

School of Computing
National University of Singapore
CS5340: Uncertainty Modeling in AI
Semester 1, AY 2018/19

Released: 28 Sep, 2018

Due: 18 Nov, 2018 (Sunday by 2359hrs)

1. Objective

In this project, you will use probabilistic graphical models to solve two real-world problems. Specifically, you will implement the Gibbs sampling and variational inference algorithms for image denoising, and the Expectation-Maximization (EM) algorithm for image segmentation.

2. Data Denoising (60%)

Images captured from low quality photo-sensors are often corrupted by noise. Figure 1-(b) shows an example of an image corrupted by Gaussian noise. Using (i) Gibbs sampling and (ii) variational inference, respectively, we aim to (partially) recover the original image shown in Figure 1-(a) from the corrupted image.

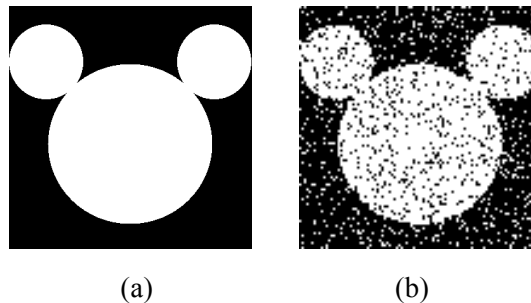
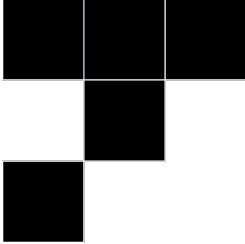


Figure 1. An example of image corruption by noise

Problem

[Description] Use the Gibbs sampling and variational inference algorithms to denoise the corrupted input data (*binary* images) we provide.

[Input] The input data is a binary image corrupted with noise. Two types of data files are provided, you may choose to read from either file as the input data. The first data file is in the PNG format (an image file format). The second data file is in the text format, where each line gives you a coordinate and the corresponding binary value denoted as $\{0,255\}$. An example is given below:

PNG format	Text format
	<pre> 0 0 255 0 1 0 0 2 255 1 0 255 1 1 255 1 2 0 2 0 255 2 1 0 2 2 0 </pre>
The input is a 3 by 3 image with only black (0) and white (255) color.	Each line is [x y value]. (x, y) is the coordinate of a pixel and value is the color of the pixel.

[Output] The output that you will generate is a binary image with the noise removed. The output file can be either in PNG or text format.

[Example] We provide an example result for you to check whether your result is reasonable. Note that your result might differ slightly from the example result due to the randomness in the sampling process. (This example result is also given in the data folder.)

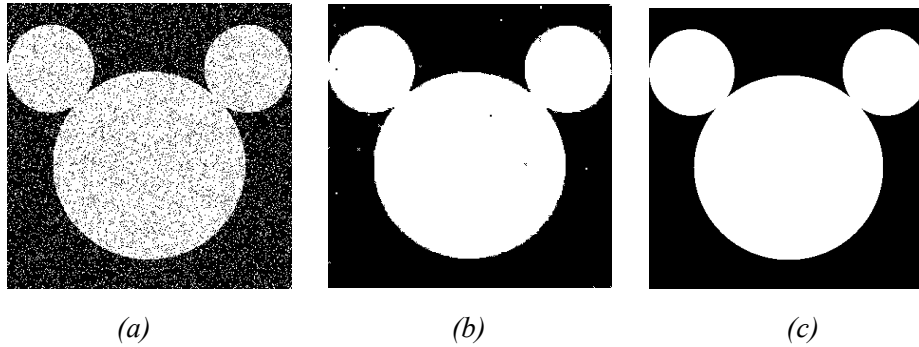


Figure 2. An example of data denoising. (a) The input corrupted image. (b) One possible result. (c) The original image without noise.

