Exercise 4: Expectation Maximization and EPLL (Due 6/6/2017)*

Statistical Methods in Image Processing 048926

Analytic Exercises (20 points)

1. The probability mass function of the variable y_i is given by:

$$p(y_i|\mu) = \begin{cases} \frac{\mu^{y_i} e^{-\mu}}{y_i!} & y_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

   Given a set of i.i.d. samples $\{y_i\}_{i=1}^N$ calculate the maximum likelihood estimator of μ.

2. The probability density function of the variable y_i is given by a d-dimension Gaussian:

$$p(y_i|\mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\mathbf{\Sigma}|^{\frac{1}{2}}} \exp(-\frac{1}{2}(y_i - \mu)^T \mathbf{\Sigma}^{-1}(y_i - \mu)).$$

   Given a set of i.i.d. samples $\{y_i\}_{i=1}^N$ calculate the maximum likelihood estimator of μ and $\mathbf{\Sigma}$. Use the following known derivatives:

$$\frac{\partial \det(\mathbf{X})}{\partial \mathbf{X}} = \det(\mathbf{X})(\mathbf{X}^{-1})^T$$

$$\frac{\partial (a^T \mathbf{X}^{-1} b)}{\partial \mathbf{X}} = -\mathbf{X}^{-T} a b^T \mathbf{X}^{-T}$$

EM-GMM (40 points)

In this section you will implement the Expectation Maximization algorithm for training Gaussian mixture models.

1. To train amixture of M Gaussians using N data examples, the EM-GMM algorithm iterates between two steps:

*Please send your solutions to Tamar

(a) E-step: write a function that accepts N examples $\{y_n\}$, and returns the relative weight of each point with respect to each Gaussian:

$$T_{m,n}(\theta) = \frac{2\pi^{d/2}|\mathbf{\Sigma}_m|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(y_n - \mu_m)^T \mathbf{\Sigma}_m^{-1}(y_n - \mu_m)\}\alpha_m}{\sum_{m'=1}^{M} 2\pi^{d/2}|\mathbf{\Sigma}_{m'}|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(y_n - \mu_{m'})^T \mathbf{\Sigma}_{m'}^{-1}(y_n - \mu_{m'})\}\alpha_{m'}}$$

(b) M-step: Implement a function which maximizes the log-likelihood with respect to the parameters of the Gaussians::

$$N_m = \sum_{n=1}^{N} T_{m,n}(\theta)$$

$$\alpha_m = \frac{N_m}{N}$$

$$\mu_m = \frac{1}{N_m} \sum_{n=1}^{N} y_n T_{m,n}(\theta)$$

$$\mathbf{\Sigma}_m = \frac{1}{N_m} \sum_{n=1}^{N} (y_n - \mu_m)(y_n - \mu_m)^T T_{m,n}(\theta)$$

(c) Note: maximizing the likelihood has an inherent problem: as a Gaussian approaches singularity, the likelihood tends to infinity. Therefore, during the EM algorithm, the covariance matrices may get singular. To avoid this undesired solution, you can use the following process: after updating the covariance matrix, find its spectral decomposition $\mathbf{\Sigma} = \mathbf{UDU^T}$ and set to ε all the eigenvalues that are smaller than ε ($\varepsilon = 10^{-3}$ should be enough).

2. Consider the distribution that was given in HW2:

$$p(x; \{\mu_i\}) = \sum_{i=1}^{N} \frac{1}{N} \frac{1}{2\pi} \exp\left\{-\frac{1}{2}||x - \mu_i||^2\right\},$$

with $N = 4$ and $\mu = \{(0,0)^T, (0,2)^T, (2,0)^T, (2,2)^T\}$

(a) draw $N = 1000$ samples y from $p(y; \{\mu_i\})$ using the function that was implemented in HW2.

(b) Apply EM-GMM: Initialize 4 Gaussians of 2 dimensions with random means μ_m, random weights α_m and $\mathbf{\Sigma}_m = \mathbf{I}$. Iterate between the E-step and the M-step until convergence. Does the estimated model equal the correct one?

(c) The EM algorithm makes a *soft* assignment based on the posterior probabilities. Another option is to preform a *hard* assignment in which each data point is associated uniquely with one Gaussian. This can be done by updating $T_{m,n}$ at each steps to be as follow:

$$\tilde{T}_{m,n} = \begin{cases} 1 & m = \arg\max_m \{T_{m,n}\} \\ 0 & \text{otherwise} \end{cases}$$

Apply this change and repeat 2b. How do the results change?

