Machine Learning Inference

37741 Conor Cullen
37748 Mike Talbot

7 December 2018

Question 1

To begin with, we take a binary image of a pug that has pixel values $y_i \in (0,1)$ and corrupt it with noise:

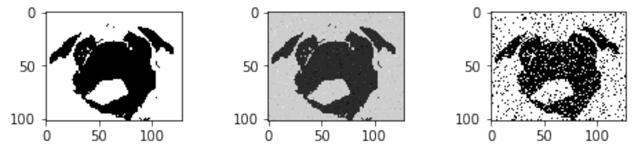

Figure 1: *From left to right: Original binary image, Image with Gaussian noise added, Image with salt and pepper noise added*

Once the pictures have been made a mess of we attempt to return the image back to the beautiful form it was originally. The first model used to attempt to fix these images is the Iterative Conditional Model (ICM). The aim of this model is to compare each pixel with each of its neighbours, either the 4 or 8 cardinal points. Then to reduce the total energy of the picture until it reaches an equilibrium. The energy of each pixel is equivalent to the sum of the pixel's value times each of its neighbours with an arbitrary weight (1).

$$E_i = -\Sigma_{j \in \mathcal{N}} w_{ij} x_i x_j \tag{1}$$

The quantity of each pixel is in {1,-1}, which means that if the pixel is the same as its neighbour the product of both would be positive, therefore increasing the sum. If we take the negative value of the sum then the more similar pixels are the lower the total energy of the picture.

To reduce the energy of the overall image, we flip each pixel one by one to see whether this change reduces the overall energy. If so the pixel is replaced! This process is ran over a number of iterations, with each iteration starting with the new picture that was outputted at the end of the previous iteration. The algorithm stops when either the energy is not changed between iterations or when the iteration limit is reached.

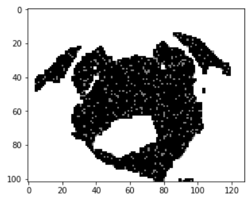

Figure 2: *The pug fixed using the ICM Ising Model. This picture took 6 iterations before reaching equilibrium. The noisy image is the middle image from Fig. 1*

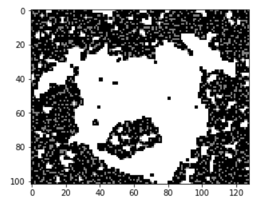

Figure 3: *The pug fixed using the ICM Ising Model. This picture took 20 iterations before reaching equilibrium. The noisy image is the right image from Fig. 1*

Working on the gaussian noise image with a proportion of 0.75 and a sigma of 0.4, the pug is practically restored, however with some sharper edges and losing some of the finer detail, as the ICM prefers larger islands of the same colour. However, working on the salt and pepper image, the method fails to remove the noise, although a white foreground is produced.

Question 2

We now implement the Gibbs sampling approach into the Ising model. First we look at the result of denoising the image with Gaussian noise added:

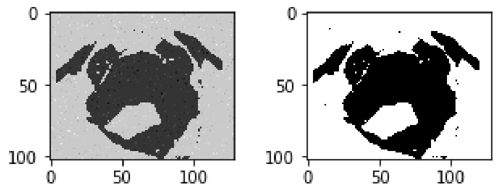

Figure 4: *The result of running the Gibbs sampling approach for 20 iterations. The left image is the image with Gaussian noise and the right image is the cleaned up version.*

The Gibbs sampler removes most of the noise in the image in Fig. 4, with only a few pixels that are not corrected, which can be seen in isolation in the background of the image. There isn't much Gaussian noise added here but with a higher variance the recovered image is similar.

