Comparison with GMM

*Sangyeon Kim

1. Aim

The aim of the research is to compare QGMM with GMM to find out the differences between them with the different experiments.

Also, we'll check the effects of the augmented Lagrange method to make the constrained optimization flexible.

2. Dataset

In this research, we generated a data set "faithful.csv" that has 2 clusters. There are 273 observations in the data set.

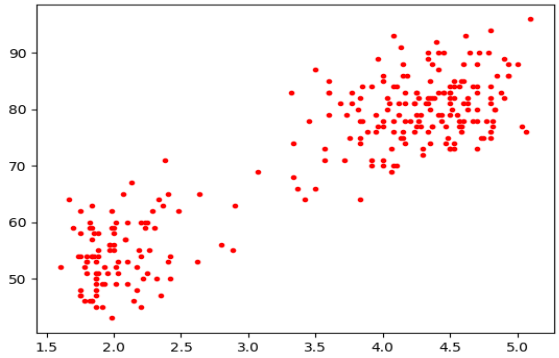

Figure 1. faithful data set that has 2 clusters, bottom-left and top-right and 2 dimensionality.

3. Experiments

In these experiments, the maximum iteration of training process is 15,000 and the tolerance is 1e-5. we used Adam optimizer with 0.01 learning rate (η). Also, as the training parameters, all the initial sequence of alpha (α_k) is [0.5, 0.5], the means are randomly or deliberately generated for check, and all the initial covariances (C_k) are

$$C_k = \begin{pmatrix} 0.08 & 0.1 \\ 0.1 & 3.3 \end{pmatrix}$$

Besides, we'll use the augmented Lagrangian method for the constrained optimization.

$$O(\theta) = f(\theta) - \sum_{i \in I} \lambda_i c_i(\theta) + \frac{1}{2\mu} \sum_{i \in I} c_i^2(\theta)$$

where $f(\theta)$: negative log-likelihood, c_i: approximation constraint, I=1.

In this experiment, we'll use the initial λ_i= 0, μ= 1 and every 1,000 iterations, λ and μ are updated.

$$\lambda^{k+1} = \lambda^k - \frac{c_i(\theta)}{\mu_k}$$
$$\mu^{k+1} = 0.7\mu^k$$

We also set the bound on λ_i and μ to prevent them to be overflowed.

$$-10{,}000{,}000 \le \lambda_i \le 0$$
$$0.0000001 \le \mu \le 1$$

3.1. Sensitivity to initial parameters

EM algorithm is known to be sensitive to the initial parameters. Among them, the means of the distributions represent the position of the distributions. Therefore, in this experiment, we'll figure out how sensitive QGMM and GMM are to the initial means.

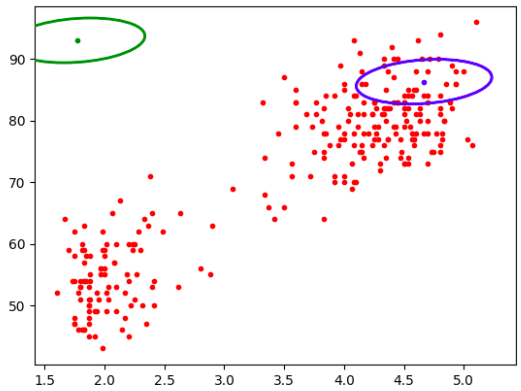

Figure 2. Initial clusters of the test case 48. The initial means are [4.6679220817969584, 86.34583189023205], [1.772306133903096, 93.05731536702366].

*This research was performed during Google Summer of Code 2019 for mlpack mentored by Sumedh Ghaisas.

In *Figure 2*, we can see the two initial distributions were positioned in a way that the blue distribution included more observations than the green distribution. In addition, the green distribution hadn't any observation.

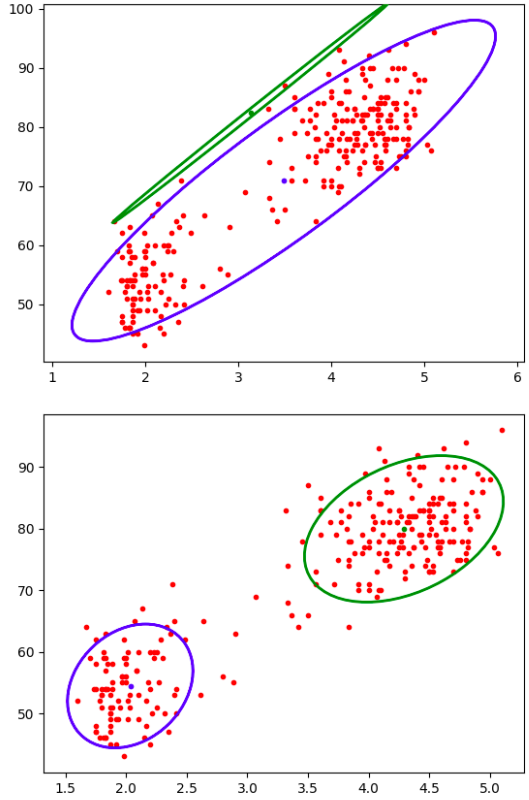

Figure 3. Training results of GMM (up) and QGMM (down).

