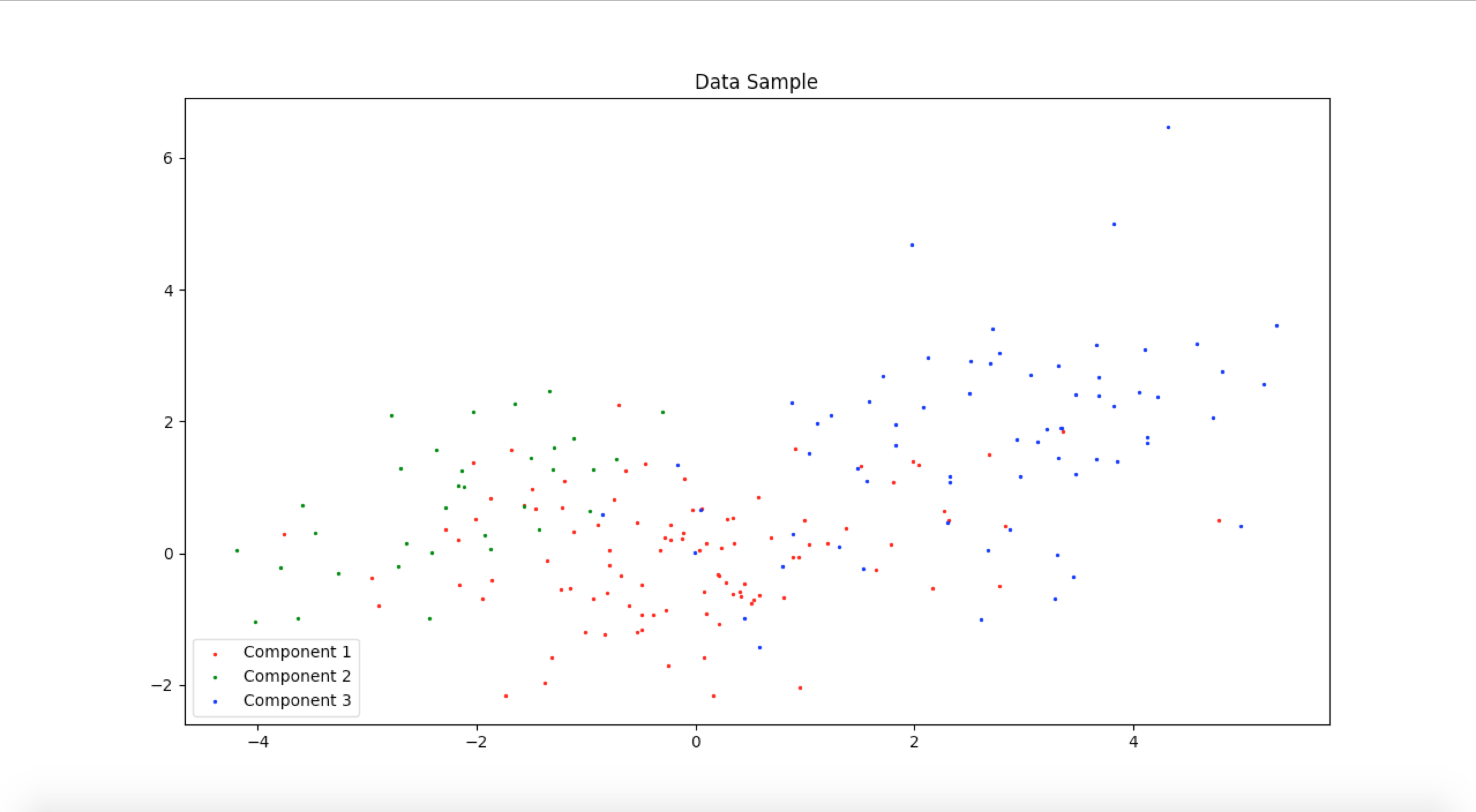
# HW3 Report

Name: Yan Kong

Data sample with three Gaussian components is shown as below, N = 200:



