Inference

Adam Sadiq - 34795, Ashwinder Khurana - 36208

Machine Learning

(a) Figure 1: Original Image

(b) Figure 2: Gaussian Noise

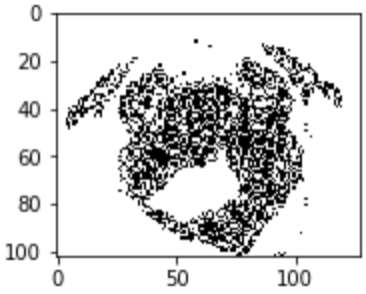
(c) Figure 3: Salt and Pepper Noise

Question 1

Implement ICM for a binary image that you have corrupted with noise, show the result for a set of images with different noise levels. How many passes through the nodes do you need until you are starting to get descent results?

Figure 2 and *Figure 3* are corrupted versions of *Figure 1* (Our original image), just with different methods of statistical corruption. Using ICM we obtain the image in *Figure 4* fairly quickly as we begin to get decent results after 1 iteration of the algorithm. The reasoning behind obtaining quick results is fairly intuitive as our Gaussian Noise image in *Figure 2* is fairly correlated with the original image despite the background. This meant that the ICM produces fairly low energy costs using the equation below (and therefore high probability) for the pixels that represent the face of the pug, and high energy costs for the noisy background pixels. However, the reconstructed image in *Figure 4* is still imperfect as there is still a huge grainy effect within the image, where the original image has a large group of solid black pixels, our reconstructed image has varying pixel values within the main body of the image.

$$E(x, y) = -\eta \sum x_i y_i + \beta \sum x_i y_i \tag{1}$$



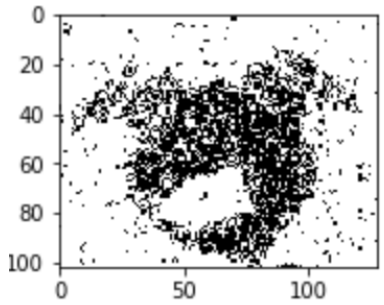
(a) Figure 4: Iterative Conditional Modes output

Question 2

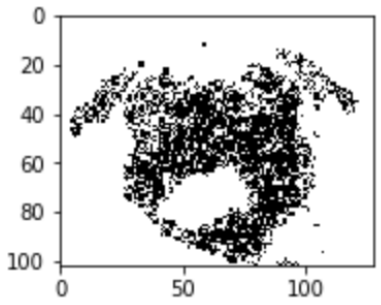
Implement the Gibbs sampler for the image denoising task. Generate images with different amount of noise and see where it fails and where it works. My result is shown in Figure 2.

The Gibbs sampler draws samples from a normal distribution, this sampled probability is then compared to a probability x that if a particular pixel ,when compared to its neighbours, is white or not. If the sample probability from the Gibbs Sampler is bigger than x, then the selected pixel is white. If the opposite is true, then the selected pixel is white. The overall effect of the Gibbs sampling method is that there are larger clusters of black or white pixel groups, when compared to the ICM denoising algorithm. This is down to the fact that Gibbs usses a distrbution to decide when to sample, whilst the ICM is a much more basic implmentation of incorporating neighbouring pixel values.

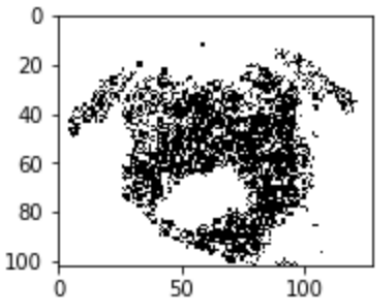
In comparison to ICM, the Gibbs sampling method works much better by clustering more black pixels together to produce a solid foreground, however, the area in and around the mouth of the pug is where the sampling fails, because there is a less defined structure for the mouth area, which is not ideal.



(b) Figure 5: Gibbs Sampling varSigma = 0.7



(c) Figure 6: Gibbs Sampling varSigma = 0.4



(d) Figure 7: Gibbs Sampling varSigma = 0.2

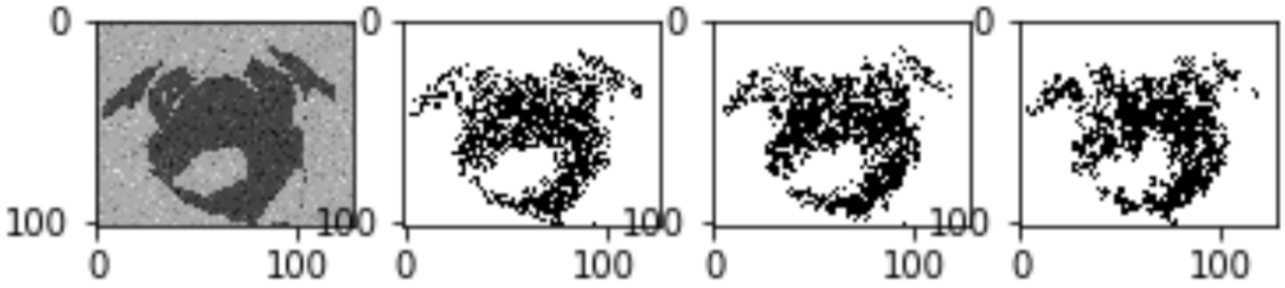
Question 3

There is nothing saying that you should cycle through the nodes in the graph index by index, you can pick any different order and you do not have to visit each node equally many times either. Alter your sampler so that it picks and updates a random node each iteration. Are the results different? Do you need more or less iterations? To get reproduceable results fix the random seed in your code with np.random.seed(42).

In *Figure 6* below, we can see an implementation of visiting nodes an unequal amount of times using a random probability. For 1 iteration, we can see that our results are not dissimilar to the initial Gibbs sampling method. As we further iterate and visit more nodes with varying probability, we can see the boundaries and edges loose accuracy. For example, the mouth boundary for the image loses its shape as the number of iterations increase. In addition, we begin to loose some detail around the eyes of the pug which is not ideal. Overall, it means that we need less iterations for to produce a more accurate representation of our original image.

Question 4

What effect does the number of iterations we run the sampler have on the results? Try to run it for different times, does the result always get better?



(e) Figure 6: Altered Gibbs Sampling with 1,5,10 iterations

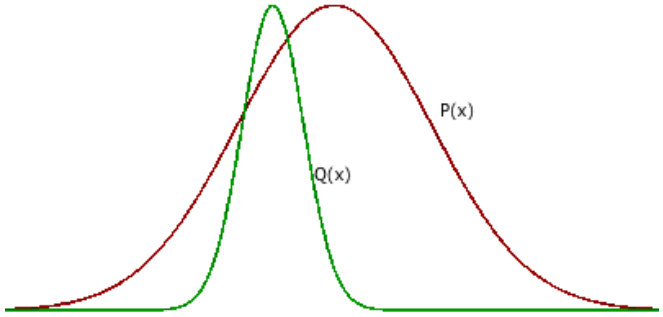
After producing the images in , it is evident that the more iterations we have on the image, the worse the image becomes. This is largely to do with the amount of accuracy and detail around the image. Each iteration very much adheres to the same general shape, however after honing in on the image we found that the detail is lost. Observing our image we see that the black pixels become more clustered together, almost leaving a more patchy outlook. This is most likely due to our likelihood of pixels tending to have the same value as neighbouring pixels. Thus with more iterations, neighbouring pixel values seems to have more weighting to decide the current pixels value.

Question 5

What will be the difference between $KL(q(x)||p(x))$ and $KL(p(x)||q(x))$? Think of where $q(x)$ would place probability mass in the different scenarios.

The KL divergence is a measure of how similar two probability distributions are to one another. If the two distributions are the exact same, the KL divergence measure will be 0. This makes our objective clear,is to then minimise the KL divergence measure in order to produce an accurate approximation to the intractable integral.

To highlight the non symmetric property of the KL divergence measure, lets assume q(x) is a narrow distribution, for example a very a peaked Gaussian, and p(x) is a very widespread distribution (visualisation shown below in *Figure 9*). In our first scenario, when integrating over x's from - ∞ to + ∞, the majority of our q(x) values will be close 0 and we would be dividing by relatively high p(x) values, therefore overall we get a fairly low divergence measure. Conversely, in scenario 2, when trying to fit p(x) onto q(x), we will be dividing by infinitesimally small values majority of the time, which will result in an extremely large divergence measure. This loose reasoning supports the idea that the KL divergence is not a valid distance measure as it does not satisfy the symmetry property.



(f) Figure 9: Example distributions to visualise both scenarios

Question 6

Implement the algorithm in Algorithm 4 and use it to denoise images as before.

(g) Figure 7: Original Image

(h) Figure 8:Variational Bayes output

Question 7

How does the result of the Variational Bayes method compare to the two previous approaches? Try to explain the results that you get by relating it to the different methods.

Both functions try to find clever ways to work around evaluating the evidence, as the integral is intractable. ICM and Gibbs look at each pixel and uses a probability to see how likely a given value should belong to that pixel. Variational Bayes however uses a probability distribution to calculate the expectation of neighbouring pixels, and gives them different weights to ensure a more consistent output, this is much more efficient and thus, the result of the Variational Bayes is much more accurate compared to the original methods. The comparison between Variational Bayes and the other method is immediate. We can see that *Figure 7* has a much more solid foreground for the face of the pug with less white noise/less grainy than the other methods. Also, in our resulting image we can see that minor details are preserved throughout the image, though being very simple, the eye reflection, the outlines of the mouth, ears and face are almost identical, and also more importantly, the algorithm is very certain on making our image fully black in areas it should be.

Question 8

Implement the image segmentation with one of the inference approaches we have gone through, either the sampling based method or the deterministic approach.

(i) Figure 9: Original Image

(j) Figure 10: Image segmentation output

To produce our segmented image, we followed these steps:

- Colored foreground and background values in green and red respectively
- Gathered original RGB values of image within coloured sections.
- Created histograms to store the individual, R,G&B values of every pixel in image
- Iterated through image and used for every pixel, determined the probability of each RGB value belonging to either the foreground or the background
- If the pixel has a higher probability of belonging to the background, it was set as white, and if not, was set as the corresponding original pixel value