Worksheet 08

Name: Xi Chen UID: U23766637

Topics

- Soft Clustering
- Clustering Aggregation

Probability Review

Read through the following

Soft Clustering

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

In [5]:
```
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)
```

```
[3.3135841172371245, 4.1691042729705945, 4.119105272712373, 6.3116319531846505, 4.47
7806784842295, 4.122060748847998, 3.045436351326088, 6.040815417225703, 5.4608658465
86146, 6.172934818572629]
```

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

In [6]:
```
mean = 8
stdev = 1
s2 = np.random.normal( mean  , stdev , 10 ).tolist()
print(s2)
```

```
[7.53368274717265, 10.163054160750784, 7.61820321307033, 8.200124515756551, 6.702033
3094577875, 7.060059709411738, 7.431116063463243, 8.960852712962737, 8.3648508420509
34, 9.014533834334113]
```

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`.

In [7]:
```
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)
```

```
[6.172934818572629, 5.460865846586146, 9.014533834334113, 6.040815417225703, 8.36485
0842050934, 8.960852712962737, 7.431116063463243, 7.060059709411738, 6.7020333094577
875, 3.045436351326088]
```

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.

- `mean_j` : the mean of the j-th component
- `variance_j` : the variance of the j-th component
- `P(S_j)` : the proportion of points in the j-th component
- `P(S_j | X_i)` : the probability of the i-th point belonging to the j-th component
- `P(X_i | S_j)` : the probability of the i-th point given the j-th component
- `P(S_j | X_i)` : the probability of the j-th component given the i-th point
- `P(X_i)` : the probability of the i-th point
- `k` : the number of components
- `n` : the number of points
- `data` : the data points
- `prob_s` : the proportion of points in each cluster
- `prob_s_j_x` : the probability of the j-th component given the i-th point
- `prob_x` : the probability of the i-th point
- `pdf_i` : the probability of the i-th point given the j-th component

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?

In [8]:

```
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

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]))
```

```
[9.014533834334113, 8.364850842050934, 8.960852712962737, 7.431116063463243]
[6.172934818572629, 5.460865846586146, 6.040815417225703, 7.060059709411738, 6.70203
33094577875, 3.045436351326088]
P(S_1) = 0.4,  P(S_2) = 0.6
mean_1 = 8.442838363202757,  mean_2 = 5.747024242096682
var_1 = 0.4062096608787438,  var_2 = 1.7140468816310719
```

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?

In [9]:

```
from scipy.stats import norm

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], var[j]))
        print("probability of observing that point if it came from cluster " + str(
        # 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]

    # 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)
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 =  6.172934818572629
probability of observing that point if it came from cluster 0 =  1.6276162053941987e
-07
probability of observing that point if it came from cluster 1 =  0.22567317147317184
point =  5.460865846586146
probability of observing that point if it came from cluster 0 =  1.9504413996103146e
-12
probability of observing that point if it came from cluster 1 =  0.22952767057492196
point =  9.014533834334113
probability of observing that point if it came from cluster 0 =  0.3647926429024162
probability of observing that point if it came from cluster 1 =  0.03782405352528412
point =  6.040815417225703
probability of observing that point if it came from cluster 0 =  2.5075642634074323e
-08
probability of observing that point if it came from cluster 1 =  0.22935482071284205
point =  8.364850842050934
probability of observing that point if it came from cluster 0 =  0.9641750157457174
probability of observing that point if it came from cluster 1 =  0.07250605515085728
point =  8.960852712962737
probability of observing that point if it came from cluster 0 =  0.4355395040986457
probability of observing that point if it came from cluster 1 =  0.04013133263520317
4
point =  7.431116063463243
probability of observing that point if it came from cluster 0 =  0.04417049941741691
5
probability of observing that point if it came from cluster 1 =  0.1436361095969605
point =  7.060059709411738
probability of observing that point if it came from cluster 0 =  0.00299142923379049
6
probability of observing that point if it came from cluster 1 =  0.1735642429448004
point =  6.7020333094577875
probability of observing that point if it came from cluster 0 =  0.00010096468193675
288
probability of observing that point if it came from cluster 1 =  0.1992862953001288
point =  3.045436351326088
probability of observing that point if it came from cluster 0 =  4.51542176049638e-3
9
probability of observing that point if it came from cluster 1 =  0.06721132048317956
6.172934818572629
Probability of coming from S_1 = 4.808178754500952e-07
Probability of coming from S_2 = 0.9999995191821245

5.460865846586146
Probability of coming from S_1 = 5.6650871903266705e-12
Probability of coming from S_2 = 0.9999999999943349

9.014533834334113
Probability of coming from S_1 = 0.8654039993798789
Probability of coming from S_2 = 0.1345960006201211

6.040815417225703
Probability of coming from S_1 = 7.288747547995184e-08
Probability of coming from S_2 = 0.9999999271125246

8.364850842050934
Probability of coming from S_1 = 0.8986339581516023
```

```
Probability of coming from S_2 = 0.10136604184839768

8.960852712962737
Probability of coming from S_1 = 0.8785705741209919
Probability of coming from S_2 = 0.12142942587900804

7.431116063463243
Probability of coming from S_1 = 0.17013213909929548
Probability of coming from S_2 = 0.8298678609007045

7.060059709411738
Probability of coming from S_1 = 0.01135966544078789
Probability of coming from S_2 = 0.9886403345592121

6.7020333094577875
Probability of coming from S_1 = 0.0003376401842750407
Probability of coming from S_2 = 0.9996623598157249

3.045436351326088
Probability of coming from S_1 = 4.478830577979216e-38
Probability of coming from S_2 = 1.0
```

