CSCI 5521: Introduction to Machine Learning (Fall 2019)[1]

Homework 3

Questions

1. (**30 points**) Derive the EM algorithm for estimating a mixture of multinomial distributions. The probability mass function of a multinomial distribution for class C_i is

$$\begin{aligned} P(x = (x_1, \ldots, x_n)|C_i) &= P(x = (x_1, \ldots, x_n)|p_{i1}, \cdots, p_{in}) \\ &= \frac{m!}{x_1! \cdots x_n!} p_{i1}^{x_1} \cdots p_{in}^{x_n} \end{aligned}$$

where $\sum_{j=1}^{n} p_{ij} = 1$ and $\sum_{j=1}^{n} x_j = m$. The mixture density of K multinomial distributions is

$$P(x) = \sum_{i=1}^{K} P(x|C_i)P(C_i) = \sum_{i=1}^{K} \pi_i \frac{m!}{x_1! \cdots x_n!} p_{i1}^{x_1} \cdots p_{in}^{x_n},$$

where $\sum_{i=1}^{K} \pi_i = 1$ and assuming $\sum_{j=1}^{n} x_j = m$ for all K distributions. Define the complete log-likelihood function and derive the EM equations including the expectation for the responsibility $\gamma(z_i^t)$ (z_i^t is the binary indicator of sample t in cluster i) and maximum likelihood learning for $\{\pi_i, p_{i1}, \cdots, p_{in}\}_{i=1,\ldots,K}$.

2. In this question, we will implement the EM algorithm to estimate a mixture of Gaussian distributions for image compression. (Please read the **Instructions** section below before you start programming)

   (a) (**20 points**) Implement the EM algorithm to estimate a mixture of k Gaussian distributions and run it on the image file "stadium.bmp". Cluster the pixels into $k = \{4, 8, 12\}$ clusters and plot the compressed images for each value of k as described below. (Note: your program might fail if Σ is singular; in this case, restart your EM again. We will fix the problem in part (d)).

[1]Instructor: Rui Kuang (kuang@cs.umn.edu). TA: Rachit Jas (jas00001@umn.edu) and Tianci Song (song0309@umn.edu)

(b) (**10 points**) Run your EM implementation on the "stadium.bmp" image for $k = \{4, 8, 12\}$ and plot the expected complete log-likelihood function $\mathcal{Q}(\Phi|\Phi^l)$ after each E-step and M-step of the EM algorithm in one curve for each value of k. Use different colors to plot the log-likelihood after the E-step and M-step in the curve. Briefly explain the results.

(c) (**10 points**) Try to run your EM implementation on the image "goldy.bmp" with $k = 7$ and report your observation (**Hint:** If your algorithm falls here, don't panic. Continue to the rest part.). Next, use the k-means function in cluster module of sklearn package [2] to cluster the pixels with $k = 7$. Plot the compressed image given by kmeans. Explain why kmeans and EM behaved differently on the image.

(d) (**20 points**) Next, implement an improved version of EM to handle singular covariance matrix. In the likelihood function, we can add the following regularization term, $-\frac{\lambda}{2}\Sigma_{i=1}^{k}\Sigma_{j=1}^{d}(\Sigma_i^{-1})_{jj}$, where $(\Sigma_i^{-1})_{jj}$ is the (j, j)-th entry of matrix Σ_i^{-1} and $\lambda > 0$. This regularization term encourages the diagonal of Σ_i^{-1} to be small such that Σ_i is not singular. After adding this regularization term, the expectation step is unchanged and in the maximization step, the maximum likelihood learning of μ_is and π_is are also unchanged. Derive the maximum likelihood learning of Σ_is using the following result,

$$\frac{\partial(-\frac{\lambda}{2}\Sigma_{i=1}^{k}\Sigma_{j=1}^{d}(\Sigma_i^{-1})_{jj})}{\partial\Sigma_i^{-1}} = -\frac{\lambda I}{2}$$

(hint: modify the derivation on slide 32 in parametric.pdf to solve the problem).

(e) (**10 points**). Implement the new model and test the new model on "goldy.bmp". Explain your observations.

Instructions

- Solutions to all questions must be presented in a report which includes results, explanations, all images and plots.
- All programming questions must be written in Python, no other programming languages will be accepted. And numpy, scipy, skimage, sklearn and matplotlib can be relied on to implement the algorithm (Note that you are only allowed to

[2] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

use KMeans function in cluster module of sklearn package in this homework). The code must be able to be executed from either terminal or PyCharm console on the cselabs machines. Each function must take the inputs in the order specified and print/display the required output to either terminal or PyCharm console. For each part, you can submit additional files/functions (as needed) which will be used by the main functions specified below. Put comments in your code so that one can follow the key parts and steps. **Please follow the rules strictly. If we cannot run your code, you will receive no credit.**

- **Question 2:**
  - hw3_Q2 a driver script which 1) uses kmeans to display a compressed image 2) runs your standard EM implementation finishing by outputting your own error message explaining why it fails 3) runs your improved EM implementation to display the compressed image. For loading the "goldy.bmp" image, you can use the following code:

    ```
    from skimage import io
    img = io.imread('./stadium.bmp')
    img = img/255
    ```

  - EMG(*image.bmp*: file path to an image, k: scalar value of the number of clusters, *flag*: a binary indicator variable). Function EMG implements the standard EM algorithm if *flag* is 0, while implements the improved EM algorithm if *flag* is 1. The function must print to either terminal or PyCharm console and return in variables the following for a single image and value of k: (1) h: a $n \times k$ matrix where n is the number of pixels, (2) m: a $k \times d$ matrix where $d = 3$ denotes the RGB values, and (3) $\mathcal{Q}$: a column vector of expected complete log-likelihood values. The outputs $\{h, m, \mathcal{Q}\}$ are defined in Section 7.4 of the textbook. The function must also display: (1) a single compressed **color** image for a single value of k and (2) a single plot for the expected complete log-likelihood function value vs iteration number for a single value of k.
  - You can use the *imread()* function in io module of skimage package [3] to convert the image into a 3D matrix in Python. You will then need to convert the 3D matrix into a 2D matrix with $d = 3$ columns, in which each row are the three RGB values of a pixel. You can use the *reshape()* function in numpy package to do this [4].

[3] https://scikit-image.org/docs/dev/api/skimage.io.html
[4] https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html

- To decide the membership of each pixel x^t, you can select $\underset{i}{\text{argmax}}\, h_i^t$.
- To visualize the compression, use the mean estimated from C_i as the color of pixels in C_i.
- To compute the component densities, $p(\mathbf{x^t}|\mathbf{\Phi})$, you may use the *multivariate_normal()* function in stats module in scipy package [5].
- In your EM implementation, you can use the function *KMeans()* in cluster module of sklearn package to find the initial clusters, however you can only run kmeans for at most 3 iterations.
- Your program must be efficient and finish running for a single image and value of k in at most a couple of minutes. You can set a maximum number of iterations for the EM algorithm however use no less than 100 iterations.

Submission

- **Things to submit:**
  1. hw3_sol.pdf: A PDF document which contains the report with solutions to all questions.
  2. EMG.py: Code for Question 2.
  3. Any other files, except the data, which are necessary for your code.
- **Submit**: All materials must be zipped in one file, and submitted electronically via canvas.

[5] https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.multivariate_normal.html