worksheet_08

February 27, 2024

1 Worksheet 08

Name: Daniyal Ahmed
UID: U11469883

1.0.1 Topics

- Soft Clustering
- Clustering Aggregation

1.0.2 Probability Review

Read through the following

1.0.3 Soft Clustering

We generate 10 data points that come from a normal distribution with mean 5 and variance 1.

[70]:
```
import random
import numpy as np
from sklearn.cluster import KMeans

mean = 5
stdev = 1

s1 = np.random.normal(mean, stdev, 10).tolist()
print(s1)
```

```
[4.7983801613986845, 6.228900940609035, 3.6463599753394185, 4.951899367216062,
5.225150676855521, 3.3464383495979786, 6.989679166006414, 4.85776652126135,
4.272780013358745, 5.238775424293421]
```

a) Generate 10 more data points, this time coming from a normal distribution with mean 8 and variance 1.

[71]:
```
s2 = np.random.normal(8,1,10 ).tolist()
print(s2)
```

```
[7.782111524577189, 10.127335677787418, 7.082620782622599, 7.862107429118189,
8.45001575362898, 7.612270721949519, 5.465169744832124, 8.241124885090787,
```

7.08190471941364, 6.693303150338769]

b) Flip a fair coin 10 times. If the coin lands on H, then pick the last data point of `s1` and remove it from `s1`, if T then pick the last data point from `s2` and remove it from `s2`. Add these 10 points to a list called `data`.

[72]:
```
data = []
for i in range(10):
    # flip coin
    coin_output = random.choice([0, 1])
    if coin_output == 0:
        p1 = s1.pop()
        data.append(p1)
    else:
        p2 = s2.pop()
        data.append(p2)
print(data)
```

```
[6.693303150338769, 7.08190471941364, 8.241124885090787, 5.238775424293421,
4.272780013358745, 4.85776652126135, 5.465169744832124, 6.989679166006414,
7.612270721949519, 8.45001575362898]
```

c) This `data` is a Gaussian Mixture Distribution with 2 mixture components. Over the next few questions we will walk through the GMM algorithm to see if we can uncover the parameters we used to generate this data. First, please list all these parameters of the GMM that created `data` and the values we know they have.

We know that this data follows Gaussian Mixture Distribution such that each data point itself is a normal distribution because of this we know that in normal distribution our parameters are mean and variance because of that we know for each P(X_i | S_j) our parameters are _j and _j

d) Let's assume there are two mixture components (note: we could plot the data and make the observation that there are two clusters). The EM algorithm asks us to start with a random `mean_j`, `variance_j`, `P(S_j)` for each component j. One method we could use to find sensible values for these is to apply K means with k=2 here.

1. the centroids would be the estimates of the `mean_j`
2. the intra-cluster variance could be the estimate of `variance_j`
3. the proportion of points in each cluster could be the estimate of `P(S_j)`

Go through this process and list the parameter estimates it gives. Are they close or far from the true values?

[73]:
```
kmeans = KMeans(2, init='k-means++').fit(X=np.array(data).reshape(-1, 1))

s1 = [x[0] for x in filter(lambda x: x[1] == 0, zip(data, kmeans.labels_))]
print(s1)
s2 = [x[0] for x in filter(lambda x: x[1] == 1, zip(data, kmeans.labels_))]
print(s2)

prob_s = [ len(s1) / (len(s1) + len(s2)) , len(s2) / (len(s1) + len(s2))]
```

```
mean = [ sum(s1)/len(s1) , sum(s2)/len(s2)]
var = [ sum(map(lambda x : (x - mean[0])**2, s1)) / len(s1) , sum(map(lambda x :
 ↪ (x - mean[1])**2, s2)) / len(s2)]

print("P(S_1) = " + str(prob_s[0]) + ",  P(S_2) = " + str(prob_s[1]))
print("mean_1 = " + str(mean[0]) + ",  mean_2 = " + str(mean[1]))
print("var_1 = " + str(var[0]) + ",  var_2 = " + str(var[1]))
```

```
[6.693303150338769, 7.08190471941364, 8.241124885090787, 6.989679166006414,
7.612270721949519, 8.45001575362898]
[5.238775424293421, 4.272780013358745, 4.85776652126135, 5.465169744832124]
P(S_1) = 0.6,  P(S_2) = 0.4
mean_1 = 7.511383066071352,  mean_2 = 4.95862292593641
var_1 = 0.4249356869669801,  var_2 = 0.20390690429150912
```

```
C:\Users\eggsc\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n
2kfra8p0\LocalCache\local-packages\Python310\site-
packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

Yes the mean is close to the true value since we have means of 5 and 8. Furthermore I would argue we are also pretty close to the true variances since Variance 1 is about .3 away from the true variance and Variance 2 is about .1 away from the true variance

e) For each data point, compute `P(S_j | X_i)`. Comment on which cluster you think each point belongs to based on the estimated probabilities. How does that compare to the truth?

[78]:
```
from scipy.stats import norm
from math import sqrt
prob_s0_x = [] # P(S_0 | X_i)
prob_s1_x = [] # P(S_1 | X_i)
prob_x = [] # P(X_i)