1) K-Means Algorithm

When K = 2:

|  |  |  |
| --- | --- | --- |
|  | k=1 | k=2 |
| l=1 | 0.16 | 0.84 |
| l=2 | 0.0 | 1.0 |
| l=3 | 0.88059701 | 0.11940299 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2

When K = 3:

|  |  |  |  |
| --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 |
| l=1 | 0.64 | 0.24 | 0.12 |
| l=2 | 0.0 | 1.0 | 0.0 |
| l=3 | 0.14925373 | 0.01492537 | 0.8358209 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3

When K = 4:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 | k=4 |
| l=1 | 0.02 | 0.24 | 0.18 | 0.56 |
| l=2 | 0.0 | 1.0 | 0.0 | 0.0 |
| l=3 | 0.53731343 | 0.01492537 | 0.35820896 | 0.08955224 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3, 4

When K = 5:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 | k=4 | k=5 |
| l=1 | 0.18 | 0.17 | 0.09 | 0.02 | 0.54 |
| l=2 | 0.0 | 0.63636364 | 0.36363636 | 0.0 | 0.0 |
| l=3 | 0.34328358 | 0.02985075 | 0.0 | 0.55223881 | 0.07462687 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3, 4, 5

From tables above, K-Means can effectively classify data points in each component and put them in different cluster.

When k = 2, this algorithm can clearly cluster component1 and component3, but cannot distinguish component1 and component2 well. It’s reasonable since component1 and component2 are pretty close to each other.

When k = 3, it can effectively classify data in each component.

When k becomes larger, it still can distinguish the majority of component, but more samples are divided into other cluster.

2) EM Algorithm

When K = 2:

|  |  |  |
| --- | --- | --- |
|  | k=1 | k=2 |
| l=1 | 0.87 | 0.13 |
| l=2 | 1.0 | 0.0 |
| l=3 | 0.14925373 | 0.85074627 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2

When K = 3:

|  |  |  |  |
| --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 |
| l=1 | 0.25 | 0.11 | 0.64 |
| l=2 | 1.0 | 0.0 | 0.0 |
| l=3 | 0.02985075 | 0.79104478 | 0.17910448 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3

When K = 4:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 | k=4 |
| l=1 | 0.6 | 0.15 | 0.08 | 0.17 |
| l=2 | 0.0 | 0.0 | 0.36363636 | 0.63636364 |
| l=3 | 0.10447761 | 0.86567164 | 0.0 | 0.02985075 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3, 4

When K = 5:

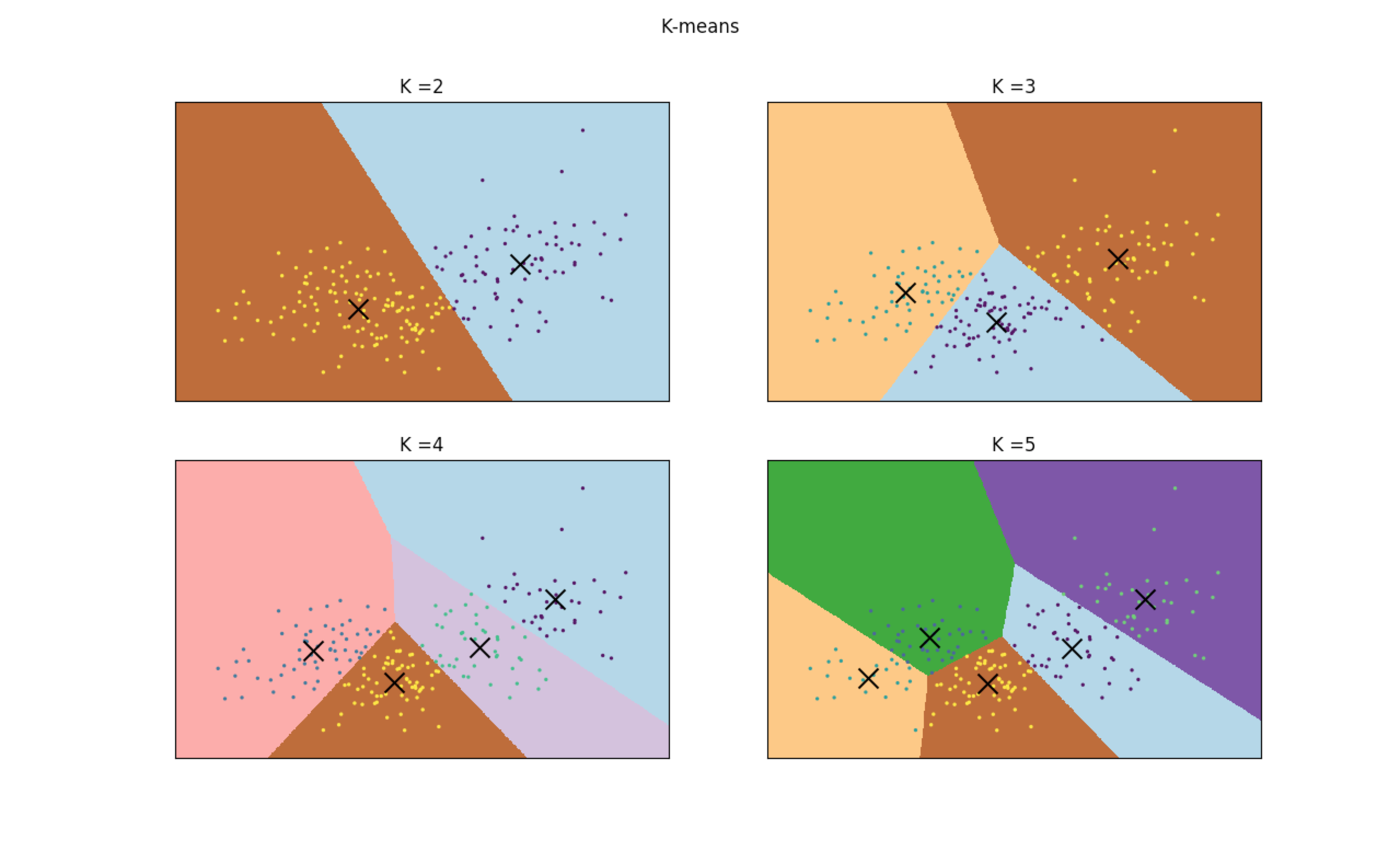
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 | k=4 | k=5 |
| l=1 | 0.05 | 0.6 | 0.2 | 0.09 | 0.06 |
| l=2 | 0.0 | 0.0 | 0.66666667 | 0.0 | 0.33333333 |
| l=3 | 0.29850746 | 0.11940299 | 0.02985075 | 0.55223881 | 0.0 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3, 4, 5

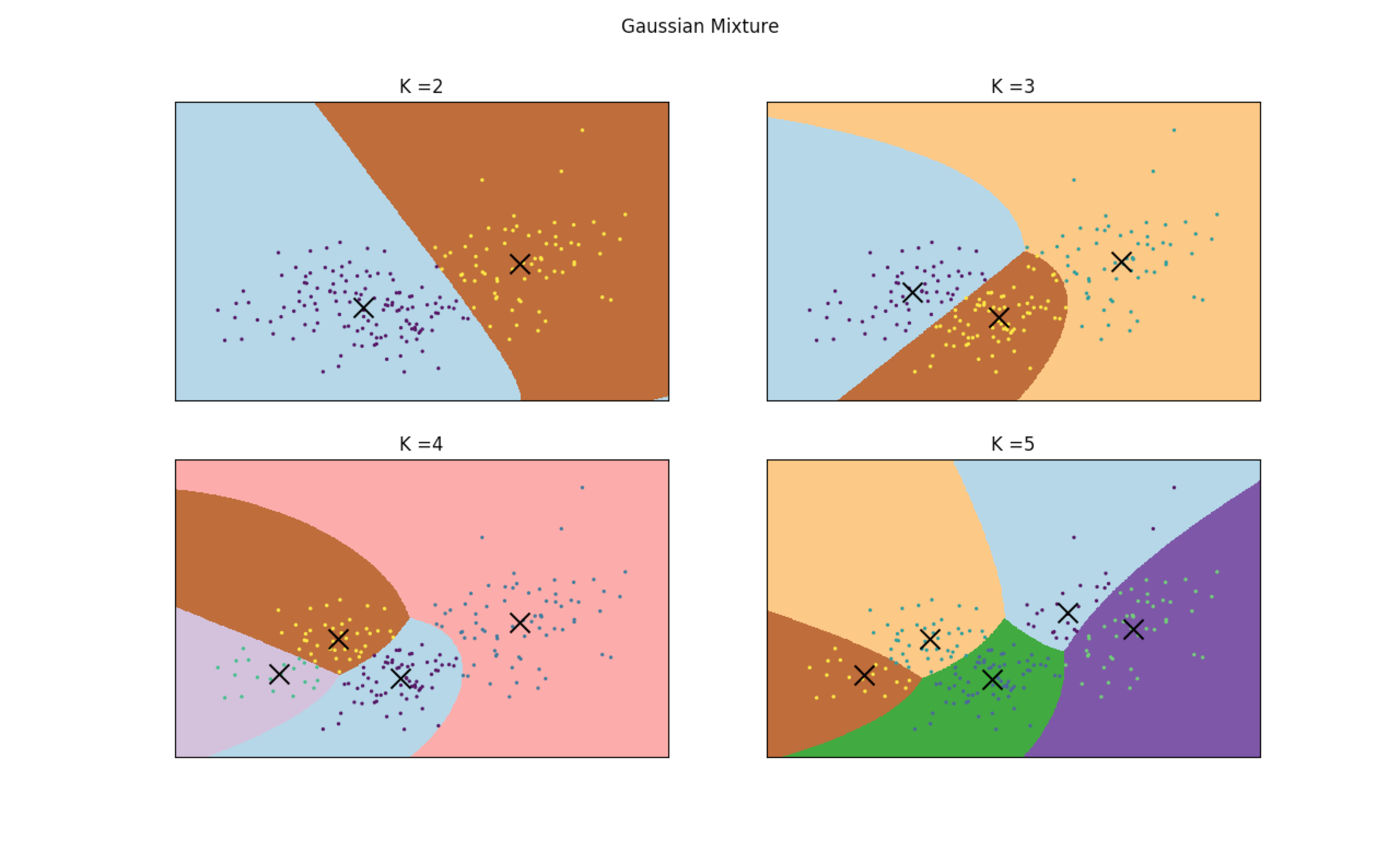
### 3)

I use diagram to represent how the mean and covariance define each cluster. Black crosses denote the mean for each cluster. Colored areas to represent each cluster’s area, which can show the direction of covariance.

For K-Means:

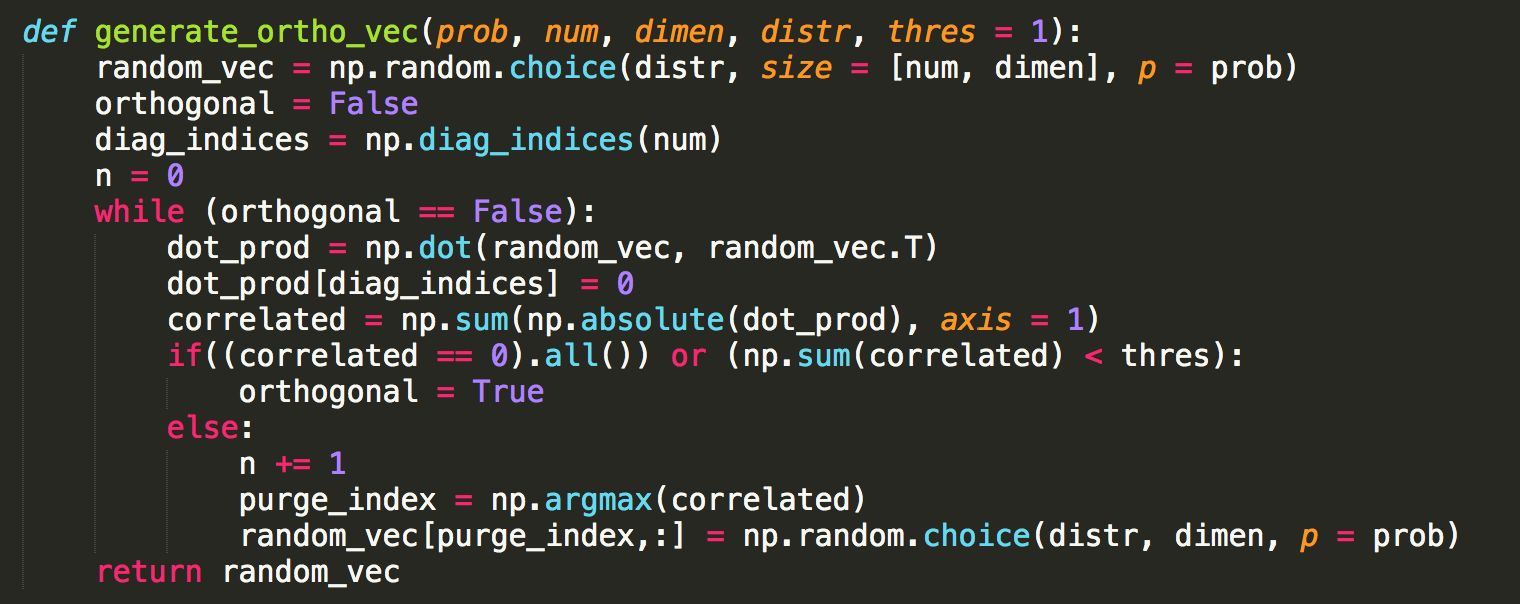


For GMM:



4) Generate Random Vector

The step is simple and straightforward. First generate random vectors, find the most correlative one, replace with a new one until it meets the threshold, which is quasi-orthogonal. Here is the function to generate quasi-orthogonal random vectors. distr = [0, -1, 1], prob = [2/3, 1/6, 1/6], dimen = 30, u = 7.



As a result, it generates a vector set in one run like this:

[[ 0 0 0 0 1 0 -1 0 0 0 0 1 0 0 0 0 0 0 1 -1 0 0 0 0 1 0 0 0 0 1]

[ 1 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 -1 0]

[ 0 0 1 1 0 0 1 0 0 1 0 1 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 -1 0]

[ 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 1 0 0 0 0 0 0 0 0]

[-1 0 -1 1 0 0 0 -1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 -1 0 0 0 0]

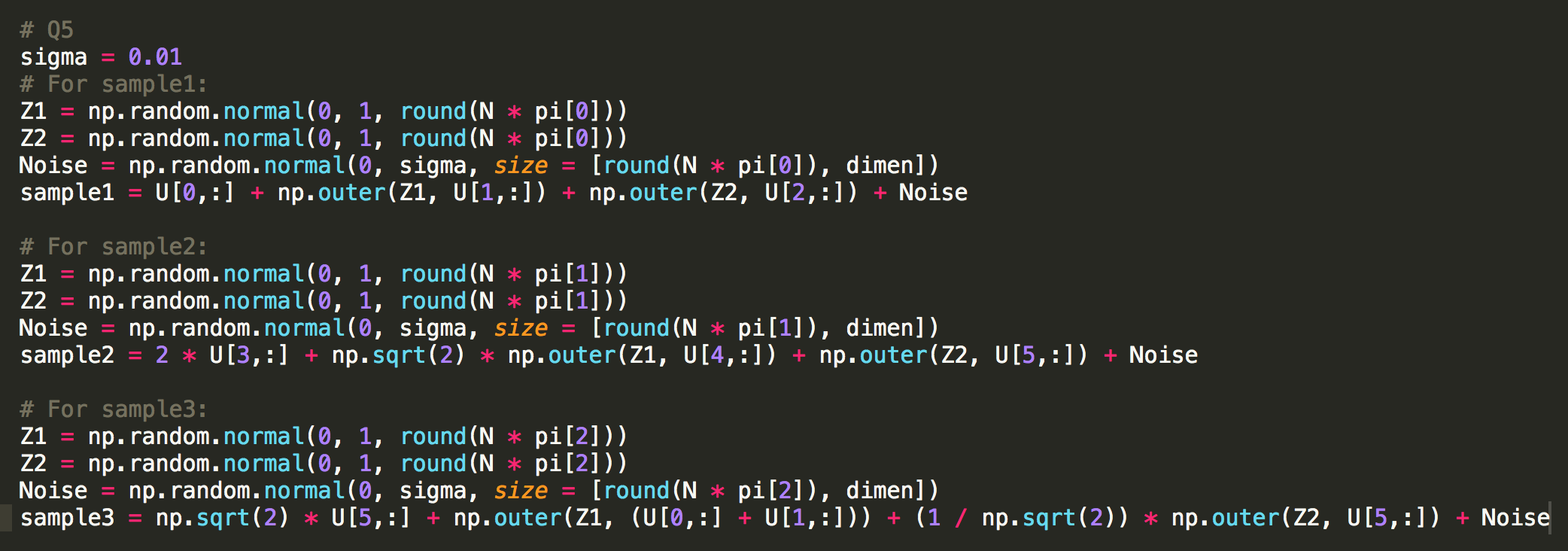
[ 0 0 0 0 0 0 1 0 0 -1 0 1 0 0 0 0 0 0 0 0 1 0 0 -1 0 0 0 0 0 0]

[ 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0]]

This vector set is obviously orthogonal.

5) Generate Data Sample with High-Dimension

This step is to generate one-dimension Gaussian sample as well as noise, then combine them with orthogonal vectors as this question requires. The code is shown as below.



6)K-Means in Higher Dimension

Here I use N = 200, dimension = 30.

When K = 2:

|  |  |  |
| --- | --- | --- |
|  | k=1 | k=2 |
| l=1 | 0.0 | 1.0 |
| l=2 | 0.27272727 | 0.72727273 |
| l=3 | 0.62686567 | 0.37313433 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2

When K = 3:

|  |  |  |  |
| --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 |
| l=1 | 0.75 | 0.0 | 0.25 |
| l=2 | 1.0 | 0.0 | 0.0 |
| l=3 | 0.02985075 | 0.79104478 | 0.17910448 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3

When K = 4:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 | k=4 |
| l=1 | 0.6 | 0.15 | 0.08 | 0.17 |
| l=2 | 0.0 | 0.0 | 0.36363636 | 0.63636364 |
| l=3 | 0.10447761 | 0.86567164 | 0.0 | 0.02985075 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3, 4

When K = 5:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | k=1 | k=2 | k=3 | k=4 | k=5 |
| l=1 | 0.05 | 0.6 | 0.2 | 0.09 | 0.06 |
| l=2 | 0.0 | 0.0 | 0.66666667 | 0.0 | 0.33333333 |
| l=3 | 0.29850746 | 0.11940299 | 0.02985075 | 0.55223881 | 0.0 |

Empirical probabilities P[ai [k] = 1|zi[l] = 1], l = 1, 2, 3, k = 1, 2, 3, 4, 5

7)

8)

Appendix