CS5340 Project

Chen Xukun A0087649J
Muthiah Nachiappan A0094582W

1 Data Denoising

1.1 Gibbs sampling algorithm

Choice of coupling strength J and σ: From experiment we see that if J is too big (i.e. > 20), whole image would become black. If σ is too big (i.e. > 10), the shape of the item in the image would begin to distort. If both J and σ are too small (i.em < 0.8), then the noise is not getting removed. We set $\sigma = 1$ and coupling strength $J = 2$.

We define $x_i \in \{-1, +1\}$ as the state of node x_i. Our algorithm works as follows: Looping through pixels one by one, calculate $p(x_t|x_{-t}, y, \theta)$ and update x_t right away by getting a random number r_i between 0 and 1, if $r_i > p(x_t|x_{-t}, y, \theta)$, $x_i = -1$, else $x_i = 1$.

After the burn-in period, we accumulate x_i for every i, at the end of the algorithm, we output 255 if the accumulated value for $x_i > 0$ or 0 if the accumulated value for $x_i < 0$.

In total we loop through the image 8 times, we treated the first 4 iteration as burn-in, and final result is taken from averaging results from 5th iteration onward.

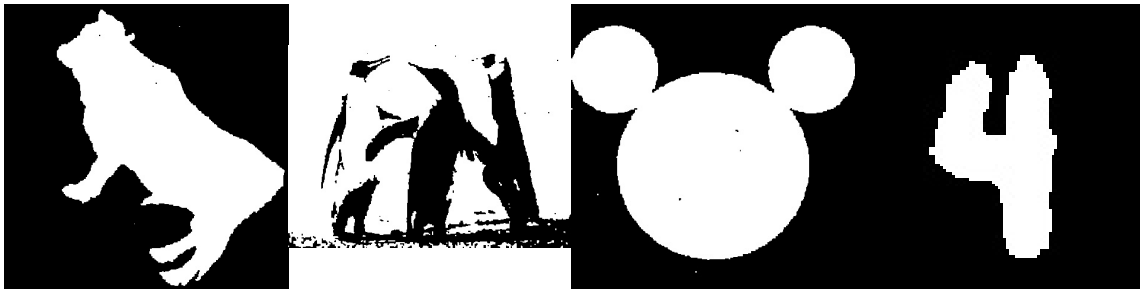


Figure 1: Gibbs Sampling Denoise: $J = 2$, $\sigma = 1$

1.2 Variational inference algorithm

We are using $q(x)$ to approximate posterior distribution $p(x|y)$ where $q(x) = \prod_i q(x_i, u_i)$ and $p(y|x) = \prod_i p(y_i|x_i)$. We define $x_i \in \{-1, +1\}$ as the state of node x_i, and $y_i \in \{-1, +1\}$ as the observed state of node x_i. Before we derieve $ELBO$, we need

to show that $p(x,y) = \prod_i p(x_i, y_i)$. We prove it as follows:
since $p(x,y) = p(x)p(y|x)$ and we know $p(y|x) = \prod_i p(y_i|x_i)$
We are also given that

$$\begin{aligned} p(x) &= \frac{1}{Z_0} exp(\sum_i \sum_j \in nbr(i) W_{ij} x_i x_j) \\ &= \prod_i \frac{1}{Z_0} exp(\sum_j \in nbr(i) W_{ij} x_i x_j) \\ &= \prod_i p(x_i) \end{aligned}$$

Hence we conclude that $p(x,y) = \prod_i p(x_i)p(y_i|x_i) = \prod_i p(x_i, y_i)$.

We defined the $ELBO$ under for this problem as:

$$\begin{aligned} ELBO &= q(x) log \frac{p(x,y)}{q(x)} \\ &= q(x) log(p(x,y) - q(x) log(q(x)) \\ &= \prod_i q(x_i, u_i) log(\prod_i p(x_i, y_i)) - \prod_i q(x_i, u_i) log(\prod_i q(x_i, u_i)) \end{aligned}$$

It is obvious that to minimze ELBO, we just need to make

$$\begin{aligned} log(\prod_i p(x_i, y_i)) &= log(\prod_i q(x_i, u_i)) \\ \sum_i log(p(x_i, y_i)) &= \sum_i log(q(x_i, u_i)) \end{aligned}$$

As $x_i \in \{-1, +1\}$, it is obvious that $q(x_i, u_i)$ is of Bernoulli distribution.

