Inference

Farrel Zulkarnaen[1] and Khai Xi Lim[2]

[1] *36772*
[2] *39084*

December 20, 2018

Introduction

To say that we have learnt implies that we have gain new found knowledge by way of inference over some observations. Such is the basis of learning, and the aim of any machine learning task. By learning, we seek to infer a conclusion based on evidence of observations, whereby then gaining an updated belief: a posterior. However, there are times where the marginal distribution of our evidence are intractable and we can no longer exploit conjugacy; as such, to reach our posterior we must settle with a convenient yet still meaningful approximation.

1 Approximate Inference

In this report we will show the utility of approximate inference in the context of image restoration for binary (black or white) images. To start, we choose an arbitrary colour image in RGB and convert it to greyscale for computational ease (a typical first approach in any image processing task). The image that we choose are shown in Figure 1.

(a) (b) (c)

Figure 1: *1(a): Original coloured image, 1(b): Greyscale representation, 1(c): Binary representation.*

Our aim is to restore the binary representation of the original image shown in Figure 1(c) after it has been corrupted by some binary noise. For consistency, the remaining of this report will follow these notation.

Let $\mathbf{y}$ be our observed noisy pixel value and $\mathbf{x}$ the underlying latent variable that have generated the observation, such that a white pixel corresponds to $\mathbf{x} = 1$ and black equates to $\mathbf{x} = -1$. We seek to regain $\mathbf{y}$ before the corruption such that it represents the pixel values of the greyscaled image i.e. $y_i \in (0, 1)$[1] .We will apply two types of noise: Gaussian noise, where a proportion of the pixel will be corrupted by a noise factor ϵ where $\epsilon \sim \mathcal{N}(0, 0.1)$, and 'salt-and-pepper' noise which flips the binary pixel value.

1.1 Question 1

Iterative conditional modes (ICM) is an algorithm that is basically coordinate-wise gradient ascent. First, we initialise the variables x_i to a deep copy of the noisy image y_i. After that, we calculate the probability of a pixel being 1 or -1, based on the values of the surrounding pixels and the initial value. It then sets the corresponding pixel to the value at which the probability is higher. Our task is then to obtain the joint probability distribution $p(\mathbf{x}, \mathbf{y})$ for each pixel, which is shown below.

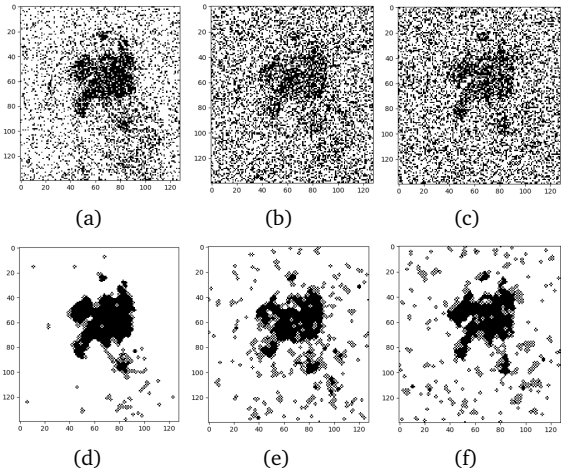
$$\begin{aligned} p(\mathbf{x}, \mathbf{y}) = \quad & p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \\ = & \frac{1}{Z_1} \prod_{i=1}^{N} e^{L_i x_i} \frac{1}{Z_0} \exp\left(\sum_{n \in \mathcal{N}(i)} w_{ij} x_i x_j \right) \end{aligned} \tag{1}$$

For the above equation, variables Z_1 and Z_0 are merely normalisation constants, N refers to the number of pixels in the image, $\mathcal{N}$ refers to neighbouring pixels, and w_{ij} refers to a weight, which defines the strength of the prior $p(\mathbf{x})$. First and foremost, we have to define a likelihood function, L_i. For the case of ICM,

[1] (0,1) is the range of values in a greyscale image where 0 corresponds to black, 1 to white, & the values in between are varying intensities of grey.

a simple function was chosen, which returns a higher probability when x_i and y_i return the same value, and a lower probability when they are different. β refers to the weight, which defines the strength of the likelihood function, $p(\mathbf{y}|\mathbf{x})$.

$$L_i(x_i) = \beta \sum_i x_i y_i \tag{2}$$



(a) (b) (c)

(d) (e) (f)

Figure 2: *2(a): Noisy Gaussian, prop = 0.7, var = 0.5, 2(b): Noisy Gaussian, prop = 0.7, var = 2, 2(c): Noisy SNP, prop = 0.3, 2(d): Result image, prop = 0.7, var = 0.5, 2(e): Result image, prop = 0.7, var = 2, 2(f): Result image, prop = 0.3.*

The results are shown in Figure 2. For low levels of Gaussian Noise, the ICM algorithm cleans the surrounding noise up pretty nicely, and maintains the detail of the face of the pug. The detail of the doughnut disappears however. For higher levels of Gaussian Noise as well as Salt and Pepper Noise, the ICM struggles to clean up the surrounding noise, although the detail of the face of the pug still maintains.