(i) Gibbs sampling algorithm for the image denoising task. **(30%)**

Gibbs sampling is one of the most popular algorithms to solve the image denoising problem. The idea behind Gibbs sampling is that we sample each variable in turn, conditioned on the values of all the other variables in the distribution. Given local evidence $\psi_t(x_t)$ and pairwise potential $\psi_{st}(x_s, x_t)$ between node X_s and X_t , the full conditional probability of a node X_t becomes

$$p(x_t | \mathbf{x}_{-t}, \mathbf{y}, \boldsymbol{\theta}) = \frac{\psi_t(x_t) \prod_{s \in \text{nbr}(t)} \psi_{st}(x_t, x_s)}{\sum_{x_t} \psi_t(x_t) \prod_{s \in \text{nbr}(t)} \psi_{st}(x_t, x_s)}, \quad (1)$$

where $nbr(t)$ means all connected nodes (neighbors) of node X_t . We use the Ising model for the edge potential, i.e., $\psi(x_s, x_t) = \exp(J x_s x_t)$, where $x_t \in \{-1, +1\}$ and J is the coupling strength. Furthermore, we use a Gaussian observation model, i.e., $\psi_t(x_t) = \mathcal{N}(y_t|x_t, \sigma^2)$, where $y_t \in \{-1, +1\}$ is the observed state on node X_t .

(ii) **Variational inference algorithm** for the image denoising task. (30%)

Variational inference is a method that seeks to approximate the posterior distribution $p(\mathbf{x}|\mathbf{y})$ with a fully factorized approximation

$$q(\mathbf{x}) = \prod_i q(x_i, \mu_i), \quad (2)$$

where μ_i is the mean value of node X_i . We use the Ising model as the prior

$$p(\mathbf{x}) = \frac{1}{Z_0} \exp(-E_0(\mathbf{x})), \quad (3a)$$

$$E_0(\mathbf{x}) = -\sum_i \sum_{j \in nbr(i)} W_{ij} x_i x_j, \quad (3b)$$

where Z_0 is the partition function and W_{ij} is the coupling strength between X_i and X_j . The likelihood has the form

$$p(\mathbf{y}|\mathbf{x}) = \prod_i p(y_i|x_i) = \exp(\sum_i -L_i(x_i)), \quad (4a)$$

$$-L_i(x_i) = \log(\mathcal{N}(y_i|x_i, c^2)), \quad (4b)$$

where c is a constant for the strength of $L_i(x_i)$ when $x_i \neq y_i$.

3. Expectation-Maximization Segmentation (30%)

Image segmentation is an important step towards scene understanding in autonomous systems. Figure 3 gives an example of image segmentation that decouples the image foreground from the background. The Expectation-Maximization (EM) algorithm is one of the earliest approaches used for image segmentation.

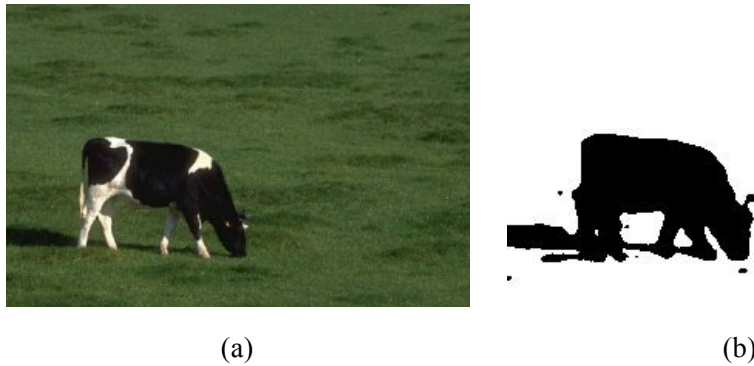


Figure 3. An example of image segmentation. (a) Original image. (b) segments of the original image

The image segmentation task can be formulated as a mixture model. Here, we assume that there are K segments and each of them is modeled as a Gaussian distribution with $\theta_k = (\mu_k, \Sigma_k)$. These segments form a Gaussian mixture model with each segment weighted by a mixing weight α_k . We encapsulate these parameters into a parameter vector $\Theta = (\alpha_1, \dots, \alpha_K, \theta_1, \dots, \theta_K)$. The Gaussian mixture model then has the form

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^K \alpha_k p_k(\mathbf{x}|\theta_k), \quad (2)$$

where

$$p_k(\mathbf{x}|\theta_k) = \frac{1}{(2\pi)^{d/2} \det(\Sigma_k)^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k) \right\}. \quad (3)$$

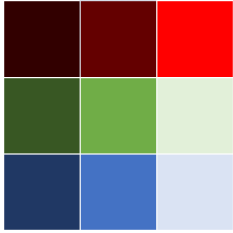
Since we know neither the component from which each data point (each pixel) came, nor the parameters of each model, we use the EM algorithm to alternatively solve the parameter vector Θ in Expectation step (E-step) and Maximization step (M-step).

In the E-step, we compute the probability that a pixel comes from a segment based on the model parameters from M-step. In the first iteration, the model parameters are initialized randomly. In the M-step, we compute the parameters θ_k of every Gaussian model based on the probabilities from E-step.

Problem

[Description] Use the EM algorithm to do image segmentation. For this project, you are required to segment the foreground from the background, i.e. the number of segments $K = 2$. We use image colors in **CIE-Lab** color space as the observed data $x_i \in \mathbb{R}^3$ for the image segmentation task. The *Lab* color space describes mathematically all perceivable colors in the three dimensions L for lightness and a and b for the color opponents green–red and blue–yellow. For more information on *Lab* color space (Nonetheless, note that you are not required to know the details of Lab color space to complete this project), please refer to https://en.wikipedia.org/wiki/Lab_color_space. We provide you the *Lab* color for each pixel as the observed data.

[Input] The input data is a RGB image. Two types of data files are provided, you may choose to read from either file as the input data. The first data file is in the JPEG format (an image file format). The second data file is in the text format, where each line gives you a coordinate and the corresponding *Lab* value. An example is given below:

PNG format	Text format
	0 0 6.1274 24.6731 9.6825 0 1 33.5168 -21.9055 26.2728 0 2 23.8528 6.5099 -28.7749 1 0 18.8440 40.3049 28.9655 1 1 64.7922 -37.7407 45.3595 1 2 48.5702 10.7215 -47.5421 2 0 53.2408 80.0925 67.2032 2 1 93.2322 -8.6952 9.5340 2 2 90.0049 0.0476 -8.8011
The input is a 3 by 3 RGB image.	Each line is [x y L a b]. (x, y) is the coordinate of a pixel and (L, a, b) is the <i>Lab</i> of the pixel.

[Output] There are three output images for an input image: mask images for foreground and background, image with only the foreground, and image with only background.

[Example] We give you an example of the output images to check whether your result is reasonable. Note that your results might differ slightly from the example results due to the choice of the initial values of the parameters. (This example result is also given in the data folder.)



(a)



(b)



(c)



(d)

Figure 2. An example of image segmentation. (a) The input image. (b) Mask image. (c) foreground image. (d) background image.

4. Implementation details

We provide the Python and MATLAB source code for file reading/writing. *read_data.m* can be used to read the text file to a matrix (2d array) and convert it to an image. *write_data.m* can be used to save a matrix (2d array) to a text file. *io_data.py* has same functions as MATLAB functions. Please read comments in the source file for details. You are recommended to use Python or MATLAB, but you can also use other programming languages.

You can use resources for mathematical operation, random number generation or data input/output. But you are not allowed to use any libraries or online codes for graphical models and the corresponding algorithms.

Before submitting to IVLE, please make sure there is no error in your source code and it is executable. If there is any special way to compile and run your code, please write it in your report or give a readme.txt file. Furthermore, adding proper comments in your source code is a good programming habit. It makes your code clear and easy to read.

5. Submission Instruction

Run your program on all images we provide to you (in “a1” and “a2” folders) and submit all the output results in a **report (10%)**.

In addition to the results, you are required to briefly introduce your algorithm and equations in the report, e.g. (1) what are the local evidence term and pairwise potential term you use in Gibbs sampling (2) what are the update rules you derive for your variational inference. (3) which data feature(s) you use in EM. If you use any 3rd party libraries or codes online, please indicate them in your report. You should also put your example output images for each problem in your report to illustrate your results. Please save your report in **pdf** format.

Please put your **codes**, **report** and **all output files** in a folder, zip the folder, and submit it on IVLE.

Other Information

This project constitutes **40%** of your final grade in CS5340. You are allowed to work in groups of **up to 2 students** per group. Note that **plagiarism** will not be condoned! You may discuss among different groups and check the internet for references, but you **MUST NOT** submit codes/reports that is copied directly from other sources!

Please indicate clearly the **names and matriculation numbers** of all the group members in the report.

Submit your project by **18th November 2018 2359HRS**. 25% of the total score will be deducted for each day of **late submission**.

If you have any question, feel free to ask TAs for help.

Xie Yaqi: yaqixie@comp.nus.edu.sg

Zhao Na: zhaona@u.nus.edu