From the *Figure 3*, we can figure out that GMM couldn't be trained well with the initial means, while QGMM could properly.

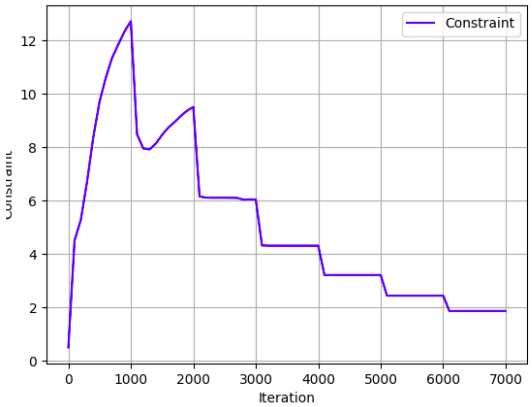

Figure 4. Constraint of QGMM using augmented Lagrange method.

From *Figure 4*, we can find out the constraint is getting stronger while training, and the training process ended before the constraint reaches 0.

As μ became smaller and smaller, lambda was updated according to the constraint and μ, allowing more flexible optimization.

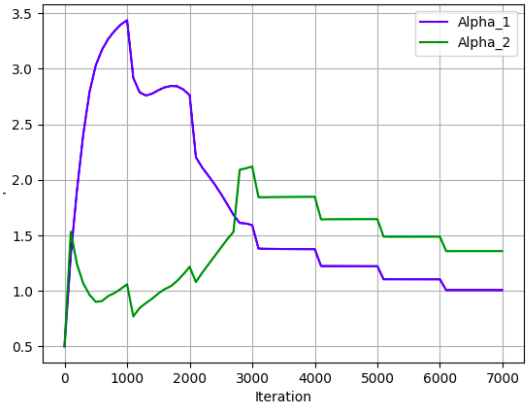

Figure 5. Alpha values of QGMM.

From *Figure 5*, we can see that the stronger the constraint become, the more constrained the the values of alpha are.

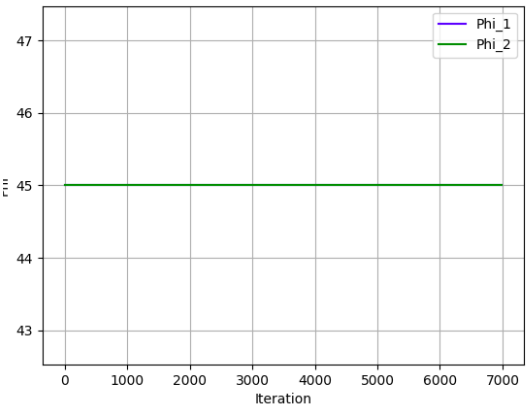

Figure 6. Phi of QGMM.

From the *Figure 6*, we can find out that the values of phi haven't been changed from the initial values, 45 and 45.

In the previous research, we figured out that when the difference of phi reaches 0, the two clusters tend to be away from each other.

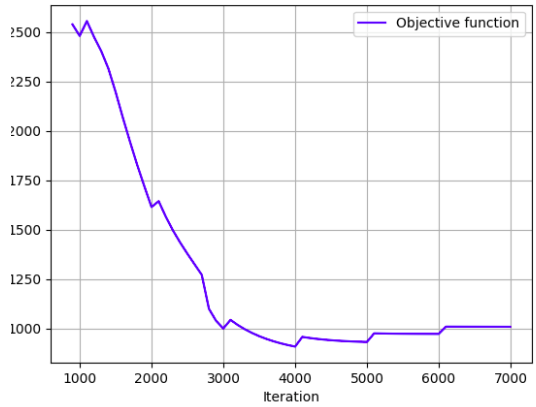

Figure 7. Objective function of QGMM.

As the constraint was decreased, NLL was increased. As a result, the variation was reflected on the objective.

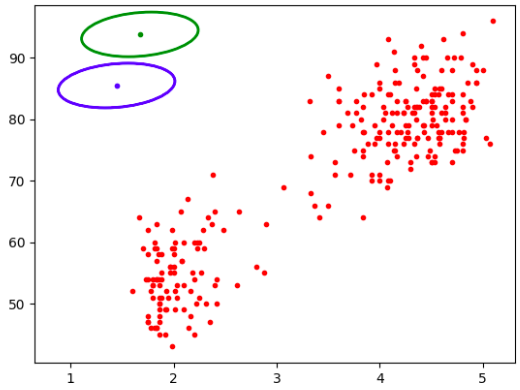

Figure 8. Initial clusters of the test case 8.

In the *Figure 8*, the two clusters were positioned initially without having observations.

