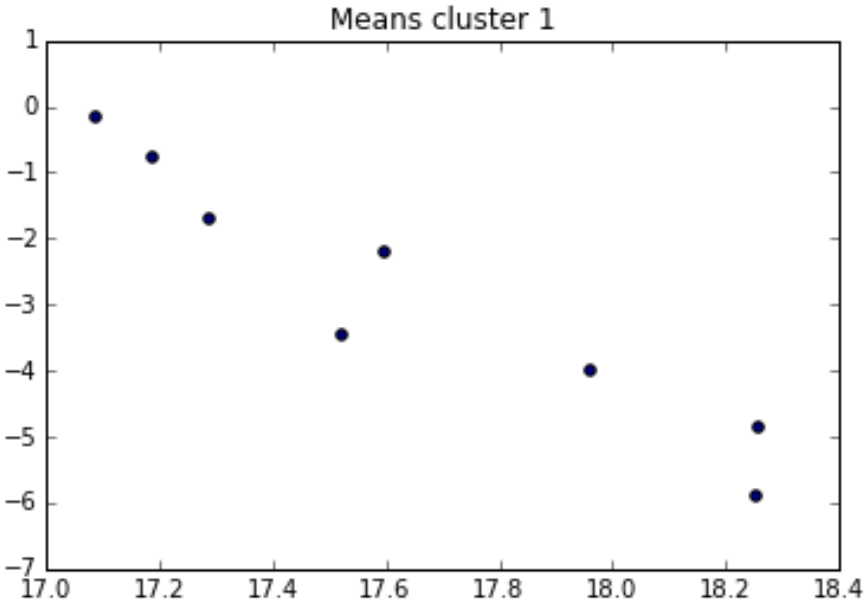

Figure 1.10: Means of cluster 2

```
fig = plt.figure()
plt.title('variances of cluster 2')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c2_variances)
ax.scatter(x, y)

fig = plt.figure()
plt.title('variances of cluster 3')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c3_variances)
ax.scatter(x, y)

fig = plt.figure()
plt.title('variances of cluster 4')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c4_variances)
ax.scatter(x, y)

fig = plt.figure()
plt.title('variances of cluster 5')
ax = fig.add_subplot(111)
x,y = get_x_y_values(c5_variances)
```

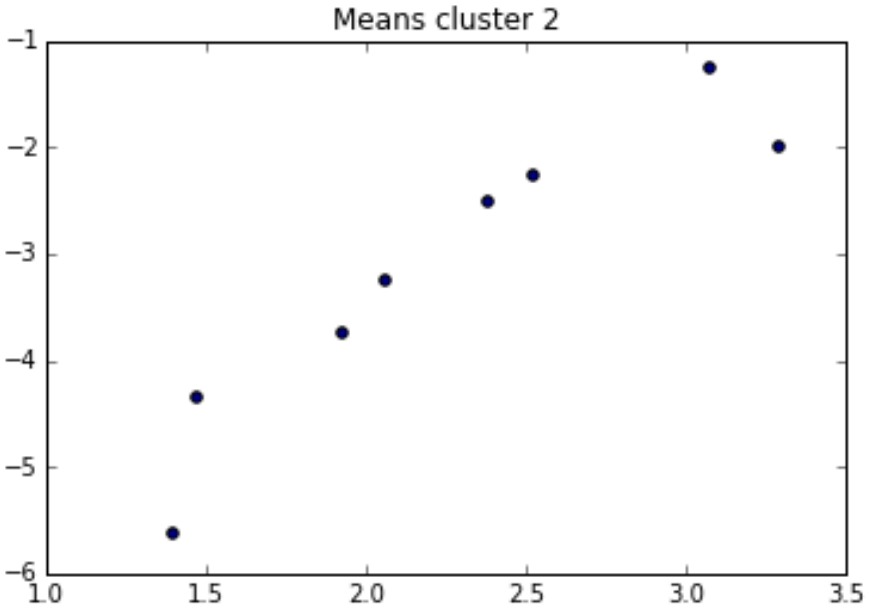

Figure 1.11: Means of cluster 3

```
    ax.scatter(x, y)

    fig = plt.figure()
    plt.title('variances of cluster 6')
    ax = fig.add_subplot(111)
    x,y = get_x_y_values(c6_variances)
    ax.scatter(x, y)

def get_x_y_values(data):
    x = []
    y = []
    for point in data:
        x.append(point[0])
        y.append(point[1])
    return x,y


def compute_mean(data):
    mean = [0,0]
    for point in data:
        mean[0] += point[0]
        mean[1] += point[1]
```

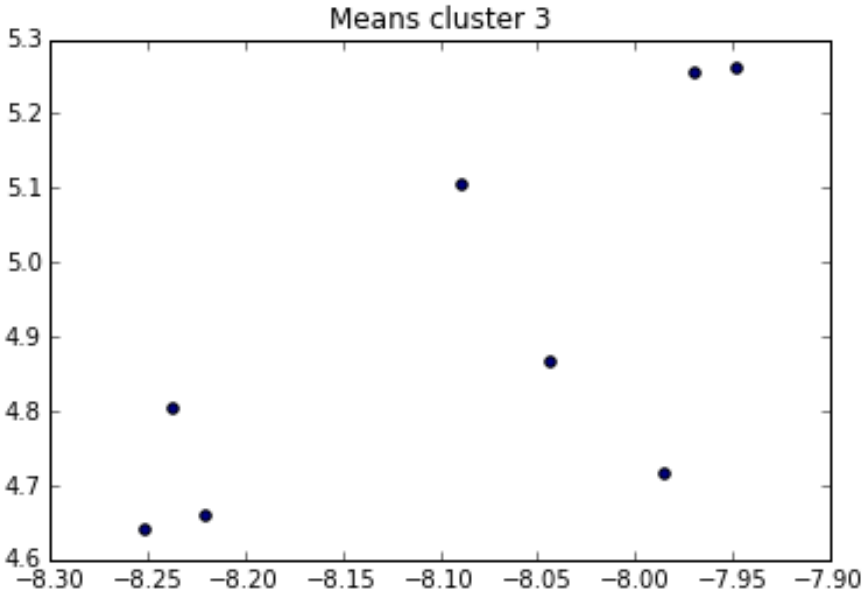

Figure 1.12: Means of cluster 4

```
    return mean[0]/len(data), mean[1] / len(data)

def compute_variance(data, mean):
    sum_squared_differences = [0,0]
    for point in data:
        squared_difference = ((point[0] - mean[0])**2, (point[1] - mean[1])**2)
        sum_squared_differences[0] += squared_difference[0]
        sum_squared_differences[1] += squared_difference[1]

    return sum_squared_differences[0]/len(data), sum_squared_differences[1] /len(data)

def compute_mean_and_variance_for_clusters(clusters):
    means = []
    variances = []
    for cluster in clusters:
        mean = compute_mean(cluster)
        means.append(mean)
        variances.append(compute_variance(cluster, mean))
    return means, variances


# Sample data points from figure
samples = []
```

