Universitat De Girona

Medical Image Segmentation and its Applications

Image Segmentation (EM Algorithm)

Students:
Ali Berrada
Ama Katseena Yawson

Supervisor:
Xavier Lladó

November 5, 2018

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Segmentation of non-trivial images is one of the most difficult tasks in image analysis. It can be defined as the partition of an image into several regions which exhibit same characteristics. This image processing technique is often the first step for image analysis and is a key basis of many higher-level activities such as visualization, compression, medical diagnosis and other imaging applications [2]. Figure 1 is an example of image segmentation of the brain. There exist several approaches to segment an image. Some of these techniques include edge based, region based, clustering based and many others.

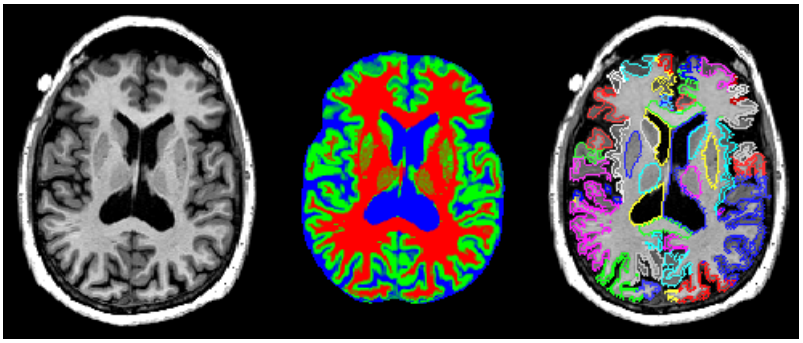

Figure 1: An example of Brain Image Segmentation [1]

1.2 Problem Definition

The quantitative evaluation of brain MRI images are associated with an estimation of tissue density and partitioning them into regions of interest. In the last decades, this field of interest has received massive attention as a result of improved accuracy and resolution of MRI systems. Additionally, MRI images provide high contrast between soft tissues in 3D. However, the amount of data obtained is very numerous for manual analysis especially in the case of diagnosis. Hence, there is a need for automated techniques of image analysis to perform segmentation of Brain MRI images into 3 different tissue classes which includes gray matter, white matter and cerebrospinal fluid. In this exercise, our main goal was to develop a clustering based segmentation approach from scratch using the Expectation-Maximization algorithm to segment brain MRI images (T1 and T2-Flair) into the three main brain tissues: white matter (WM), gray matter (GM) and cerebrospinal fluid (CSF).

2 EM Algorithm

Expectation Maximization (EM) algorithm is a clustering based segmentation approach which is developed and employed by several different researchers [2]. This produces good outputs when the input image is less noisy. It can be defined as an effective iterative approach to determine the Maximum Likelihood (ML) estimate in the presence of missing or hidden data [3]. In this algorithm, the input image is regarded as a gaussian mixture model. Each class and pixel intensity is represented as a gaussian model and observed value respectively.

For a given set of observed data (pixel intensities):

$$X = \{ x_i \mid i = 1, 2, ..., N \} \tag{1}$$

From Equation 1, this can be modelled to generate a gaussian mixture of random processes as shown in Equation 2 below,

$$X = \{ X_1, X_2, ..., X_K \} \tag{2}$$

with the corresponding joint probabilities (Equation 3)

$$JointProbability = f(X_1, X_2, ..., X_K) \tag{3}$$

where K is the number of classes. Equation 3 represent independent identically distributed random variables and can be written as:

$$f(X_1, X_2, ..., X_K) = \sum_{k=1}^{K} p_k G(x_i \mid \theta_k^t) \tag{4}$$

where $f(x, \theta_k)$ for all $k = 1, 2, 3, ...K$ represent probability distribution function for the random variables X_k and $\theta_k = \{ \mu_k, \sigma_k) \}$. This defines the input parameters for the distribution function k.