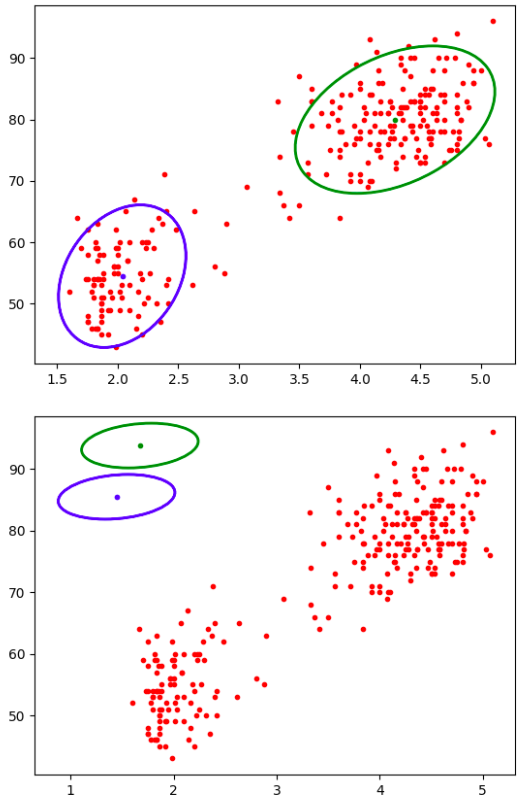

Figure 9. Training results of GMM (up) and QGMM (down).

Likewise, the training results were different totally. From the *Figure 9*, we can see the initial means have an effect on the training performance. In this time, GMM could train the distributions using the initial parameters, while QGMM couldn't.

Therefore, we tried to analyze result and figure out why QGMM can't be trained well to find possible improvements.

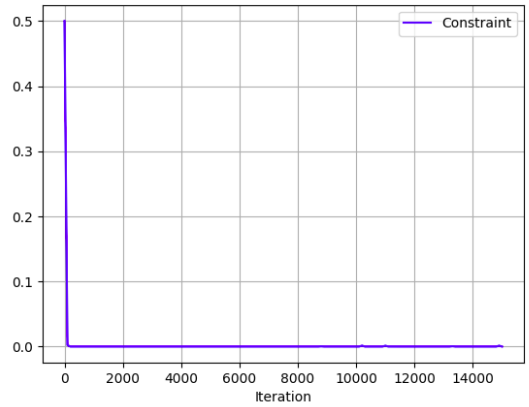

Figure 10. Constraint of QGMM.

From *Figure 10*, we can check the constraint has been decreased to zero.

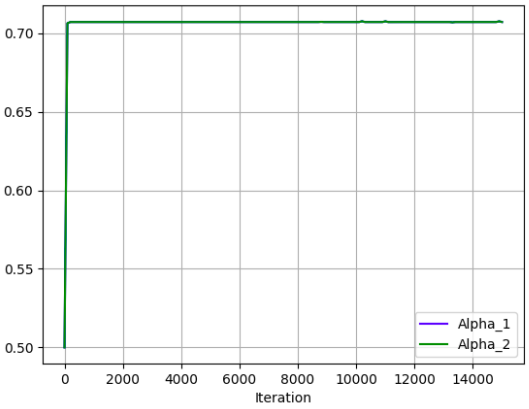

Figure 11. Alpha of QGMM.

From the *Figure 11*, we can see the values of alpha have been increased a bit from 0.5.

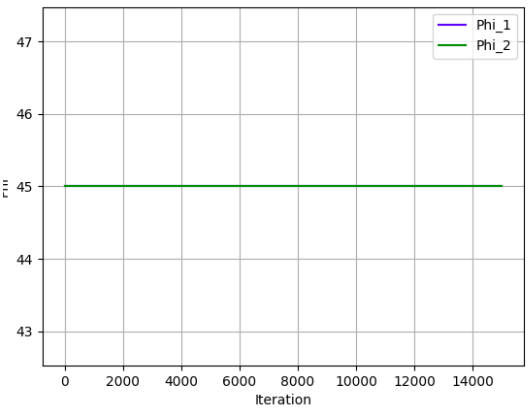

Figure 12. Phi of QGMM.

Besides, the values of phi haven't been changed from the initial values, 45 and 45.

Although the constraint has been decreases to zero, the values of alpha has been increased, and the difference between the values of phi is 0, the training result was bad.

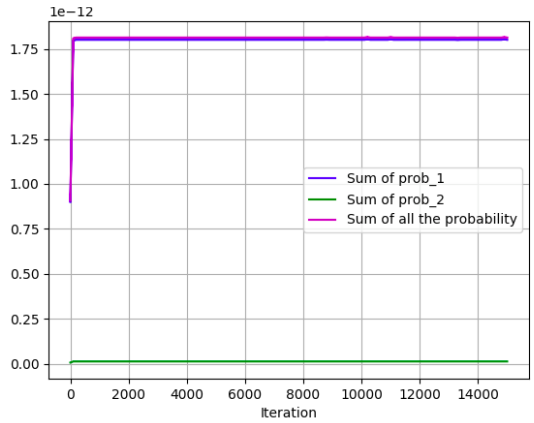

Figure 13. Probabilities of QGMM.

