Image Segmentation: E-M Algorithm

Lab 2

Franky E., Gonzalez J.

Prepared for Medical Imaging Segmentation and Applications, MAIA/MIC MASTERS
October 2023

Contents

1 Introduction

The segmentation is an essential part of many computer vision systems and medical applications. The goal is to divide an input image into a set of non-overlapping regions which union is the entire image. Clustering-based methods are a well-known example to perform this task. These techniques (i) are fast and easy to implement and (ii) provide reasonable results even for complex medical applications.

In this coursework, the primary goal is to develop from scratch an Expectation-Maximization algorithm for segmenting brain MRI images (T1-w image) into the three main tissues: white matter (WM), grey matter (GM) and cerebrospinal fluid (CSF). The implemented algorithm should (i) assume mixture of Gaussians is a suitable model and (ii) be extended to process multivariate data (i.e. different image modalities: T1-w, T2-w, PD-w, ...). The algorithm will be evaluated using the provided data (ground truth for the three tissue classes is available). In the analysis, use the Dice Similarity measure to report the results quantitatively.

1.1 Problem Definition

Neuroimaging plays an important role in the study of brain structure and function, enabling researchers and clinicians to gain critical insights into neurological disorders, brain development, and cognitive processes. However, the analysis of neuroimaging data is full of challenges, including the accurate segmentation and classification of brain tissues from magnetic resonance images (MRI). This laboratory called Image Segentation E-M Algorithm focuses on understanding, analyzing and implementing the Expectation-Maximization algorithm addressing the specific problem of tissue classification in brain.

The development of an Expectation - maximization algorithm (EM algorithm) from

scratch to segment brain images in multi-modalities techniques, taking into account a mixture of Gaussians as a suitable model, refers to an unsupervised clustering problem since the intention is to detect the missing patterns, or hidden variables, within the data distribution that best predict the tissue differentiation of the brain. EM is applicable to latent variables, which are those that are not directly observable but can be inferred from the observed variables, in our case voxel intensity values. Given a general form of the probability distribution that these latent variables (tissue differentiation variables) could have, one can predict their values. In this case the probability distribution is modeled as a uni variate (mean, variance and weights) for only one input image segmentation, or multivariate (covariance) if the segmentation have two input images to segment, of Gaussian distributions for each hidden variable (tissues). Is known that there are 3 hidden variables (CSF, white & gray matter) on the images since they have gone across a skull stripping process before by SPM, and are the resulting images extracted from masking the original MR images.

The algorithm aims to accurately classify brain tissues into distinct categories, including gray matter GM, white matter WM, and cerebrospinal fluid CSF, while accounting for variations in image intensities and ensuring a high level of accuracy.

We intend to assess its accuracy, efficiency, and reliability in segmenting brain tissues. One way to compare is changing the way the parameters are initialized. As we are working with a clustering method, one way to assess is initializing the parameters of the algorithm with K-means algorithm and comparing to random initialization.

To gauge the algorithm's effectiveness, we plan to compare its segmentation results with ground truth provided. This comparison will allow us to quantify the algorithm's precision and recall in identifying brain tissues. Finally we will compare the segmentation results against the best cases of the SPM lab, using the Dice Score Coefficient as indicator.

2 Algorithm analysis

2.1 Design

The EM algorithm consists of two steps that iterate between them until convergence is achieved. These are Expectation, on which the posterior probabilities of the data points belonging to each cluster (estimated latent variable - tissues) is calculated based on the current parameters of the latent variables (1) for all mixture components, in our case; mean, covariance and weights for CFS, WM and GM. Next, it estimates the membership (2) of each point of the data to the latent variables based on the probabilities made.

$$p_k(\underline{x}|\Theta_k) = \frac{e^{-\frac{1}{2}(\underline{x}-\underline{\mu})^t \Sigma_k^{-1}(\underline{x}-\underline{\mu_k})}}{(2\pi)^{\frac{d}{2}}|\Sigma_k|^{\frac{1}{2}}} \tag{1}$$

where:

- $p_k(\underline{x}|\Theta_k)$ is the mixed probability distribution of the data $\underline{x}$ given the mixture components (k) sets of multivariate parameters Θ_k.
- $\underline{x}$ data measurements, in this case intensity level of each pixel.
- $\underline{\mu}$ mean of the data measurements.
- $\underline{\mu_k}$ mean of each mixture components.
- Σ_k covariance matrix of each mixture component.

$$w_{ik} = \frac{p_k(\underline{x_i}|z_k, \Theta_k) \cdot \alpha_k}{\Sigma_{m=1}^{K} p_m(\underline{x_i}|z_m, \Theta_m) \cdot \alpha_m}, 1 \leq k \leq K, 1 \leq i \leq N, \tag{2}$$

where:

- w_{ik} is the weight of the given mixture component k to point data i.
- $p_k(\underline{x_i}|z_i, \Theta_k)$ is the probability of the data point $\underline{x_i}$ given the mixture components (k) sets of multivariate parameters Θ_k extracted from the mixed distribution by the binary indicator variable z_i.
- α_k is the mixture weights for a given mixture component k.
- $\Sigma_{m=1}^{K} p_m(\underline{x_i}|z_m, \Theta_m) \cdot \alpha_m$ is the sum of the different probabilities times its weights of all the mixture components k for the given point i.
- $\underline{\mu_k}$ mean of each mixture components.
- k is one mixture component of the total K.
- i is a data point of the total dataset N.

In the maximization step the parameters of the latent variables are updated accordingly to the latent variables clustered in the previous step.

- The means are updated as: $\mu_k^{new} = \frac{\sum_{i=1}^{N} w_{ik} \cdot x_i}{N_k}, 1 \leq k \leq K$, where w_{ik} represent the belonging of each pixel x_i to certaing cluster k, and N_k the sum of all the belongings for that cluster.
- The wheights are update as: $\alpha_k^{new} = \frac{\Sigma_{i=1}^{N} w_{ik}}{N}, 1 \leq k \leq K$, where the new wheights are calculated for the k mixing component as the ratio between the sum of its a significance, understand as the wheight of that pixel times the intensity level (value) it has.

- the covariance matrix is updated as:

  $\Sigma_k^{new} = (\frac{1}{N})\Sigma_{i=1}^{N} w_{ik} \cdot (\underline{x_i} - \underline{\mu}_k^{new})(\underline{x_i} - \underline{\mu}_k^{new})^t, 1 \leq k \leq K$, where the new covariance matrix is equal to the standard deviation of each pixel times the standard deviation of the transpose data, the covariance of each pixel, times the weights of the mixture components, all normalized over the total number of pixels.