Next, we look at the result when the image has more noise added to it. Namely, we try to recover the image from having salt and pepper noise added to it:

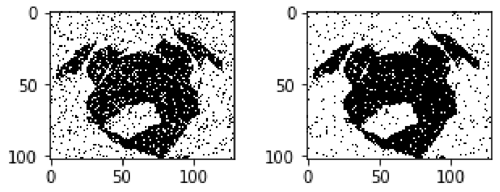

Figure 5: *The result of running the Gibbs sampling approach for 20 iterations. The left image has 10% pixels flipped and the right image is the cleaned up version.*

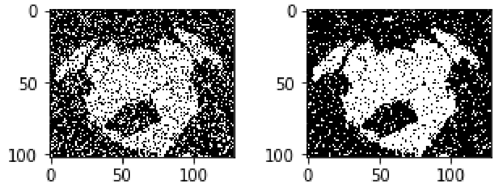

Figure 6: *The result of running the Gibbs sampling approach for 20 iterations. The left image has 80% pixels flipped and the right image is the cleaned up version.*

From Fig. 5, we can see that when 10% of the pixels have been flipped, there is some background noise removed but not enough to reach an acceptable level. However, the noise in the foreground is removed to a good extent. With 80% of pixels flipped by the noise, the Gibbs sampler fails to remove much noise at all.

Question 3

The Gibbs sampler is now altered so that it picks and updates a random node (pixel) each iteration. The result shown in Fig. 4 now looks like:

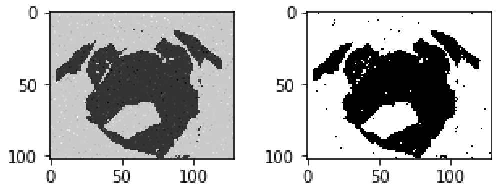

Figure 7: *The result of running the Gibbs sampling approach for 20 iterations with a random pixel chosen in each loop.*

This gives a similar result to that in Fig. 4, except there are more pixels that have not been corrected. The only difference made when choosing random pixels is that you need to iterate through the pixels more times to get the optimal version of the image as shown in Fig. 7.

Question 4

Running through the sampler for more iterations has a negative effect on the results of the denoising. Taking the image with Gaussian noise and running through for 100 iterations yields the image shown below:

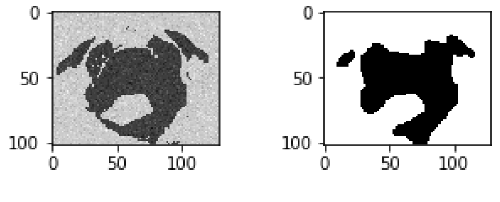

Figure 8: *Gibbs sampler ran on the Gaussian noise image for 100 iterations.*

The image quality has been lowered as a result of the image becoming 'too smooth'. A similar result to Fig. 6 is shown with the salt and pepper noise added. It would seem that the optimal result for denoising the image comes from using the lowest possible number of iterations. This is because of the geometric rate of convergence of the Markov Chain to its stationary distribution. Sampling after the chain has converged results in the representation being closer to the true posterior distribution.

Question 5

There are three scenarios of importance with $KL(q(\mathbf{x})||p(\mathbf{x}))$:

- If $q(\mathbf{x})$ is high and $p(\mathbf{x})$ is high then we are happy as we have a low KL-divergence. This is because there is little evidence that $q(\mathbf{x})$ is not a good representation of $p(\mathbf{x})$.
- If $q(\mathbf{x})$ is high and $p(\mathbf{x})$ is low then we have a high KL-divergence, which we are not happy with. This is because there is a lot of evidence that $q(\mathbf{x})$ is not a good representation of $p(\mathbf{x})$.
- If $q(\mathbf{x})$ is low then we don't care because there is a low divergence regardless of what $p(\mathbf{x})$ is.