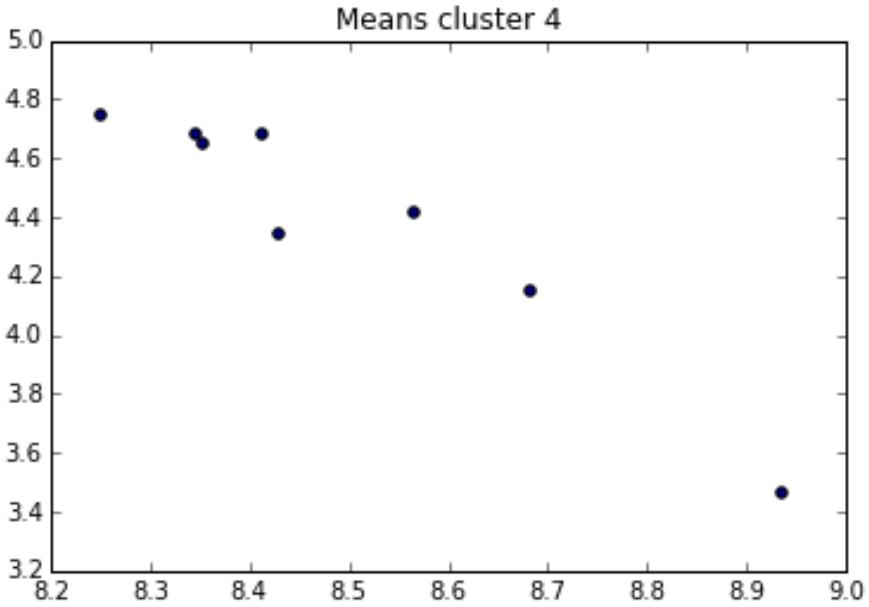

Figure 1.13: Means of cluster 5

```
for i in range(1000):
    samples.append(sample_figure(RADIUS, WIDTH, 0))
# Cluster data points using kmeans
X,Y = get_x_y_values(samples)
kmeans = KMeans(n_clusters = 6)
kmeans.fit(samples)
labels = kmeans.labels_
fig1 = plt.figure()
plt.title('vertical separation = 0')
ax1 = fig1.add_subplot(111)
ax1.scatter(X, Y, c=labels.astype(np.float))

# Sample data points for varying vertical separation
all_means = []
all_variances = []
for i in range(-6,2):
    samples = []
    for j in range(1000):
        samples.append(sample_figure(RADIUS, WIDTH, i))
    labels = kmeans.predict(samples)
    X,Y = get_x_y_values(samples)
    fig1 = plt.figure()
    plt.title('vertical separation = ' + str(i))
```

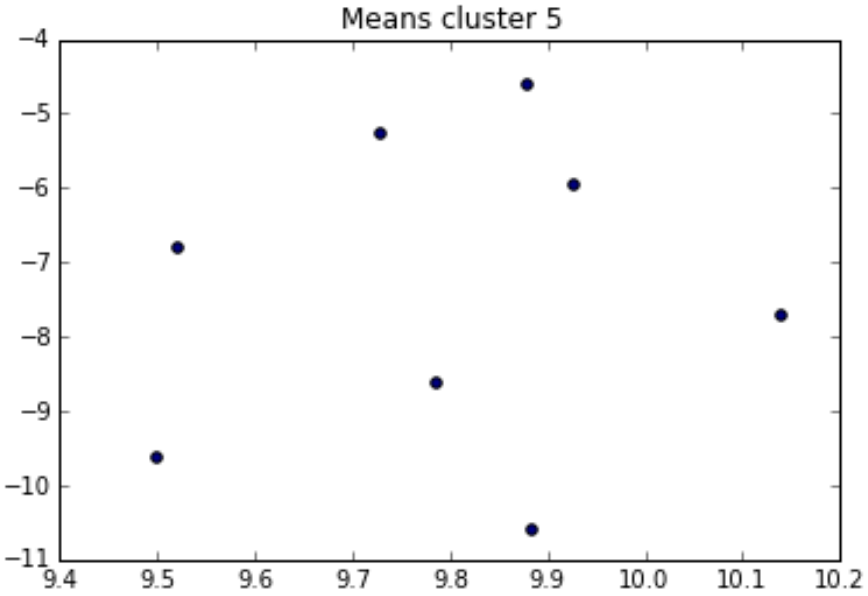

Figure 1.14: Means of cluster 6

```
    ax1 = fig1.add_subplot(111)
    ax1.scatter(X, Y, c=labels.astype(np.float))

    clusters = get_clusters(samples, labels)
    means, variances = compute_mean_and_variance_for_clusters(clusters)
    all_means.append(means)
    all_variances.append(variances)

# Print means
for i in range(6):
    fig1 = plt.figure()
    plt.title('Means cluster ' + str(i))
    ax1 = fig1.add_subplot(111)
    for mean in all_means:
        ax1.scatter(mean[i][0], mean[i][1], c= float(i) / 6.0 )

# Print variances
for i in range(6):
    fig1 = plt.figure()
    plt.title('Variances cluster ' + str(i))
    ax1 = fig1.add_subplot(111)
    for variance in all_variances:
        ax1.scatter(variance[i][0], variance[i][1], c= float(i) / 6.0 )
```