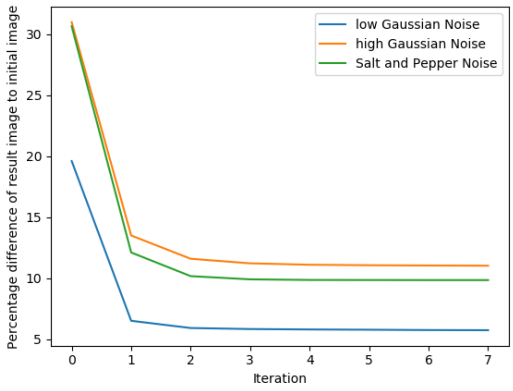


Figure 3: *Graph of difference in percentage for the final, resulting image compared to the initial, clean image for each iteration*

An interesting thing to note about ICM is that the algorithm converges to a limit after a certain number of iterations, whereby subsequent iterations will not improve the image clarity. This can be shown in Figure 3, where after the 1st iteration, a large majority of the noise has been cleared out and we are getting decent results. Subsequent iterations show a small reduction in noise, until there is no change in reduction of noise. A convergence limit was also set for the algorithm in order to prevent unnecessary iterations.

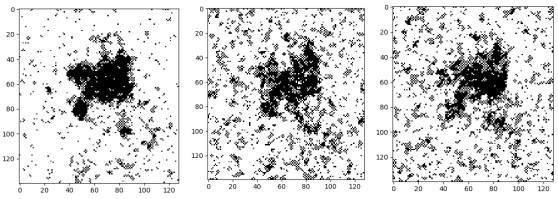
1.2 Question 2

The general idea behind Gibbs sampling is that for each pixel in the image, we calculate the probability of the pixel having value of $x = 1$, given the value of its neighbours as well as its initial value. A random sample, t, is then obtained from a uniform distribution of 0 to 1, and compared to $p(x = 1)$. If the value of $p(x = 1)$ is higher than t, the pixel is flipped to -1. If it $p(x = 1)$ is lower than t, the pixel is kept at -1. Calculations of $p(x = 1)$ on subsequent pixels would then take into account the previously updated pixels, which is the core concept of Gibbs Sampling.

For the purposes of this project, the calculation of $p(x_i = 1|x_{\neg i}, \mathbf{y})$ was given by the formula:

$$\frac{p(y_i|x_i = 1)p(x_i = 1, \mathbf{x}_{\mathcal{N}(i)})}{p(y_i|x_i = 1)p(x_i = 1, \mathbf{x}_{\mathcal{N}(i)}) + p(y_i|x_i = -1)p(x_i = -1, \mathbf{x}_{\mathcal{N}(i)})} \tag{3}$$

where $\mathbf{x}_{\neg i}$ means for all dimensions of x except for i, and $\mathbf{x}_{\mathcal{N}(i)}$ refers to the neighbouring pixels for each pixel i.



(a) (b) (c)

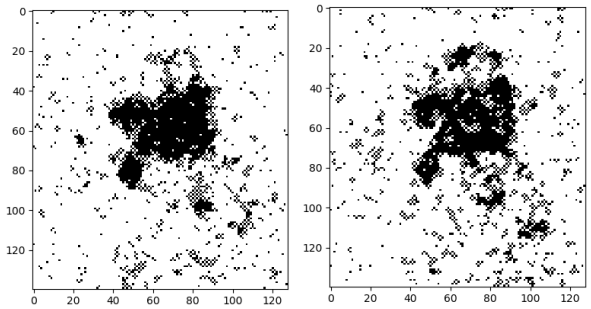
Figure 4: *4(a): Noisy Gaussian, prop = 0.7, var = 0.5. Percentage difference = 9% 4(b): Noisy Gaussian, prop = 0.7, var = 2. Percentage difference = 18% 4(c): Noisy SNP, prop = 0.3. Percentage difference = 16%*

Figure 4 shows the results of Gibbs Sampling after 1 iteration. If we compare the results of Gibbs Sampling to ICM after 1 iteration, we can see that generally, ICM performs better in cleaning up noise after the 1st iteration. This is of course, an unfair comparison as different values of the weight of the prior were used for both algorithms. A smaller weight would result in more iterations required, and values that are too small would result in the algorithm not being able to denoise image at all, as we are no longer comparing between 2

different possibilities of pixels, but we are comparing the probability that a pixel is 1 to a value randomly sampled from a uniform distribution.

