Exercise 1 - Unsupervised Image Denoising

Lirane Bitton - 200024677

8/11/18

Part I
Theoretical Questions

1 MLE in the EM algorithm

$$\mathbb{E}[l(S,\Theta)] = \sum_{i=1}^{N}\sum_{y=1}^{k} c_{i,y}\log(\pi_y \mathcal{N}(x_i : \mu_y, \Sigma_y))$$

Show that the MLE of the multinomial distribution of our mixture weights is indeed:

$$\pi_y = \frac{1}{N}\sum_{i=1}^{N} c_{i,y}$$

Since it's a mixture of multinomial distribution hence the weights are distributed as multinomial distribution and need to satisfy:

$$\pi_y \geq 0$$

$$\sum_{y=1}^{k}\pi_y = 1 \rightarrow \sum_{y=1}^{k}\pi_y - 1 = 0$$

As learned in TA we'll solve the problem of maximisation by defining the Lagrangian for our and derive it w.r.t π.

$$\begin{aligned}\mathcal{L}(S,\Theta,\lambda) = & \sum_{i=1}^{N}\sum_{y=1}^{k} c_{i,y}\log(\pi_y \mathcal{N}(x_i : \mu_y, \Sigma_y)) - \lambda(\pi^T\mathbf{1} - 1) \\ = & \sum_{i=1}^{N}\sum_{y=1}^{k} c_{i,y}(\log(\pi_y) + \log(\mathcal{N}(x_i : \mu_y, \Sigma_y)) - \lambda(\pi^T\mathbf{1} - 1)\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\pi_y} &= \frac{\sum_{i=1}^{N} c_{i,y}}{\pi_y} - \lambda = 0 \\ \rightarrow \pi_y &= \frac{\sum_{i=1}^{N} c_{i,y}}{\lambda}\end{aligned}$$

Since π is a distribution vector it might be normalize hence $\lambda = N$ is a normalization factor and we get:

$$\pi_y = \frac{1}{N}\sum_{i=1}^{N} c_{i,y}$$

2 MLE in the GSM Model

$$\begin{aligned}
\mathbb{E}[l(S,\Theta)] =& \sum_{i=1}^{N}\sum_{y=1}^{k}\left[c_{i,y}\log(\pi_y \mathcal{N}(x_i : 0, r_y^2\Sigma_y))\right] \\
=& \sum_{i=1}^{N}\sum_{y=1}^{k}\left[c_{i,y}(\log(\pi_y) + \log(\mathcal{N}(x_i : 0, r_y^2\Sigma_y))\right] \\
=& \sum_{i=1}^{N}\sum_{y=1}^{k}\left[c_{i,y}\left(\log(\pi_y) + \log(\frac{1}{\sqrt{\det(2\pi_y r_y^2\Sigma_y)}}\exp(-\frac{1}{2}x^T\cdot\left(r_y^2\Sigma_y\right)^{-1}\cdot x)\right)\right] \\
=& \sum_{i=1}^{N}\sum_{y=1}^{k}\left[c_{i,y}\left(\log(\pi_y) - \log\left(\sqrt{\det(2\pi_y r_y^2\Sigma_y)}\right) + (-\frac{1}{2}x^T\cdot\left(r_y^2\Sigma_y\right)^{-1}\cdot x)\right)\right] \\
=& \sum_{i=1}^{N}\sum_{y=1}^{k}\left[c_{i,y}\left(\log(\pi_y) - \log\left(\sqrt{(2\pi_y r_y^2)^d\det(\Sigma_y)}\right) + (-\frac{1}{2}x^T\cdot(r_y^2)^{-1}\Sigma_y^{-1}\cdot x)\right)\right] \\
=& \sum_{i=1}^{N}\sum_{y=1}^{k}\left[c_{i,y}\left(\log(\pi_y) - \log(r_y^d) + \log\left(\sqrt{(2\pi_y)^d\det(\Sigma_y)}\right) - \frac{1}{2r_y^2}x^T\cdot\Sigma_y^{-1}\cdot x\right)\right] \\
=& \sum_{i=1}^{N}\sum_{y=1}^{k}\left[c_{i,y}\left(\log(\pi_y) + \log\left(\sqrt{(2\pi_y)^d\det(\Sigma_y)}\right)\right)\right] + \\
& \sum_{i=1}^{N}\sum_{y=1}^{k}\left[c_{i,y}\left(-d\log(r_y) - \frac{1}{2r_y^2}x^T\cdot\Sigma_y^{-1}\cdot x\right)\right] \\
\frac{\partial\mathbb{E}[l(S,\Theta)]}{\partial r_y^2} =& \frac{\sum_{i=1}^{N}\sum_{y=1}^{k} c_{i,y}(-\frac{1}{2}x^T\cdot\Sigma_y^{-1}\cdot x)r_y^{-2} + c_{i,y}(-d\log(r_y))}{\partial r_y^2} = 0 \\
=& \sum_{i=1}^{N} c_{i,y}(x^T\cdot\Sigma_y^{-1}\cdot x)r_y^{-4} - \sum_{i=1}^{N} c_{i,y}dr_y^{-2} = 0 \\
\rightarrow& \sum_{i=1}^{N} c_{i,y}(x^T\cdot\Sigma_y^{-1}\cdot x)r_y^{-4} = \frac{d}{r_y^2}\sum_{i=1}^{N} c_{i,y} \\
\rightarrow& r_y^2 = \frac{\sum_{i=1}^{N} c_{i,y}(x^T\cdot\Sigma_y^{-1}\cdot x)}{d\sum_{i=1}^{N} c_{i,y}}\blacksquare
\end{aligned}$$