The parameter vector is given as $\phi = \{ p_1...p_k, \mu_1...\mu_k, \sigma_1...\sigma_k \}$. p_k is the mixing proportion, $(0 \leq p_k \leq 1, \forall k = 1...K, \sum_k p_k = 1)$

The EM algorithm is made up of two phases known as the expectation and maximization. In the expectation phases, the hidden data are estimated using the observed data and current estimate of the model parameters. This is done using the conditional expectation. The maximization phase is done immediately after the expectation phase. The likelihood function is maximized with the assumption that the hidden data is known. The estimate of the missing data from the expectation phase are used in place of the actual missing data. This procedure is continued until convergence is reached at a maximum likelihood.

The "estimation" part of the algorithm refers to the computation of the "membership weight" for every data point in every cluster (5) given the cluster parameters. To be able to start the very first "estimation" step, the cluster parameters are estimated either arbitrarily of heuristically. The "maximization" part, on the other hand, refers to the computation of the cluster parameters —cluster weights (7), means (8), covariances (9) —given the "membership weights" found in the "estimation" step. The algorithm converges when the log-likelihoods between 2 iterations display virtually no difference.

$$w_{ik} = p(\Theta_k \mid \underline{\mathrm{x}}_i) = \frac{p_k(\underline{\mathrm{x}}_i \mid \theta_k).\alpha_k}{\sum_{m=1}^{k} p_m(\underline{\mathrm{x}}_i \mid \theta_m).\alpha_m} \tag{5}$$

where: $1 \leq k \leq K$, $1 \leq i \leq N$, and

$$p_k(\underline{\mathrm{x}}_i \mid \theta_k) = \frac{1}{(2\pi)^{\frac{d}{2}} \left|\sum_k\right|^{\frac{1}{2}}} e^{-\frac{1}{2}(\underline{\mathrm{X}}-\mu_i)^t \sum_k^{-1}(\underline{\mathrm{X}}-\mu)} \tag{6}$$

$$a_k^{new} = \frac{N_k}{N}, 1 \leq k \leq K \tag{7}$$

$$\mu_k^{new} = (\frac{1}{N_k}) \sum_{i=1}^{N} w_{ik}.\underline{\mathrm{x}}_i, 1 \leq k \leq K \tag{8}$$

$$\sum_k^{new} = (\frac{1}{N_k}) \sum_{i=1}^{N} w_{ik}.(\underline{\mathrm{x}}_i - \mu_k^{new})(\underline{\mathrm{x}}_i - \mu_k^{new})^t, 1 \leq k \leq K \tag{9}$$

$$logl(\Theta) = \sum_{i=1}^{N} log(p(\underline{\mathrm{x}}_i \mid \Theta) = \sum_{i=1}^{N} (log \sum_{k=1}^{K} a_k p_k(p_k(\underline{\mathrm{x}} \mid z_k, \theta_k)) \tag{10}$$

3 Implementation

The algorithm was implemented in MATLAB R2018b and divided into several files, in line with the principle of separation of concerns.

3.1 main.m

This script file is the starting point of the program. It reads the brain data, namely T1, T2-FLAIR and ground truth volumes using MATLAB's built-in function "niftiread()". Some pre-processing steps are done such as building a mask image that will serve to ignore non-ROI pixels such as the background and the skull contours, in addition to smoothing the T2-FLAIR volume with a Gaussian filter to improve the segmentation results. Also, it builds the Nx2 feature vector X by reshaping all elements of T1 and T2-FLAIR into single column vectors and assigning them, respectively, as the first and second columns of X. Afterwards, the script passes the vector X and the mask to the function "expect_max()" to obtain the labelled pixels as a Nx1 array. The latter is then passed by the script to the "relabel()" method to ensure the same regions from the segmentation result and the ground truth have the same label tags. The vector of labels is finally reshaped into the original volume shape. Additionally, this script computes the Dice similarity index and total elapsed time, and saves the segmented volume into a NifTi file.

3.2 expect_max.m

The function "expect_max()" is the core and most important part of the segmentation algorithm. It iterates between a series of expectation and maximization steps, trying to find the best parameters of the Gaussian mixture models representing the data. This method requires primarily the feature vector X but we also provide the mask vector to exclude non-ROI pixels from the clustering.

To initialize the cluster means, we have used MATLAB's "kmeans()" function. With the clustering performed by this function, we computed a simple covariance for each group of feature points using MATLAB's "cov()" and set these matrices as the initial covariances for our algorithm. This initialization of covariances is important as random initialization may result in having likelihoods equal to 0 for some features under one or more density functions which eventually breaks the code due to divisions by zero. The cluster weights are initialized equally as 1/3.

Now that we have the initial cluster parameters, the E-step starts. We compute how likely each data point came from each cluster (5) and store the result as an Nx3 matrix called R. Each row of R contains the membership weight of a feature point for cluster 1, 2 and 3 (hence the existence of 3 columns for the R matrix). We leveraged MATLAB's "mvnpdf()" function to compute the probability density (6).

Given the cluster responsibilities (i.e. matrix R), we move to the M-step where the new cluster parameters are computed, namely the cluster weights (7), means (8) and covariance matrices (9).

As the EM algorithm is an iterative process, the stopping criteria was set to either a maximum of 10 iterations (enough for our scenario) or a barely changing log likelihood value between 2 successive iterations. In the implementation, the log-likelihood was not literally computed as in (10) as part of its computation is already embedded in the R matrix.

3.3 label.m

The function "label()" helps in assigning labels (1, 2 or 3) to every feature point given the matrix of responsibilities R. A feature point will be labeled according to the cluster which holds the highest responsibility over it. For simplicity, the label of a cluster, and hence the label of feature points it has most responsibility over, is the column index in the R matrix. Implementation-wise, we first convert the R matrix into one-hot encoding. This makes it easier to assign the labels since each column of the R matrix contains now binary values which act as indices of which feature points are going to be assigned the column index as the label.

3.4 relabel.m

The purpose of the function "relabel()" is to swap the label tags, if necessary, of a segmented image called target. In addition to the target image, this function requires a reference image, which normally is the ground truth image, to ensure that the same regions in both images have the same label tags. Since, there are 3 clusters, the swapping of labels might be needed for 2 or 3 regions. We detect this need by computing the Dice score between the 2 images; the result is a 3-element vector, one element for each cluster. If exactly 2 elements have a value below 0.5, then the labels of the corresponding regions are swapped. On the other hand, if the Dice score is low for every class, then a swap of labels must happen among the 3 classes and in this situation only 2 possibilities are to be exploited: what used

to be labelled region 1, region 2 and region 3, respectively, should change to either region 2, region 3 and region 1 or region 3, region 1 and region 2. Therefore, we try with the first labels remapping option and recompute the Dice score, if all values are still low we remap following the second option. Naturally, if the Dice index is not improving, this indicates that the problem is with the segmentation result rather than a mismatch of labeling tags.

4 Results and Discussion

Table 1 shows the results obtained from running our algorithm over the 5 brain data. The segmentation takes about 5 seconds on an ordinary modern-day computer which is relatively fast. The Dice similarity index for each region is at least 80%. From the scores, the segmentation of the CSF is often better than other components. As a note, these scores are obtained with the T2-FLAIR images being smoothed prior to segmentation. Smoothing helped to slightly improve the scores, but for Brain 2 the improvement was very large as the Dice indices obtained without smoothing were 0.85, 0.40 and 0.57 for CSF, GM and WM respectively as the T2-FLAIR slices in Brain 2 display poor contrast between the white matter and grey matter. For qualitative comparison, we displayed the slice 25 of each Brain data in Figure 2. The overall glance without stopping at details gives close look between the ground truth and segmented images.

	Dice				
Images	CSF	Gray Matter	White Matter	Average Dice	Execution time (sec)
Brain 1	0.91	0.82	0.87	0.866666667	4
Brain 2	0.87	0.8	0.82	0.83	5
Brain 3	0.89	0.81	0.85	0.85	4
Brain 4	0.89	0.83	0.86	0.86	4
Brain 5	0.88	0.86	0.9	0.88	4

Table 1: Quantitative evaluation and execution time for 5 Brain data

The implementation of our EM algorithm gives initial satisfactory results given only 2 sources of data images are being exploited for the segmentation. Greater improvements can be made by extending this algorithm into a larger one that incorporates additional methods such as an atlas-based method.

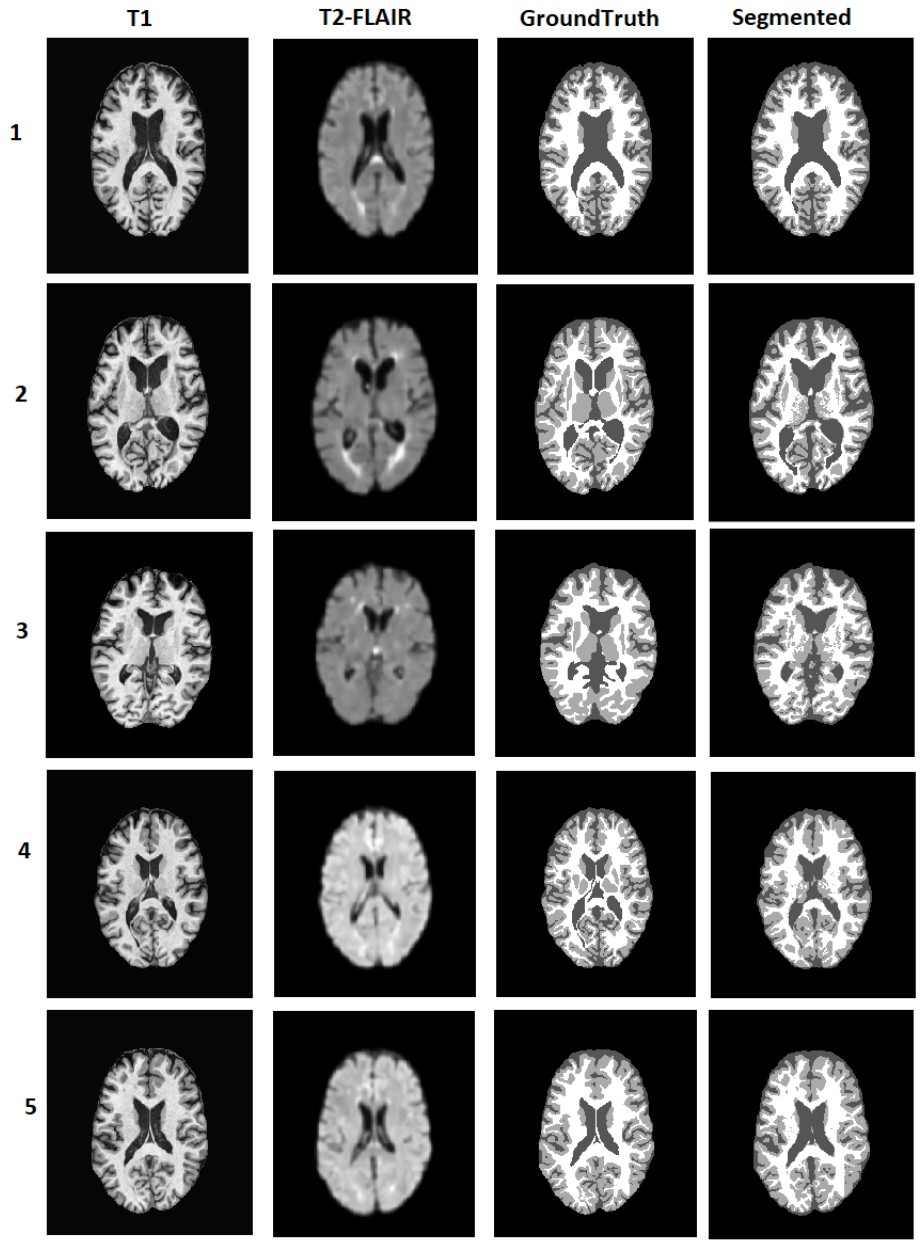

Figure 2: Qualitative evaluation of slice 25 for 5 Brain data

5 Project Management

As a way of catching up with deadline in this short period, proper planning was managed using the Gantt chart as shown in Figure 2.

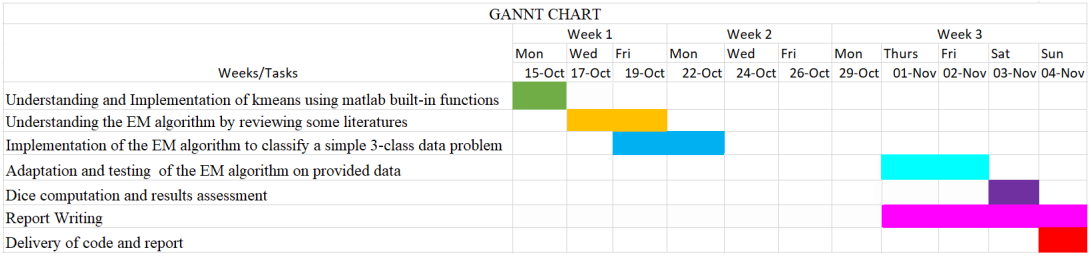

Table 2: Project Management using the Gannt chart

This exercise was divided into 7 tasks. The details of each task are explained below.

1. **Understanding and experimenting with k-means clustering**
   The purpose is to familiarize with one of the simplest methods for clustering data. It can also help in giving a better insight and understanding of the EM algorithm. Furthermore, the intention was also to be able to later on use the results from k-means as the initialization part of our EM algorithm. MATLAB's "kmeans()" function was used in test examples.

2. **Understanding the EM algorithm**
   Initially, our grasp of the ins and outs of the EM algorithm was limited even though the whole picture seemed simplistic. Reviewing some literature works and online materials was deemed necessary. By the end of this task, our understanding of the EM algorithm has enhanced and our confidence to start the coding phase has increased.

3. **Implementation of the EM algorithm for a simple 3-class data problem**
   In this task, we generated data from 3 different normal distributions and we sought to cluster the data by implementing the EM algorithm. The purpose here was to focus on correctly implementing the EM algorithm therefore we exercised on a simple problem of which we know the expected outcome. With some help from the Professor, we were able to achieve the correct results (see "Examples\em.m" in the submitted code folder).

4. **Adaptation the EM algorithm to the provided data**

This task was planned for a period with less stress on other courses to give it more attention given we had solid grounds from achieving the previous important tasks at a relatively early period; hence the gap between this task and the one preceding it (in addition to balance the needs from other courses). The actual time taken to achieve this task was 1 day more than initially speculated. This is partly due to a review of the code after which parts to be optimized were identified and amended. With the extra time spent on this task, we were able to achieve a 300% increase in speed. Additionally, the mediocre results obtained with Brain 2 data pushed for more trials to improve the algorithm until adopting the solution of smoothing the T2-FLAIR slices.

5. **Report Writing**
   With enough understanding of the theory and successful implementation of the algorithm for a small scale problem, the report redaction started simultaneously with the adaptation of the algorithm to the real data.

6. **Dice computation and results assessment**
   This step naturally comes at the end when we are able to run our algorithm on the data and make numerical and visual assessments.

7. **Delivery of code and report**
   As indicated on the moodle, we set the delivery of our code and report inline with the given deadline.

6 Conclusion

Image segmentation is often the one of the initial steps for image analysis and is a key basis of many higher-level activities such as visualization, compression, medical diagnosis and other imaging applications. There exist several approaches to segment an image. Some of these techniques include edge based, region based, clustering based and many others. In this exercise, we developed a clustering based segmentation approach from scratch using the Expectation-Maximization algorithm to segment brain MRI images (T1 and T2-Flair) into the three main brain tissues: white matter (WM), gray matter (GM) and cerebrospinal fluid (CSF). Based on the qualitative and quantitative evaluation, our algorithm showed overall acceptable results given only 2 types of input images guided the segmentation process. We were also able to overcome the mediocre results from Brain 2 data by smoothing the T2-FLAIR slices with a Gaussian filter. Nevertheless, greater improvements can be made by extending this algorithm into a larger one that incorporates additional methods such as an atlas-based method.

References

[1] Rolf Heckemann. Segmenting brain images with maper. https://soundray.org/maper/. Accessed: 2018-11-02.

[2] Anita Khanna, Meenakshi Sood, and Swapna Devi. Us image segmentation based on expectation maximization and gabor filter. *International Journal of Modeling and Optimization*, 2(3):230, 2012.

[3] Sean Borman. The expectation maximization algorithm-a short tutorial. *Submitted for publication*, pages 1–9, 2004.