3. MNIST:

   (a) Download the MNIST database from `http://yann.lecun.com/exdb/mnist/`. This database contains images of hand written digits of size 28×28 pixels. Use the training dataset ($N = 60000$ image) and resize all the digit to be of size 10×10 pixels.

   (b) Apply EM-GMM on the 10×10 dataset. Initialize $M = 5$ Gaussians with $\theta_m = \{\mu_m, \mathbf{\Sigma}_m, \alpha_m\}$. Initialize μ_m randomly, $\mathbf{\Sigma}_m = \mathbf{I}$ and the weights $\alpha_m = \frac{1}{M}$. Apply EM-GMM. Draw images from each of the trained Gaussians (using the function `mvnrnd` for example). What did each Gaussian capture? Repeat the process with $M = 10, 20$ Gaussians. How do the results change?

   (c) Repeat the process again with $M = 10$ Gaussians. This time initialize the mean of each Gaussian to be a different digit from the training set, picked randomly. How do the results change?

   (d) Reduce the implementation into hard assignment, as described in 2c. Repeat 3b with $M = 10$ and 3c. How do the results change?

EPLL(40 points)

The trained GMM model can be used as a prior for restoring 10×10 digit images. To use this model for restoring larger images, we can employ the Expected Patch Log Likelihood (EPLL) (*1*) method. As mentioned in class, solving the EPLL restoration problem can be done using Half Quadratic Splitting. This reduces to solving the following optimization problem:

$$\min_{x,\{z_i\}} \frac{1}{2\sigma^2}\|Hx - y\|^2 - \sum_i \log p(z_i) + \frac{\beta}{2}\sum_i \|Q_i x - z_i\|^2,$$

where Q_i is a matrix that extracts the i-th patch from the image and $\{z_i\}$ is a set of auxiliary vectors that approximate the image's patches. $p(\cdot)$ is the trained GMM model of 3c. This problem can be solved alternately for x and $\{z_i\}$ while gradually increasing β.

1. <u>z-step:</u> updating $\{z_i\}$ can be done by solving MAP estimation for each patch z_i separately:

$$\min_{z_i} \frac{\beta}{2}\|Q_i x - z_i\|^2 - \log p(z_i).$$

   Using the GMM model, the MAP estimate can not be calculate in a closed form. To tackle this use the following approximation:

   (a) Given a patch $Q_i x$ calculate its probability according to each of the Gaussians:

$$\alpha'_k = \frac{\alpha_k}{(2\pi)^{\frac{d}{2}}|\mathbf{\Sigma}_k|^{\frac{1}{2}}} \exp(-\frac{1}{2}(Q_i x - \mu_k)^T \mathbf{\Sigma}_k^{-1}(Q_i x - \mu_k))$$

(b) Choose the Gaussian with the maximal value $k_{\max} = \arg\max_k \alpha'_k$

(c) Compute the MAP estimator according to the $k_{\max}$ Gaussian. This is given by the Weiner filter:

$$\hat{z}_i = (\mathbf{\Sigma}_{k_{\max}} + \frac{1}{\beta}\mathbf{I})^{-1}(\mathbf{\Sigma}_{k_{\max}} Q_i x + \frac{1}{\beta}\mu_{k_{\max}})$$

Implement a function that accepts the image x and returns $\{\hat{z}_i\}$.

2. x-step: updating x is done by solving

$$\hat{x} = \arg\min_x \frac{1}{2\sigma^2}\|Hx - y\|^2 + \frac{\beta}{2}\sum_i \|Q_i x - z_i\|^2.$$

Implement a function that accepts an image y and a set of patches $\{z_i\}$ and returns $\hat{x}$. As shown in class, this can be done in the frequency domain:

$$\hat{X}(\omega) = (\frac{1}{\sigma^2}|H(\omega)|^2 + \beta M)^{-1}(\frac{1}{\sigma^2}H^*(\omega)Y(\omega) + \beta M Z(\omega)).$$

M is the number of pixels in a patch and $z = \sum_i Q_i z_i$ is an image constructed from $\{z_i\}$ by averaging overlapping patches. You may use the HQS x-step function you implemented in HW2. Pay attention to the modifications. As in HW2, use replicate padding.

3. Implement the full HQS scheme. Given an image y, initialize x to be y. Iterate between the z-step, x-step and increasing β.

4. Debluring: show the debluring results for the given image and blur kernels, and with a noise variance of $\sigma = 1$. Present the results and their PSNR values.

5. The proposed implementation is not the exact way of using the EPLL method, why? discuss the differences and explain how would you implement EPLL.

References

1. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. In: IEEE International Conference on Computer Vision (ICCV), IEEE (2011) 479–486