3 EM Initialization

E-step

$$
\begin{aligned}
c_{i,y} =& \frac{\pi_y \mathcal{N}(x_i : \mu_y, \Sigma_y)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(x_i : \mu_l, \Sigma_l)} \\
=& \frac{\frac{1}{k}\mathcal{N}(x_i : \mu, \Sigma)}{\sum_{l=1}^{k} \frac{1}{k}\mathcal{N}(x_i : \mu, \Sigma)} \\
=& \frac{\frac{1}{k}\cancel{\mathcal{N}(x_i : \mu, \Sigma)}}{\frac{k}{k}\cancel{\mathcal{N}(x_i : \mu, \Sigma)}} \\
=& \frac{1}{k}
\end{aligned}
$$

M-step

$$
\begin{aligned}
\pi_y =& \frac{1}{N}\sum_{i=1}^{N} c_{i,y} \\
=& \frac{1}{N}\sum_{i=1}^{N} \frac{1}{k} \\
=& \frac{1}{k}
\end{aligned}
$$

$$
\begin{aligned}
\mu_y =& \frac{\sum_{i=1}^{N} c_{i,y} x_i}{\sum_{i=1}^{N} c_{i,y}} \\
=& \frac{\frac{1}{k}\sum_{i=1}^{N} x_i}{N\frac{1}{k}} \\
=& \frac{1}{N}\sum_{i=1}^{N} x_i
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_y =& \frac{\sum_{i=1}^{N} c_{i,y}(x_i - \mu_y)(x_i - \mu_y)^T}{\sum_{i=1}^{N} c_{i,y}} \\
=& \frac{\frac{1}{k}\sum_{i=1}^{N}(x_i - \mu_y)(x_i - \mu_y)^T}{\frac{N}{k}} \\
=& \frac{\sum_{i=1}^{N}(x_i - \mu_y)(x_i - \mu_y)^T}{N}
\end{aligned}
$$

