Inference

Samia Mohinta(97293) and Tushar Bhatnagar (97607)

07 December 2018

Approximate Inference

Question 1

We have implemented Iterative Conditional Modes for Ising Model to recover a noise-free binary image (latent x) from a noisy binary image (observed y). We have formulated our prior with the knowledge that unless the percentage of noise is too high, there exists a strong correlation between the observed y_i and the latent x_i as well as a spatial correlation exists between the neighbouring pixels (x_i, x_j) in an image. The energy functions are formulated in a way such that they render lower energy when there is a strong correlation between the variables[1]. In other words, low energy means that the probability of the pixel values to have the same label either 1 or -1 is high. The energy function defined is as below:

$$E(x, y) = -\beta \sum_{i,j} x_i, x_j - \eta \sum_i x_i y_i \qquad (1)$$

The number of iterations required to reach a local maxima of the probability distribution depends on the choice of β and η values. Large values of beta and eta will result in reaching the local optimum faster and thereby require lesser number of iterations. We also assume that the prior probabilities of x_i taking values 1 and -1 are equal. Keeping β =2.1 and η=1.5 values constant, we vary the noise percentage in the images and try to recover the original image by feeding the model with a noisy version of it.

De-noising Image corrupted with Gaussian Noise: Just 1 iteration using 4 neighbors for images corrupted with Gaussian noise (proportion of noise = 0.7 and sigma=0.9) effectively eliminates most of the noise and gives decent output(**Figure 1**). This happens because Gaussian noise is a smoother noise,which does not result in flipping the pixel values in the latent image. After 15 iterations (**Figure 2**), we are left with almost no noise in the image. Using 8 neighbors, helps us eliminate noise even faster (lesser number of iterations), but the recovered image lacks fine detail and is smoothed out (blurred). This is believed to happen because with 8 neighbours, the model now looks at a larger area of pixels that have the same value and clumps them together. In the code, we also maintain a flag which highlights the number of iterations required to reach a local maxima.

Figure 1: *ICM :Original(left),Noisy(center),Denoised(right). Gaussian Noise* $70\%\sigma = 0.9$*, Iterations=1.*

Figure 2: *ICM :De-noised Image containing Gaussian Noise 70%,*σ*=0.9.Iterations=15.Neighbours=8*

De-noising Image corrupted with Salt N Pepper Noise

Noise proportion = 0.7.

Images containing a high proportion of Salt n Pepper noise require more numbers of iterations to reach acceptable results. **Figure 4** shows the recovered image after 150 iterations using 4 neighbours. High noise percentage hinders the correlation between latent x values and observed y values. Even after 150 iterations, the local optima was not reached and took about 250 iterations to get the outline of the image foreground.

Noise proportion = 0.4
We could considerably recover the latent image after 5 iterations using 4 and 8 neighbours. **Figure 3(R)** shows the recovered image after 5 iterations using 4 neighbours.

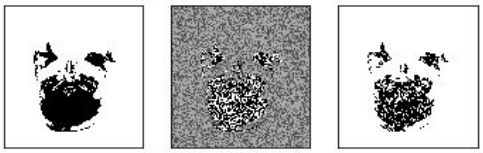

Figure 3: *ICM :Original(left),Noisy(center),Denoised(right). Noise Type-Salt n Pepper, Noise proportion= 40% Iterations=5 Neighbours=4*

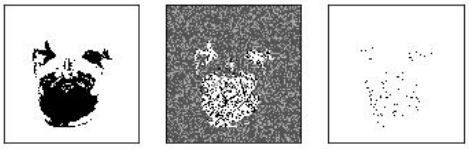

Figure 4: *ICM :Original(left),Noisy(center),Denoised(right). Noise Type-Salt n Pepper, Noise proportion= 70% Iterations=150 Neighbours=4*

Question 2

We have implemented Gibbs sampler for de-noising images containing different proportions of Salt n Pepper noise. The recovered de-noised images are given in **Figure 5** for 50% Salt n Pepper noise after 1 burn-in and 1 iteration. The image recovered is completely black because when we increase the proportion of noise, the Gibbs sampler struggles to recover the latent variables. An image corrupted with 30% of Salt n Pepper noise is substantially recovered after 15 burn-in and 5 iterations using 4 neighbours (**Figure 6**) and 8 neighbours(**Figure 7**).

Figure 5: *Gibbs:Original(Left),Noisy(Center),Denoised-Image(Right). 50% Salt n pepper noise, 4 neighbours, 1 burn-in and 1 iteration*

Figure 6: *Gibbs:Original(Left),Noisy(Center),Denoised Image(Right).30% Salt n pepper noise, 4 neighbours, 15 burn-in and 5 iterations*

Figure 7: *Gibbs:Original(Left),Noisy(Center), Denoised Image(Right). 30% Salt n pepper noise, 8 neighbours, 15 burn-in and 5 iterations*

Question 3

If we pick a random node for each iteration, it is observed that the local maxima of the distribution is reached faster than when we cycle through every node in a sequential manner. In other words, the recovery of the image requires lesser number of iterations(**Figure 8**) than is required with a sequential approach(**Figure 9**). A benefit with this approach is that no potential 'directional' effects occur. However, there is a chance that some pixels will be updated only a few times but does not affect the output if the sampler is run for long[6].

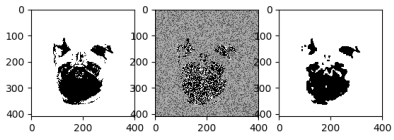

Figure 8: *Gibbs with Random Node :Original(Left), Noisy(Center), Randomly Denoised Image(Right).30% Salt n pepper noise, 4 neighbours, 50 burn-in and 10 iterations.*

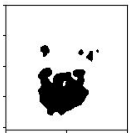

Figure 9: *Gibbs:Sequentially Denoised Image. 30% Salt n pepper noise, 4 neighbours, 80 burn-in and 20 iterations.*

Question 4

After a certain number of iterations,the algorithm converges to give a Markov Chain sample size adequate enough to meet the required accuracy i.e. it would reach a stationary distribution and the further iterations from this distribution will give us an approximation to the true posterior. In order to ensure that the sample is approximately independent from the target distribution, we need to ignore the first M samples, called burn-in. Thus, any noisy artifacts from the initial state is now avoided. Burn-in time is extremely useful in order to obtain a more accurate representation of the image thus ignoring initial conditions and correlation between samples. Therefore, increasing the number of iterations in burn-in helps the sampler to reach a better equilibrium w.r.t to the stationary distribution and hence restores the images more accurately **Figure 10**.

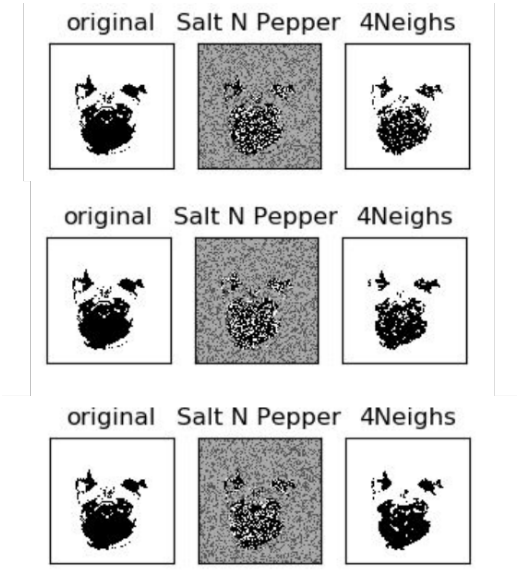

Figure 10: *Gibbs with Varying Iterations:Top to Bottom : Images recovered (Right) after Burn-in= 3,6,12 and iterations=5,10,20*

Question 5

KL divergence is a measure of the similarity between two probability distributions $p(x)$ and $q(x)$ and is calculated by calculating the distance between the two distributions. We know KL divergence is not a symmetric measure, that is, $KL(p\|q) \neq KL(q\|p)$, where $q(x)$ is our approximate distribution, which we are trying to fit to our true distribution $p(x)$. $KL(p\|q)$ is called Forward KL,while $KL(q\|p)$ is called Reverse KL.

In forward KL, the difference between the distributions ($p(x)$ and $q(x)$) is weighted by $p(x)$. That is, the KL divergence is totally dependent on the value of $p(x)$ and it does not matter what the value of $q(x)$ is. As a result, when $p(x) = 0$, then $q(x)$ is completely ignored. However, when $p(x) > 0$, the optimization algorithm forces $q(x)$ to spread its probability mass so that it covers all $p(x) > 0$, to lower the KL divergence (**Figure 11 [left image]**). Forward KL is zero avoiding, since it avoids $q(x) = 0$ whenever $p(x) > 0$.

In reverse KL, we switch the position of the distributions in the equation, while still keeping $q(x)$ as the approximate distribution and $p(x)$ as the true distribution. The equation now becomes :

$$D_{KL}(q\|\|p) = \sum q(x) ln \frac{q(x)}{p(x)} \tag{2}$$

Thus, now $q(x)$ becomes the weight. So when $q(x) = 0$, we can ignore $p(x) > 0$. Also, with $q(x) > 0$, it can be assumed that the KL divergence will be very low, because the $q(x)$ will be best approximated to fit $p(x)$. As a result, Reverse KL states that it is better to fit $q(x)$ to just some portion of $p(x)$ as long as the approximate is good. In other words, $q(x)$ does not spread its probability mass over all of $p(x) > 0$ (**Figure 11 [right image]**). Consequently, Reverse KL is zero forcing, that is, it forces $q(x) = 0$ in some portions of the probability distribution $p(x)$, even if $p(x) > 0$.

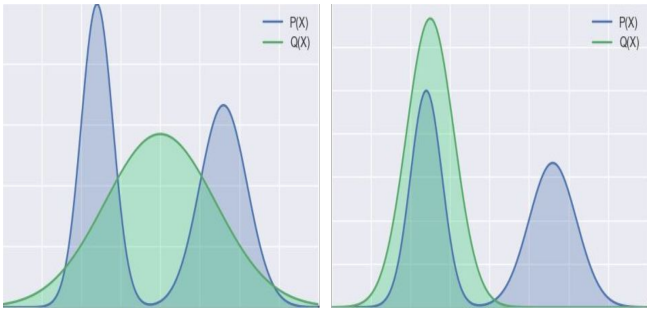

Figure 11: *Forward KL[Left] : $q(x)$ has distributed its mass over all of $p(x)$.Reverse KL[RIGHT]: $q(x)$ has confined its probability mass over a portion of $p(x)$ Reference :[4]*

Question 6

De-noised images obtained after implementing Mean Field Variational Bayes in **Figures 12,13,14,15,16**. An image with high proportion of noise cannot be recovered with low number of iterations (used 5) even with MFVB as seen in **Figure 16**.

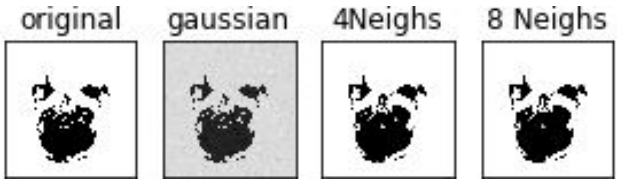

Figure 12: *MFVB :De-noising Image with Gaussian Noise (prop=0.3 σ=0.1) using 4 and 8 neighbours after 1 iteration.*

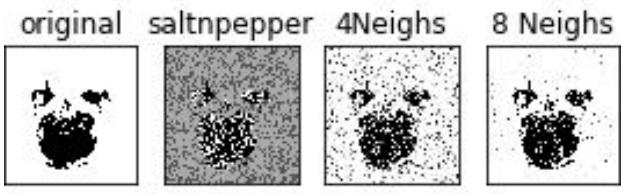

Figure 13: *MFVB: De-noising Image with SaltNPepper Noise (prop=0.3) using 4 and 8 neighbours after 1 iteration.*

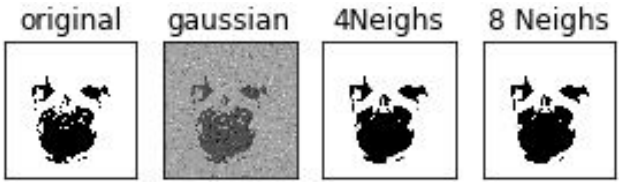

Figure 14: *MFVB: De-noising Image with Gaussian Noise (prop=0.3 σ=0.1) using 4 and 8 neighbours after 5 iterations.*

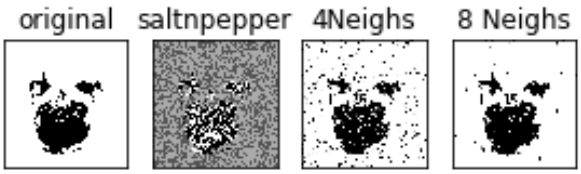

Figure 15: *MFVB: De-noising Image with SaltNPepper Noise (prop=0.4) using 4 and 8 neighbours after 5 iterations.*

Figure 16: *MFVB: De-noising Image with SaltNPepper Noise (prop=0.5) using 4 and 8 neighbours after 5 iterations*

Question 7

The recovered image from Mean Field Variational Bayes algorithm has minimal noise with an almost clear foreground and background outline after only 5 iterations. However, with ICM the images recovered contain speckle noise distributed on the foreground (face), thereby displaying breaks in between parts having the same colour. These differences can be visualized from **Figures 3** and **Figures 15**. Furthermore, when compared with Gibbs sampling,it is observed that MFVB is faster and renders a less noisy image in less number of iterations. **Figure 6** shows the recovered image after 20 iterations (15 burn-in) using 4 neighbours, while **Figure 13** almost recovers the structural definitions of the latent image after only 1 iteration using 4 neighbours. Also, the images recovered with Gibbs sampling have smoother shape edges than the images restored using MFVB. This is because MFVB places less uncertainty at the shape boundaries. However, variational inference is irrecoverably biased and will give an approximate of the true distribution, whereas Gibb's (MCMC) bias approaches 0 as the Markov chain is ran for longer durations[7]. So, with unlimited computational resources, Gibbs Sampling will give us better results.

Question 8

We have used Gibbs Sampling to segment an image into its foreground and background. We used an RGB image and partitioned it into a foreground image and a background image depending on the colour distributions. The resultant images after segmentation are as in **Figure 17**.

A chosen threshold value helped us determine our masks for foreground and background. The masks were used to produce RGB histograms for the colour space and later we normalized these histograms. The likelihood of a specific pixel value was determined by its joint probability with respect to the normalized histograms. The posterior probability (true pixel value) was calculated by traversing through the image and comparing it against the masks.

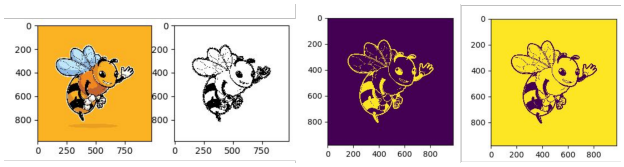

Figure 17: *Image Segmentation with Gibbs Sampling. L-R :Original Image, Binary Image generated with Gibbs Sampling, Foreground, BackGround (both highlighted in yellow).*

Variational Auto-Encoder

Question 9

Variational Auto-Encoders are generative models, which means they can learn a probability distribution p(x) over some input x and after training such models can create data that will resemble the input. As for example,the VAE learns a distribution over our MNIST data-set, which is a data-set of handwritten 0-9 digit images and after training is able to create images which look like handwritten digits, even though they're not "copies" of the images in the training set. The first layer of the VAE is the encoder, which takes the input and convert its into a latent vector. This is usually

done by reducing the mean squared error of the input and output, like a standard auto-encoder. However, to make the auto-encoder variational, the encoding is forced to roughly follow a Gaussian distribution.This feature of VAE allows to generate an output similar to the database the VAE was trained on by inputting a latent vector[2] straight to the decoder. For computing image reconstruction loss, squared difference is used. This loss is combined with KL divergence which makes sure that the latent values are sampled from the Gaussian distribution. Now the decoder shall be able to generate new characters.Sampled values from a Gaussian distribution are fed to the decoder and the decoder then uses these latent variables to generate an image similar to the input data (in this case digits between 0-9).

Standard variational Bayes approach involves the optimization of an approximation to the intractable posterior. However, MFVB approach requires analytical solutions of expectations with respect to the approximate posterior, which are also intractable usually[3]. VAE uses reparameterization of the variational lower bound to yield a simple differentiable unbiased estimator of the lower bound called Stochastic Gradient Variational Bayes. This estimator is able to efficiently approximate posterior inference in almost any model with continuous latent variables and parameters, makes use of the standard stochastic gradient ascent techniques for optimisation [3]. This reparameterization facilitates back-propagation through a random node in a VAE.

Question 10

We integrated the VAE to run over the MNIST data-set comprised of noisy images of 0-9 hand-written digits. The encoder(recognition model)was successful in learning a distribution over the data-set.Thus, when random samples drawn from this distribution were passed to the decoder (generative model), it was able to create images that looked similar to the input (0-9 digit) data-set (**Figure 18**).

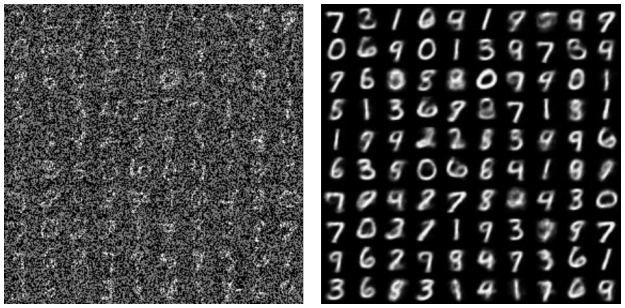

Figure 18: *Implemented VAE:Input Noisy MNIST Data[5]. Generated Output from VAE after Training (L-R)*

We also observed that the total loss and likelihood dips at a certain point and then increases while running for 10 epochs. The plot of KL divergence alternately decreases and increases across the 10 epochs. A graph,

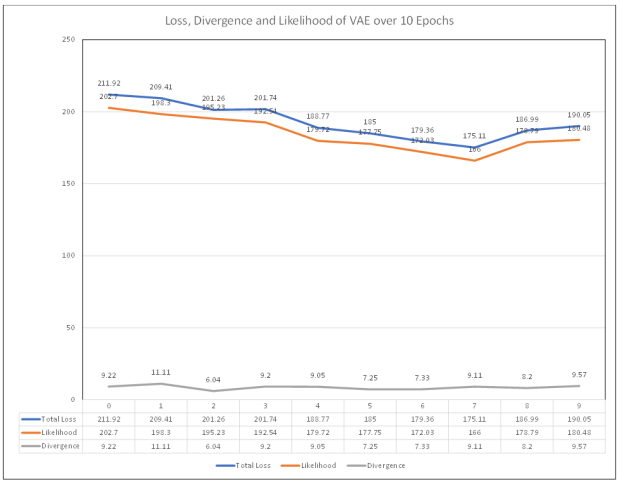

Figure 19: *Plot showing Total Loss, Likelihood and KL Divergence for each epoch (0-9).*

Figure 19, has been plotted to represent the total loss, likelihood and KL divergence for each epoch.

References

[1] Christopher Bishop. *Pattern Recognition and Machine Learning* (2006)

[2] Carl Doersch. *Tutorial on Variational Autoencoder*(2016)

[3] D.P. Kingma and M. Welling. *Auto-encoding variational Bayes. In Proceedings of the International Conference on Learning Representation,*(2014)

[4] Augustus Kristiadi. *KL Divergence: Forward vs Reverse?*(2016)

[5] Yann LeCun and Corinna Cortes. *MNIST handwritten digit database.*(2016)

[6] Jesper Moller. *Spatial Statistics and Computational Methods.*(2003)

[7] David Blei, Alp Kucukelbir, Jon D. McAuliffe. *Variational Inference: A Review for Statisticians.*(2018)