Problem 1: Gaussian Mixtures vs. k-means

1. Please take source file 1.1.py for reference.  
   The mean and variance of the value of k-mean loss function obtained for each k:

12 mean: 1.5135364920150756 variance: 0.07039856121642735

18 mean: 1.1773336266985777 variance: 0.14341386651063143

24 mean: 0.96639020451292 variance: 0.16108154181311818

36 mean: 0.8551377348161255 variance: 0.1651459573614382

42 mean: 0.6202332332648471 variance: 0.15840263914372837

1. Please take source file 1.2.py for reference.  
   The mean and variance of the value of k-mean loss function obtained for each k:

12 mean: 25636.195945553805 variance: 62242.01030089415

18 mean: 25863.714977174684 variance: 21645.990028544195

24 mean: 25957.138904054158 variance: 31032.079093634235

36 mean: 26112.20898890987 variance: 4986.656952573681

42 mean: 26151.29832605709 variance: 15058.916593071222

1. Please take source file 1.31.py for reference.

Please take source file 1.32.py for reference.  
True labels, when k=36.

[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], [12, 13, 14, 15, 16, 17, 18, 19, 20, 21], [22, 23, 24, 25, 26, 27, 28, 29, 30, 31], [32, 33, 34, 35, 36, 37, 38, 39], [40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51], [52, 53, 54, 55, 56, 57, 58, 59], [60, 61, 62, 63, 64, 65, 66, 67, 68, 69], [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80], [81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94], [95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107], [108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123], [124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135], [136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148], [149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160], [161, 162, 163, 164, 165, 166, 167, 168, 169, 170], [], [], [], [], [], [], [171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182], [183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193], [194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206], [207, 208, 209, 210, 211, 212, 213, 214, 215], [216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227], [228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238], [239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250], [251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262], [263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274], [275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285], [286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296], [297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307], [308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318], [319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329], [330, 331, 332, 333, 334, 335, 336, 337, 338, 339]]

Predicted labels, for k-means.

[[52, 110, 112, 114, 118], [204, 222, 226], [32, 39, 140, 141, 143, 146, 199, 306], [1, 2, 3, 5, 6, 7, 10, 11, 139, 201, 206, 227, 254, 296, 300], [81, 89, 90, 97, 98, 101, 102, 103, 105, 106, 263, 267, 269, 272, 273], [24, 26, 28, 30, 35, 82, 85, 86, 87, 92, 94, 266, 268, 270], [22, 148, 208, 301, 303, 305], [33, 36, 83, 93, 142, 197, 203, 218, 221, 225, 228, 229], [53, 57, 109, 111, 113, 115, 116, 117, 119, 120, 122, 339], [14, 15, 17, 19, 47, 153, 172, 174, 177, 224, 286, 288, 290, 291], [21, 66, 135, 157, 234, 242, 302, 326], [40, 42, 43, 44, 45, 46, 48, 49, 50, 51, 62], [126, 134, 158, 159, 249], [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80], [147, 215, 298, 307], [207, 209, 210, 211, 212, 213, 214], [107, 264, 265, 271], [84, 95, 96, 99, 100, 104, 162, 163, 164], [58, 165, 185, 187, 188, 330], [27, 38, 202, 205, 219, 220], [186, 191, 192, 193], [131, 244, 247, 319], [41, 60, 61, 63, 64, 65, 68, 69, 91], [0, 37, 145, 198, 216, 230, 231, 232, 233, 235, 304], [9, 18, 67, 179, 180, 217, 287, 289, 292, 293, 294], [277, 280, 282, 283], [54, 55, 56, 59, 108, 121, 123, 338], [161, 166, 167, 168, 169, 170, 331, 332, 333, 334, 335, 336, 337], [275, 276, 278, 279, 281, 284, 285, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318], [128, 129, 133, 155, 156, 171, 176, 239, 240, 248, 250, 325], [124, 125, 127, 132, 149, 150, 151, 152, 154, 160, 241, 243, 245, 246, 320, 321, 322, 323, 324, 327, 328], [12, 13, 16, 20, 130, 173, 178, 182, 329], [23, 25, 29, 31, 88, 183, 184, 189, 190, 195, 196, 200, 223, 238, 274], [4, 8, 251, 252, 253, 255, 256, 257, 258, 259, 260, 261, 262, 295], [175, 181], [34, 136, 137, 138, 144, 194, 236, 237, 297, 299]]

Predicted labels, for Gaussian mixture model.

