CSE 4020 - Machine Learning

Experiment 9

15BCE0329

Harshit Kedia

L19+L20

CODE:

```
import numpy as np
import pandas as pd
import random as rand
import matplotlib.pyplot as plt
from scipy.stats import norm
from sys import maxint

rand.seed(42)

mu1 = [0, 5]
sig1 = [ [2, 0], [0, 3] ]
mu2 = [5, 0]
sig2 = [ [4, 0], [0, 1] ]

x1, y1 = np.random.multivariate_normal(mu1, sig1, 100).T
x2, y2 = np.random.multivariate_normal(mu2, sig2, 100).T

xs = np.concatenate((x1, x2))
ys = np.concatenate((y1, y2))
```

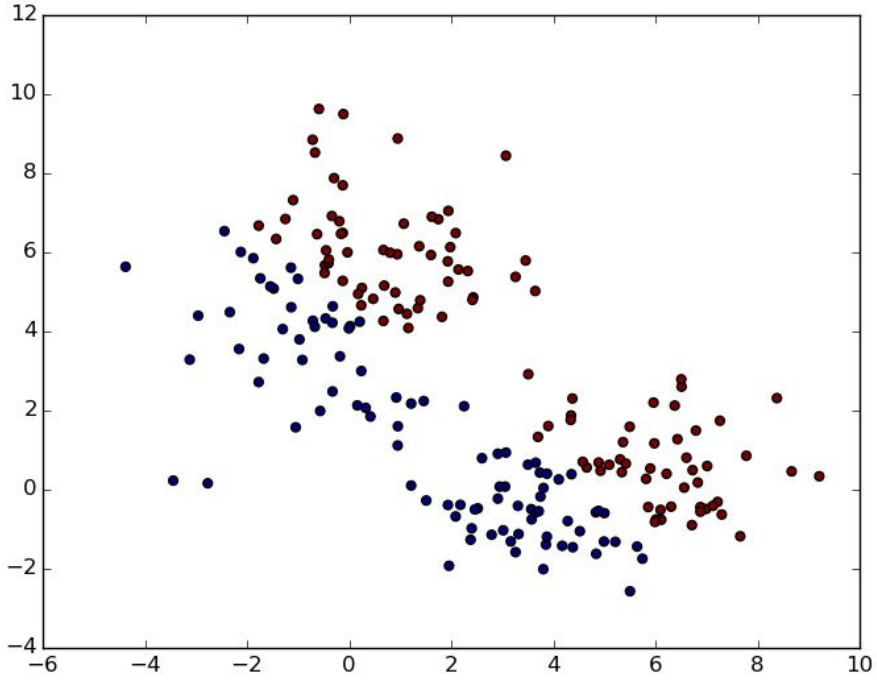
```
labels = ([1] * 100) + ([2] * 100)

data = {'x': xs, 'y': ys, 'label': labels}
df = pd.DataFrame(data=data)

df.head()
df.tail()

fig = plt.figure()
plt.scatter(data['x'], data['y'], 24, c=data['label'])
fig.savefig("true-values.png")

guess = { 'mu1': [1,1],
          'sig1': [ [1, 0], [0, 1] ],
          'mu2': [4,4],
          'sig2': [ [1, 0], [0, 1] ],
          'lambda': [0.4, 0.6]
        }

def prob(val, mu, sig, lam):
  p = lam
for i in range(len(val)):
 p *= norm.pdf(val[i], mu[i], sig[i][i])
return p

def expectation(dataFrame, parameters):
fori in range(dataFrame.shape[0]):
    x = dataFrame['x'][i]
```

```
    y = dataFrame['y'][i]
    p_cluster1 = prob([x, y], list(parameters['mu1']), list(parameters['sig1']), parameters['lambda'][0]
)
    p_cluster2 = prob([x, y], list(parameters['mu2']), list(parameters['sig2']), parameters['lambda'][1]
)
if p_cluster1 > p_cluster2:
 dataFrame['label'][i] = 1
else:
 dataFrame['label'][i] = 2
return dataFrame


def maximization(dataFrame, parameters):
  points_assigned_to_cluster1 = dataFrame[dataFrame['label'] == 1]
  points_assigned_to_cluster2 = dataFrame[dataFrame['label'] == 2]
  percent_assigned_to_cluster1 = len(points_assigned_to_cluster1) / float(len(dataFrame))
  percent_assigned_to_cluster2 = 1 - percent_assigned_to_cluster1
parameters['lambda'] = [percent_assigned_to_cluster1, percent_assigned_to_cluster2 ]
parameters['mu1'] = [points_assigned_to_cluster1['x'].mean(),
points_assigned_to_cluster1['y'].mean()]
parameters['mu2'] = [points_assigned_to_cluster2['x'].mean(),
points_assigned_to_cluster2['y'].mean()]
  parameters['sig1'] = [ [points_assigned_to_cluster1['x'].std(), 0 ], [ 0,
points_assigned_to_cluster1['y'].std() ] ]
  parameters['sig2'] = [ [points_assigned_to_cluster2['x'].std(), 0 ], [ 0,
points_assigned_to_cluster2['y'].std() ] ]
return parameters


def distance(old_params, new_params):
dist = 0
for param in ['mu1', 'mu2']:
for i in range(len(old_params)):
```