To simplify this problem even more, we just need to calculate u_i for all i such that $p(x_i, y_i) == q(x_i, u_i)$. More explicitly, to calculate:

$$\frac{1 - u_i}{u_i} = \frac{p(y_i, x_i = 0)}{p(y_i, x_i = 1)}$$

For each iteration, we loop through all pixels one by one, calculate u_i. At the end of each iteration, we calculate x_i for all i using corresponding Bernoulli distribution: get a random number r_i between 0 and 1, if $r_i > u_i$, $x_i = 0$, else $x_i = 1$.

We set coupling strength $W_{i,j} = 2$ and $c = 1$ to be consistent with the parameters used in Gibbs sampling algorithm.

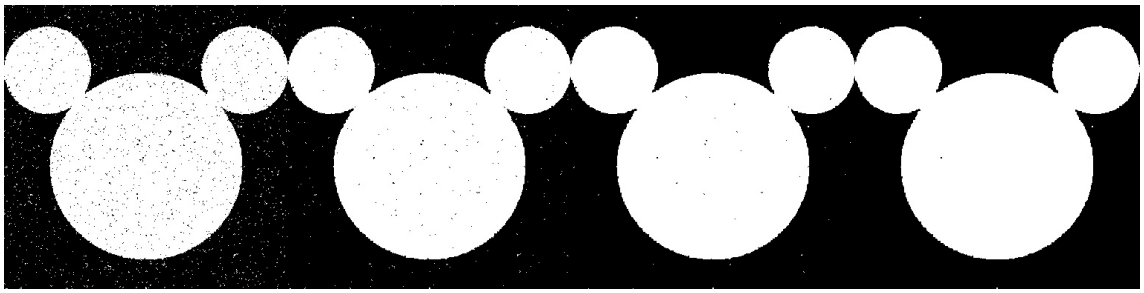
In total we loop through the image 5 times, and take the output from 5th iteration as the output.

(a) 1^{st} iteration (b) 2^{nd} iteration (c) 3^{rd} iteration (d) Final Result

Figure 2: Variational Inference Denoise: $J = 2$, $\sigma = 1$

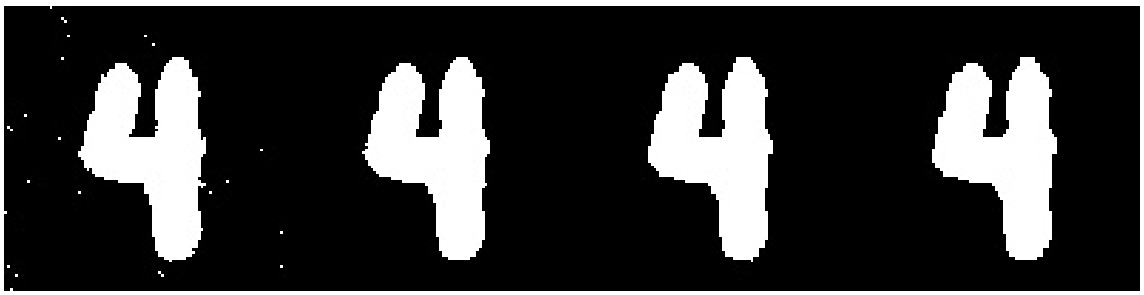
(a) 1^{st} iteration (b) 2^{nd} iteration (c) 3^{rd} iteration (d) Final Result