k = 2

for p in data:
    print("point = ", p)
    pdf_i = []

    for j in range(k):
        # P(X_i | S_j)
        pdf_i.append(norm.pdf(p, mean[j], sqrt(var[j])))
        print("probability of observing that point if it came from cluster " +␣
 ↪str(j) + " = ", pdf_i[j])
        # P(S_j) already computed
        prob_s[j]
```

```
    # P(X_i) = P(S_0)P(X_i | S_0) + P(S_1)P(X_i | S_1)
    prob_x = prob_s[0] * pdf_i[0] + prob_s[1] * pdf_i[1]


    #pdf_i[0] is p(p | s_0)

    # P(S_j | X_i) = P(X_i | S_j)P(S_j) / P(X_i)
    prob_s0_x.append(pdf_i[0] * prob_s[0] / prob_x)
    prob_s1_x.append(pdf_i[1]* prob_s[1] / prob_x)

probs = zip(data, prob_s0_x, prob_s1_x)

cluster_1 = []
cluster_2 = []
for p in probs:
    print(p[0])
    print("Probability of coming from S_1 = " + str(p[1]))
    print("Probability of coming from S_2 = " + str(p[2]))
    print()
    if(p[1] > p[2]):
        cluster_1.append(p[0])

    else:
        cluster_2.append(p[0])

print("points belonging to Cluster 1: ", cluster_1)
print("points belonging to Cluster 2: ", cluster_2)
```

```
point =  6.693303150338769
probability of observing that point if it came from cluster 0 =
0.28155834023538384
probability of observing that point if it came from cluster 1 =
0.0005453943033818891
point =  7.08190471941364
probability of observing that point if it came from cluster 0 =
0.4954473723033904
probability of observing that point if it came from cluster 1 =
1.3773550100874141e-05
point =  8.241124885090787
probability of observing that point if it came from cluster 0 =
0.3238221911905236
probability of observing that point if it came from cluster 1 =
2.8997890599845373e-12
point =  5.238775424293421
probability of observing that point if it came from cluster 0 =
0.0014490263088537214
probability of observing that point if it came from cluster 1 =
```

```
0.7274472397778581
point =  4.272780013358745
probability of observing that point if it came from cluster 0 =
2.795491599533792e-06
probability of observing that point if it came from cluster 1 =
0.2800581611303317
point =  4.85776652126135
probability of observing that point if it came from cluster 0 =
0.00016005227062484775
probability of observing that point if it came from cluster 1 =
0.862280489179299
point =  5.465169744832124
probability of observing that point if it came from cluster 0 =
0.004564018156172851
probability of observing that point if it came from cluster 1 =
0.46933326698178796
point =  6.989679166006414
probability of observing that point if it came from cluster 0 =
0.44742191255360625
probability of observing that point if it came from cluster 1 =
3.526326485517866e-05
point =  7.612270721949519
probability of observing that point if it came from cluster 0 =
0.6038354992562937
probability of observing that point if it came from cluster 1 =
2.7511244786624614e-08
point =  8.45001575362898
probability of observing that point if it came from cluster 0 =
0.21428748783580734
probability of observing that point if it came from cluster 1 =
9.013471239585647e-14
6.693303150338769
Probability of coming from S_1 = 0.9987102947473724
Probability of coming from S_2 = 0.0012897052526276926

7.08190471941364
Probability of coming from S_1 = 0.9999814668578962
Probability of coming from S_2 = 1.853314210371024e-05