1.3 Question 3

In this section we focus on the aspect of cycling through the pixels one by one. The order of pixels was randomised and the result is shown in Figure 5. Generally, randomising the order of pixels at which the algorithm cycles through does not help with the clarity of image. The main effective difference is that since the order is random, certain pixels could have been cycled through more than others, while other pixels could be not updated at all. Therefore, it is clear that the order of pixels is not a vital factor.

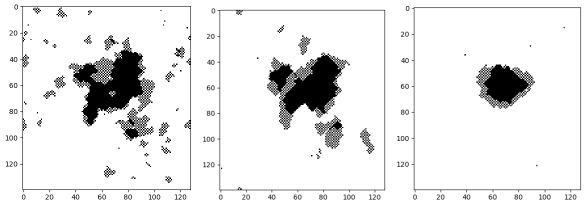


(a) (b)

Figure 5: *5(a): Result image of normal iteration 5(b): Result image of random iteration*

1.4 Question 4

When we are running the Gibbs Sampler for multiple iterations, it is obvious that more iterations of Gibbs Sampling would lead to a decrease of detail. As seen in Figure 6 we can see that as the number of iterations increase, the detail of the pug's face disappears into a black round figure. Furthermore, if we continue to run the Gibbs sampler on a clean image, the algorithm is merely removing original detail from the image. This is because diagonal lines are not accepted by the sampler after many iterations, and are gradually sacrificed. Therefore, it is a delicate balance to settle on the number of iterations for the most clean image. A general observation for our results was that after around 5 iterations, we can see that the percentage difference of resulting image to the noisy image becomes more inconsistent, and subsequent iterations result in slight increases and decreases in percentage difference. Therefore, the best method to determine the ideal number of iteration could perhaps be stopping the iteration when the percentage difference starts to show a slight uptick.



(a) (b) (c)

Figure 6: *6(a): Result image after 5 iterations 6(b): Result image after 10 iterations 6(c): Result image after 100 iterations*

1.5 Question 5

To use Variational Bayes for our model as inference, we must use the property of The Kullback-Leibler divergence (KL) as a measure of 'similarity between' two distributions; specifically the distribution of our latent variable $q(\mathbf{x})$ and the distribution of our posterior $p(\mathbf{x}|\mathbf{y})$ noted in Equation 4.

$$KL(q(\mathbf{x})||p(\mathbf{x})) = \int q(\mathbf{x}) \log\left(\frac{q(\mathbf{x})}{p(\mathbf{x}|\mathbf{y})}\right) \quad (4)$$

Notice that KL is not a symmetric measure, an intrinsic property to note to estimate the lower bound for our marginal likelihood. Therefore, we seek to minimise the bound by having $q(\mathbf{x})$ as close to $p(\mathbf{x}|\mathbf{y})$ as possible, i.e. when $q(\mathbf{x}) = p(\mathbf{x}|\mathbf{y})$, $KL = 0$ because log of 1 implies KL to converge to zero. Likewise, on the other hand if the denominator in the logarithmic tends to 0 whilst numerator remains high, then the KL will diverge to infinity, showing that KL is not a symmetric.

$$KL(q(\mathbf{x})||p(\mathbf{x})) \neq KL(p(\mathbf{x})||q(\mathbf{x}))$$

1.6 Question 6

We now know that the KL divergence determines the similarity between our evidence p($\mathbf{y}$) and the $ELBO$ as seen in Equation 5.

$$\log p(\mathbf{y}) = KL(q(\mathbf{x})||p(\mathbf{x}|\mathbf{y})) \quad + \underbrace{\mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{x},\mathbf{y})] - H(q(\mathbf{x})}_{\text{ELBO}} \quad (5)$$

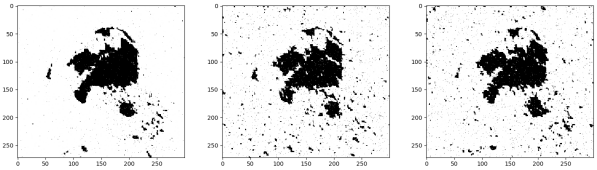
Which implies that maximising the $ELBO$ would be the same as minimising the KL divergence to preserve the equality of the equation. And one way to find $q(\mathbf{X})$ that maximises the $ELBO$ is to use Mean Field Approximation on Variational Inference (Mean Field Variational Bayes).

