AI in Medical Image Processing

HW2 - Theory Questions

Dr.Fatemizadeh

Sharif University of Technology

Electrical Engineering

Tina Halimi _ 400101078

Date: September 6, 2025

Contents

1 Q1 **2**

1.A The GMM algorithm and its limitations 2

1.B SVGMM and DeepG, and how they improve on GMM 2

1.B.a SVGMM: Spatially Variant Gaussian Mixture Model 2

1.B.b DeepG: CNN-based Gaussian Mixture Model 3

1.C DeepSVG and its advantages over previous models 3

1.D Why replace the Expectation step with a gradient-based CNN update 4

2 Q2 **6**

2.A Fuzzy C-Means (FCM) Algorithm – Full Explanation 6

2.B Derivation of the Fuzzy C-Means Membership Formula 7

1 Q1

1.A The GMM algorithm and its limitations

The Gaussian Mixture Model (GMM) is a classical probabilistic approach for unsupervised image segmentation. It assumes that the intensity of each pixel in an image is generated from one of several Gaussian distributions, each corresponding to a different class or region. Mathematically, for K classes, each class k is described by a mean vector μ_k, a covariance matrix Σ_k, and a mixing coefficient π_k, which represents the prior probability of class k. The probability density of a pixel $I(x)$ is thus expressed as a mixture of Gaussians:

$$p(I(x)|\pi, \mu, \Sigma) = \sum_{k=1}^{K} \pi_k \mathcal{N}(I(x)|\mu_k, \Sigma_k)$$

where $\mathcal{N}$ is the multivariate Gaussian density. The overall likelihood of the image is obtained by multiplying these pixel-wise probabilities. In practice, we minimize the negative log-likelihood (NLL) over all pixels to estimate the model parameters.

To perform this optimization, GMM uses the Expectation-Maximization (EM) algorithm. The EM algorithm alternates between two steps. In the Expectation step (E-step), it computes the posterior probability that each pixel belongs to each class, given the current parameter estimates. Then in the Maximization step (M-step), it updates the parameters π_k, μ_k, Σ_k based on these probabilities. This process is repeated until convergence, after which each pixel is assigned the class with the highest posterior probability.

However, GMM suffers from several important limitations when used for image segmentation. First, it assumes that pixels are statistically independent, which means it ignores the strong spatial correlations between neighboring pixels. As a result, GMM often produces noisy, speckled segmentation masks without spatial coherence. It is also highly sensitive to initialization, since the likelihood surface can have many local maxima, and different random starting points can lead to different results. Moreover, the method is computationally demanding for large images because the EM algorithm must iterate many times. Finally, GMM assumes that each class follows a Gaussian distribution, which may not always accurately model the true intensity distributions found in real-world images, especially in medical imaging.

1.B SVGMM and DeepG, and how they improve on GMM

1.B.a SVGMM: Spatially Variant Gaussian Mixture Model

One improvement over the basic GMM is the Spatially Variant Gaussian Mixture Model (SVGMM). Unlike the standard GMM, which uses a single global mixing coefficient π_k for each class across the entire image, SVGMM introduces pixel-specific mixing proportions $\Pi_k(x)$. This allows the model to adapt more flexibly to local variations in the image. The negative log-likelihood for SVGMM becomes:

$$NLL_V = -\frac{1}{|\Omega|} \sum_x \log \Big(\sum_k \Pi_k(x) \mathcal{N}(I(x)|\mu_k, \Sigma_k) \Big)$$

In this model, the EM algorithm works similarly to GMM, but in the M-step, the pixel-specific label probabilities $\Pi_k(x)$ are updated directly to match the computed posteriors. An interesting property of SVGMM is that, under certain mild conditions, these label probabilities converge toward binary values, meaning the final segmentation masks can be obtained directly without an extra labeling step.

While SVGMM improves flexibility and produces better segmentation than the classical GMM, it still treats pixels independently. It does not explicitly enforce spatial smoothness or take advantage of information from neighboring pixels. So, although it helps adapt to local class proportions, it still cannot fully resolve the issue of noisy segmentations.

1.B.b DeepG: CNN-based Gaussian Mixture Model

Another approach to improving GMM is DeepG, which integrates deep learning into the GMM framework. In DeepG, the E-step of the EM algorithm is replaced by a convolutional neural network (CNN), which predicts the soft label probabilities for all pixels in a single forward pass. The CNN architecture, often a U-Net, naturally incorporates spatial context because convolutional layers aggregate information from neighboring pixels and larger receptive fields. After the CNN outputs the label probabilities, the M-step is still used to update the Gaussian parameters μ_k and Σ_k.

This approach has two major advantages over classical GMM. First, it introduces spatial regularization, since CNNs inherently learn smooth and spatially coherent representations. This significantly reduces the random noise typical of GMM segmentations. Second, once the CNN is trained, it can produce label probabilities for new images without requiring multiple EM iterations, making inference much faster and more practical.

However, DeepG still uses global mixing coefficients, which means it lacks the local adaptability of SVGMM. Thus, while it improves spatial coherence, it does not fully capture spatial variations in label proportions.

In summary, SVGMM improves GMM by allowing spatially adaptive label probabilities, whereas DeepG improves GMM by introducing spatial awareness and faster inference through CNNs. Both methods partially address GMM's weaknesses, but each has limitations that are resolved when they are combined in DeepSVG.

1.C DeepSVG and its advantages over previous models

DeepSVG combines the strengths of both SVGMM and DeepG. It retains the pixel-specific label proportions of SVGMM, which makes the model locally adaptive, but it also replaces the E-step with a CNN-based gradient descent step, as in DeepG. In DeepSVG, the CNN predicts label probabilities for the entire image, but unlike DeepG, these probabilities are not constrained to be global—they vary spatially just like in SVGMM. After each CNN forward pass, a conventional M-step updates the Gaussian parameters μ_k and Σ_k. Then the CNN parameters are updated by gradient descent to minimize the negative log-likelihood of the spatially variant model. This iterative combination continues until convergence.

The key advantage of DeepSVG is that it achieves the best of both worlds. By leveraging a CNN, it introduces spatial regularization and learns a "deep image prior," meaning that the segmentation respects local image structure such as edges and textures. At the same time, by allowing spatially variant label proportions, it avoids the rigid assumption of global class probabilities. As a result, DeepSVG produces smoother, more coherent segmentation masks and improves segmentation accuracy over both SVGMM and DeepG. In the experiments presented in the paper, DeepSVG achieved higher Dice similarity coefficients compared to all other models, especially in medical image segmentation tasks where spatial consistency is crucial.

Moreover, DeepSVG can easily incorporate additional regularization terms into its loss function. For example, the authors showed that including a simple prior on the mean tissue intensities further improved performance. After training on multiple images, the resulting CNN can generalize to unseen images without needing further optimization, which makes DeepSVG not only more accurate but also more practical for real applications.

1.D Why replace the Expectation step with a gradient-based CNN update

In the classical EM algorithm, the E-step computes the label probabilities for each pixel based solely on its intensity, assuming that each pixel is independent of its neighbors. This makes it impossible to capture the natural spatial correlations that exist in real images, especially in medical scans where neighboring pixels often belong to the same tissue. By replacing the E-step with a CNN gradient step, the model can process the entire image holistically. CNNs are inherently spatial because convolutional filters aggregate information from local neighborhoods. This means that the CNN can implicitly learn to enforce spatial smoothness and respect image boundaries.

Another reason for replacing the E-step is that it enables end-to-end optimization with gradient descent. Instead of updating label probabilities pixel by pixel, the CNN parameters are optimized globally to minimize the negative log-likelihood of the mixture model. This allows the model to incorporate additional regularization terms easily and adapt more flexibly to the training data. Once trained, the CNN can directly predict label probabilities in a single forward pass, eliminating the need for iterative EM optimization for each new image.

Thus, the gradient-based CNN update replaces a local, memoryless computation with a learned, spatially aware predictor, which significantly improves the quality and efficiency of unsupervised segmentation.

Comparison Table

Summary

In summary, GMM provides a simple unsupervised segmentation method but suffers from noisy results due to its pixel independence assumption. SVGMM improves flexibility by allowing spatially variant label probabilities, while DeepG uses a CNN to introduce spatial awareness and speed up inference. DeepSVG combines both improvements by integrating a CNN with SVGMM, resulting in smoother, more accurate segmentation. Replacing the E-step with a CNN gradient step allows the model to exploit neighborhood information, learn deep image priors, and achieve better performance, as demonstrated in medical image segmentation experiments.

Method	**Main Idea**	**Advantages**	**Limitations**
GMM	Mixture of Gaussians with EM algorithm	Simple, fully unsupervised, no training required	Ignores spatial correlation, noisy segmentation, sensitive to initialization
SVGMM	Pixel-specific label proportions $\Pi_k(x)$	Adapts locally, sometimes avoids final labeling step	Still treats pixels independently, no explicit smoothing
DeepG	CNN replaces E-step for global GMM	Introduces spatial awareness, smooth results, faster inference after training	Still uses global mixing weights, less flexible than SVGMM
DeepSVG	CNN + SVGMM pixel-wise + spatial prior	Best segmentation quality, smooth + adaptive, can add regularization, generalizes to new images	More complex, requires CNN training

Table 1: Comparison of GMM-based segmentation methods

.

2 Q2

2.A Fuzzy C-Means (FCM) Algorithm – Full Explanation

Fuzzy C-Means (FCM) is a soft clustering algorithm that extends the traditional K-Means approach by allowing each data point to belong to multiple clusters simultaneously with varying degrees of membership. Unlike K-Means, which performs hard assignments, FCM provides a more flexible solution that is especially useful when cluster boundaries overlap.

The algorithm seeks to minimize the **fuzzy objective function**:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} \mu_{ij}^m \, \|x_i - c_j\|^2,$$

where x_i is the i-th data point, c_j is the center of cluster j, μ_{ij} is the membership degree of x_i in cluster j, and $m > 1$ is the **fuzziness parameter** controlling the softness of clustering. When m approaches 1, the clustering becomes more like hard clustering (K-Means), while larger m produces more fuzzy memberships.

Minimizing this function requires finding both the optimal **cluster centers** and the **membership matrix**. FCM achieves this through an iterative process: it alternates between updating the cluster centers as weighted means of the data points and recalculating the memberships based on the distances to the updated centers. The memberships satisfy the constraint $\sum_{j=1}^{C} \mu_{ij} = 1$ for each data point, ensuring that every point's memberships collectively sum to a full degree of belonging.

How FCM differs from K-Means

Aspect	**K-Means**	**Fuzzy C-Means**
Clustering type	Hard clustering: each point belongs only to one cluster	Soft clustering: each point can belong to multiple clusters
Membership values	Binary (0 or 1)	Continuous between 0 and 1
Objective function	Minimize sum of squared distances	Minimize fuzzy-weighted sum of squared distances
Cluster overlap	Cannot handle overlap	Naturally handles overlapping clusters
Flexibility	Less flexible, rigid boundaries	More flexible, smooth transitions
Complexity	Slightly lower	Slightly higher due to fuzzy memberships

Thus, while K-Means is simpler and faster, FCM provides a richer representation of clustering, especially in ambiguous regions.

Algorithm Steps

1. **Initialization**

   - Choose the number of clusters C and set the fuzziness parameter $m > 1$.
   - Initialize the **membership matrix** μ_{ij} randomly, making sure that $\sum_{j=1}^{C} \mu_{ij} = 1$ for every data point i.

2. **Compute Cluster Centers**

   - For each cluster j, compute the updated center using the weighted average of all points:

$$c_j = \frac{\sum_{i=1}^{N} \mu_{ij}^m \, x_i}{\sum_{i=1}^{N} \mu_{ij}^m}.$$

3. **Update Membership Values**

   - For each data point x_i and cluster j, update the membership degree:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}.$$

4. **Check Convergence**

   - Compare the updated membership matrix with the previous iteration.
   - If the change is smaller than a predefined threshold, stop; otherwise, repeat from Step 2.

Intuition

This algorithm ensures that a data point closer to a cluster center has a higher membership for that cluster and a lower membership for more distant clusters. The fuzziness parameter m controls how strongly the membership is influenced by distance: when m is closer to 1, the memberships tend toward crisp values (0 or 1, like K-Means), while higher values of m spread the memberships more evenly across clusters.

For example, if a data point lies exactly between two cluster centers, FCM might assign it a membership of 0.5 in each cluster, whereas K-Means would arbitrarily choose just one cluster.

2.B Derivation of the Fuzzy C-Means Membership Formula

The Fuzzy C-Means algorithm minimizes the objective function

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} (\mu_{ij})^m \, \|x_i - c_j\|^2,$$

where $m > 1$ is the fuzziness parameter, x_i is the i-th data point, c_j is the center of cluster j, and μ_{ij} is the degree of membership of x_i in cluster j.

The memberships for each point must satisfy the constraint

$$\sum_{j=1}^{C} \mu_{ij} = 1, \quad \forall i = 1, \ldots, N.$$

1. Introduce the Lagrangian

To minimize J_m with the membership constraint, we build a **Lagrangian function** for each data point x_i:

$$\mathcal{L} = \sum_{j=1}^{C} \mu_{ij}^{m} \|x_i - c_j\|^2 - \lambda_i \left(\sum_{j=1}^{C} \mu_{ij} - 1 \right),$$

where λ_i is a Lagrange multiplier for the constraint of point x_i.

2. Differentiate with respect to μ_{ij}

We now take the partial derivative of $\mathcal{L}$ with respect to μ_{ij} and set it to zero:

$$\frac{\partial \mathcal{L}}{\partial \mu_{ij}} = m\, \mu_{ij}^{m-1}\, \|x_i - c_j\|^2 - \lambda_i = 0.$$

From this, we can solve for λ_i:

$$\lambda_i = m\, \mu_{ij}^{m-1} \|x_i - c_j\|^2.$$

Since λ_i is the same for all clusters j for a fixed i, it means for any two clusters j and k:

$$m\, \mu_{ij}^{m-1}\, \|x_i - c_j\|^2 = m\, \mu_{ik}^{m-1}\, \|x_i - c_k\|^2.$$

We can cancel m from both sides, giving:

$$\mu_{ij}^{m-1} \|x_i - c_j\|^2 = \mu_{ik}^{m-1} \|x_i - c_k\|^2.$$

3. Express the ratio of memberships

Rearranging the above gives the ratio between two memberships:

$$\frac{\mu_{ij}^{m-1}}{\mu_{ik}^{m-1}} = \frac{\|x_i - c_k\|^2}{\|x_i - c_j\|^2}.$$

Taking both sides to the power $\frac{1}{m-1}$, we get:

$$\frac{\mu_{ij}}{\mu_{ik}} = \left(\frac{\|x_i - c_k\|}{\|x_i - c_j\|} \right)^{\frac{2}{m-1}}.$$

So the membership ratio depends on the relative distances of the point to the two cluster centers.

4. Apply the sum-to-one constraint

We also know that the memberships for a given data point must sum to one:

$$\sum_{j=1}^{C} \mu_{ij} = 1.$$

From the ratio relationship, we can express any membership μ_{ik} in terms of μ_{ij}:

$$\mu_{ik} = \mu_{ij} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}.$$

Now substitute this into the sum constraint:

$$\sum_{k=1}^{C} \mu_{ik} = \sum_{k=1}^{C} \mu_{ij} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}} = 1.$$

Factor μ_{ij} out since it's independent of k:

$$\mu_{ij} \sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}} = 1.$$

5. Solve for μ_{ij}

Finally, solving for μ_{ij}, we get the well-known FCM membership formula:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}.$$

Interpretation

This formula tells us that the membership of a data point in a cluster is determined **relatively** to the distances to all other clusters. If a point is very close to a given cluster center compared to others, the fraction becomes small, making its membership larger. Conversely, if it's far from a cluster, the fraction becomes larger, reducing its membership. The fuzziness parameter m controls how sharply these distances influence the memberships—values close to 1 behave more like hard assignments (K-Means), while larger values spread the memberships more evenly across clusters.

Summary of the Proof

1. Start from the objective function with the membership constraint.
2. Use a Lagrange multiplier to include the constraint in the optimization.
3. Differentiate with respect to memberships and relate memberships between clusters.
4. Express the ratio of memberships in terms of distances.
5. Use the sum-to-one constraint to normalize the memberships and obtain the final closed-form formula.

Thus, the derivation shows that **the optimal membership values are normalized inverse-distance ratios raised to a fuzziness-dependent power**, completing the proof.

DATE : SUBJECT :

$Q = XW^Q \longrightarrow$ Query

سوال سوم :

$K = XW^k \longrightarrow$ key

$X = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \end{bmatrix}$

$V = XW^V \longrightarrow$ Value

$W^Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad W^k = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \quad W^V = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

$\Rightarrow Q = XW^Q = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 3 & 1 \end{bmatrix}$

$K = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 2 & 2 \end{bmatrix}$

$V = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 3 & 3 \end{bmatrix}$

$\text{scores} = \dfrac{QK^T}{\sqrt{d_k}}$ ⟶ keys dimension $= 2$

$QK^T = \begin{bmatrix} 3 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 6 \\ 7 & 8 \end{bmatrix} \Rightarrow \text{scores} = \begin{bmatrix} 3\sqrt{2} & 3\sqrt{2} \\ \frac{7\sqrt{2}}{2} & 4\sqrt{2} \end{bmatrix} = \begin{bmatrix} 4.24 & 4.24 \\ 4.95 & 5.66 \end{bmatrix}$

$\text{weights} = \text{softmax}(\text{scores}) \Rightarrow \text{weights}(i,j) = \dfrac{\exp(\text{scores}(i,j))}{\sum_j \text{scores}(i,j)}$

$\Rightarrow \text{weights} = \begin{bmatrix} 0.5 & 0.5 \\ \frac{e^{4.95}}{e^{4.95}+e^{5.66}} & \frac{e^{5.66}}{e^{4.95}+e^{5.66}} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.33 & 0.67 \end{bmatrix}$

$\text{output} = \text{weights}\, V \Rightarrow \text{output} = \begin{bmatrix} 0.5 & 0.5 \\ 0.33 & 0.67 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 3 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 3 & 2.34 \end{bmatrix}$

Self-Attention Output

FABER-CASTELL