8.241124885090787
Probability of coming from S_1 = 0.9999999999940301
Probability of coming from S_2 = 5.969920404582177e-12

5.238775424293421
Probability of coming from S_1 = 0.002978998851350189
Probability of coming from S_2 = 0.9970210011486498

4.272780013358745
```

```
Probability of coming from S_1 = 1.4972513563157548e-05
Probability of coming from S_2 = 0.9999850274864369

4.85776652126135
Probability of coming from S_1 = 0.00027834513744919655
Probability of coming from S_2 = 0.9997216548625508

5.465169744832124
Probability of coming from S_1 = 0.014376994794639613
Probability of coming from S_2 = 0.9856230052053604

6.989679166006414
Probability of coming from S_1 = 0.999947459864125
Probability of coming from S_2 = 5.254013587488603e-05

7.612270721949519
Probability of coming from S_1 = 0.9999999696261161
Probability of coming from S_2 = 3.037388381977668e-08

8.45001575362898
Probability of coming from S_1 = 0.9999999999997196
Probability of coming from S_2 = 2.8041678434314304e-13

points belonging to Cluster 1:  [6.693303150338769, 7.08190471941364,
8.241124885090787, 6.989679166006414, 7.612270721949519, 8.45001575362898]
points belonging to Cluster 2:  [5.238775424293421, 4.272780013358745,
4.85776652126135, 5.465169744832124]
```

"'Funny enough I believe it got all the points in the right clusters I don't think this is suppossed to happen at this stage in the worksheet but sometimes randomness may be a little helpful. :)"'

f) Having computed `P(S_j | X_i)`, update the estimates of `mean_j`, `var_j`, and `P(S_j)`. How different are these values from the original ones you got from K means? briefly comment.

[75]:
```
prob_c = [sum(prob_s0_x)/ len(prob_s0_x), sum(prob_s1_x)/ len(prob_s1_x)]
mean = [sum([x[0] * x[1] for x in zip(prob_s0_x, data)]) / sum(prob_s0_x),␣
 ↪sum([x[0] * x[1] for x in zip(prob_s1_x, data)]) / sum(prob_s1_x)]
var = [ sum(map(lambda x : (x - mean[0])**2, s1)) / len(s1) , sum(map(lambda x :
 ↪ (x - mean[1])**2, s2)) / len(s2)]


print("P(S_1) = " + str(prob_s[0]) + ",  P(S_2) = " + str(prob_s[1]))
print("mean_1 = " + str(mean[0]) + ",  mean_2 = " + str(mean[1]))
print("var_1 = " + str(var[0]) + ",  var_2 = " + str(var[1]))
```

```
P(S_1) = 0.6,  P(S_2) = 0.4
mean_1 = 7.505612492951281,  mean_2 = 4.95726925225798
var_1 = 0.4249689864811142,  var_2 = 0.20390873672393678
```

These variance are somewhat higher than the ones we calculated previously although they aren't higher by huge portion. The mean is somewhat the opposite, though it is very close to the values we already calculated they are somewhat lower but not by a lot.

g) Update `P(S_j | X_i)`. Comment on any differences or lack thereof you observe.

[76]:
```
from scipy.stats import norm
from math import sqrt
prob_s0_y = [] # P(S_0 | X_i)
prob_s1_y = [] # P(S_1 | X_i)
prob_x = [] # P(X_i)

k = 2

for p in data:
    print("point = ", p)
    pdf_i = []

    for j in range(k):
        # P(X_i | S_j)
        pdf_i.append(norm.pdf(p, mean[j], sqrt(var[j])))
        print("probability of observing that point if it came from cluster " +␣
 ↪str(j) + " = ", pdf_i[j])
        # P(S_j) already computed
        prob_c[j]

    # P(X_i) = P(S_0)P(X_i | S_0) + P(S_1)P(X_i | S_1)
    prob_x = prob_c[0] * pdf_i[0] + prob_c[1] * pdf_i[1]


    #pdf_i[0] is p(p | s_0)

    # P(S_j | X_i) = P(X_i | S_j)P(S_j) / P(X_i)
    prob_s0_y.append(pdf_i[0] * prob_c[0] / prob_x)
    prob_s1_y.append(pdf_i[1]* prob_c[1] / prob_x)

probs = zip(data, prob_s0_y, prob_s1_y)
for p in probs:
    print(p[0])
    print("Probability of coming from S_1 = " + str(p[1]))
    print("Probability of coming from S_2 = " + str(p[2]))
    print()
```