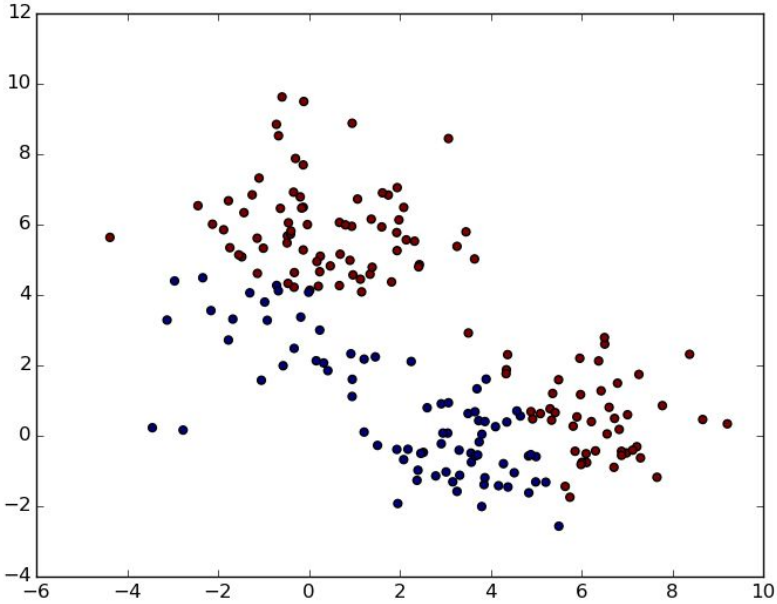
```
dist += (old_params[param][i] - new_params[param][i]) ** 2
return dist ** 0.5



shift = maxint
epsilon = 0.01
iters = 0
df_copy = df.copy()
df_copy['label'] = map(lambda x: x+1, np.random.choice(2, len(df)))
params = pd.DataFrame(guess)

while shift > epsilon:
iters += 1
updated_labels = expectation(df_copy.copy(), params)
updated_parameters = maximization(updated_labels, params.copy())

shift = distance(params, updated_parameters)
print("iteration {}, shift {}".format(iters, shift))
df_copy = updated_labels
params = updated_parameters

fig = plt.figure()
plt.scatter(df_copy['x'], df_copy['y'], 24, c=df_copy['label'])
fig.savefig("iteration{}.png".format(iters))
```