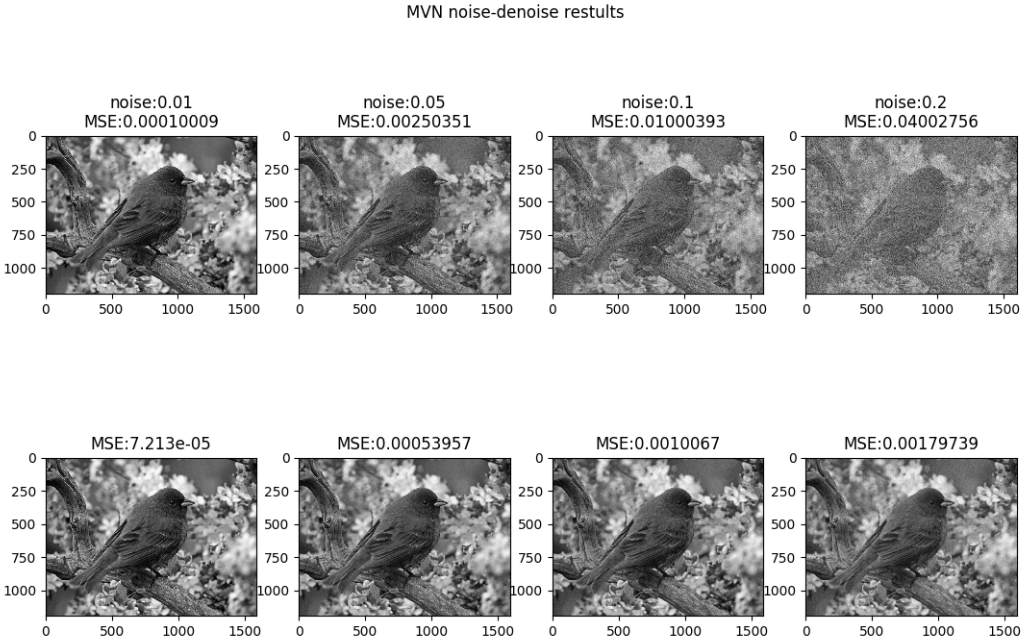
In the next iteration $c_{i,y}$ will remain the same as in the first iteration hence the update in M-step won't change anything for all $y \in [k]$.

Therefore after one iteration we'll get the same values and so on the same log likelihood meaning that the EM algorithm has converged with $\pi_y = \frac{1}{k}$ uniform accross the mixture with gaussian having as parameter as mean the entire data mean and covariance the covariance of all the data. Actually we got a mixture of k gaussian that are the same and distributed uniformly losing the expressiveness of mixture of multiple different gaussians.

Part II
Practical Exercise

3.1 Results

For k=2



Output of the program for this run for MVN denoising:
MVN LogLikelihood: 2869898.932001023
noisy MSE for noise = 0.01: 0.0001000879505727387
denoised MSE for noise = 0.01: 7.212656311621659e-05
noisy MSE for noise = 0.05: 0.0025035061731658445
denoised MSE for noise = 0.05: 0.0005395680215644455
noisy MSE for noise = 0.1: 0.010003927855669061
denoised MSE for noise = 0.1: 0.001006702516746113
noisy MSE for noise = 0.2: 0.04002756239146701
denoised MSE for noise = 0.2: 0.001797389222342733

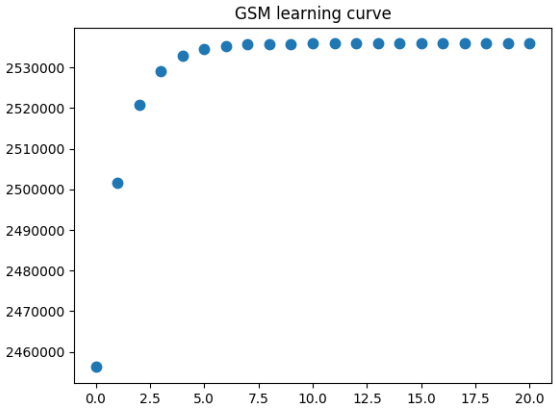
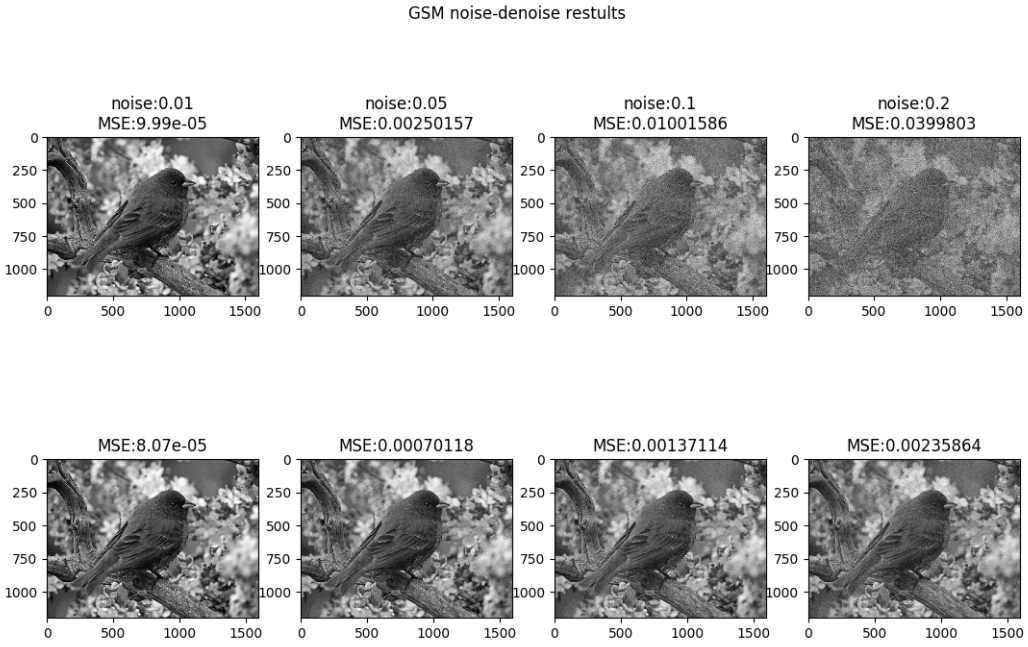