```
point =  6.693303150338769
probability of observing that point if it came from cluster 0 =
0.28155834023538384
```

```
probability of observing that point if it came from cluster 1 =
0.0005453943033818891
point =  7.08190471941364
probability of observing that point if it came from cluster 0 =
0.4954473723033904
probability of observing that point if it came from cluster 1 =
1.3773550100874141e-05
point =  8.241124885090787
probability of observing that point if it came from cluster 0 =
0.3238221911905236
probability of observing that point if it came from cluster 1 =
2.8997890599845373e-12
point =  5.238775424293421
probability of observing that point if it came from cluster 0 =
0.0014490263088537214
probability of observing that point if it came from cluster 1 =
0.7274472397778581
point =  4.272780013358745
probability of observing that point if it came from cluster 0 =
2.795491599533792e-06
probability of observing that point if it came from cluster 1 =
0.2800581611303317
point =  4.85776652126135
probability of observing that point if it came from cluster 0 =
0.00016005227062484775
probability of observing that point if it came from cluster 1 =
0.862280489179299
point =  5.465169744832124
probability of observing that point if it came from cluster 0 =
0.004564018156172851
probability of observing that point if it came from cluster 1 =
0.46933326698178796
point =  6.989679166006414
probability of observing that point if it came from cluster 0 =
0.44742191255360625
probability of observing that point if it came from cluster 1 =
3.526326485517866e-05
point =  7.612270721949519
probability of observing that point if it came from cluster 0 =
0.6038354992562937
probability of observing that point if it came from cluster 1 =
2.7511244786624614e-08
point =  8.45001575362898
probability of observing that point if it came from cluster 0 =
0.21428748783580734
probability of observing that point if it came from cluster 1 =
9.013471239585647e-14
6.693303150338769
```

```
Probability of coming from S_1 = 0.9987187048745743
Probability of coming from S_2 = 0.0012812951254257904

7.08190471941364
Probability of coming from S_1 = 0.9999815878647507
Probability of coming from S_2 = 1.8412135249326996e-05

8.241124885090787
Probability of coming from S_1 = 0.9999999999940691
Probability of coming from S_2 = 5.93094079787409e-12

5.238775424293421
Probability of coming from S_1 = 0.0029985188598842688
Probability of coming from S_2 = 0.9970014811401157

4.272780013358745
Probability of coming from S_1 = 1.5070915134558832e-05
Probability of coming from S_2 = 0.9999849290848654

4.85776652126135
Probability of coming from S_1 = 0.00028017397785234
Probability of coming from S_2 = 0.9997198260221477

5.465169744832124
Probability of coming from S_1 = 0.014470116683598322
Probability of coming from S_2 = 0.9855298833164018

6.989679166006414
Probability of coming from S_1 = 0.9999478028983357
Probability of coming from S_2 = 5.2197101664264e-05

7.612270721949519
Probability of coming from S_1 = 0.9999999698244375
Probability of coming from S_2 = 3.017556258104399e-08

8.45001575362898
Probability of coming from S_1 = 0.9999999999997213
Probability of coming from S_2 = 2.7858584938466113e-13
```

There is barely any difference between the two for example for s2 there is about a .05 difference in the values for total probability. The same thing goes for the individual probabilities as in there is any difference

h) Use `P(S_j | X_i)` to create a hard assignment - label each point as belonging to a specific cluster (0 or 1)

[77]:
```
cluster_1= []
cluster_2 = []


probs = zip(data, prob_s0_y, prob_s1_y)
for p in probs:
    if(p[1] > p[2]):
        cluster_1.append(p[0])

    else:
        cluster_2.append(p[0])

print("points belonging to Cluster 1: ", cluster_1)
print("points belonging to Cluster 2: ", cluster_2)
```

```
points belonging to Cluster 1:  [6.693303150338769, 7.08190471941364,
8.241124885090787, 6.989679166006414, 7.612270721949519, 8.45001575362898]
points belonging to Cluster 2:  [5.238775424293421, 4.272780013358745,
4.85776652126135, 5.465169744832124]
```

”‘’Once again it seems like the clusters are correct. I’m sure this time it had something to do with our algorithm and not just randomness.”’