From Figure 13, we can check the probabilities are too low initially and they hasn't been updated while training.

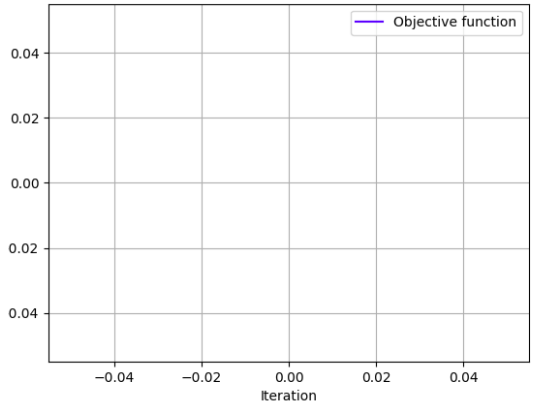

Figure 14. Objective function of QGMM

In addition, the values of the objective function were NaN (Not a Number), so we figured out the training didn't work well with the initial parameters including hyper parameters.

Therefore, we changed the step to update lambda from 1000 to 30 and the learning rate from 0.01 to 0.0001.

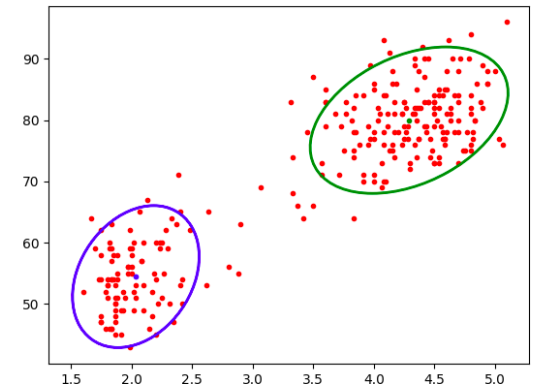

Figure 15. Training result with the changed hyper-parameters.

The reason we selected the values is that through several tests we found out the values are proper as an initial hyper parameters empirically.

As a result, we found out that the two distributions were moved little and little. Besides, at iterations 1000, we changed the learning rate to 0.01.

The training result can be seen in *Figure 15,* and we can see the training was good like GMM.

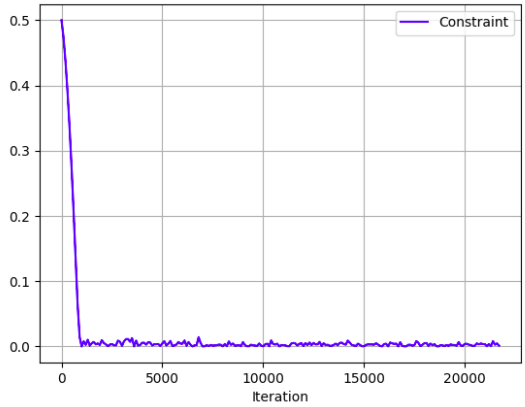

Figure 16. Constraint of QGMM.

The constraint has been decreased to almost zero.

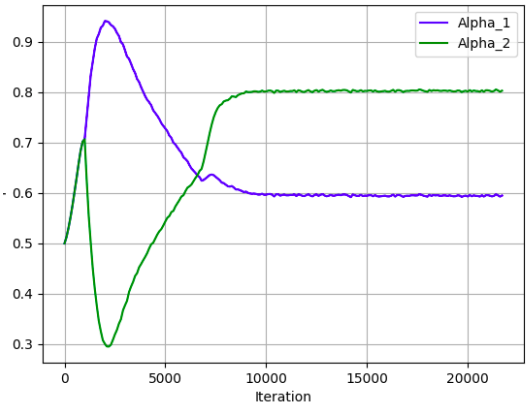

Figure 17. Alphas of QGMM.

Initially, the values of alpha have been changed sharply and after iteration 10,000, they converged.

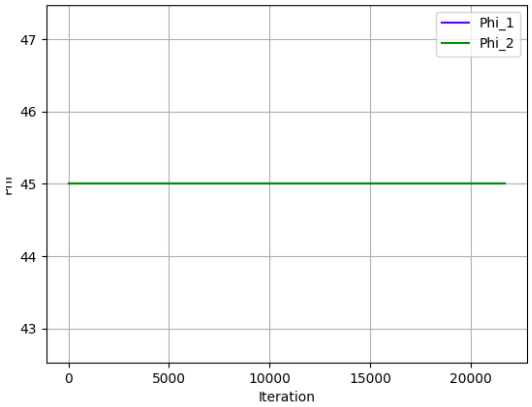

Figure 18. Phis of QGMM.

The values of phi haven't been changed from 45.

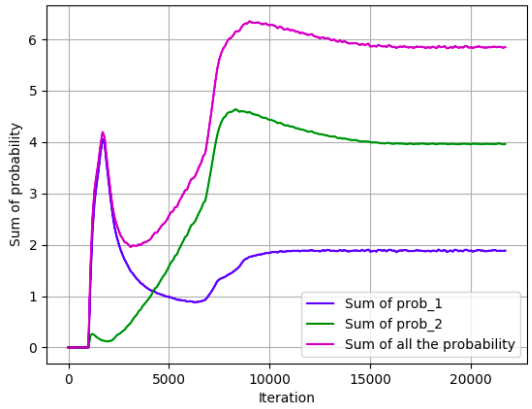

Figure 20. Probabilities of QGMM.

From *Figure 20*, we can see the probabilities are almost zero initially, however they have been increased at the specific iteration and converged after iteration 15,000.

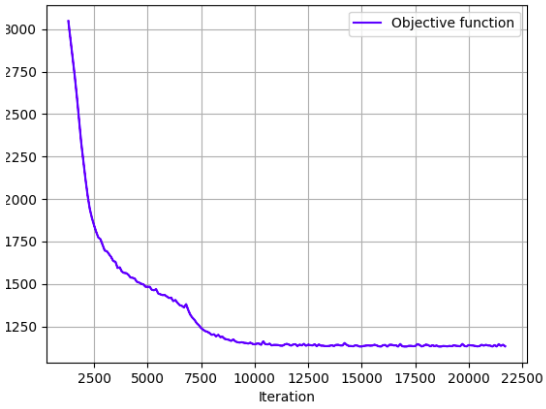

Figure 19. Objective function of QGMM.

In this time, the value of the objective function has been decreased unlike the previous experiments.

3.2. Convergence rates

We figured out the initial means had an effect on the training performance of both QGMM and GMM. Therefore, in this experiment, we'll check the percent of the convergence to each clusters by conducting 100 experiments with the different initial means generated randomly.

The means were generated from the maximum and minimum of x coordinates ± 1, and the maximum and minimum of y coordinates ± 10. In addition, the rest of other hyper parameters are the same with the experiment 3.1.

Model	**QGMM (ϕ=0)**	**QGMM (ϕ=90)**	**GMM**
Convg. (%)	**96**	80	94

Table 1. Convergence rates of QGMM and GMM.

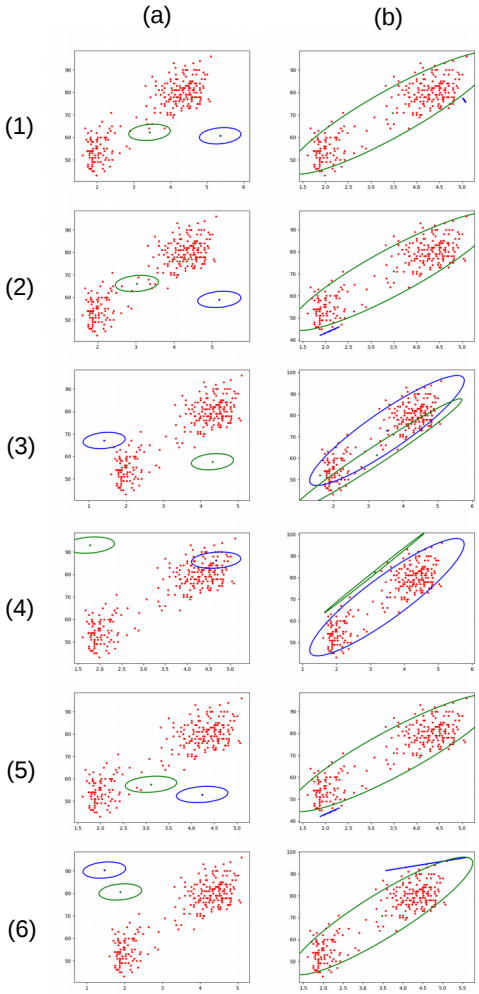

Figure 13. 6 failed cases for GMM. (a) Initial clusters. (b) Final clusters.

Out of 100, there were 4 and 21 failed cases for QGMM with the initial phi 0 and 90 respectively, and 6 failed cases for GMM.

With respect to the failed cases, the training results can be seen in the following figures. The notable point is that there was little intersection for the failed cases between QGMM and GMM. Especially, there was no intersection between QGMM with the initial phi 0 and GMM.

On the other hand, there were intersection between QGMMs with the initial phi 0 and 90. When failed with the initial phi 0, the convergence failed with the initial phi 90 as well.

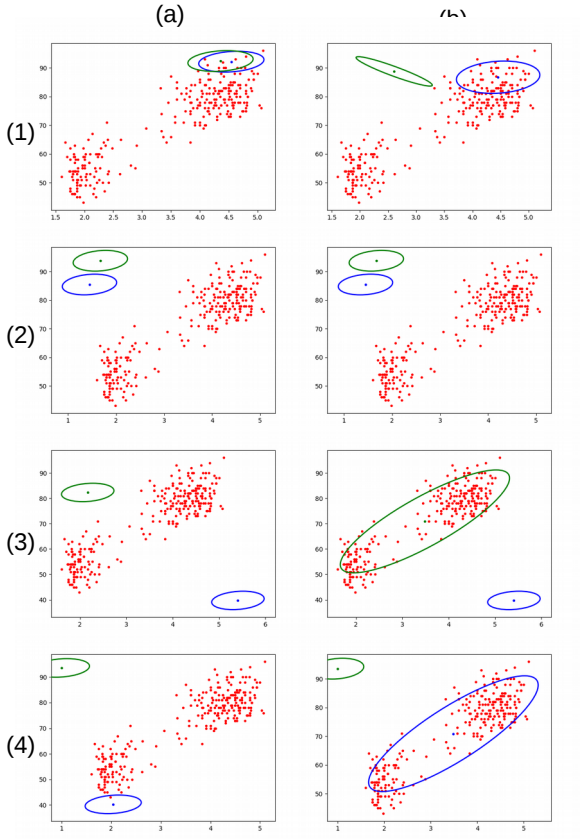

Figure 14. 4 failed cases for QGMM with the initial phi 0. (a) Initial clusters. (b) Final clusters.

For QGMM, there were failed cases when the covariance calculation didn't work correctly while optimizing, and results in the Cholesky's decomposition error (1, 2), and when the initial Gaussians were calculated to almost zero and there are no updates for them (3, 4, 5). Of course, in the previous experiment, we figured out that these results can be changed with the different hyper parameters.

In addition, we observed that for QGMM with the initial phi is 0, the two distributions pushed each other away when converging to the same cluster of the observations.

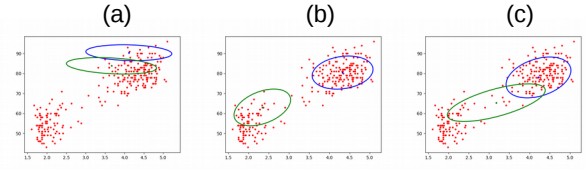

Figure 15. Push-away of QGMMs with the initial phi 0 and 90 in test case 22. (a) Common state. (b) cos(phi) is positive. (c) cos(phi) is negative

In *Figure 15* (b), we figured out that the two distributions of QGMM pushed each other away, while the two distributions were a bit mixed when the cos(phi) was negative.

The graphs about the variation progresses of phi can be seen in Figure 16.

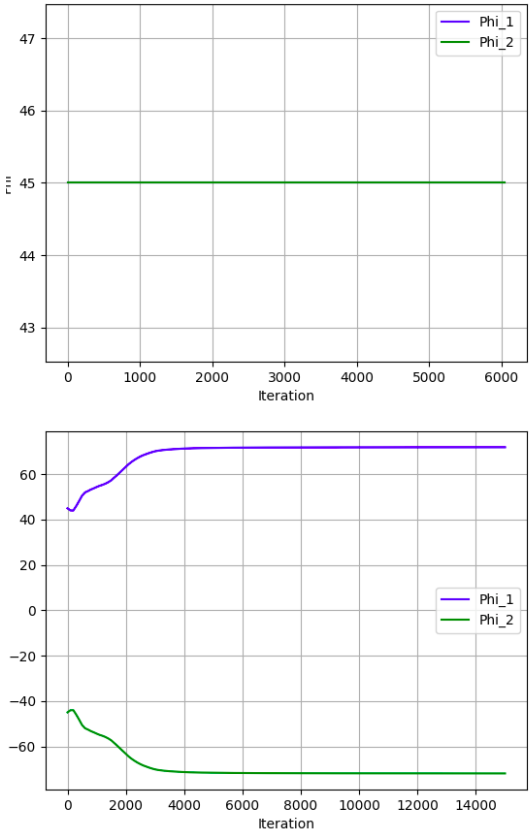

Figure 16. Phi variation of the initial phi 0 (up) and 90 (down).

In *Figure 16,* we can figure out that the cohesion of the two distributions varies according to the value of cos(phi). When cos(phi) is negative, the two distributions tend to mixed, while they tend to push each other away when cos(phi) is positive.

From the experiment that we just saw, we found out that the cases to fail between QGMM and GMM are totally different. In addition, when the initial values of Gaussians are almost

zero, QGMM tends to be hard to be trained well. Therefore, we conducted another experiment with the initial means generated from the maximum and minimum of x coordinates ±0.5, and the maximum and minimum of y coordinates ±5 to generate the initial means close to the clusters of the observations.

Model	QGMM (ϕ=0)	QGMM (ϕ=90)	GMM
Convg. (%)	**98**	80	94

Table 2. Convergence rates of QGMM and GMM.

Compared with the previous initial means, the convergence rate of QGMM with the initial phi 0 was increased.