[[112, 181, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318], [], [], [], [], [], [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 156, 172, 287, 289, 290, 329], [], [], [], [], [], [], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 30, 31, 32, 33, 34, 35, 36, 37, 39, 41, 42, 43, 44, 46, 47, 48, 49, 51, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 84, 89, 91, 95, 96, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 157, 158, 159, 160, 162, 163, 164, 166, 167, 168, 169, 170, 178, 179, 182, 194, 199, 201, 202, 203, 204, 207, 208, 210, 211, 212, 213, 214, 215, 216, 217, 218, 221, 222, 224, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 269, 271, 273, 286, 288, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 332, 333, 334, 335, 336, 337], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [175, 176], [], [], [], [], [22, 26, 27, 28, 29, 38, 40, 45, 50, 52, 53, 54, 55, 56, 57, 58, 59, 81, 82, 83, 85, 86, 87, 88, 90, 92, 93, 94, 97, 108, 109, 110, 111, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 161, 165, 171, 173, 174, 177, 180, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 195, 196, 197, 198, 200, 205, 206, 209, 219, 220, 223, 225, 248, 268, 270, 272, 274, 330, 331, 338, 339], [], []]

So from above observation, we can see the k-mean perform much more separately than GMM,. K-mean also has a higher accuracy than GMM, So I prefer k-mean better.

1. Please take source file 1.41.py for reference.

Please take source file 1.42.py for reference.

Yes. This procedure result in an improvement in both k-mean and GMM. Firstly, it will avoid some clusters contain no values. Secondly, it has more reasonable initial value which needs less rounds to converge to result. Thirdly, it need less computation or consideration how to select initial value.

1. Define the diagonal matrix M by M[k,k] = variance(Data[ : , k ]). It means calculate each attributes variance and set it as one diagonal matrix element. So, we get a n\*n positive diagonal matrix. For this special case, we calculate each of 14 attributes’ variance and set them to diagonal matrix.

Modify EM algorithm from

𝛴MLE = 1/(N−1) \* 𝛴 (𝑥(𝑖) − 𝜇MLE)\*(𝑥(𝑖) − 𝜇MLE)T

To

𝛴MLE = [σ2 1, 0, 0, ……

0, σ2 2, 0, ……

……

…… 0, 0, σ2 n]

Problem 2: Logistic Regression

1. Please take source code 2.1.py for reference.

w: [ -0.05682072 -1.15888705 -3.53976016 1.00146392 1.00001001 1.00098653

1.00091525 1.00295884 1.00986217 1.09747883 1.00528497 1.00598831

1.00773614 1.01585594 1.00816069 -0.88929404 1.01817415 1.00897506

1.60151307 1.03676384 1.1658984 1.04859023]

b: 1.4599628106759468

The accuracy of on the testing dataset is **0.7796610169491526**

Our objective function is the maximum probability function: ℓ(𝑤,𝑏) = ∑ ln 𝑝(𝑦|𝑥,𝑤,𝑏).

It’s object is trying find w and b that gives the result most highest probability for prediction. Take derivation for this function, we cannot find a closed form solution. It means we cannot reach a specific maximum value. So our result will wander around the result.

1. Please take source code 2.2.py for reference.

Select λ as 0.1

Weight: [-0.20020617 -1.17950683 -3.6317561 0.09988819 0.09845176 0.09942027 0.09934795 0.101376 0.10821101 0.19508726 0.10367768 0.10437434 0.10610387 0.11415005 0.10648949 -1.77381102 0.11585484 0.10704305 0.68897318 0.1347506 0.26420778 0.14625264]

Bias: 1.4594142653541857

The accuracy of on the testing dataset is **0.7796610169491526**

1. Please take source code 2.3.py for reference.

Select λ as 0.1

Weight: [-2.19284616e-01 -1.18354672e+00 -3.64712252e+00 1.44456424e-03 9.71071448e-06 9.77540974e-04 9.05095601e-04 2.93186872e-03 9.75980980e-03 9.65672978e-02 5.23081062e-03 5.92669506e-03

7.65450868e-03 1.56934927e-02 8.03654581e-03 -1.87069059e+00

1.73417991e-02 8.56098514e-03 5.89430611e-01 3.62593640e-02

1.65747911e-01 4.77317342e-02]

Bias: 1.4656145826293663

The accuracy of on the testing dataset is **0.7796610169491526**

1. Yes. Compared with weight vector in 1, the l1 and l2 loss function tends to be sparser in values.

For maximum probability function ℓ2(𝑤,𝑏) = ∑ ln 𝑝(𝑦|𝑥,𝑤,𝑏) – λ/2(||w||2 + b2) and ℓ1(𝑤,𝑏) = ∑ ln 𝑝(𝑦|𝑥,𝑤,𝑏) – λ(||w|| + b), after introduced an additional penalty value, it required the ||w|| to be smaller to get larger summary of probability. So, the weight vectors tend to be sparser for smaller ||w||.