Inference

Mark Ergus Nicholl[1] and Faizaan Sakib[2]

University of Bristol
[1] *35358*
[2] *38655*

December 7, 2018

Certain models will have a posterior that is intractable, meaning we have to take a different approach in order to make use of the data. In this case, inference is used to approximate the intractable integrals found when computing the marginal likelihood or evidence for the model. To do this, we must combine our assumptions with the observed data. In the case of images, the marginal likelihood is often computationally intractable. This might not be the case in our 128x128 image, but it definitely is when the scale increases. For this paper, we explore the amazing ways we can reach a posterior distribution without explicitly calculating the marginal probability of the distribution

1 Approximate Inference

1.1 Iterative Conditional Modes (ICM)

Q1 After implementing ICM for binary images, the algorithm was tested on the chosen input image with varying noise proportions. By using the chain rule, we can deduce that :

$$p(x_i = 1, x_{\neg i}, y) = p(y|x_{\neg i}, x_i = 1)p(x_{\neg i}|x_i = 1)p(x_i = 1) \tag{1}$$

$$p(x_i = 0, x_{\neg i}, y) = p(y|x_{\neg i}, x_i = 0)p(x_{\neg i}|x_i = 1)p(x_i = 0) \tag{2}$$

And, since we are comparing values and :

$$p(x_{\neg i}|x_i = 1) = p(x_{\neg i}|x_i = 0) \tag{3}$$

We can ignore the above terms. While taking the properties of each pixel being in a Markov blanket, we only need to compute :

$$p(y|x_j, x_i = 1)p(x_i = 1) \tag{4}$$

and

$$p(y|x_j, x_i = 0)p(x_i = 0) \tag{5}$$

to compare, where j are neighbouring nodes of i. As we only need the information in the Markov blanket of each node, we do not need to factorise the likelihood over the whole image, which reduces a lot of calculations. The second term is the prior of the pixel at i. To compute our likelihood, we needed to choose an L function. For this purpose, we used :

$$L(x_i) = -(|x_i - (2y_i - 1)|^2) \tag{6}$$

This function has a nice feature of being upper bounded by 0, and the ability to go to very small values. This makes it ideal as the likelihood it remains a distribution at its upper bound and can give a small likelihood compared to 1, where appropriate.

Figure 1: *In order of position: Original input image, Image with added Gaussian noise of proportion 0.4, Image with added Salt and Pepper noise of proportion 0.4*

For the Gaussian noise images, we obtained very clean output similar to the input for different noise levels just with a single iteration. To reach the minima it took 4 iterations. The 'Salt and Pepper' noise images did not fare as well though. However, considering the

much higher amount of distortion, it did manage to produce an image resembling the original one after 9 iterations. A higher noise level of 0.4 did take significantly longer at 17 iterations and produced a less cohesive image.

Figure 2: *The left two images were derived from a noise proportion of 0.4 and 0.7 respectively. The image on the right is the result of just one iteration of ICM from noise of 0.7.*

Figure 3: *Output from SnP images with noise of 0.2, 0.4 and 0.7, along with iterations of 9, 17 and 18 respectively.*

It is important to note for SnP images with a higher proportion of noise, it seems that the output is becoming inverted. This makes sense as when the proportion is greater than a half, more of the black pixels invert to white and vice versa. Hence, the appropriate action to take is to alter our L function, as now a -1 latent variable has a high chance of generating a white, instead of a black pixel.

In order to produce these results from the latent variable, we need to encode our prior which involves using the information we can get about the current pixel, from its surrounding pixel values.

$$p(x) = \frac{1}{Z_0} e^{E_0(x)} \tag{7}$$

where

$$E_0(x) = \sum_{i=i-1}^{i+1} \sum_{j=j-1}^{j+1} w x_{ij} \tag{8}$$

Here, a sum of the product of w, which is the current pixel being evaluated, and the available neighbours is taken. As a result, each value within the summation will either be +1 or -1, which means the sum will be positive indicating the current pixel is more likely to be white and vice-versa.

2 Stochastic Inference

2.1 Gibbs Sampling in an Ising Model

Q2 Once again, the Gaussian noisy images have come out very close to the original image through the Gibbs sampler, while also taking fewer iterations. Only 1 iteration for noise proportion of 0.4 and 0.7, while 3 iterations for 0.9.

Figure 4: *Output with Gibbs Sampler from Gaussian noisy images with noise proportions of 0.4, 0.7 and 0.9.*

Apart from the edges being slightly less smoother compared to the original (same with ICM), the output is identical to the original.

The Gibbs sampler performs better on the SnP images compared to ICM. It does well on images with lower noise, getting slightly worse as more noise is added. It seems to have difficulties with small areas of black/white within regions of the opposite colour. Unlike the Gaussian noise images, the SnP ones doesn't reach a minima for many iterations.

As done in ICM, we use a similar method in the Gibbs sampler of obtaining the likelihood and the prior to use in our posterior, where we use the neighbours $\mathbf{x}_{\mathrm{N(i)}}$ of the current pixel x_{i} instead of the entire image as this would require a lot more of computation time on top of the existing time.

Figure 5: *Output with Gibbs Sampler from SnP images with noise proportions of 0.05, 0.15 and 0.35. All ran for 20 iterations.*

Q3 When cycling through the nodes in the graph randomly, the images are mostly recovered quickly within an iteration or two, for Gaussian noise. But since every iteration it means certain pixels will be updated more times than others, it means some may be left having not converged to the actual value. After 2 iterations, there seems to be a couple of pixels out of place, notably the white pixel on the glasses. This is corrected by the 10^{th} iteration, but leaves behind another incorrect black pixel on the hat. By the 30^{th} iteration, the image is at the same level as what the Gibbs Sampler produced without a random index.

For SnP noise, this approach gives results of the same kind, where it seems to be converging to the image it would without random traversal, but it will take more iterations to get to the same level of recovery.

Figure 6: *Output with Gibbs Sampler using random node traversal on image with Gaussian noise. Run at 2, 10 and 30 iterations from left to right.*

Figure 7: *Output with Gibbs Sampler using random node traversal on image with SnP noise. Run at 2, 20 and 40 iterations from left to right.*

Q4 It's known due to its high computational requirements that the Gibbs Sampler using the MCMC method is fairly slow. As such, getting to the correct image may take a long time. We tested on SnP images, since the Gaussian images were recovered almost immediately. On images with a noise proportion of 0.15 and 0.35, generally, using more iterations has shown improvements on the results. This is evident in the images below. However, it can be seen that the images do converge to a point of close resemblance to the recovered image, but iterating past this can cause the image to steer away from what the original image was. The number of iterations taken to converge to best image is dependent on the amount of noise that was in the input image.

Figure 8: *Output from SnP images using Gibbs Sample with noise proportion of 0.35, running for 20, 100, 300 and 450 iterations respectively.*

The images above have been iterated through 20, 100, 300 and 450 times. As mentioned before, the Gibbs Sampler seems to converge at around the 300th iterations, but at 450 incorrect pixels start to appear. Continuing on from this, the image does not return to its previous, more correct version.

3 Deterministic Inference

3.1 Variational Bayes with Mean Field Approximation

Q5 There are typically 2 forms of KL-divergence we could use, forward KL-divergence $(KL(p(x)||q(x)))$ and reverse KL-divergence $(KL(q(x)||p(x)))$. Since they are not symmetric, there are different ways in which we fit q(x) to p(x) as well as possible.

When using forward KL, $(\int p(x)log\frac{p(x)}{q(x)})$, if $p(x)$ increases and $q(x) \to 0$, the integral becomes very large. This happens when $q(x)$ fits over a mode and not over other parts of $p(x)$ This is not suitable as we want to mimimise the function. As that is the case, we need to fit $q(x)$ over as much of the probability mass as possible. Hence the name mean seeking/zero avoiding.

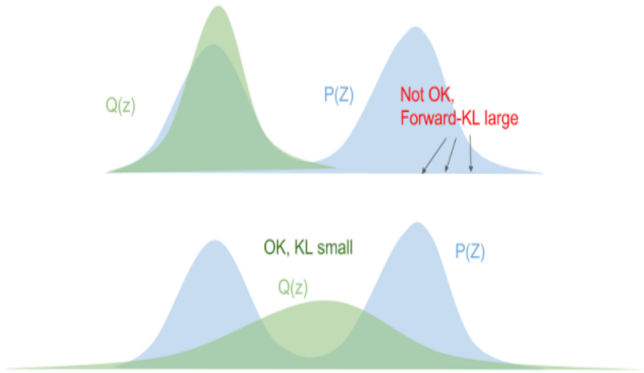

Figure 9: *Optimizing forward KL fit.*

Conversely, in reverse KL, $(\int q(x)log\frac{q(x)}{p(x)})$,as the integral is weighted over $q(x)$, we can minimise the function by seeking $q(x) = 0$. By forcing this, we can minimise the function. Therefore, it is also known as zero forcing/mode seeking.

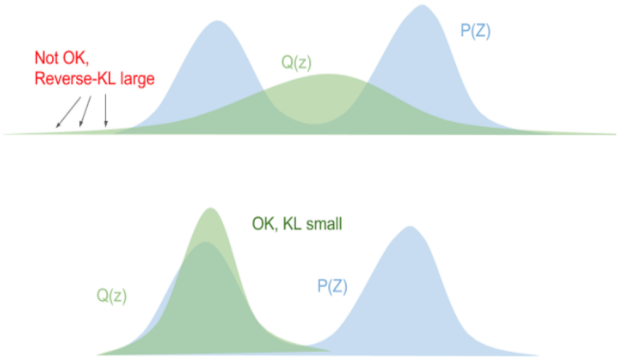

Figure 10: *Optimizing reverse KL fit.*

Q6 Denoising the images as before, but now with the Variational Bayes method, we get these results :

Figure 11: *The Gaussian image denoised with variational Bayes on the left, followed by the SnP image on the right*

Iterations/Time(s)	ICM	Gibbs	V. Bayes
1	1.275	1.4	0.203
10	11.008	11.646	2.303
50	54.729	63.08	9.778

Table 1: *Run-times for different inference methods for different number of iterations*

Q7 The most noticeable difference between the Variational Bayes method and the the two previous method is the runtime. Per iteration, the Variational Bayes method is significantly faster, as shown in Table 1. In terms of 'accuracy' with respect to the initial image, there is not much difference in the outcome, as both were largely accurate with Gaussian noise and slightly off for SnP. We personally did not expect this result, we were expecting the MCMC method to produce a better result, given that MCMC is asymptotically exact whereas variational Bayes can only provide an approximative posterior. This result could be caused by a few underlying issues. As the model is rather simple, the variational Bayes could have approximated fairly close to the actual posterior. If our Gibbs sampling result was accurate, this would show that this hypothesis is correct as both are fairly similar. In the case that there is a less than perfect implementation of our Gibbs model, it would not converge on the actual posterior. In that case, the variational Bayes could have done just as well by chance. It would be interesting, given the time, to look into this further with more complex models and explore where these 2 methods diverge in terms of accuracy.

3.2 Image Segmentation

Q8 For this part of the assignment, we decided to use the Variational Bayes method to perform image segmentation. Choosing an image with clear foreground and background makes it easier to display our results. Also, we performed the segmentation to an RGB image to further explore what we can do with the method. For this purpose, we choosed figure 10.

The first step was to decide what was considered foreground and background. With Gimp, we annotated the image in order to mask the foreground and

Figure 12: *Image chosen for segmentation*

background. In order to perform image segmentation,

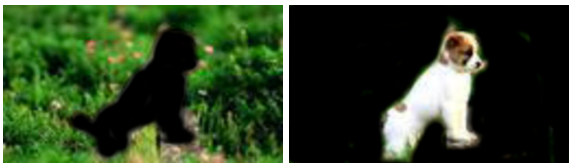

Figure 13: *Image background on the left and image foreground on the right*

we now have to encode new semantics into our latent variable. Now, 1 indicates a pixel that belongs to the foreground class whereas -1 indicates that the pixel belongs to the background. From the background and foreground images, we can create histograms of the red, blue and green channel of the image. The histogram acts as a counter table of the pixel values in each channel. Once we normalise the histogram to integrate to a value of 1, we now have the probability of each pixel colour value, given foreground or background. We now have $P(R|foreground)$, $P(G|foreground)$, $P(B|foreground)$ and vice versa. For instance, the first bin in the red foreground histogram now stores the value of $P(R = 1|foreground)$. Now, in order to calculate the likelihood that the pixel is a foreground, we take the observed pixel's colour channel values, and look up the histogram, sort of a lookup table of probabilities. Hence, $L(1) = P(Yr, Yg, Yb|foreground)$ and $L(-1) = P(Yr, Yg, Yb|background)$ where Yr, Yb and Yg are the respective channels of the image. With this information, we can now run the variationa Bayes algorithm as before, yielding the following segmentation :

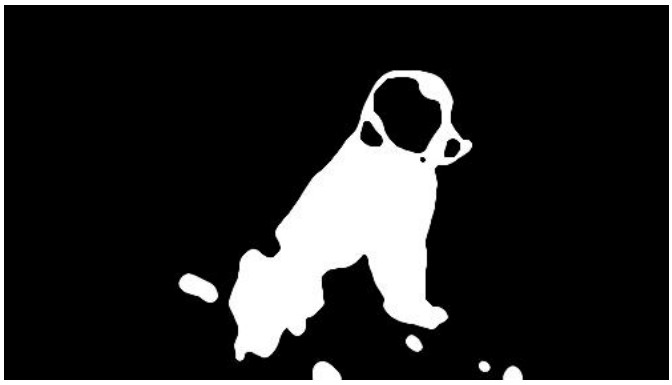

Figure 14: *Segmented image*

4 Sources Used

[1]Brian Keng (2017) - Variational Bayes and The Mean-Field Approximation http://bjlkeng.github.io/posts/variational-bayes-and-the-mean-field-approximation/
[2]Agustinus Kristiadi - KL Divergence: Forward vs Reverse? https://wiseodd.github.io/techblog/2016/12/21/forward-reverse-kl/
[3]Tuan Anh Le (2017) - Reverse vs Forward KL http://www.tuananhle.co.uk/notes/reverse-forward-kl.html