3.3. Mixed clusters

In this experiment, we'll check the training results between QGMM and GMM in the mixed clusters of the observations. The data sets are the same with them in the document, "Mixture case"

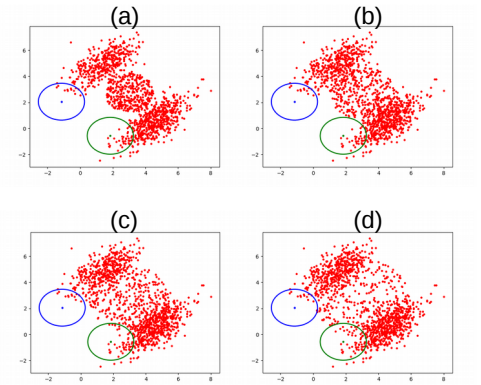

Figure 17. Initial states. (a) Radius 1.5. (b) Radius 2. (c) Radius 2.5. (d) Radius 3.

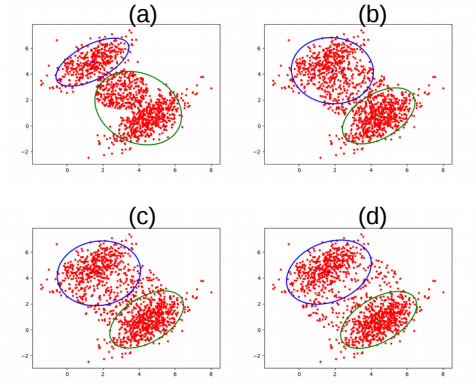

Figure 18. Training results of GMM. (a) Radius 1.5. (b) Radius 2. (c) Radius 2.5. (d) Radius 3.

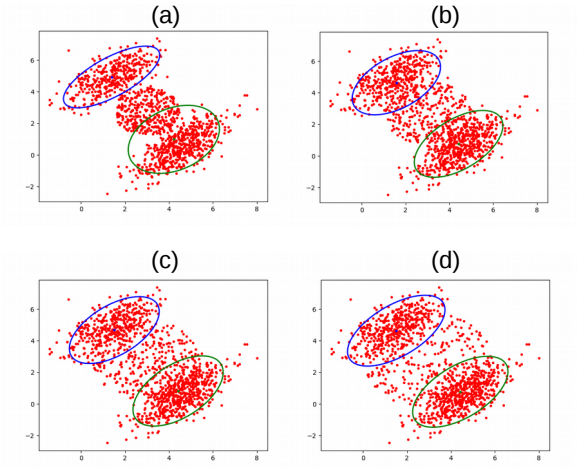

Figure 19. Training results of QGMM with the initial phi 0. (a) Radius 1.5. (b) Radius 2. (c) Radius 2.5. (d) Radius 3.

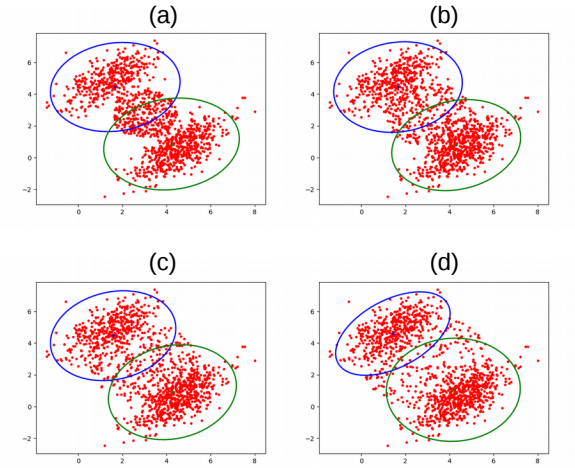

Figure 20. Training results of QGMM with the initial phi 90. (a) Radius 1.5. (b) Radius 2. (c) Radius 2.5. (d) Radius 3.

In *Figure 17*, we can see the initial states that we'll use for the experiment. In addition, from the *Figure 18, 19, and 20*, we can check the differences between QGMM and GMM.
In GMM, the final two distributions were not overlaid, but close to each other. On the other hand, in QGMM with the initial phi 0, they were positioned in some distance as the value of cosine was fixed 1. Lastly, in QGMM with the phi 90, they were mixed a bit as the value of cosine was negative.

3.4. Impact of the initial phi

From various experiment, we figured out that the values of phi represent the mixed state between the distributions. Therefore, we'll conduct an experiment to check if the impact of

phi setting to 180 when the two distributions were overlaid.

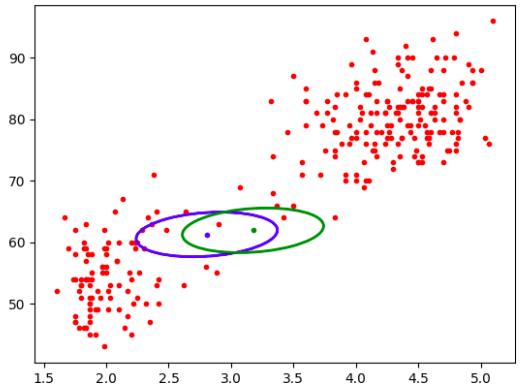

Figure 21. Overlaid initial state of the test case 43.

