Image Segmentation using Expectation Maximization

Medical Image Registration and Applications

Kaouther Mouheb Joaquin Oscar Seia

I. Introduction

Image segmentation is a major task in Computer Vision which aims to divide an image into mutually exclusive regions which union provides the image itself. Segmentation can be performed in both supervised and unsupervised manners. One of the most widely used unsupervised segmentation techniques is clustering. The latter is particularly useful in tasks such as brain tissue segmentation where having a manual annotation is very time and effort consuming. Expectation-Maximization (EM) combined with a Gaussian Mixture Model (GMM) is one way to perform clustering-based image segmentation. In this approach, the features coming from each of the different classes composing the image are assumed to follow a Gaussian (Normal) probability distribution. Mean and covariance matrix parameters for each class are estimated using the EM algorithm.

II. Objective

The objective of this laboratory work is to understand the theoretical basis of image segmentation using a Gaussian Mixture Model and Expectation-Maximization, implementing a segmentation framework that uses these techniques to perform multivariate 3D volume segmentation, with an application on brain tissue segmentation from MRI scans using T1 and T2 FLAIR modalities.

III. Theoretical Background

A. Gaussian Mixture Model

A Gaussian Mixture Model is defined as a combination of K components (clusters) where each component k is a Gaussian density characterized -in its multivariate version- with the following 3 parameters:

- A mean μ_k defining its center.
- A Covariance matrix Σ_k defining its spread.
- A prior probability α_k defining the weight of the cluster k in the mixture. That is to say, the probability that a randomly selected sample x was generated by the component k. This implies that α_k values sum to 1.

Each component is defined by a Gaussian density function (equation 1):

$$p_k(x|\theta_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \quad (1)$$

Where d is the dimension of the point x, and θ_k is the set of parameters of the Gaussian component k (μ_k and Σ_k).

The final density of the mixture model of K Gaussian components is given by equation 2.

$$p(x|\Theta) = \sum_{k=1}^{K} \alpha_k p_k(x|z_k, \theta_k) \quad (2)$$

Where z_k is a latent variable representing the identity of the mixture component c that is presumed to have generated the sample x. It is a vector of K elements where the c^{th} element is one and the remaining elements are zero. In other terms, it is a one-hot encoding indicating the belonging of point x to cluster k.

The goal of the clustering-based segmentation method is to determine Θ the set of optimal values for the three parameters μ, Σ and α for each Gaussian component that best fits the available data points. This is achieved using Maximum Likelihood.

Gaussian mixture models can be seen as a generalization of the k-means algorithm that not only considers the centers of the latent Gaussians but also incorporates the covariance structure of the points belonging to each of them.

B. Expectation-Maximization for GMM

As previously mentioned, EM is an algorithm used to estimate the optimal parameters of the Gaussian mixture densities that best fit the data samples. The algorithm is an iterative approach that starts from some initialization of the parameters and keeps updating them until a stopping criteria is met. Each iteration has two steps: an Expectation step (E-step) and a Maximization one (M-step).

1) E-Step

In this step, the current set of parameters are used to determine the belonging of each point (voxel) to one of the components. Mean, covariance and prior probabilities of each Gaussian components are used in a Bayesian approach to determine the posterior probability, indicating how probable is that the given point is a member of a particular component. Posterior probabilities, also called membership weights are calculated for each data point and for each mixture component according to equation 3.

$$w_{ik} = \frac{p_k(x_i|z_k, \theta_k).\alpha_k}{\sum_{m=1}^{K} p_m(p_m(x_i|z_m, \theta_m).\alpha_m)} \quad (3)$$

The weight w_{ik} denotes the probability that the data sample x_i belongs to the component k given the data sample itself and the current set of parameters Θ. For each point x_i, the sum of the weights across all Gaussian components must sum to 1.

2) M-Step

In the maximization step, the weights calculated in the E-Step and the data samples are used to calculate the new parameters of the Gaussian components following formulas 4, 5 and 6.

$$\alpha_k^{new} = \frac{N_k}{N} \quad (4)$$

$$\mu_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} w_{ik}.x_i \quad (5)$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} w_{ik}.(x_i - \mu_k^{new})(x_i - \mu_k^{new})^T \quad (6)$$

where N is the total number of samples and N_k the sum of membership weights across all data samples for component k.