Finally, the convergence of the algorithm is obtained from the log-likelihood value after each iteration when its value does not have significant change between iterations. The log likeli-hood value can be calculated under the independent and identically distributed random variables as (3).

$$Logl(\Theta) = \Sigma_{i=1}^{N}(Log\Sigma_{k=1}^{K}\alpha_k p_k(\underline{x_i}|z_k, \theta_k)) \tag{3}$$

2.2 Implementation

Firstly, the image's data was prepared by flattening the image's pixels, extracting those with relevant information (intensity levels) into a 1D array, and saving the indices of the image's 1D array of those pixels. The process loops across all the input image modalities, generating different feature columns for different modalities on the resulting array. Then the object class of the EM algorithm is defined as the execution of 10 functions, or stages, correlated between them to **fit** the data and iterate it over the expectation and maximization stages, calculating for each iteration the log-likelihood, until the maximum of iteration or convergence is achieved, then **predict** the final cluster from the last membership obtained. The object initializes the parameters randomly or as the result of a heuristic classification from k means. Also, it lets the user select the implementation of the mixture distribution

(Gaussian mixture model) or the implementation of Scipy. Last, the script defines a function to reconstruct the image segmentation from the index of each element on the labeled array according to the indices of the original array.

Our experimentation with various random initialization methods shed light on the critical role of this process in the E-M algorithm for brain tissue segmentation. Initially, we adopted a random initialization strategy that utilized the random.rand function, producing values within the 0 to 1 range. However, this approach proved to be detrimental as it introduced a high degree of randomness and chaos to the segmentation process. We quickly realized that for effective segmentation, the initialization values needed to be derived from within the minimum and maximum pixel intensity range, aligning more closely with the actual image data.

In the case of single MRI modality segmentation, this new approach yielded promising results. However, when dealing with multi-modal images, a new set of challenges emerged. The unique characteristics of T1 and T2 FLAIR images meant that the six possible cluster centers were vastly different, introducing a high level of complexity. The random differences in pixel intensities between the modalities were often significant, further complicating the clustering process.

To address this, we refined our random initialization technique by generating only three random clusters and extending these values across the required number of columns. This approach provided more stability and consistency in our results. Yet, we encountered situations where random values like 200, 210, and 215 were generated. In such cases, the segmentation results were less reliable. We even contemplated the idea of generating random numbers in intervals of `1/n_components` to ensure reasonably separate differences among the clusters. However, this approach was not ultimately implemented in our methodology.

In the initial stages of our study, our results were relatively satisfactory, but they bore a striking resemblance to the outcomes of a previous laboratory's work. Upon closer examination, a quick visual inspection unveiled an important revelation - the ground truth data lacked certain critical brain structures, such as the cerebellum. This discovery prompted us to reevaluate our approach. The decision was made to create a mask derived from the ground truth data to address this issue, a step that ultimately led to a significant improvement in the quality of our segmentation results.

One notable consideration was whether to apply the masking process at the beginning or end of the segmentation pipeline. After some deliberation, we reached a consensus to mask our skull-stripped image before the clustering process began. This choice offered the advantage of a more controlled and stable segmentation process. Nevertheless, it's worth mentioning that we observed marginally superior results when the masking was performed at the conclusion of the segmentation process.

3 Experimental Section

Here we will test different parameters and results, patient by patient. All the obtained results will be explained, plotting all the iterations and differences among different tissue segmentation. The 4 different scenarios are k-means initialization using our gmm implementation (km_our), k-means initialization using scipy implementation (km_sc), random initialization and our gmm implementation (rn_our) and random initialization using scipy implementation(rn_sc).

3.1 Patient 1

Case 1 was selected according to the best result of the SPM lab. In which segmentation results, in Dice Score Coefficient are: 0.7547 for GM segmentation, 0.8284 for White Matter and 0.8030 for CSF.

After executing the "testing" function, we collect various outcomes. These include the log-likelihood maximization graph showing how it evolves across the EM algorithm iterations, the time it takes for each execution method, as well as visual comparisons of random slices selected from the range 0 to 47 (considering each NIFTI file's dimensions are (240, 240, 48)). These visual comparisons involve the image, the corresponding ground truth, and a difference map between the two.

Using again the Dice score coefficient, we can asses the different obtained results.

In terms of execution time, the results presented in Tab 1 clearly indicate that the algorithm exhibits superior speed when applied to a single MRI modality as opposed to the multi-modal scenario. Specifically, for the single modality, the employment of T2 FLAIR in combination with k-means and the scipy multivariate normal function leads to a noteworthy 58.82% reduction in processing time compared to our implementation of the mixture of Gaussian distributions. Conversely, within the multi-modal context, it is observed that Random Initialization outperforms K-means in terms of speed. Furthermore, the discrepancy in execution time between our implementation and the scipy counterpart is relatively minimal, with the latter being only 14.28% faster. Interestingly, our implementation surpasses the scipy method in terms of speed in half of the tested cases.

	T1		T2		T1+T2	
	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal
K-means	17s	19s	17s	7s	35s	36s
Random	12s	9s	18s	20s	21s	18s

Table 1: Patient 1: execution time per parameter initialization method and per gaussian generator. Our method and scipy one.

As per the different required iterations fig 1 shows the different log-likelihoods plotted against the iterations for T1 case. It can be seen that, logically, the fastest performers required less iterations, which means that their stop flag was the tolerance.