OUTPUT

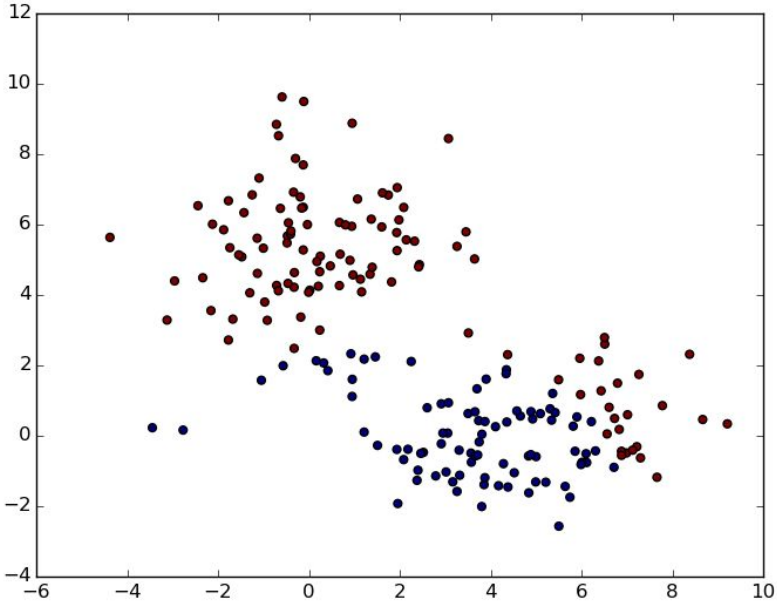
Iteration 1



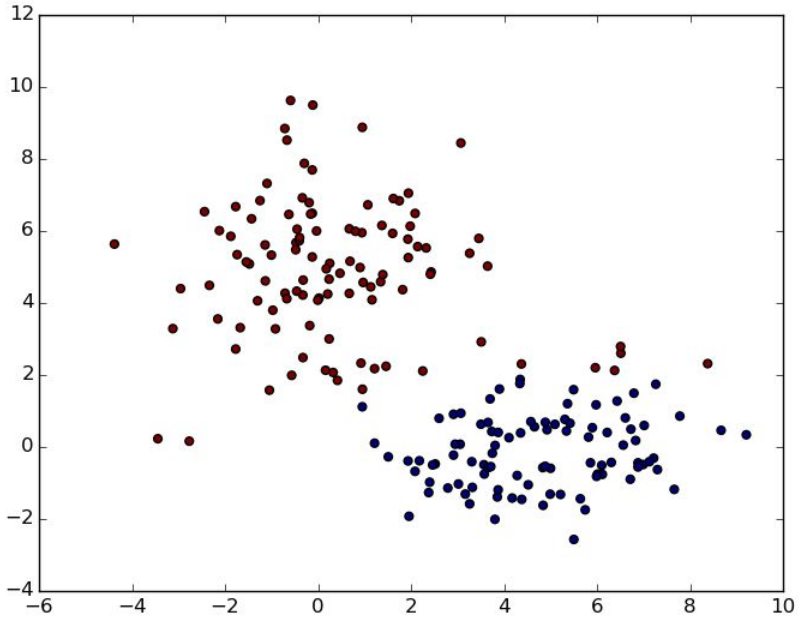
Iteration 2



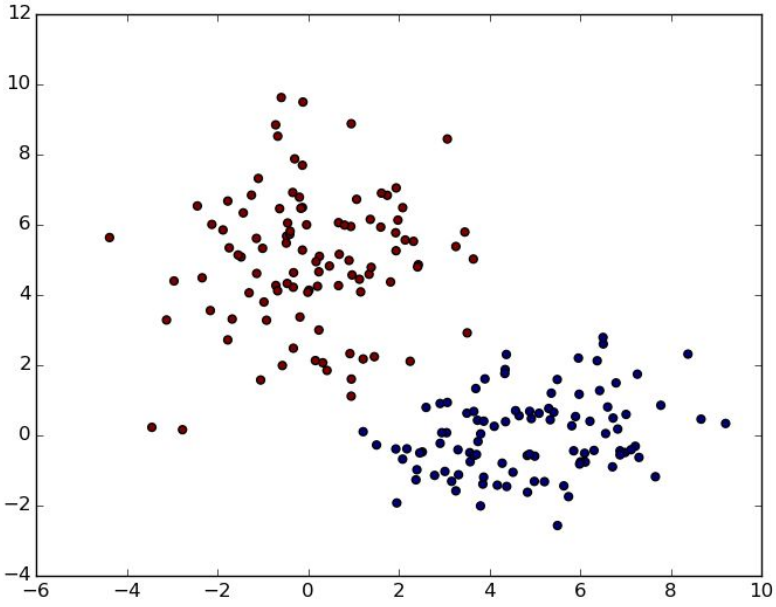
Iteration 3



Iteration 4



Iteration 5



Iteration 6