Figure 3: Variational Inference Denoise: $J = 2$, $\sigma = 1$



(a) 1^{st} iteration (b) 2^{nd} iteration (c) 3^{rd} iteration (d) Final Result

Figure 4: Variational Inference Denoise: $J = 2$, $\sigma = 1$



(a) 1^{st} iteration (b) 2^{nd} iteration (c) 3^{rd} iteration (d) Final Result

Figure 5: Variational Inference Denoise: $J = 2$, $\sigma = 1$

2 Expectation-Maximization Segmentation

2.1 Problem Statement

In this section, we focus on using probabilistic graphical models to segment an image into 2: foreground and background. Each pixel belongs to one of the segments and the assignment of each pixel can be modelled as a categorical latent variable (Z), while the pixel value is the observed variable (X). The observed variables can be modelled as a multivariate normal distribution. The observations are 3-dimensional and each pixel can either be represented in the RGB space or the Lab space.

2.2 Approach

This is a clustering problem with 2 clusters, each represented as a Gaussian. The input is an image with N pixels and each pixel is represented as a 3-dimensional random variable $\boldsymbol{x}_n$ where, $n = 1, ..., N$. $\boldsymbol{z}_n$ is a latent random variable that denotes the segment each observation belongs to and follows a categorical distribution that can be represented as:

$$p(Z) = \prod_{k=1}^{2} \pi_k^{z_k}$$

where the parameter $\pi = [\pi_1, \pi_2]$ represent the probability of Z taking each of the 2 states. This paramater also represents mixing coefficients of each cluster's Gaussian. The Gaussian mixture model is given by:

$$p(\boldsymbol{x}) = \sum_{k=1}^{2} \pi_k \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$$

The responsibility that each segment k takes for explaining each observation $\boldsymbol{x}_n$ is given by $\gamma(\boldsymbol{Z})$ which is an N x 2 table. We aim to find the parameters, θ, that maximize the log-likelihood of the model, which is given by:

$$lnp(\boldsymbol{x}_1, ..., \boldsymbol{x}_N|\theta) = \sum_{n=1}^{N} ln \sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\theta), \theta = \{\pi_1, \pi_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\} \quad (1)$$

Since we cannot obtain a closed form solution of the unknown parameters by setting the derivatives to 0, we initialize the parameters to sensible initial values and iteratively calculate new responsibilities and new parameters until the log-likelihood converges. To obtain sensible initializations, we first perform K-means clustering to estimate π_k, μ_k and Σ_k for each cluster. Then we proceeded to the expectation step where we calculated the responsibilities as:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}\left(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{2} \pi_j \mathcal{N}\left(\boldsymbol{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}$$

Next, in the maximization step, we re-estimate the parameters using the current responsibilities as such:

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new}\right) \left(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new}\right)^T$$

$$\boldsymbol{\pi}_k^{new} = \frac{N_k}{N}$$

where,

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

After every iteration, we calculate the log-likelihood as specified in Equation **??**. The difference in the increase in log-likelihood after every iteration was computed. If the difference was less than 1×10^{-6}, the iteration was stopped and the final responsibilities were calculated. The segment that had the highest responsibility for each observation was assigned as the label.

2.3 Results and Discussion

Using EM algorithm, we successfully segmented all 4 test images into distinct foreground and background segments, with minimal noise. We experimented with using 3 features (the L, a, b channels) and 5 features (x-coordinate, y-coordinate and the L, a, b channels). We found that the results were better when 5 features were used. The input data was scaled along each feature independently using their respective means and standard deviations. We did not notice any difference in result between using the raw data and scaled data. However, the runtime was faster when scaled data was used.

(a) original Image (b) mask (c) background (d) foreground

Figure 6: Segmentation

(a) original Image (b) mask (c) background (d) foreground

Figure 7: Segmentation

(a) original Image (b) mask (c) background (d) foreground

Figure 8: Segmentation

(a) original Image (b) mask (c) background (d) foreground

Figure 9: Segmentation

3 Code Instruction

Before running the code, please make sure all input images required are inside $a1$ and $a2$ folder.

1. To run Gibbs Sampling: **python gibbs_sampling.py**
2. To run Variational Inference: **python variational_inference.py**
3. To run Image Segmentation: **python ImageSegmentation.py**