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.

In [10]:

```python
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
var = [ sum([x[0] * (x[1] - mean[0])**2 for x in zip(prob_s0_x, data)]) / sum(prob_

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]))


# The difference is that the original ones are calculated from the centroids of the
```

```
P(S_1) = 0.4,  P(S_2) = 0.6
mean_1 = 8.6876141143737,  mean_2 = 6.092326997618799
var_1 = 0.1953412055222592,  var_2 = 2.111104579535503
```

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

In [11]:

```python
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], var[j]))
        print("probability of observing that point if it came from cluster " + str(
        # 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]

    # 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)
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()
# The probabilities are updated based on the new estimates of the parameters.
# differences are observed in the probabilities of the points belonging to the clus
# lack: the probabilities are not updated much, because the parameters are already
```

```
point =  6.172934818572629
probability of observing that point if it came from cluster 0 =  2.1099068068956695e
-36
probability of observing that point if it came from cluster 1 =  0.18883553969143
point =  5.460865846586146
probability of observing that point if it came from cluster 0 =  1.1454325428137703e
-59
probability of observing that point if it came from cluster 1 =  0.18070589402055928
point =  9.014533834334113
probability of observing that point if it came from cluster 0 =  0.5033995562895269
probability of observing that point if it came from cluster 1 =  0.07250026428694185
point =  6.040815417225703
probability of observing that point if it came from cluster 0 =  2.7770796607346498e
-40
probability of observing that point if it came from cluster 1 =  0.18891699724670902
point =  8.364850842050934
probability of observing that point if it came from cluster 0 =  0.521530695651218
probability of observing that point if it came from cluster 1 =  0.10587087868698168
point =  8.960852712962737
probability of observing that point if it came from cluster 0 =  0.7678059357932356
probability of observing that point if it came from cluster 1 =  0.07507326586612595
point =  7.431116063463243
probability of observing that point if it came from cluster 0 =  2.116861563639704e-
09
probability of observing that point if it came from cluster 1 =  0.15455077529379196
point =  7.060059709411738
probability of observing that point if it came from cluster 0 =  1.7210761865063765e
-15
probability of observing that point if it came from cluster 1 =  0.170126051668008
point =  6.7020333094577875
probability of observing that point if it came from cluster 0 =  7.487001736061535e-
23
probability of observing that point if it came from cluster 1 =  0.18125412938666616
point =  3.045436351326088
probability of observing that point if it came from cluster 0 =  1.4158956102575618e
-181
probability of observing that point if it came from cluster 1 =  0.0666925678562826
6.172934818572629
Probability of coming from S_1 = 7.448833732405103e-36
Probability of coming from S_2 = 1.0

5.460865846586146
Probability of coming from S_1 = 4.225770826945475e-59
Probability of coming from S_2 = 1.0

9.014533834334113
Probability of coming from S_1 = 0.822346802532277
Probability of coming from S_2 = 0.17765319746772304

6.040815417225703
Probability of coming from S_1 = 9.799999298485665e-40
Probability of coming from S_2 = 1.0

8.364850842050934
Probability of coming from S_1 = 0.7665769834113815
Probability of coming from S_2 = 0.23342301658861866
```

```
8.960852712962737
Probability of coming from S_1 = 0.8720946462175797
Probability of coming from S_2 = 0.12790535378242032

7.431116063463243
Probability of coming from S_1 = 9.13124522900254e-09
Probability of coming from S_2 = 0.9999999908687548

7.060059709411738
Probability of coming from S_1 = 6.744317599144874e-15
Probability of coming from S_2 = 0.9999999999999932

6.7020333094577875
Probability of coming from S_1 = 2.7537769801976573e-22
Probability of coming from S_2 = 1.0

3.045436351326088
Probability of coming from S_1 = 1.4153457231883362e-180
Probability of coming from S_2 = 1.0
```

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

In [12]:
```
labels = [0 if x[1] > x[2] else 1 for x in zip(data, prob_s0_x, prob_s1_x)]
print(labels)
```

```
[1, 1, 0, 1, 0, 0, 1, 1, 1, 1]
```

In [ ]: