Portland State University

CS 4/510: Computer Vision & Deep Learning

Programming Assignment #2

A. Rhodes

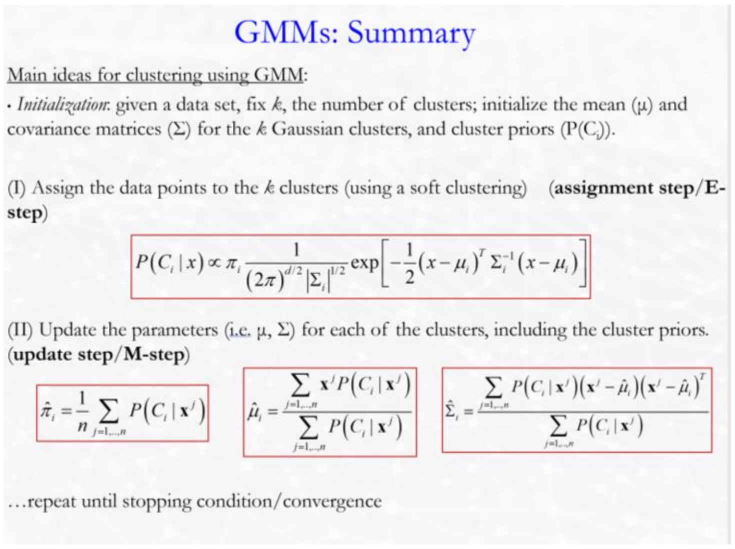
Note: This assignment is **due by Monday, 4/26 at 800pm**; you will turn in the assignment by email to our TA.

Note that there are (2) exercises listed below. **<u>You will complete one of the exercises of your choosing</u>** (please don't do both).

<u>Exercise #1: GMM Clustering</u>

Dataset: The data set is 2d data (for ease of visualization) simulated from 3 Gaussians, with considerable overlap. There are 500 points from each Gaussian, ordered together in the file. For the GMM RGB space clustering two test images are provided: GMM_test1.jpg and GMM_test2.jpg.

Use the following general outline to execute the EM algorithm for GMM (please refer to lecture slides as needed for details of the EM algorithm for GMMs as needed).



GMMs: Summary

<u>Main ideas for clustering using GMM</u>:

• *Initialization*: given a data set, fix *k*, the number of clusters; initialize the mean (μ) and covariance matrices (Σ) for the *k* Gaussian clusters, and cluster priors (P(C$_i$)).

(I) Assign the data points to the *k* clusters (using a soft clustering) (**assignment step/E-step**)

$$\boxed{P(C_i \mid x) \propto \pi_i \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)\right]}$$

(II) Update the parameters (i.e. μ, Σ) for each of the clusters, including the cluster priors. (**update step/M-step**)

$$\boxed{\hat{\pi}_i = \frac{1}{n} \sum_{j=1,\dots,n} P(C_i \mid \mathbf{x}^j)} \quad \boxed{\hat{\mu}_i = \frac{\sum_{j=1,\dots,n} \mathbf{x}^j P(C_i \mid \mathbf{x}^j)}{\sum_{j=1,\dots,n} P(C_i \mid \mathbf{x}^j)}} \quad \boxed{\hat{\Sigma}_i = \frac{\sum_{j=1,\dots,n} P(C_i \mid \mathbf{x}^j)(\mathbf{x}^j - \hat{\mu}_i)(\mathbf{x}^j - \hat{\mu}_i)^T}{\sum_{j=1,\dots,n} P(C_i \mid \mathbf{x}^j)}}$$

…repeat until stopping condition/convergence

To reasonably initialize your GMM algorithm, **first run k-means (reuse your code from programming #1) on your dataset** prior to executing the GMM algorithm <u>on the cluster dataset provided on D2L</u> (same as k-means assignment dataset). Use K=3 for k-means and for your GMM algorithm execution. At the conclusion of k-means, your dataset will be clustered into K=3 clusters. From these cluster assignments, compute the centroid (μ_i) for each cluster $1 \leq i \leq k$, this gives you a set of initial $\{\mu_i\}$ parameter values to initialize for the

GMM algorithm. In addition, using the clustering result from k-means, calculate the empirical covariance matrix for each cluster (Σ_i). You are welcome to use a built-in library function to calculate each of these covariance matrices (one for each cluster) – or if you prefer, you may code this calculation by hand; this gives you a set of initial $\{\Sigma_i\}$ covariance matrices for the GMM algorithm. Finally, using your centroid (μ_i) and covariance (Σ_i) estimates, calculate the prior for each cluster $\{\pi_i\}$ using the formula shown above.

The purpose in running k-means here is to "seed" the GMM with reasonable initial parameters values. Now we execute GMM:

• Begin with your initial parameter values determined by k-means for each cluster i: $\{\mu_i\}$ $\{\Sigma_i\}$, and $\{\pi_i\}$.

Execute E and M steps as long as the convergence condition is not satisfied:

– **E-step**: compute membership probabilities using the current θ current values (step (I) in the GMM summary box above)

– **M-step**: compute new parameters θ_{new} using the membership probabilities from the E-step (step (II) in the GMM summary box above).

You should run your algorithm multiple times from *r* different randomly-chosen starting conditions (e.g. r = 10) and pick the solution that results in the best clustering (eye-balling is fine).

***Note that The EM algorithm for Gaussian mixtures is a non-trivial algorithm to get working properly: please try and debug it carefully**.

At each step of the GMM algorithm, please report the *negative log likelihood* (NLL), defined:

$$NLL = -\sum_{i=1}^{n} \log\left\{\sum_{k=1}^{K} \pi_k N\left(x_i \mid \mu_k, \Sigma_k\right)\right\}$$

Where the sum from $1 \leq i \leq n$ is over the dataset, $1 \leq k \leq K$ denotes the sum over the clusters, and $N(x \mid \mu_k, \Sigma_k)$ represents the Gaussian pdf evaluated at point x_i with mean μ_k and covariance Σ_k. The NLL is **guaranteed to decrease** at each step of the GMM algorithm, so this can be used to monitor whether you are executing things correctly. If it is decreasing at each step, you're good; otherwise, some debugging of your implementation of the GMM algorithm is necessary.

Please print out your final parameters for the Gaussians in your GMM in addition to the NLL at each step. For this assignment you could also impose a maximum number of iterations to halt the algorithm (e.g., 500) if it gets that far and still has not converged. Report your results for each and include a 2D plot, showing several steps of your GMM algorithm beginning with the initial step, some intermediate steps, and the final result. This 2D plot can be color coded according to cluster assignment so that it is easy to interpret.

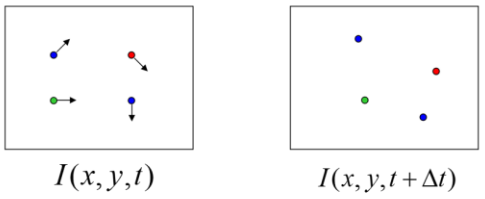
Finally, with your working GMM algorithm, we can perform clustering in the RGB space of an image! Using the two images provided: GMM_test1.jpg and GMM_test2.jpg (found on D2L), perform clustering in the RGB space for separate experiments with K=3, K=5, and K=10 on each image. Provide visual results for each of these experiments.

Exercise #2: Optical Flow

Dataset: Two pairs of images from video clips: frame1_a.png, frame1_b.png and frame2_a.png and frame2_b.png.

This exercise requires you to implement Lucas-Kanade Optical Flow method (by hand).

Recall that optical flow aims to estimate motion from one frame to the next in a video sequence.



If your image is not already grayscale (i.e. one channel), convert it to grayscale. For each pair of images: (frame$_1$, frame$_2$), we want to compute optical flow, a 2D vector (V_x,V_y) with respect to each pixel (x,y) in frame$_1$. To do this, we solve the following system of equations for each pixel (x,y) in frame$_1$ for V_x and V_y:

$$\underbrace{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_9) & I_y(p_9) \end{bmatrix}}_{9\times 2} \underbrace{\begin{bmatrix} V_x \\ V_y \end{bmatrix}}_{2\times 1} = -\underbrace{\begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_9) \end{bmatrix}}_{9\times 1}$$

where p$_1$,…,p$_9$ is a set of nine points surrounding the central pixel (x,y) in a 3x3 grid:

p_1	p_2	p_3
p_4	$p_5 = (x, y)$	p_6
p_7	p_8	p_9

Above, I_x and I_y (on the left-hand side of the system of equations) denote the result of convolving the input image (frame_1) with the x and y gradient filters, respectively:

$$g_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, g_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Note that you should zero-pad (width=1) your input image, so you are able to perform convolution over pixels on the periphery of the image.

The notation $I_t(p)$ above (right-hand side of system of equations) stands for the "temporal" derivative. To estimate this we simple take the difference of the image intensities between the frames: $I(\text{frame}_2) - I(\text{frame}_1)$ where $I(\bullet)$ denotes pixel intensity; so then $I_t(p)$ is the difference in pixel intensities between the frames with respect to pixel *p*.

Finally, for each pixel in frame_1 we can generate the 9x2 system of equations shown above. Next, we leverage OLS to solve each 9x2 system:

$$\rightarrow \underbrace{\begin{bmatrix} V_x \\ V_y \end{bmatrix}}_{x} = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

Giving us V_x and V_y , the x and y components of optical flow, respectively, for each pixel in the original image, as desired.

For each pair of images provided on D2L, calculate V_x and V_y and visualize each of these values (in their own 2D image, over all pixels); also include a 2D visualization of $\sqrt{(V_x)^2 + (V_y)^2}$.

Report: Your report should include a short description of your experiments, along with the plots and discussion paragraphs requested above and any other relevant information to help shed light on your approach and results.

Here is what you need to turn in:

- Your report.
- Readable code.