$KL(p(\mathbf{x})||q(\mathbf{x}))$ (Fig. 9) gives us a measure of the information gained when we revise our beliefs from the approximating distribution $q(\mathbf{x})$ to the true distribution $p(\mathbf{x})$. More specifically, this is a measure of the information lost when using $q(\mathbf{x})$ to approximate $p(\mathbf{x})$. The equation in this case is:

$$KL(p(\mathbf{x})||q(\mathbf{x})) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \tag{2}$$

Now if $q(\mathbf{x})$ is very close to zero and $p(\mathbf{x}) \neq 0$, we have that the divergence is ∞. However, for $KL(q(\mathbf{x})||p(\mathbf{x}))$ (Fig. 10), the case is swapped so that if $p(\mathbf{x})$ is very close to zero and $q(\mathbf{x}) \neq 0$, we have that the divergence is ∞. In other words, the difference between $KL(p(\mathbf{x})||q(\mathbf{x}))$ and $KL(q(\mathbf{x})||p(\mathbf{x}))$ is that the former wants to avoid $q(\mathbf{x})$=0 whenever $p(\mathbf{x})$>0, whereas the latter forces $q(\mathbf{x})$ to be 0 on some areas, even if $p(\mathbf{x})$>0. A good way to picture this is with the two figures below:

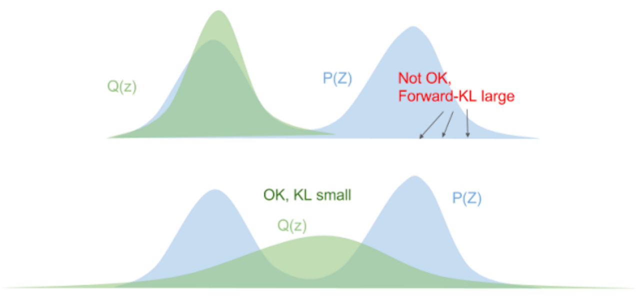

Figure 9: *Forward KL divergence, https://blog.evjang.com/2016/08/variational-bayes.html*

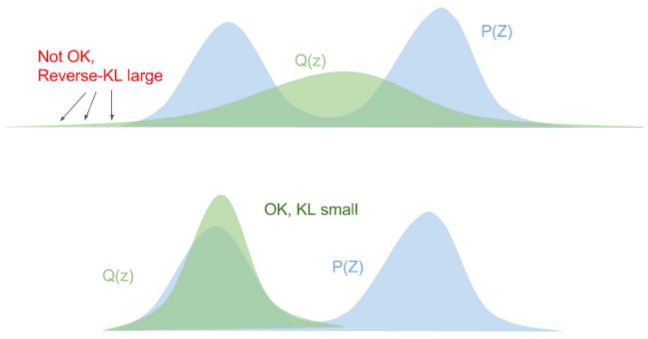

Figure 10: *Reverse KL divergence, https://blog.evjang.com/2016/08/variational-bayes.html*

Question 6

Implementing the variational inference method yields this result for the image denoising task with Gaussian noise:

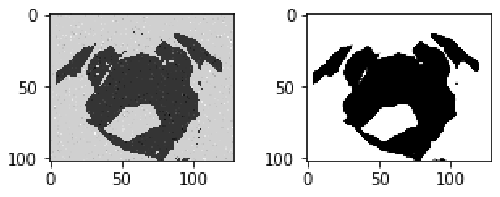

Figure 11: *The result of the Variational Bayes method. The left image is the noisy image and the right image is the cleaned up version.*

The result shows that the Ising Prior does a good job of cleaning up the image, retaining the quality of the facial features within only one iteration. The result for the salt and pepper image is shown below:

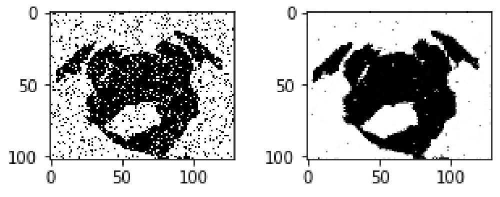

Figure 12: *The result of the Variational Bayes method where 10% of the pixels of the image are flipped. The left image is the noisy image and the right image is the cleaned up version.*

The result of running the variational inference for an image corrupted by a lot of noise is shown below. The reason for the discolouration of the recovered image is due to 80% of pixels being flipped.