We can see that the two distributions were overlaid initially.

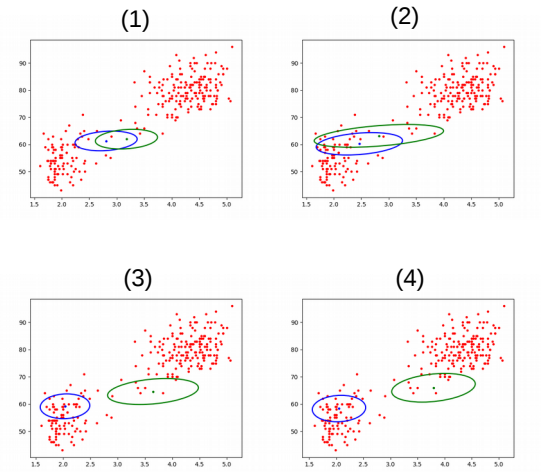

Figure 22. Training process with phi 0 in test case 43.

With the initial phi 0, the two distributions moved to the same cluster of the observations, and then, the green distribution moved to the upper cluster.

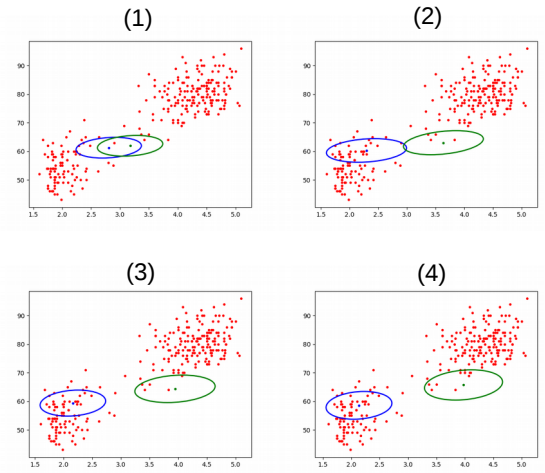

Figure 23. Training process with phi 180 in test case 43.

On the other hand, with the initial phi 180, the two distributions moved in an opposite direction, not moving to the same cluster as the value of cosine is about 0.5.

3.5. Augmented vs normal Lagrangian multiplier

In this research, we used the augmented Lagrangian multiplier to make our constraint more robust and flexible. In this experiment, we'll check the differences between the augmented and normal Lagrangian multiplier with the fixed lambda 500.

We conducted 100 experiments with the augmented and normal Lagrangian multiplier respectively, and the experiments were same with them used in 3.2. Convergence rates.

Model	**Aug. (ϕ=0)**	**Aug. (ϕ=90)**	**Normal (ϕ=0)**	**Nornal (ϕ=90)**
Convg. (%)	**96**	80	94	85

Table 3. Convergence rates of Augmented and normal Lagrangian multiplier. (x: ± 1, y: ± 10)

Model	**Aug. (ϕ=0)**	**Aug. (ϕ=90)**	**Normal (ϕ=0)**	**Nornal (ϕ=90)**
Convg. (%)	**98**	80	**98**	90

Table 4. Convergence rates of Augmented and normal Lagrangian multiplier. (x: ± 0.5, y: ± 5)

From the results, in the training with the initial phi 90, the normal Lagrangian multiplier showed the better rates. On the other hand, in the training with the initial phi 0, the augmented Lagrangian multiplier showed better results.

If we can get the proper lambda when training, it is a good option to use the normal Lagrangian method. However, it is not easy to set the initial value of lambda for every case regarding the normal method.

Therefore, we also can use the augmented Lagrangian method to approximate the proper initial lambda for the specific case.

4. Conclusions

In this research, we did compare QGMM with GMM, and check the training performance. From several experiments, we found out that QGMM can be trained more flexibly in the various environment when we get the proper initial parameters, while GMM showed the simple and consistent training performance.

Also, we figured out that QGMM can optimize the training performance according to the data set by adjusting the parameters with the means and covariances fixed.

Additionally, we checked the performance between the augmented and normal Lagrangian multiplier. Although we can decide which one is fine because it depends on the initial parameters and data set. We can get the optimized results from the fixed data set by changing the optimization method to use.

Consequently, if someone needs to use mixture model in the specific environment for the special purpose, QGMM would be a good alternative because it can optimize its performance according to the data set by adjusting the parameters, of course, it showed good result in the general environment as well.

References

[1] Curtin, R.; Edel, M.; Lozhnikov, M.; Mentekidis, Y.; Ghaisas, S.; Zhang, S. mlpack 3: A fast, flexible machine learning library. J. Open Source Softw. 2018, 3, 726 [Ref]