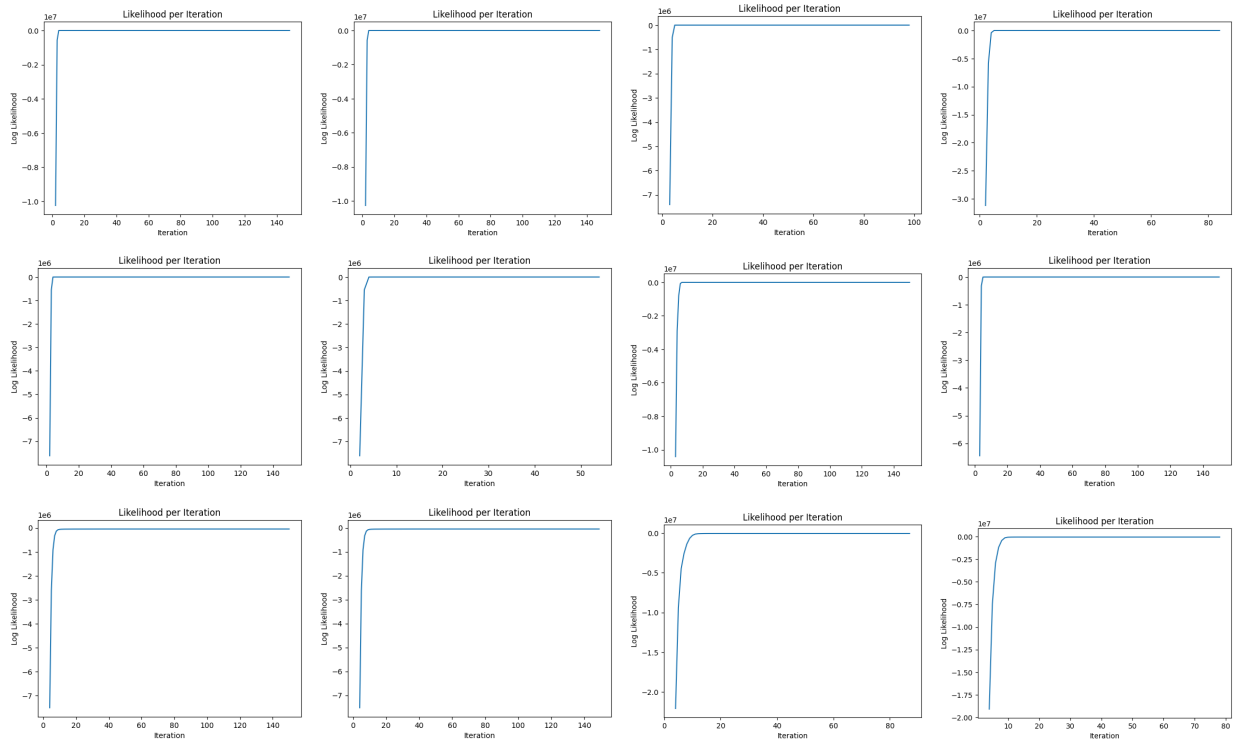

Figure 1: Log-Likelihood vs Iterations in the different cases for patient 1. From left to right km_our, km_sc, rn_our and rn_sc. Top: T1, Middle T2, Bottom: T1+T2

	T1				**T2**				**T1+T2**				**Lab 1**	
	km_{our}	km_{sc}	rn_{our}	rn_{sc}	km_{our}	km_{sc}	rn_{our}	rn_{sc}	km_{our}	km_{sc}	rn_{our}	rn_{sc}	T1	T2
CSF	0.80	0.80	0.79	0.79	0.76	0.76	0.74	0.73	**0.90**	**0.90**	**0.55**	**0.90**	0.80	0.70
GM	0.78	0.78	0.78	0.78	0.39	0.35	0.58	0.61	**0.83**	**0.83**	**0.16**	**0.83**	0.75	0.63
WM	0.86	0.86	**0.87**	**0.87**	0.10	0.12	0.02	**0.01**	0.86	0.86	0.61	0.86	0.83	0.70

Table 2: Overall performance of the E-M algorithm for every tissue in patient 1.

In tab 2 we can see that there is a slight improvement of the segmentation on T1 compared with the segmentation Dice score obtained in the previous lab. However, the real improvement comes when segmenting the brain using an ensemble of T1 and T2, providing a better results when handling the problem as a multi modal problem. In tab 2 we highlight in blue the best results and in red the worst result. If a decision should be made on this one, definitively using T1+T2, k-means initialization and our implementation of gmm is the way to go.

Finally, the following figure 2 will display the segmentation results, the ground truth and the difference among them.

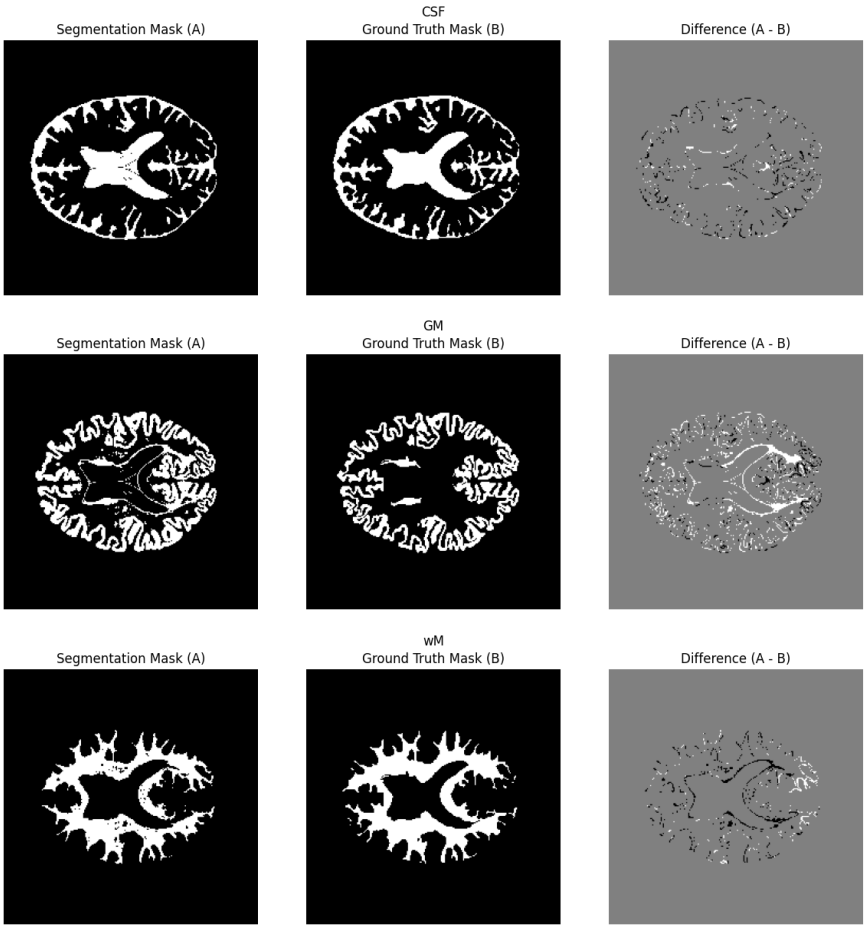

Figure 2: Top segmentation results per tissue for patient 1: Top CSF. Middle GM. Bottom WM

3.2 Patient 2

The Overall performance shows results seen in 4 where we can see that the best results come from using just T1 as the input. In case of T2, results under-perform the results from the previous lab. The best set of results, overall, are the random initialization using our GMM implementation. In this case, while T1+T2 somehow obtain similar values to either T1 or T2 from previous lab, they are just in CSF and there is no K-means or random initialization

set that works perfect.

	T1		T2		T1+T2	
	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal
K-means	22s	19s	21s	22s	38s	38s
Random	12s	7s	20s	11s	39s	14s

Table 3: Patient 2: execution time per parameter initialization method and per gaussian generator. Our method and scipy one.

When it comes to execution time, it can be seen in table 3 that random initialization and Scipy GMM are again the fastest method, however it does not produce the best results. The iterations can be seen in the log-likelihood graph in figure 3.

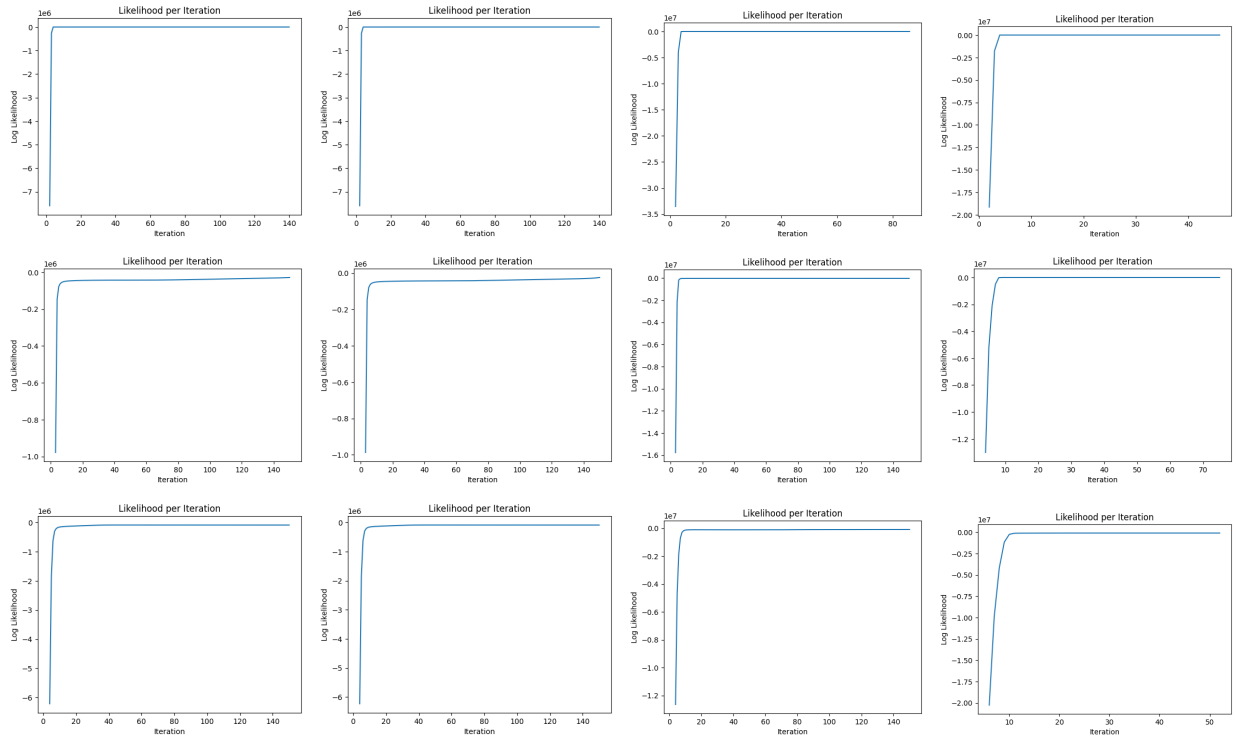

Figure 3: Log-Likelihood vs Iterations in the different cases for patient 2. From left to right km_our, km_sc, rn_our and rn_sc. Top: T1, Middle T2, Bottom: T1+T2

	T1				**T2**				**T1+T2**				**Lab 1**	
	km_{our}	km_{sc}	rn_{our}	rn_{sc}	km_{our}	km_{sc}	rn_{our}	rn_{sc}	km_{our}	km_{sc}	rn_{our}	rn_{sc}	T1	T2
CSF	**0.87**	**0.87**	**0.87**	0.71	0.49	0.49	**0.44**	0.78	0.73	0.73	0.70	0.86	0.77	0.73
GM	**0.81**	**0.81**	**0.81**	0.28	0.13	0.13	**0.11**	0.69	0.34	0.34	0.32	0.70	0.72	0.70
WM	0.78	0.78	**0.79**	0.59	0.52	0.52	0.52	**0.00**	0.56	0.56	0.56	0.00	0.76	0.51

Table 4: Overall performance of the E-M algorithm for every tissue in Patient 2

One thing that is worth noticing is that the worst cases come from using random initialization and the scipy implementation of a normal multivariate distribution, normally using just T2 FLAIR as an input. Even if we obtained low results with our implementation, this leads us to think that a wrong set of initial parameters won't allow to create a proper set of gaussian mixtures to improve via E-M algorithm. The following figure shows the best performers for each tissue.

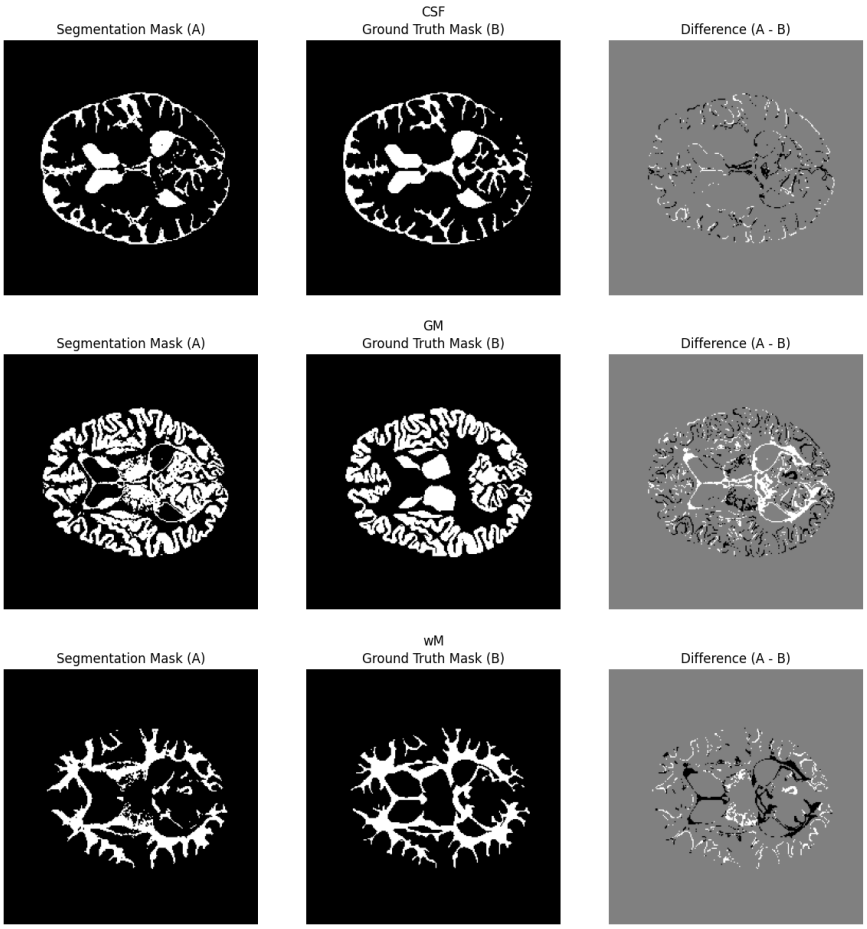

Figure 4: Top segmentation results per tissue for patient 2: Top CSF. Middle GM. Bottom WM

3.3 Patient 3

We have been seeing the under-perform delivered by the combination of random initialization and scipy multivariate normal distribution. Interestingly, in this patient data, this was precisely the best combination. Is worth noting that the multi-modal version improves the best the results from previous lab. Again, as in the previous patient, the worst performance are present using just T2 as an input and this leads os tu think about the possible reasons.

Is it possible to see that it again performs worst than the results from previous labs and one reason may be due to inherent properties of T2 FLAIR imaging, where WM and GM are intermediate to bright.

	T1		T2		T1+T2	
	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal
K-means	16s	19s	18s	18s	34s	35s
Random	15s	5s	18s	11s	23s	35s

Table 5: Patient 3: execution time per parameter initialization method and per gaussian generator. Our method and scipy one.

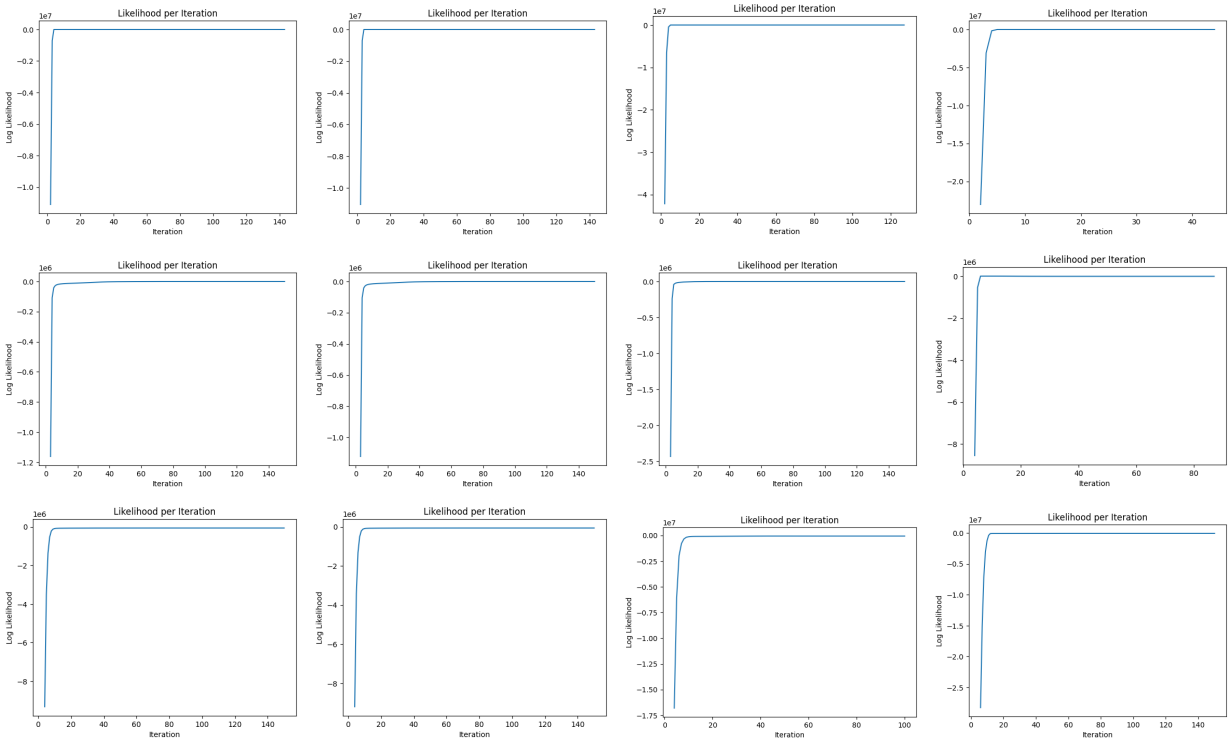

Figure 5: Log-Likelihood vs Iterations in the different cases for patient 3. From left to right km_our, km_sc, rn_our and rn_sc. Top: T1, Middle T2, Bottom: T1+T2

	T1				**T2**				**T1+T2**				**Lab 1**	
	km$_o$*ur*	km$_s$*c*	rn$_o$*ur*	rn$_s$*c*	km$_o$*ur*	km$_s$*c*	rn$_o$*ur*	rn$_s$*c*	km$_o$*ur*	km$_s$*c*	rn$_o$*ur*	rn$_s$*cipy*	T1	T2
CSF	0.83	0.83	0.63	0.78	0.45	0.45	0.41	**0.09**	0.87	0.87	0.57	**0.88**	0.78	0.68
GM	0.79	0.79	0.36	0.77	0.61	0.61	0.61	**0.03**	0.80	0.80	0.25	**0.81**	0.73	0.51
WM	0.85	0.85	0.70	**0.86**	**0.12**	**0.12**	**0.12**	0.55	0.85	0.85	0.65	**0.86**	0.83	0.68

Table 6: Overall performance of the E-M algorithm for every tissue in Patient 3

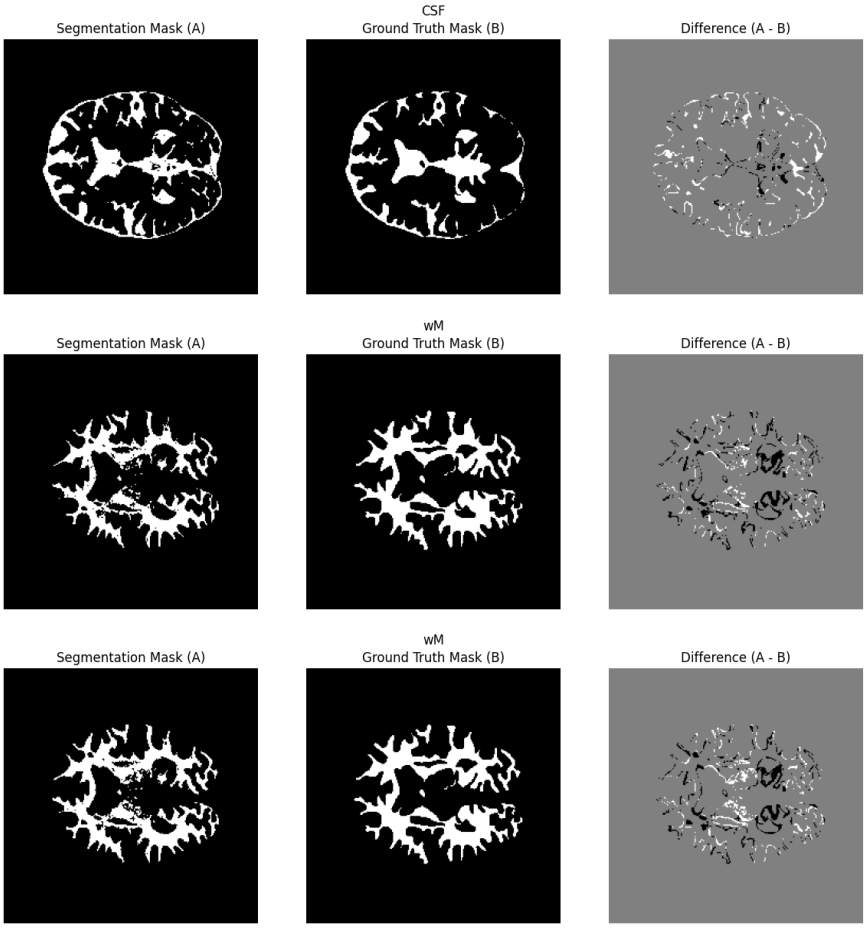

Figure 6: Top segmentation results per tissue for patient 3: Top CSF. Middle GM. Bottom WM

	T1		T2		T1+T2	
	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal
K-means	19s	15s	17s	18s	29s	30s
Random	6s	12s	14s	10s	19s	22s

Table 7: Patient 4: execution time per parameter initialization method and per gaussian generator. Our method and scipy one.

3.4 Patient 4

In patient 4 we see again a critical 0.00 number, this time in multimodal, using random initialization and scipy multivariate normal. In any case, using T1 and T2 FLAIR, with k-means and our multivariate implementation we find notable improvement in the Dice Score. According to table 8 we see 12.82% improvement in CSF, 7.59% in GM and 6.09% in GM compare to the best performance in last practice. Execution time presents more or less the same pattern of the previous patients.

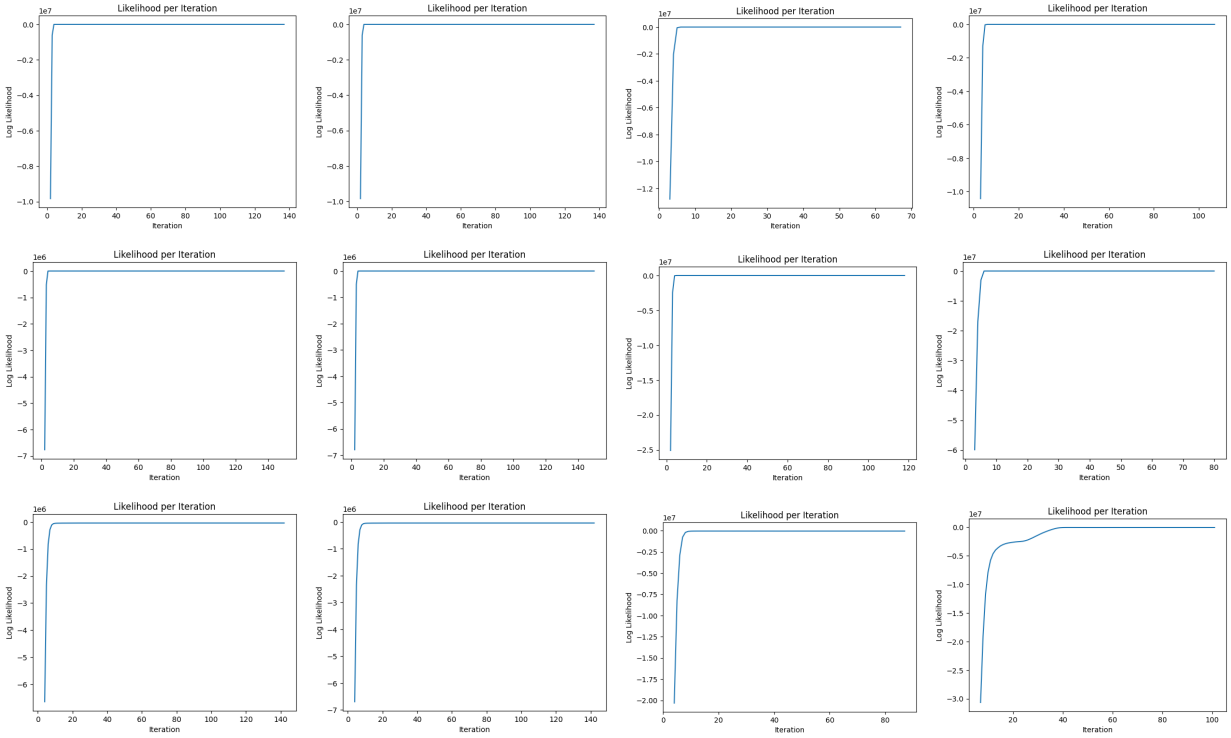

Figure 7: Log-Likelihood vs Iterations in the different cases for patient 4. From left to right km_our, km_sc, rn_our and rn_sc. Top: T1, Middle T2, Bottom: T1+T2

	T1				**T2**				**T1+T2**				**Lab 1**	
	km$_our$	km$_{sc}$	rn$_{our}$	rn$_{sc}$	km$_{our}$	km$_{sc}$	rn$_{our}$	rn$_{sc}$	km$_{our}$	km$_{sc}$	rn$_{our}$	rn$_{sc}$	T1	T2
CSF	0.84	0.84	0.83	0.83	0.76	0.76	0.42	**0.12**	**0.90**	**0.90**	0.66	0.89	0.80	0.71
GM	0.82	0.82	0.82	0.82	0.37	0.37	**0.03**	0.02	**0.84**	**0.84**	0.23	0.69	0.77	0.61
WM	**0.87**	**0.87**	**0.87**	**0.87**	0.17	0.16	0.48	0.47	**0.87**	**0.87**	0.58	**0.00**	0.82	0.65

Table 8: Overall performance of the E-M algorithm for every tissue in Patient 4

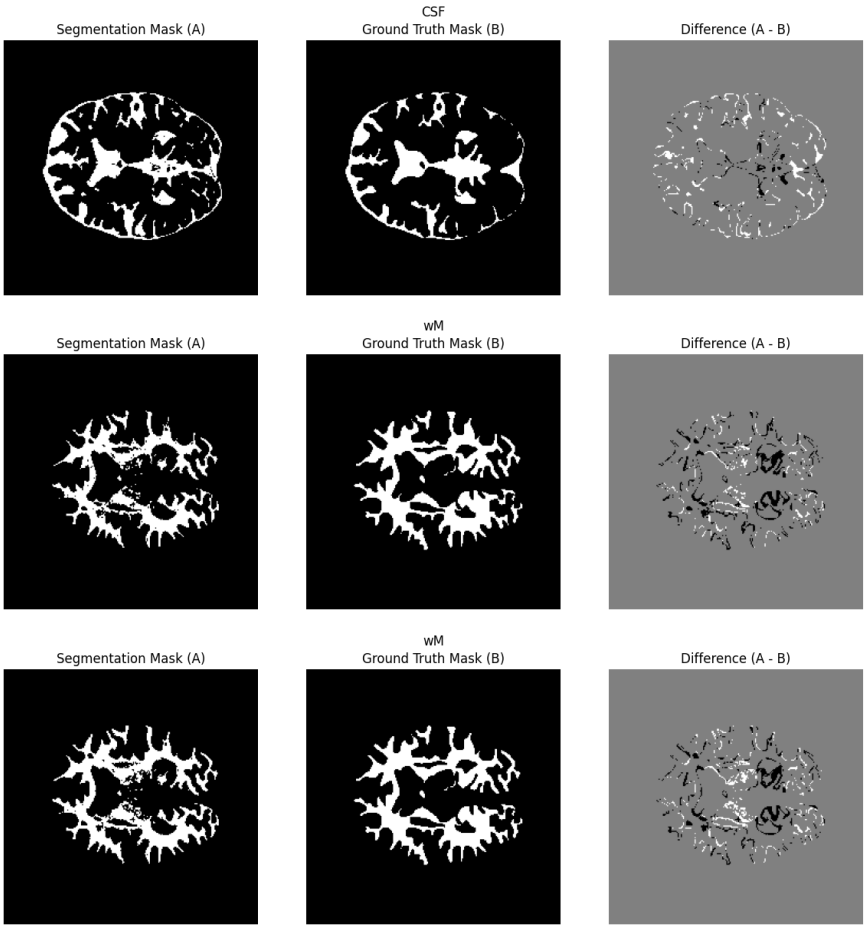

Figure 8: Top segmentation results per tissue for patient 4: Top CSF. Middle GM. Bottom WM

3.5 Patient 5

Patient 5 repeats the case in which just T1 itself gives the best performance. Random initialization with our GMM implementation seems the most suitable in terms of Dice Score Coefficient. In terms of log likelihood, multimodal segmentation shows a different pattern compared to the ones we have seen so far. Nevertheless, it provided a black image for WM tissue, which make it less reliable, even though CSF was considerably improved.

	T1		T2		T1+T2	
	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal	Our GMM	Scipy Multivariate Normal
K-means	16s	16s	17s	19s	33s	31s
Random	10s	6s	17s	19s	22s	11s

Table 9: Patient 5: execution time per parameter initialization method and per gaussian generator. Our method and scipy one.

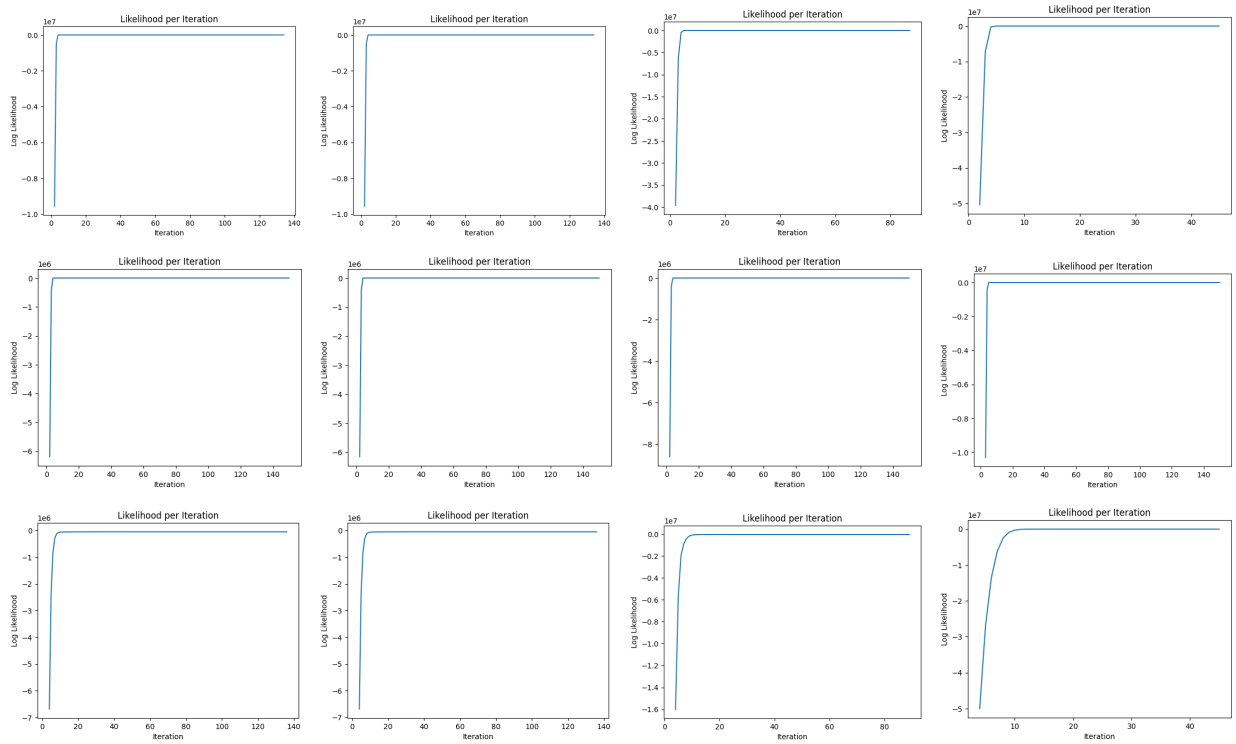

Figure 9: Log-Likelihood vs Iterations in the different cases for patient 5. From left to right km_our, km_sc, rn_our and rn_sc. Top: T1, Middle T2, Bottom: T1+T2

	T1				**T2**				**T1+T2**				**Lab 1**	
	km_{our}	km_{sc}	rn_{our}	rn_{sc}	km_{our}	km_{sc}	rn_{our}	rn_{sc}	km_{our}	km_{sc}	rn_{our}	rn_{sc}	T1	T2
CSF	0.82	0.82	0.82	0.22	0.80	0.80	0.28	**0.23**	**0.87**	**0.87**	0.73	0.86	0.72	0.71
GM	**0.86**	**0.86**	**0.86**	0.06	0.40	0.38	0.05	**0.04**	0.85	0.85	0.28	0.68	0.79	0.61
WM	0.89	0.89	**0.90**	0.54	**0.12**	0.14	0.51	0.51	0.89	0.89	0.60	0.00	0.84	0.67

Table 10: Overall performance of the E-M algorithm for every tissue in Patient 5

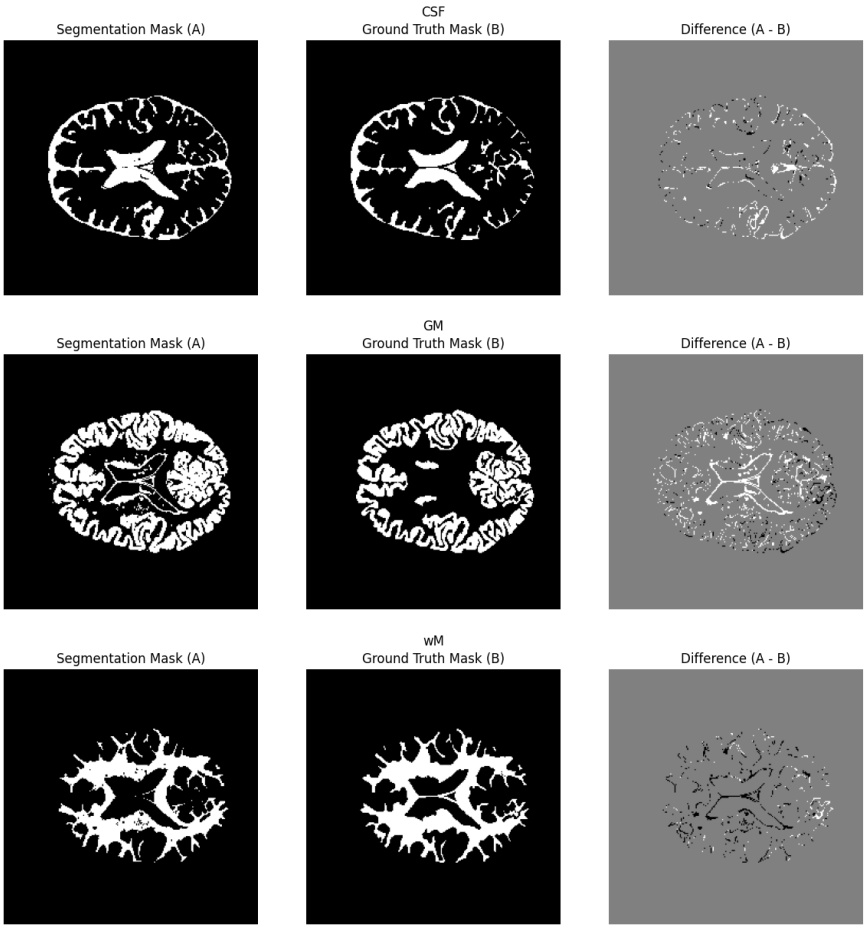

Figure 10: Top segmentation results per tissue for patient 5: Top CSF. Middle GM. Bottom WM

3.6 Drawbacks and solutions

One of the challenges we encountered during the implementation of the E-M algorithm was the issue of zero divisions and zero matrices. These problems can arise when the data distribution is particularly skewed or when certain clusters have very low data points. To mitigate this, we introduced a small regularization term in both the expectation step and our custom multivariate function. This regularization term added a small constant to the

diagonal of the covariance matrix, ensuring that it remains well-conditioned even in cases where the data distribution might otherwise lead to numerical instability. By addressing the zero division and zero matrix issues in this manner, we improved the robustness and stability of our algorithm, allowing it to handle a better range of data distributions and clustering scenarios.

We also observed a common issue when assessing the quality of our segmentation, particularly in cases where the Dice similarity coefficient fell within the range of 0.50 to 0.60. In these instances, the resulting segmentation exhibited a characteristic pattern. They tended to merge multiple distinct tissue types into a single cluster as shown in figure 11, and the Dice score, while seemingly high, was essentially a reflection of the coincidental alignment of pixels rather than an accurate segmentation. We encountered situations where the algorithm assigned all pixel values to a single tissue, resulting in a perfect Dice score of 1.00 for that particular tissue. This was clearly not a meaningful segmentation, but rather an artifact of the clustering process, highlighting the importance of carefully interpreting segmentation results and considering the quality of the output beyond simple metrics. Finally we saw cases as well in which the output was zero. This last ones were found mostly in random initialization, which points toward the importance of initialization points.

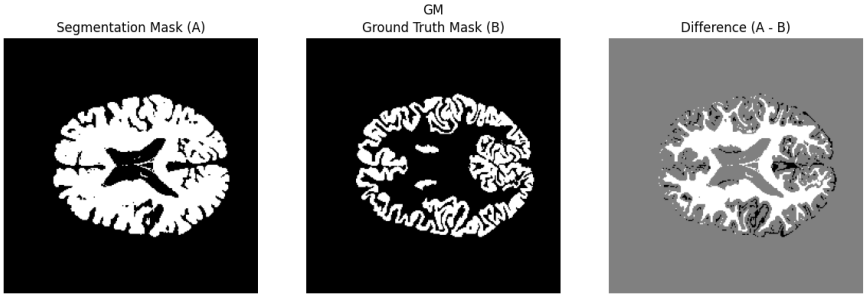

Figure 11: Bad segmentation example

4 Project Management

The first two objectives of the lab were, in our case, the most time-consuming. We looked for information while trying to understand the scope of the E-M algorithm. We started with the info presented in the classroom. Sometimes happened that we were searching for something already explained. This taught us to recheck our notes and provided material as well as the recommended literature before going down the rabbit hole of searching for more information.

This can be easily compared to a proper initialization. Code was also a challenge, finding the best numpy practices to achieve the best possible calculations. Testing required less time as we started to comprehend the algorithm. However, dealing with the randomness of the initialization was a bit hard. We estimated delivering 2 days before the deadline but the testing part and other coursework made us use the real deadline as the one to go.

Around forty percent of the time was used in the "now what?" part of the understanding, another forty percent in dealing with the code and testing, and a final 20 percent for reporting purposes.

5 Conclusion

In conclusion, our exploration of the Expectation-Maximization (E-M) algorithm for brain tissue segmentation has yielded valuable insights and lessons. Random initialization emerged as a crucial factor in the success of our segmentation. Initially, we experimented with random values between 0 and 1 for initialization, which resulted in chaotic outcomes. It became evident that initializing with random values within the range of minimum and maximum pixel intensities made more sense, providing stability and robustness to the algorithm. In

addition, we explored the potential to introduce significant differentiation among the initial parameters; however, this idea, although considered, has not yet been implemented in this version. The primary objective is to optimize the distribution of the initial centroids, aiming to better generalize the overall data distribution. This initiative stems from observations indicating that in certain random initializations, the initial centroids tend to be positioned too closely, resulting in a deterioration of the overall performance.

Additionally, we noted the significance of random initialization in the K-means clustering algorithm. While using a specified `random_state` value can yield consistent results, omitting it allowed us to observe the algorithm's sensitivity to initialization, highlighting the stochastic nature of the process.

Another aspect worth mentioning is the challenge posed by incomplete ground truth data. We noticed that not all brain structures, such as the cerebellum, were included in the ground truth data. To address this, we initially utilized skull-stripped brain images from previous work. Eventually, we adopted a mask from the ground truth, a decision we initially hesitated to make. This experience underscored the importance of thorough and accurate ground truth data for precise segmentation.

Through our journey, we gained a deeper understanding of the E-M algorithm's strengths and benefits in clustering brain tissue. Despite initial challenges and unexpected turns, we have harnessed the power of this algorithm for segmentation tasks and have witnessed its potential to make significant contributions in the field of neuroimaging.