3) Convergence

As EM is an iterative approach, a convergence criterion is required to stop the iterations.

One way to monitor the convergence of the EM algorithm is by computing the log-likelihood after each iteration and stopping the algorithm when there is no significant change in this metric between two consecutive iterations. The log-likelihood is given by formula 7.

$$logl(\Theta) = \sum_{i=1}^{N} (log \sum_{k=1}^{K} \alpha_k p_k(x_i|z_k, \theta_k)) \quad (7)$$

IV. Implementation

In this work, we implemented a segmentation framework based on GMM and EM in Python following the OOP paradigm. The code of this framework can be found in https://github.com/kakou34/misa_lab. In this section, the implementation details of our work are described. All the following mentioned line numbers are from *src.py* file unless stated differently.

A. Inputs

To perform the segmentation, the framework needs to be provided with the volume to segment as well as the number of Gaussian Mixture Components to be used. The framework supports multivariate inputs, therefore each voxel can have one or more channels. Before testing our segmentation method on the provided brain MRI scans, we first perform skull-stripping to keep only the brain tissues. This is achieved using the provided ground truth mask by combining the 3 tissue classes GM, WM and CSF in a single class. Then, this mask is used before giving the MRI scan as an input to the segmentation method to filter out the background voxels. This way, the flattened input vector will only contain the voxels belonging to one of the 3 tissues.

B. Initialization

The first step of the GMM-EM segmentation method is to initialize the set of parameters Θ. We provide multiple initialization options for each parameter.

- Means: they can be initialized using 3 different approaches: by using random values selected from the given data samples, by first running k-means [1] on the data and using the resulting centroids as the initial means for EM, or by using predefined means provided as a parameter by the user. The size of the means matrix will be Kxd where K is the number of components and d is the dimension of the data (number of features) [lines 122-144]
- Covariance: when the means are initialized randomly or manually the same covariance matrix -the one obtained from all data points- is used for all the components. If k-means algorithm was run, then the labels assigned to each point are used to compute the covariance matrix of each of the clusters independently. Covariance matrices for each component will be of size dxd. [lines 147-152]
- Priors: priors can be non informative by giving equal probabilities to all components ($\alpha = \frac{1}{N}$) for each class, or using predefined priors provided by the user as parameters. The priors vector is of size 1xK. [lines 116-119]

[1]The Sci-kit learn version of k-means was used in this lab work

C. E-Step

In the expectation method [lines 234-243] the posterior probabilities (membership weights) are computed based on the current parameters following equation 3. The data samples vector x will have a dimension of Nxd with N being the total number of voxels to be classified and the resulting weights will have a dimension of NxK. It is important to highlight one more time that only the voxels inside the brain tissues mask are classified. To calculate the density of each component, the multivariate_normal_pdf function of the SciPy package was used to avoid long computation times in the case of for-loop-based implementation and the overflow/underflow errors in the case of matrix-based implementation of the Gaussian likelihood.

D. M-Step

In the maximization method [lines 245-268] the new parameters (means, covariance matrices and priors) are calculated for each component using the weights obtained from the E-Step. In this function we implemented equations 5, 6 and 7 as given in the previous section.

In order to better follow the optimization process, we created scatter plots (histograms in the case of 1D feature vectors) that show the current assignment of the data every few iterations [lines 313-348]. An example is presented in figure 1. It can be seen that unlike in the case of k-means (starting point), EM supports non-circular shapes of the clusters (correlation among variables in the Gaussian model) which helps to better model the data distributions.

E. Stopping Criteria

To decide when to stop the EM process, we used 2 stopping criteria:

- The log-likelihood: [lines 223-227] as described in the previous section, if the change in the log-likelihood between two consecutive iteration is lower than a user-defined tolerance parameter then the EM procedure stops.
- The maximum number of iterations: [line 210] if the algorithm does not converge according to the previous criterion after a predefined number of iterations then the EM procedure stops.