Figure 1: Convergence example for GSM model



Output of the program for this run for GSM denoising:
LogLikelihood 2535313.798501323
noisy MSE for noise = 0.01: 9.989894443463817e-05
denoised MSE for noise = 0.01: 8.069911415811826e-05
noisy MSE for noise = 0.05: 0.002501573571852609
denoised MSE for noise = 0.05: 0.0007011815341521101
noisy MSE for noise = 0.1: 0.010015862255031703

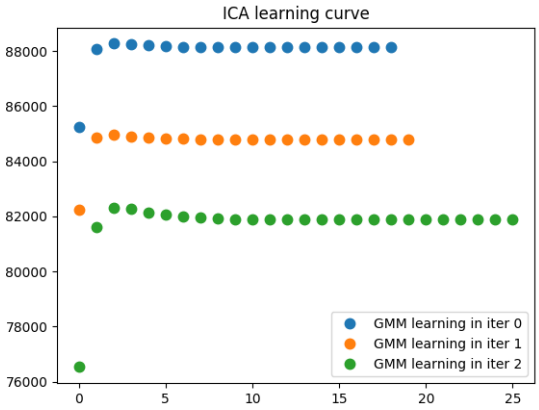
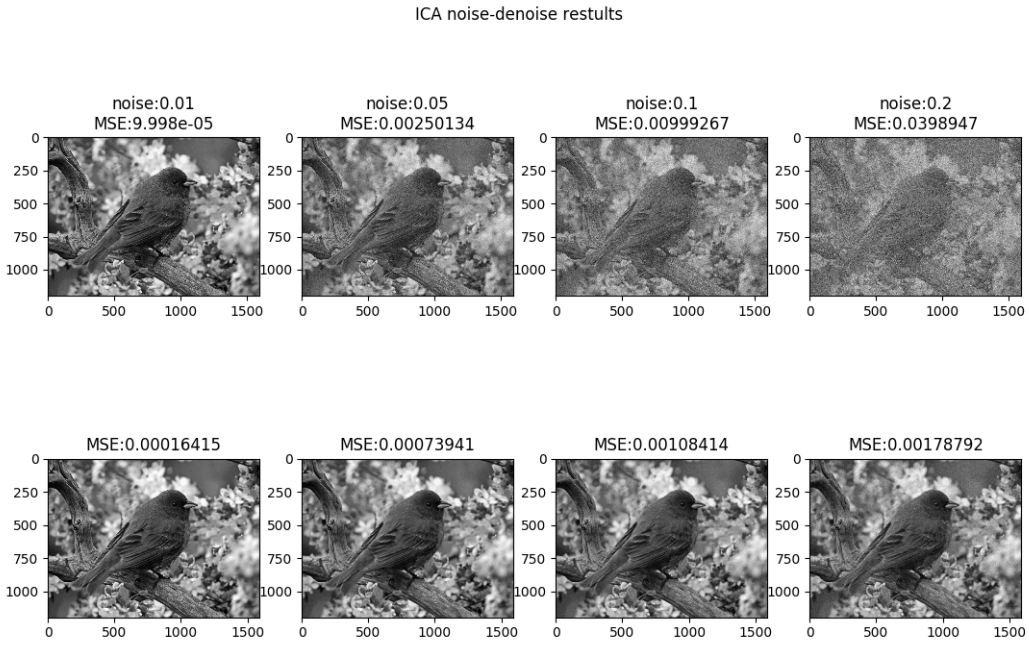


Figure 2: Convergence example in ICA model of the first three GMM

denoised MSE for noise = 0.1: 0.0013711417424166975
noisy MSE for noise = 0.2: 0.03998029720420108
denoised MSE for noise = 0.2: 0.0023586389880232306



Output of the program for this run for ICA denoising:
ICA LogLikelihood sum 3740548.660549269
ICA LogLikelihood mean 58446.07282108233
noisy MSE for noise = 0.01: 9.997822542537602e-05

denoised MSE for noise = 0.01: 0.00016415495268484715
noisy MSE for noise = 0.05: 0.0025013432964240442
denoised MSE for noise = 0.05: 0.0007394115530265786
noisy MSE for noise = 0.1: 0.00999267206188311
denoised MSE for noise = 0.1: 0.0010841424210256433
noisy MSE for noise = 0.2: 0.039894695714116225
denoised MSE for noise = 0.2: 0.0017879167011375465

The code was tested on a machine with a Xeon CPUmade of 16 cores and 64 GB ram.

Profiling results of the program:

learn_MVN = 20ms
learn_GSM = 1768ms
learn_ICA = 29995ms
MVN_denoise = 3650ms
GSM_denoise = 12805ms
ICA_denoise = 12857ms

For k=1

Here I just attach the output of each run to avoid generate a huge pdf (the result can be seen by running the program)

Output of the program for this run for MVN denoising:

MVN LogLikelihood: 2858487.3207302056
noisy MSE for noise = 0.01: 9.999399252991564e-05
denoised MSE for noise = 0.01: 7.255220773884023e-05
noisy MSE for noise = 0.05: 0.0024981219637112534
denoised MSE for noise = 0.05: 0.0005337829437560284
noisy MSE for noise = 0.1: 0.010012520709544155
denoised MSE for noise = 0.1: 0.0009974998689489861
noisy MSE for noise = 0.2: 0.03992480837342372
denoised MSE for noise = 0.2: 0.0017730978589110809

Output of the program for this run for GSM denoising:

LogLikelihood 2858409.2093455833
noisy MSE for noise = 0.01: 9.999045441026111e-05
denoised MSE for noise = 0.01: 7.255673327727255e-05
noisy MSE for noise = 0.05: 0.0025005512673189918
denoised MSE for noise = 0.05: 0.0005341103531239731
noisy MSE for noise = 0.1: 0.009985021258583231
denoised MSE for noise = 0.1: 0.0009962492577834886
noisy MSE for noise = 0.2: 0.03994248666585434
denoised MSE for noise = 0.2: 0.0017810856241731049

Output of the program for this run for ICA denoising:

ICA LogLikelihood sum 2858487.3215302317
ICA LogLikelihood mean 44663.86439890987
noisy MSE for noise = 0.01: 9.980403419332787e-05
denoised MSE for noise = 0.01: 7.243187328756275e-05
noisy MSE for noise = 0.05: 0.0025019983593264367
denoised MSE for noise = 0.05: 0.0005345214745598893
noisy MSE for noise = 0.1: 0.010005560603579357

denoised MSE for noise = 0.1: 0.0009963166417395924
noisy MSE for noise = 0.2: 0.03996530024900678
denoised MSE for noise = 0.2: 0.0017826890912257549

The code was tested on a machine with a Xeon CPUmade of 16 cores and 64 GB ram.

Profiling results of the program:
learn_MVN = 17ms
learn_GSM = 1151ms
learn_ICA = 1344ms
MVN_denoise = 3828ms
GSM_denoise = 12800ms
ICA_denoise = 8548ms

For k=3

Output of the program for this run for MVN denoising:
MVN LogLikelihood: 2852235.2794499015
noisy MSE for noise = 0.01: 0.00010004006478502549
denoised MSE for noise = 0.01: 7.242383744651062e-05
noisy MSE for noise = 0.05: 0.002499217219222088
denoised MSE for noise = 0.05: 0.0005429898548041234
noisy MSE for noise = 0.1: 0.010013950426765948
denoised MSE for noise = 0.1: 0.0010108390012673705
noisy MSE for noise = 0.2: 0.04002386156722221
denoised MSE for noise = 0.2: 0.00179184394939436
Output of the program for this run for GSM denoising:
LogLikelihood 2635313.298501323
noisy MSE for noise = 0.01: 9.998552358142195e-05
denoised MSE for noise = 0.01: 7.541403798284629e-05
noisy MSE for noise = 0.05: 0.002496821371151996
denoised MSE for noise = 0.05: 0.0005389861201522224
noisy MSE for noise = 0.1: 0.00999207869948621
denoised MSE for noise = 0.1: 0.001004074282139285
noisy MSE for noise = 0.2: 0.03999167072538547
denoised MSE for noise = 0.2: 0.0018030725434344116
Output of the program for this run for ICA denoising:
ICA LogLikelihood sum 3818469.8521888386
ICA LogLikelihood mean 59663.591440450604
noisy MSE for noise = 0.01: 9.988401923791658e-05
denoised MSE for noise = 0.01: 0.0001711023523556134
noisy MSE for noise = 0.05: 0.0024975643021563003
denoised MSE for noise = 0.05: 0.0007179338879808184
noisy MSE for noise = 0.1: 0.00999817591664311
denoised MSE for noise = 0.1: 0.0010898565640358533
noisy MSE for noise = 0.2: 0.04000392008381399
denoised MSE for noise = 0.2: 0.0018031623921915963

The code was tested on a machine with a Xeon CPUmade of 16 cores and 64 GB ram.

Profiling results of the program:
learn_MVN = 21ms
learn_GSM = 5035ms
learn_ICA = 66081ms
MVN_denoise = 3900ms
GSM_denoise = 12620ms
ICA_denoise = 17949ms

3.2 Discussions

Above I attached the output of one run of learning and denoising the first image in the data set as well as the profiling runtime results. You can get the same results by running the main function where it's possible also to edit the k parameter for number of mixture and also the number of graph plotted during the ICA learning step.

From the results above we can see that the MSE got in the ICA and GSM model are slightly better than the MVN model. But without a clear advantage for one of this models.

The MVN model took in average 20ms to learn and 3900ms to denoise not depending on the number of mixture define as k parameter.

The GSM model showed different learning time accross the different runs (k=1, ~1000ms, k=2, ~2000ms, k=3, ~5000ms) but the denoise time remain the same accross the different mixture (~12000ms) while the ICA model also got different running time depending on mixture number (growing more in term of runtime than in GSM) but also different times in denoising step. In case that $k \leq 2$ the ICA model perform better in runtime than GSM (8000ms against 12000ms) but in general case the GSM model got constant runtime while ICA denoising grows with the number of mixture.

Since training is normally performed once and the trained model is used for many task hence the denoising time is crucial in consideration for model choice.

Hence if the running time is important for me I'll choose the MVN model for denoising. In the lab I research I showed all the results to everybody and all of them chosed the images denoisedwith ICA as best one or cleaner one hence if someone doesn't care about runtime in denoise probably the ICA model can be a good choice.

P.S.: just for the fun I runned a profiling with k=10 to see how the runtime growth with k and the results are attached here and i was even surprised to see that the MSE got in MVN model is even better that the ICA and really close to the GSM one for a runtime clealy advantageous in learning and denoising.

Profiling results of the program with k=10:
MVN:
noisy MSE for noise = 0.2: 0.04005438056543772
denoised MSE for noise = 0.2: 0.0017784414970012789
GSM
noisy MSE for noise = 0.2: 0.03996931618501308
denoised MSE for noise = 0.2: 0.0017724070531022
ICA
noisy MSE for noise = 0.2: 0.039991983639477306

denoised MSE for noise = 0.2: 0.001818472060841697
learn_MVN = 17ms
learn_GSM = 41038ms
learn_ICA = 311894ms
MVN_denoise = 3857ms
GSM_denoise = 12538ms
ICA_denoise = 42616ms