$$q(\mathbf{x})|_{x_i=1} = q(x_i = 1|\mu_i) \quad (6)$$

where

$$\mu_i = \tanh\left(\sum_{j\in\mathcal{N}(i)} w_{ij}x_i x_j + \frac{1}{2}(L_i(1) - L_i(-1))\right)$$

And this goes the same for calculating $q(x_i = -1)$. By updating μ_i we can now define the probability $q(x_i = 1 \quad \text{or} -1)$ for every pixel in the image. The hyperbolic tangent function ensures for every μ_i, the a pixel will either be 1 or -1 depending on if $L_i(1)$ is larger than $L_i(-1)$ or vice versa respectively. Below are the results of applying the Mean Field Variational Bayes to gain the $q(\mathbf{x})$ that minimises the KL divergence



(a) (b) (c)

Figure 7: *Results of mean field approximation given 7(a): Gaussian noise with prop = 0.7 & varSigma 0.5, 7(b): Gaussian noise with prop = 0.7 & varSigma = 2, 7(c) SaltPeppar noise with prop = 0.3*

The images from Figures 7 shows the denoised image for 3 different configurations of parameters $prop$ and $varSigma$ for just 1 iteration. For Figures, 7(a), 7(b), 7(c), the percentage of pixels that have been denoised are approximately 95.7%, 92.3%, and 92.8% respectively.

1.7 Question 7

Previously we have utilised stochastic approximation methods in the case of ICM and Gibbs Sampling, whereas Variational Bayes is a form of a non-stochastic approximation, i.e it is deterministic. The advantage of Variational Bayes is that it an alternative to stochastic approximation method like Markov Chain Monte Carlo algorithms such as Gibbs Sampling, as it provide a locally-optimal deterministic (exact) posterior using a set of samples. It is easy to gauge convergence in Variational Bayes, implying that it will take less iterations (typically less than dozens. On that note, we can view Variational Bayes as an extension of Expectation Maximisation (EM)[**1**] for finding the most likely hidden parameter, in our case μ_i. We can see evidence of this if you compare the results given by Mean Field Variational Bayes with those of ICM and Gibbs Sampling. For Figures 7, with just 1 iteration about 90% of all pixels have been denoised.

1.8 Question 8

This section aimed at using Gibbs Sampling in order to perform image segmentation of a full coloured (RGB) image. A different image was also used as the image in Figure 1 did not have a clear colour contrast. Our end result should be segmenting the background and foreground, so we replace our latent variables with background being 1 and foreground being -1. First and foremost, the foreground and background masks were labelled, and for the pixels in each mask, the RGB colour density was measured in the form of 3 histograms, representing each colour. Next, the histograms were normalised, which enables us to effectively treat the histograms as a probability distribution. This helps us to define our likelihood function, where our main process will be that given a new pixel, calculate the probability that the pixel comes from the foreground $p(y_i|x_{fg})$, or background, $p(y_i|x_{bg})$. The prior is still defined by neighbouring pixels and the foreground and background histograms are updated after every iteration. As the number of bins for the foreground and background histograms decreases, the histogram is increasingly smoothed out.

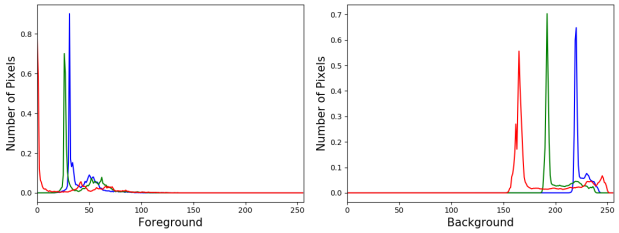


Figure 8: *Graph showing the colour distribution histogram in the foreground 8(a) and background 8(b)*

Figure 9: *Original pic (left) and segmented pic (right)*

2 Variational Auto-Encoders

Auto-Encoders are a means of reconstructing an input data from a small compressed space back into a new representation of the original input with minimum loss of information. It consist of two main layers, an encoder layer that encodes and compresses all information from the input space into a 'bottle-neck' hidden layer that feeds it to the 'decoder' layer which tries to gain back the original input in a new representation.