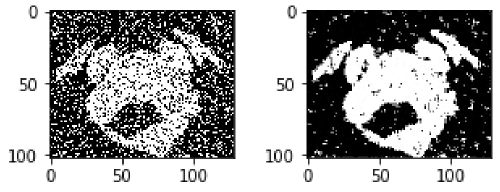

Figure 13: *The result of the Variational Bayes method where 80% of the pixels of the image are flipped. The left image is the noisy image and the right image is the cleaned up version.*

By using the stated equation of

$$m = \Sigma_{j \in \mathcal{N}(i)} \mu_j i, \tag{3}$$

$$\mu_i = \tanh\left(m + \frac{1}{2}\mathcal{L}_i\right), \tag{4}$$

this assumes that the learning rate of the Variational Bayes is set to 1. Doing this means that the algorithm is more likely to make wild guesses at what the pixel values should be, which means that the finished product is complete in fewer iterations, however it is more likely to get some values incorrect due to the definite guesses it makes. Decreasing the learning rate increases the effect the previous value of the pixel influences what the outcome should be, therefore a learning rate of 1 disregards the current state entirely and focuses on the hyperbolic tan function.

Question 7

The variational inference method gives us better results than the other two methods. Firstly, the resulting image in Fig. 11 has no background pixels, unlike images such as Fig. 2 and Fig. 4. The foreground is also completely black, which is a better result than that from Fig. 2. The downside of variational inference is that you can't converge to a global maximum like MCMC, but it is a lot easier to compute.

Variational Bayes also creates a much sharper image (i.e. better defined edges) than that from the Gibbs sampler. This is to do with the mean field approximation; at the edge boundaries the sampler tends to overestimate the pixel values, which results in the overly smoothed image shown in Fig. 8. On the other hand, variational inference is a strict approximation so this issue does not arise. ICM is a very greedy algorithm and gets stuck in local optima, which is why an image such as Fig. 2 is produced. Gibbs sampling is less greedy but is a lot slower due to sampling from any state as opposed to picking the best one.

In conclusion, variational inference is generally the fastest method compared to ICM and Gibbs sampling. However, variational inference is biased, whereas the bias of MCMC approaches 0 as the Markov Chain converges. The main difference is that variational inference is deterministic whereas MCMC is stochastic. This determinism is more useful as you can use back propagation to tune the model parameters to get the correct estimate of the image training data.

Question 8

Image segmentation is the aim of separating an image into different groups. For example, to remove the object in the foreground from the background or remove certain colours altogether. The way in which this is achieved is by making a mask for the picture. This mask denotes what you would like to keep in that group (Fig. 14). Once this is done the colours in the picture are separated into smaller clusters using a K-means algorithm. By separating the colours into smaller groups ($\sim$20) rather than the $256 \times 256 \times 256$ different possible groups from RGB, we can group similar coloured areas to be in the same cluster (Fig. 15).

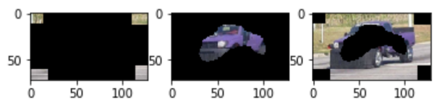

Figure 14: *A figure showing the two masks used, in the order of the foreground, background and the remaining pixels. These masks were picked arbitrarily.*

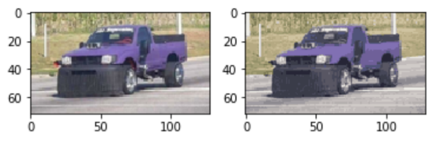

Figure 15: *The before and after of the K-means clustering using 20 colours*

To see what values this mask is picking out, a histogram of the RGB values in the two masks is made. Here we can see that within the foreground mask area the values tend to be a lot darker with the majority of the values being below 0.5. The only colour that surpasses the 0.5 boundary is blue. Comparing that to the histogram on the left shows that many more colours are lighter with a clear positive trend and with a modal intensity of around 0.75 for all colours (Fig. 16). An assumption before applying the image segmentation is that the first mode at the darker ranges of colour is due

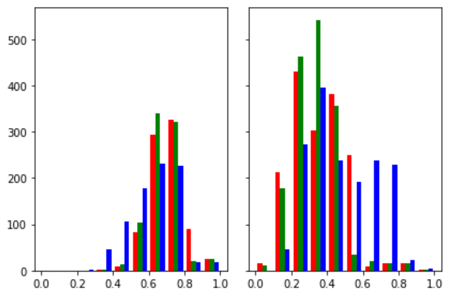

Figure 16: *Histograms depicting the colour intensity of the pixels within the two masks. Left: Foreground Mask. Right: Background Mask*

to the colours of the foreground mask and the second mode in higher intensities is attributed the background.

Once applying the masks, we can apply the clusters again and compute a histogram to see which colours are taken by each mask. Looking at Fig. 17 we can see the split of colours used by each mask. This will help with the segmentation of the image as we can use a direct comparison with the colours of each pixel to say which group they ought to be in.

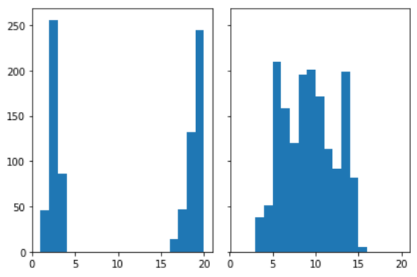

Figure 17: *How the 20 colours separated in K-Means were distributed between the masks. Left: the foreground mask. Right: the background mask.*

In order to sort which pixels are owned by each mask, we will use one of the denoising methods we used above. In this example three values would be used, 1 to represent the front mask, -1 to represent the back and 0 for any pixels that are unaligned. Using the energy equation for each pixel (1), we can say that $x_i \in \{-1, 0, 1\}$. The weights is where the clusters come into the equation. If two neighbouring pixels belong to the same group then we want a high chance of them being in the same mask. A metric of comparing the pixels is by using their colours. The difference between their red, green, blue values is taken, and then the Pythagorean distance between them, is measured.

$$\text{Distance} = \sqrt{\Delta R^2 + \Delta G^2 + \Delta B^2} \tag{5}$$

So that w_{ij} is large for a small difference in colour this is the equation that will be used:

$$w_{ij} = \frac{1}{1 + \mathcal{D}}; \tag{6}$$

where $\mathcal{D}$ is equal to the difference of the two pixels' colour values. The one in the denominator is so that a division by zero does not occur.

As there are now three options the pixels can take, the simple flip method of ICM cannot be used, however a variation can be applied. The options the pixels can take now are as follows. If a pixel is a 1 or a -1 then they can either swap or stay as they are. A pixel selected by a mask will not change back into an unassigned pixel (0). Where as a 0 has the chance to stay as a 0 or swap to a -1 or a 1. This is done by an extension to the if statement that allocates the new values, having specific options for the unassigned pixels. Running this algorithm is slow, and converges after 32 iterations. A convergence is achieved when changing the pixel assignment does not decrease the total energy of the picture (in the code the negative of the energy is calculated and used so the signs are swapped).

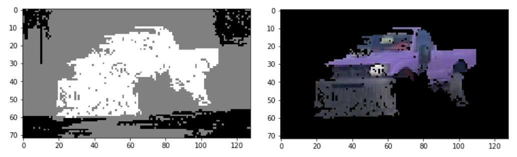

Figure 18: *Running the altered ICM with 8 neighbours produces this after 32 iterations. Left: depicting the mask assignments with {-1, 0, 1} as {Black, 50% Grey, White} respectively. Right: applying the foreground mask to the image*

Comparing the clusters once again we can see that using this weighting has increased the number of similar pixels in each mask, while also having practically zero of the alternate mask colours.

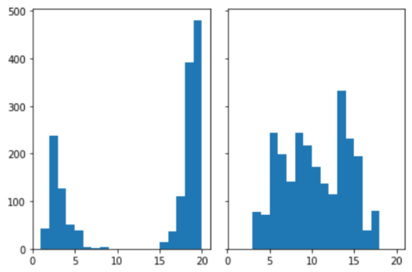

Figure 19: *After producing the new masks, these histograms depict the division of the 20 clusters. Again the background clusters are on the left and foreground on the right.*