Figure 2 shows an example of the evolution of the log-likelihood during an EM process. It can be seen that the log-likelihood increases during the optimization process until it reaches a plateau and the improvement becomes ignorable.

F. Final Result

In the predict method [lines 164-177] the final segmentation is obtained by assiging each voxel to the GMM component that has the maximum final posterior probability.

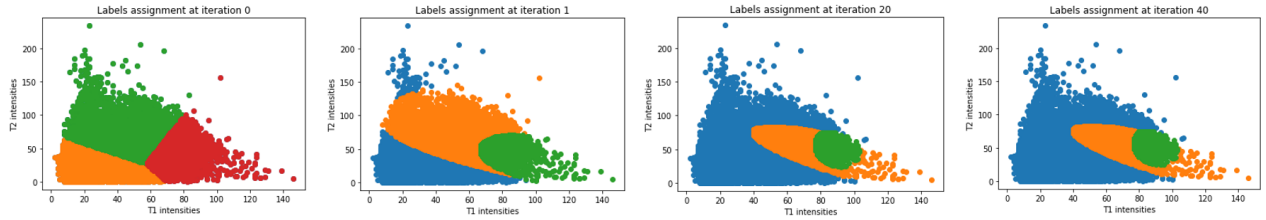

Fig. 1. Scatter plots of the data distribution of each cluster per iteration.

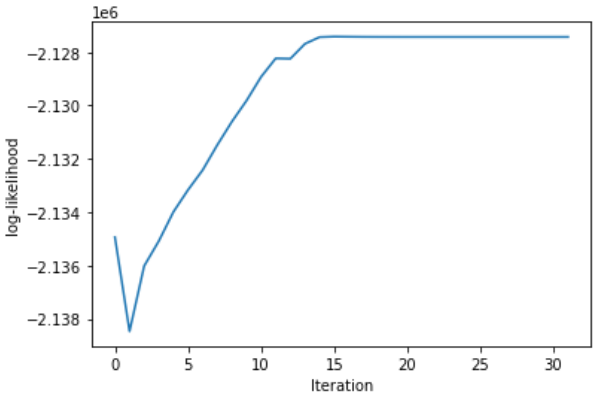

Fig. 2. Log-Likelihood per iterations.

G. Label Matching

In order to evaluate the segmentation results we need the label given to each tissue by the segmentation model to match the label given to the same tissue in the ground truth. To achieve this we created the $match\_pred\_w\_gt$ function [lines 33-92 in utils.py]. The mean intensities in each cluster are calculated for both the segmentation result and the ground truth and each cluster's label is matched with the one having the closest center in the ground truth.

V. Experimental Results

In this section we present the results of the multiple experiments that we conducted on the given dataset to solve the task of brain tissue segmentation using the GMM+EM segmentation framework that we implemented. The Dice score was used to assess the quality of the segmentation quantitatively. For all the following experiments, a tolerance of 10^{-5} and a maximum number of iterations of 100 were used.

A. Initialization

To investigate the effect of the different initialization methods we performed a set of experiments on the provided brain images. Table I summarizes the findings. The results are better appreciated in figure 3. By comparing the k-means initialization with the random one, we can see that both methods provide very similar results both in terms of means and standard deviation. This is an illustration that the EM algorithm is robust to random initialization. It it also important to highlight that using T1 scans provides a higher median dice score compared to the results obtained with the combination of both T1 and T2 modalities. However, if we see the mean values in Table I, we can see the opposite behaviour, this is caused by the larger dispersion we can find across the segmentation results for T1.

It is natural that using both modalities needs more computational time due to the higher dimension on the data. If we compare the two initialization methods, the extra time needed to compute the initial k-means is equivalent on average to the extra time needed by the random initialization case to converge. This is evidenced by comparing time and iterations for both algorithms provided in Table I. Figure 7 in the Appendix shows example segmentation results for each subject.

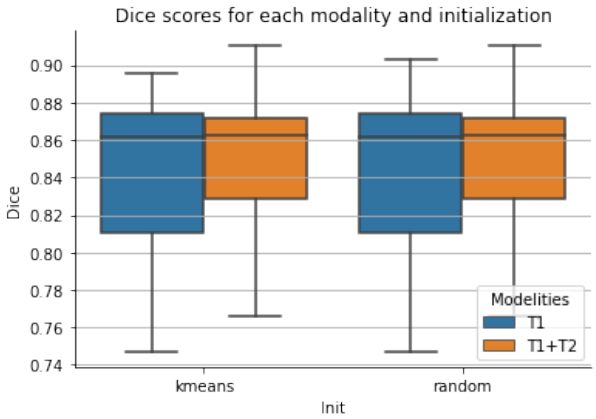

Fig. 3. Dice score boxplot across tissues and subjects for each modality and initialization type

B. Comparison with k-means and SPM

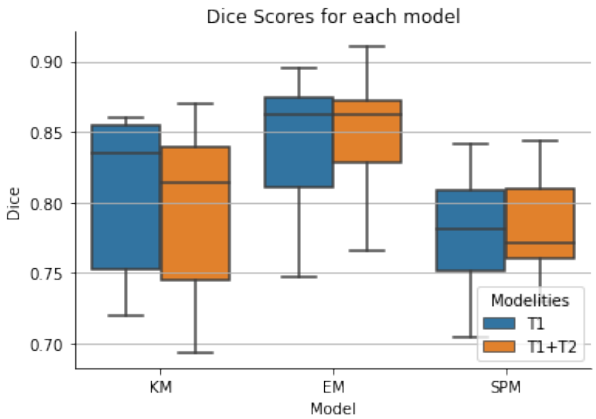

Fig. 4. Dice score boxplot across tissues and subjects for EM, K-means and SPM for each modality.

In a second experiment, we compared the results obtained using our best EM model obtained from the previous experiments with the results obtained using the SPM tool (Atlas-based

Initialization	Modality	Time [s]	Iterations	CSF Dice	GM Dice	WM Dice
K-means	T1	**7.4212** ± 0.7655	**30** ± 3	0.8723 ± 0.0134	0.8004 ± 0.0336	0.8595 ± 0.0403
	T1 + T2	12.8812 ± 1.9178	41 ± 4	**0.8745** ± 0.0327	**0.8109** ± 0.0311	**0.8641** ± 0.0339
Random	T1	**6.4090** ± 1.1694	**40** ± 4	0.8720 ± 0.0148	0.7991 ± 0.0344	0.8610 ± 0.0421
	T1 + T2	12.8445 ± 4.1135	55 ± 13	**0.8745** ± 0.0327	**0.8109** ± 0.0311	**0.8641** ± 0.0339

TABLE I
EXPERIMENTAL RESULTS FOR DIFFERENT INITIALIZATION OF THE EM ALGORITHM.
MEAN AND STANDARD DEVIATION ACROSS SUBJECTS ARE PROVIDED.

Model	Modality	Time[s]	Iterations	CSF Dice	GM Dice	WM Dice
GMM+EM	T1	7.3843 ± 0.7575	30 ± 3	0.8723 ± 0.0134	0.8004 ± 0.0336	0.8595 ± 0.0403
	T1 + T2	12.8716 ± 1.9321	41 ± 4	**0.8745** ± 0.0327	**0.8109** ± 0.0311	**0.8641** ± 0.0339
K-Means	T1	**2.7580** ± 0.5392	6 ± 4	0.8542 ± 0.0077	0.7447 ± 0.0326	0.8205 ± 0.0369
	T1 + T2	3.2609 ± 0.6968	**6** ± 3	0.8449 ± 0.0242	0.7314 ± 0.0466	0.8106 ± 0.0346
SPM	T1	-	-	0.7702 ± 0.0338	0.7483 ± 0.0316	0.8123 ± 0.0325
	T1 + T2	-	-	0.7759 ± 0.0256	0.7550 ± 0.0222	0.8143 ± 0.0318

TABLE II
EXPERIMENTAL RESULTS FOR EM, K-MEANS AND SPM
MEAN AND STANDARD DEVIATION ACROSS SUBJECTS ARE PROVIDED.

segmentation from Laboratory Work 1) and the segmentation obtained using k-means. The results of these experiments are presented in details in table II and summarized in figure 4.

As seen from the results, the k-means algorithm is the fastest in terms of computation time but it provides lower dice scores and suffers from a larger dispersion across subjects compared to the other models. EM provided better results compared to SPM with the latter being slightly less dispersed according to inter-quartile ranges. We can see that the previous observation between the use of one or two modalities still holds for GMM-EM and SPM. However, we can appreciate that in the case of K-Means algorithm, using only T1 information leads to better results.

To have a deeper look into the differences between the three models we provide figure 5 that summarizes the results obtained for each tissue type using the T1 scans only, and figure 6 that provides the results obtained when using both modalities.

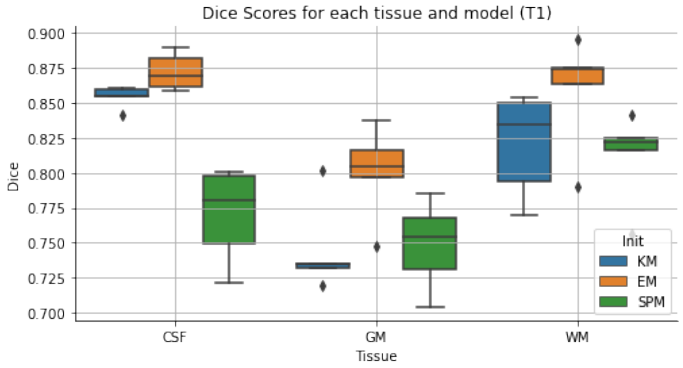

Fig. 5. Dice score across subjects for EM, K-means and SPM algorithms (T1)

It can be seen from the plots that the white matter tissue is the easiest to segment for all algorithms with EM outperforming both k-means and SPM whereas the gray matter is the hardest especially in the case of k-means. Figure 8 in the appendix provides visual segmentation examples for K-means and EM for each model (slice 25).

However it is important to say that the extension of each tissue type is not equivalent inside the brain might bias the obtained results. the small/thin and very tortuous nature of GM

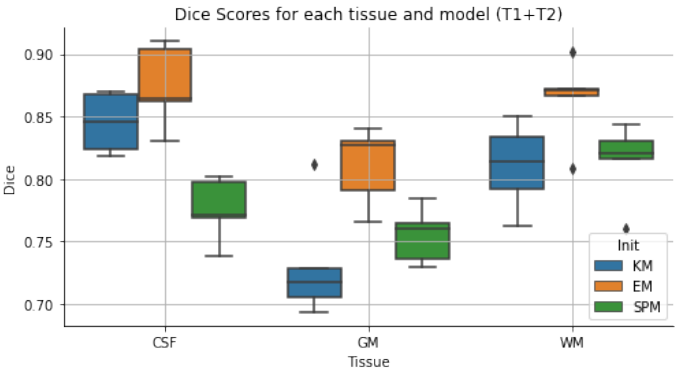

Fig. 6. Dice score across subjects for EM, K-means and SPM algorithms (T1+T2)

with very winding contours both on the CSF side and WM side, might have an impact on the dice scores. Inclusion of other metrics, such as Hausdorff distance could give us a richer comprehension of the segmentation results obtained.

Finally, it is necessary to state that the comparison against SPM results is not a fair one. The results obtained with the private software Matlab, were computed using 5 components and the complete volume. In order to correctly compare the performances the developed algorithm should had been run in the same conditions.

VI. PROJECT MANAGEMENT DETAILS

The project was executed in not much more time than the assigned one. We highly exploited the Laboratory Work classes, using a pair programming approach in order to code the required methods. We first conceived a working version of the algorithm in a function based paradigm and later reformulated our code to object oriented paradigm for better clarity and usability of the algorithm. We mainly mimic the scikit-learn API fashion, implementing the methods fit, predict, fit_predict for our ExpectationMaximization class. The low computational time allowed us to run several times the code in search for solutions to some bugs. In addition to this, the scatter plots served us a lot to understand were our problems where.

APPENDIX

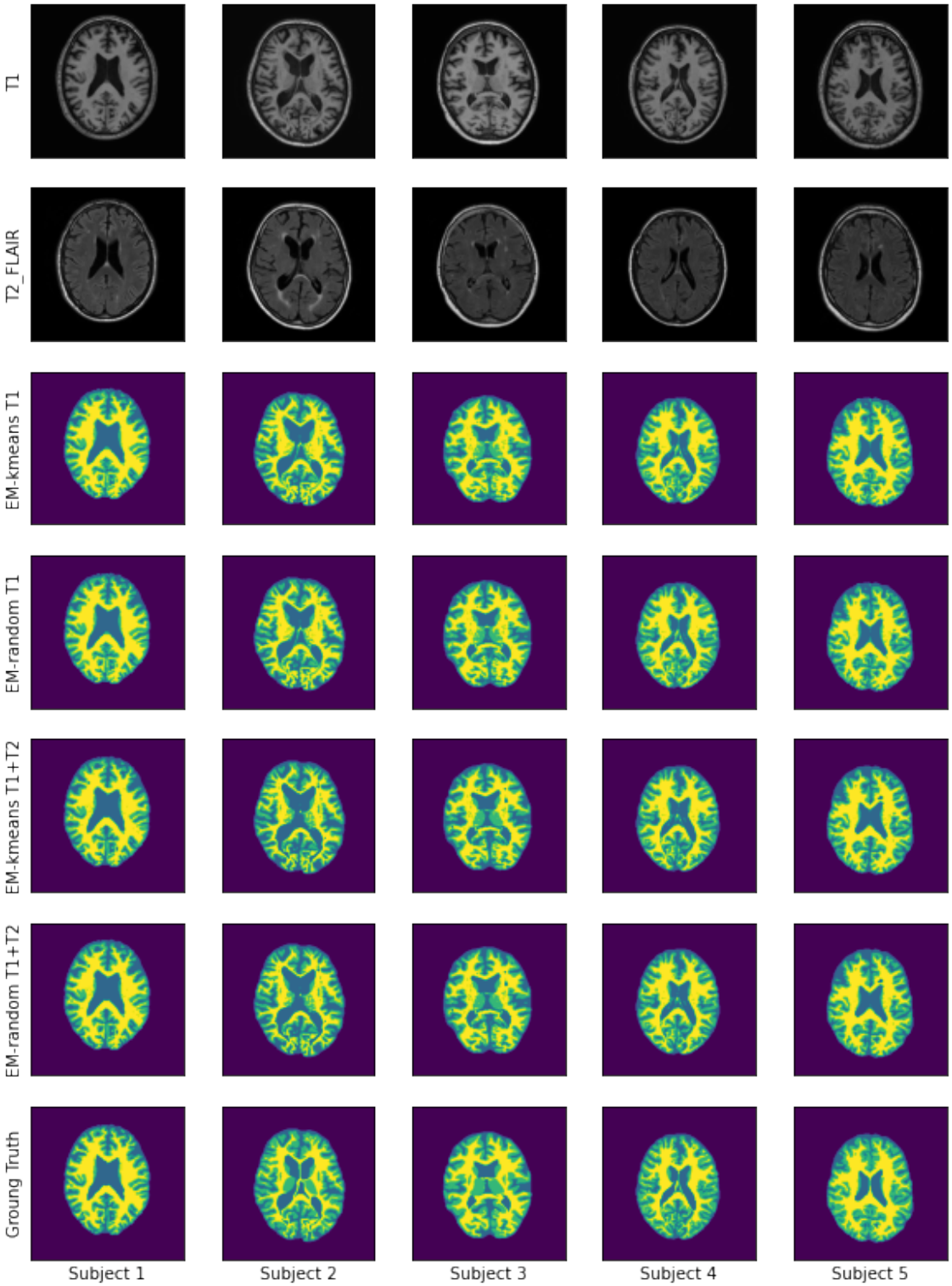

Fig. 7. Segmentation examples for each subject using different initialization, axial slice n°25

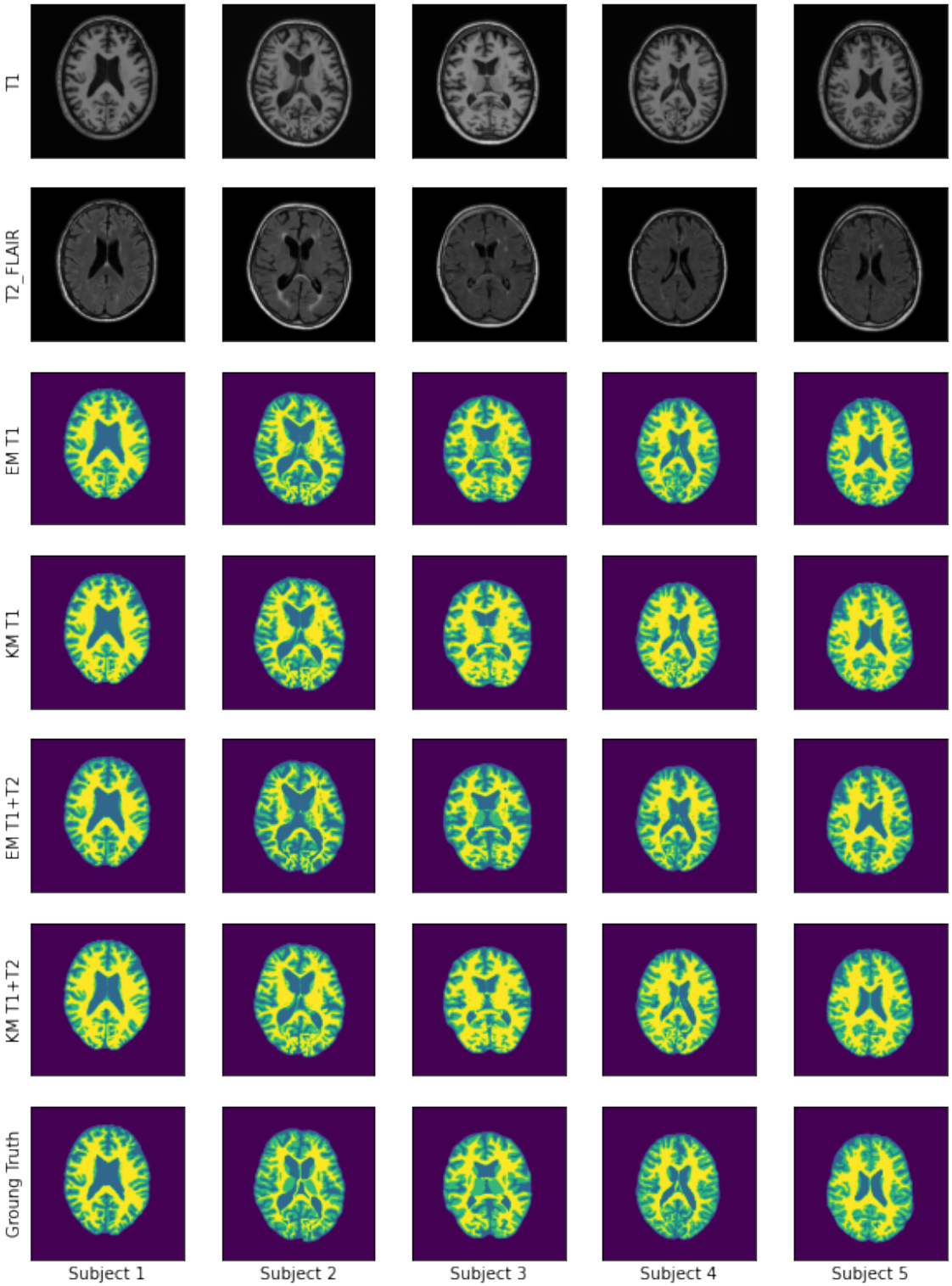

Fig. 8. Segmentation examples for each subject using different models, axial slice n°25