A particular example would be in the context of internet bandwidth. For a a given high resolution image, to send the image in an uncompressed stage would require a larger bandwidth, therefore using an auto-encoder ensures that the image can be send between devices with minimum bandwidth without loss of resolution. However, we don't actually need to output a representation of our original input data, we could in fact utilise the neural network in an auto-encoder in the same context as we have been previously been doing, to decode an original unmasked image from latent noises and/or to produce a segmented copy of the original image.

2.1 Question 9

We have seen previously that we can determine the most likely $q(\mathbf{x})$ by Mean Field Variational Bayes, but we can also do so utilising artificial neural networks like Variational Auto-Encoders. Whereby the objective now would be to minimise the loss function so that whatever the decoder outputs would resemble the encoded input as much as possible, which is practically the same setting as we have been doing so far, trying to maximise $q(\mathbf{x})$ so that it resembles our posterior $p(\mathbf{x}|\mathbf{y})$ as much as possible. This is more clearly illustrated in Figure 10.

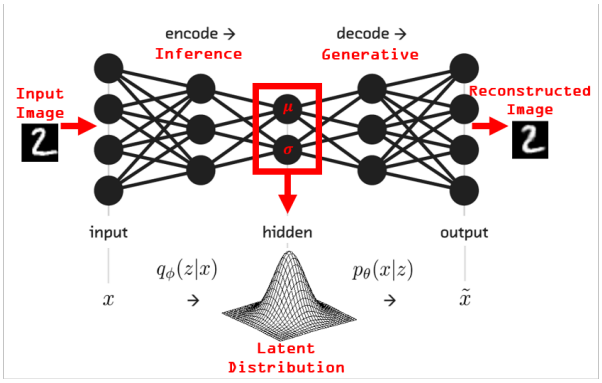


Figure 10: *A graphical model of an Variational Auto-encoder [2].*

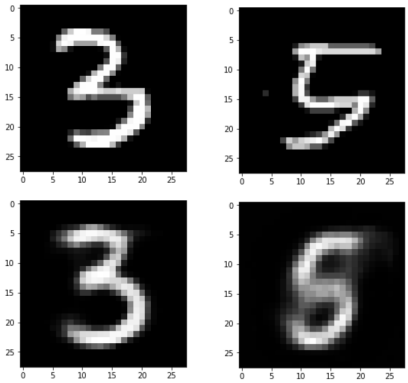
The main difference between a general Auto-Encoder and a variational one is that a Variational Auto-Encoder presents two variables in the 'bottle-neck' hidden layer of the network, corresponding to the mean and variance of a latent distribution. This implies that the inference ends as we make the assumption that every information regarding the input data is govern by some latent distribution with just two parameterising variables: μ and σ that corresponds to the hidden layer of the network. Then the decoder works as a generative model to generate the encoded input back by sampling from those two parameters. And because this is a deep learning model, the loss function given by the decoder would be back propagated to continuously update the result. You can see further similarity between Variational Bayes and Variational Auto-Encoders as the loss function of our network is given by Equation 7

$$loss function = -ELBO = -(\mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{x}, \mathbf{y})] - H(q(\mathbf{x})) \tag{7}$$

Hence, to minimise the loss function would equate to maximising the $ELBO$,and therefore minimising the KL divergence of our variational latent distribution with our posterior.

2.2 Question 10

Given some mean μ and variance σ that represents latent distribution of the encoded input, the decoder will sample from the hidden layer randomly given those two parameters, and will output a copy of the input as best as possible. However, as the encoder tries to compress all information into a lower dimension, which then would be parameterised by μ and σ, it would then pose a slight difficulty for the decoder to generate back the compressed information in to the original dimensional size. It would still succeed in outputing the same dimension as the original input, but depending on how small of a reduced dimension that the encoder tries to seek, there could potentially be loss of detail. This can be seen in the MNIST example.



(a) (b)

Figure 11: *Variational Auto-Encoder on MNIST example.*

As seen in Figure 11, the digit 5 is more fuzzy compared to 3. This is because as for 5, the encoder tries to reduced the information for that digit into a much lower dimension than that of 3. Because we are forcing all the information carrying the structure of that digit into a lower compressed dimension, the decoder will find it harder to uncompressed it without losing detail, hence the fuzziness. Additionally, you can see the 5 'morphing' into either 6 or 8, and this is because the decoder for digit 5 is potentially sampling from values of μ and σ that is closer to those for the latent distribution of 6 or 8.

References

[1] Neiswanger, Willy; Liu, Min Xing. *Variational Bayes* https://www.cs.cmu.edu/~epxing/Class/10708-17/notes-17/10708-scribe-lecture13.pdf

[2] Wade W., Rebeccal; *Variational Auto-Encoders.* https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder