Worksheet 08

Name: Jeong Yong Yang, Junho Son UID: U95912941, U64222022

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.

```python
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)
```

```
[4.31061139517483, 4.583571606979407, 3.922930492955666, 5.32087544662834, 4.706804917425918, 5.282289634353272, 3.4710783781531482, 5.07
```

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

```python
mean = 8
stdev = 1
s2 = np.random.normal( mean , stdev , 10 ).tolist()
print(s2)
```

```
[10.738940883258707, 8.434553897762486, 8.136873818772713, 6.87017272740713, 7.773071981188038, 8.187751318382452, 6.340975729049986, 7.8
```

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`.

```python
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)
```

```
[9.440283336122544, 8.786630939737213, 7.886084359813896, 6.340975729049986, 3.3906494685571773, 4.1491393354386785, 8.187751318382452, 7
```

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.

Probability

P(`s1`) = 0.5

P(`s2`) = 0.5

Variance

Var(`s1`) = 1

Var(`s2`) = 1

Mean

Avg(s1) = 5

Avg(s2) = 8

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?

```
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]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10
  warnings.warn(
[3.3906494685571773, 4.1491393354386785, 5.0790113424947325, 3.4710783781531482]
[9.440283336122544, 8.786630939737213, 7.886084359813896, 6.340975729049986, 8.187751318382452, 7.773071981188038]
P(S_1) = 0.4,  P(S_2) = 0.6
mean_1 = 4.022469631160934,  mean_2 = 8.069132944049022
var_1 = 0.45888865838402415,  var_2 = 0.9194355546505552
```

They are close from the true values.

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?

```
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(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]

    # 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 =  9.440283336122544
probability of observing that point if it came from cluster 0 =  4.68725506329445e-31
```

```
probability of observing that point if it came from cluster 1 =  0.1427122651064448
point =  8.786630939737213
probability of observing that point if it came from cluster 0 =  3.4197199416445984e-24
probability of observing that point if it came from cluster 1 =  0.32000129369143593
point =  7.886084359813896
probability of observing that point if it came from cluster 0 =  3.516091041984081e-16
probability of observing that point if it came from cluster 1 =  0.4253847506342204
point =  6.340975729049986
probability of observing that point if it came from cluster 0 =  2.48919459428751e-06
probability of observing that point if it came from cluster 1 =  0.07417241467457988
point =  3.3906494685571773
probability of observing that point if it came from cluster 0 =  0.33694122801200077
probability of observing that point if it came from cluster 1 =  1.035109613379593e-06
point =  4.1491393354386785
probability of observing that point if it came from cluster 0 =  0.8368681516672145
probability of observing that point if it came from cluster 1 =  4.9006032522650756e-05
point =  8.187751318382452
probability of observing that point if it came from cluster 0 =  1.1180414841448051e-18
probability of observing that point if it came from cluster 1 =  0.4303031674292418
point =  7.773071981188038
probability of observing that point if it came from cluster 0 =  2.712700866538803e-15
probability of observing that point if it came from cluster 1 =  0.4119776581037071
point =  5.0790113424947325
probability of observing that point if it came from cluster 0 =  0.061390756509752786
probability of observing that point if it came from cluster 1 =  0.002191630111291801
point =  3.4710783781531482
probability of observing that point if it came from cluster 0 =  0.42236406042086544
probability of observing that point if it came from cluster 1 =  1.6092991796249037e-06
9.440283336122544
Probability of coming from S_1 = 2.189606272826126e-30
Probability of coming from S_2 = 1.0

8.786630939737213
Probability of coming from S_1 = 7.12438774271976e-24
Probability of coming from S_2 = 1.0

7.886084359813896
Probability of coming from S_1 = 5.510448343907989e-16
Probability of coming from S_2 = 0.9999999999999993

6.340975729049986
Probability of coming from S_1 = 2.2372548389062752e-05
Probability of coming from S_2 = 0.999977627451611

3.3906494685571773
Probability of coming from S_1 = 0.9999953919047709
Probability of coming from S_2 = 4.608095228965517e-06

4.1491393354386785
Probability of coming from S_1 = 0.999912169447126
Probability of coming from S_2 = 8.783055287400747e-05

8.187751318382452
Probability of coming from S_1 = 1.7321763952677292e-18
Probability of coming from S_2 = 1.0
```

Based on the the probability,

1. Point 9.440283336122544 belongs to cluster 1.
2. Point 8.786630939737213 belongs to cluster 1.
3. Point 7.886084359813896 belongs to cluster 1.
4. Point 6.340975729049986 belongs to cluster 1.
5. Point 3.3906494685571773 belongs to cluster 0.
6. Point 4.1491393354386785 belongs to cluster 0.
7. Point 8.187751318382452 belongs to cluster 1.
8. Point 7.773071981188038 belongs to cluster 1.
9. Point 5.0790113424947325 belongs to cluster 0.
10. Point 3.4710783781531482 belongs to cluster 0.

In reality,

1. Point 9.440283336122544 belongs to cluster 1.
2. Point 8.786630939737213 belongs to cluster 1.
3. Point 7.886084359813896 belongs to cluster 1.
4. Point 6.340975729049986 belongs to cluster 1.
5. Point 3.3906494685571773 belongs to cluster 0.
6. Point 4.1491393354386785 belongs to cluster 0.
7. Point 8.187751318382452 belongs to cluster 1.
8. Point 7.773071981188038 belongs to cluster 1.
9. Point 5.0790113424947325 belongs to cluster 0.
10. Point 3.4710783781531482 belongs to cluster 0.

They match 100%.

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.

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

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) = 0.4,  P(S_2) = 0.6
mean_1 = 4.008883031771101,  mean_2 = 8.043957444697222
var_1 = 0.45028095445011657,  var_2 = 0.9864187325310322
```

They are very similar to the original values from K means. The probability is the same while the mean and variance is different by small amounts.

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

```
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(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]

    # 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 )


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

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]))
```

```
point =  9.440283336122544
probability of observing that point if it came from cluster 0 =  2.2541229388788283e-32
probability of observing that point if it came from cluster 1 =  0.1485020784827171
point =  8.786630939737213
probability of observing that point if it came from cluster 0 =  3.162083882599154e-25
probability of observing that point if it came from cluster 1 =  0.30461929057483844
point =  7.886084359813896
probability of observing that point if it came from cluster 0 =  7.038838942258483e-17
probability of observing that point if it came from cluster 1 =  0.39928825738539925
point =  6.340975729049986
probability of observing that point if it came from cluster 0 =  1.3263450954454118e-06
probability of observing that point if it came from cluster 1 =  0.0911232393966319
point =  3.3906494685571773
probability of observing that point if it came from cluster 0 =  0.34520638755003746
probability of observing that point if it came from cluster 1 =  5.950222267360344e-06
point =  4.1491393354386785
probability of observing that point if it came from cluster 0 =  0.8440303612864287
probability of observing that point if it came from cluster 1 =  0.00016652499534014278
point =  8.187751318382452
probability of observing that point if it came from cluster 0 =  1.7567789103709353e-19
probability of observing that point if it came from cluster 1 =  0.4001606540530076
point =  7.773071981188038
probability of observing that point if it came from cluster 0 =  5.920813583345314e-16
probability of observing that point if it came from cluster 1 =  0.389469033907699
```

```
point =  5.0790113424947325
probability of observing that point if it came from cluster 0 =  0.052595794107278895
probability of observing that point if it came from cluster 1 =  0.004415714748578017
point =  3.4710783781531482
probability of observing that point if it came from cluster 0 =  0.4341688766058164
probability of observing that point if it came from cluster 1 =  8.712351710014864e-06
P(S_1) = 0.4,  P(S_2) = 0.6
mean_1 = 3.992078688952148,  mean_2 = 8.014186732164532
var_1 = 0.4391001684793039,  var_2 = 1.0640856661027474
```

The probability, mean, and the variance values are similiar or the same as the previous iteration. One thing that it lacks is the mean value and the variance from the first cluster, which is ideally 5 and 1, respectively, but from the iteration, we continue to get values around 4 for the mean and 0.4 of the variance. This might be because of the values we got from the actual data and the fact that we only collected 10 total data.

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

```
probs = zip(data, prob_s0_x, prob_s1_x)
for p in probs:
    if p[1] > p[2]:
      print(f"Point {p[0]} belongs to cluster 0")
    else:
      print(f"Point {p[0]} belongs to cluster 1")


```

```
Point 9.440283336122544 belongs to cluster 1
Point 8.786630939737213 belongs to cluster 1
Point 7.886084359813896 belongs to cluster 1
Point 6.340975729049986 belongs to cluster 1
Point 3.3906494685571773 belongs to cluster 0
Point 4.1491393354386785 belongs to cluster 0
Point 8.187751318382452 belongs to cluster 1
Point 7.773071981188038 belongs to cluster 1
Point 5.0790113424947325 belongs to cluster 0
Point 3.4710783781531482 belongs to cluster 0
```