CS688: Graphical Models - Spring 2016

Assignment 3

Shamya Karumbaiah

Collaborated with Rafael Lizarralde

Note - For most of the code(Question*.m) corresponding to different sections in this report, there is a provision at the beginning of the code to uncomment the setting you need. Code runs with defaults otherwise.

1. Derivation for Binary Image Model: The conditional distribution over de-noised pixel values given noisy pixel values is -

$$P(\mathbf{Y}=\mathbf{y}|\mathbf{x}) = \frac{\exp\left(\sum_{(i,j),(k,l)\in E} W^P_{ijkl}[y_{ij}=y_{kl}] + \sum_{(i,j)\in V} W^L[y_{ij}=x_{ij}]\right)}{\sum_{\mathbf{y}\in\mathcal{Y}} \exp\left(\sum_{(i,j),(k,l)\in E} W^P_{ijkl}[y'_{ij}=y'_{kl}] + \sum_{(i,j)\in V} W^L[y'_{ij}=x_{ij}]\right)} \tag{1}$$

Using the gibb's sampler, we follow sequential updates making this algorithm non parallel. Thus, the above equation must be rewritten with updates to single y_{ij}.Also, in the given CRF model, a de-noised pixel value Y_{ij} depends only on its markov blanket which in this case is Y_{ij}'s pixel neighbors (A_{ij}) and the corresponding pixel value in the noisy image (x_{ij}). Therefore the above equation can be simplified as -

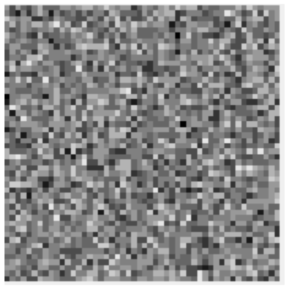
$$P(Y_{ij}|\mathbf{y}_{\mathbf{A}_{ij}}, x_{ij}) = \frac{\exp\left(\sum_{(k,l)\in A_{ij}} W^P_{ijkl}[y_{ij}=y_{kl}] + W^L[y_{ij}=x_{ij}]\right)}{\sum_{y\in\mathcal{Y}_{ij}} \exp\left(\sum_{(k,l)\in A_{ij}} W^P_{ijkl}[y'_{ij}=y'_{kl}] + W^L[y'_{ij}=x_{ij}]\right)} \tag{2}$$

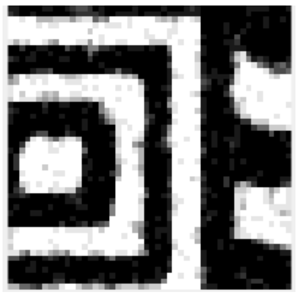
where $\mathbf{y}_{\mathbf{A}_{ij}}$ denotes the values of the neighbors.

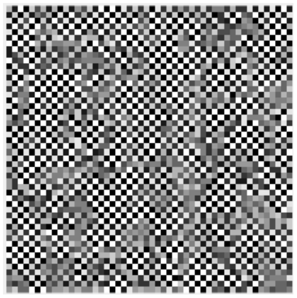
Since, this is a binary image model and the pixels can only take values $\{0,1\}$, this can be further simplified to get -

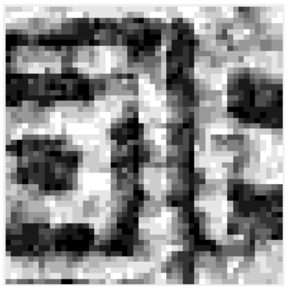
$$P(Y_{ij}=1|\mathbf{y}_{\mathbf{A}_{ij}}, x_{ij}) = \frac{\exp\left(\sum_{(k,l)\in A_{ij}} W^P[y_{kl}=1] + W^L[x_{ij}=1]\right)}{\sum_{y=0}^{1} \exp\left(\sum_{(k,l)\in A_{ij}} W^P[y_{kl}=y] + W^L[x_{ij}=y]\right)} \tag{3}$$

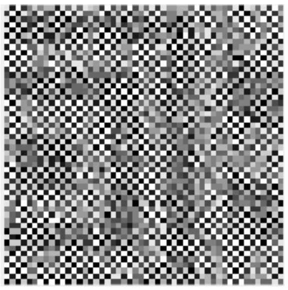
2. Monte Carlo De-noising for Binary Images:
(a) Qualitative analysis: Below are the output images after 10 sweeps-


(a) WP = 0, WL = 0


(b) WP = 1, WL = 1

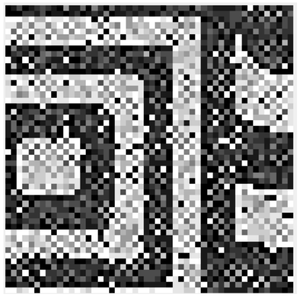

(c) WP = -1, WL = 1


(d) WP = 1, WL = -1


(e) WP = -1, WL = -1

In all the below examples, the absolute values of the weights matter. For convenience, I have taken unit value to observe the variation in terms of the sign of the weight.

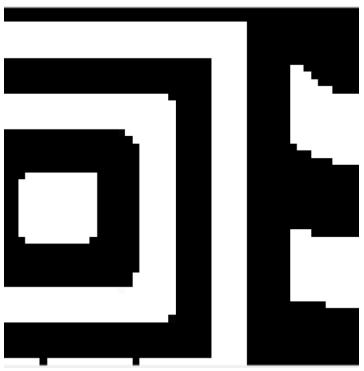
- When the parameters are zero, the model is not learning from the neighbours or from the noisy image data. All exp terms in the conditional probability become 0 and always returns $\frac{1}{2}$, thus doing the pixel assignment based on the α, the random number generated. This can be seen as the complete noise in Figure a.
- When both are positive(Figure b), the model learns positively correlate from both neighbors and the noisy image data. Since, we initialized to start with the noisy data, we can see an output thats closer to the true image just with 10 iterations.
- When one of them is negative, the model negatively correlates with it either in smoothness or in being close to the noisy image. When $W^P = -1$ and $W^L = 1$, the model prefers to negatively correlate with smoothness and hence tries to contrast between the neighbors (Figure c) . Since most pixels have 4 neighbors, $W^P = -1$ and $W^L = 1$ is giving more weights to the neighbor. To visualize this better, below is the output when $W^P = -1$ and $W^L = 4$. Now, the model learns almost equally from the noisy image data. But in the mean time, tries to maximize the contrast between neighbors especially around noise.



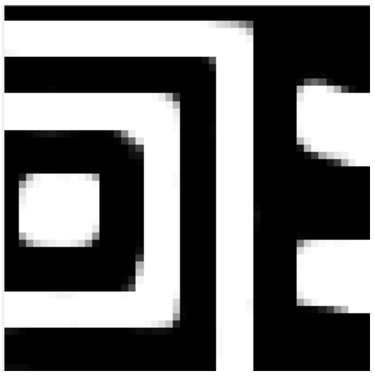
When $W^P = 1$ and $W^L = -1$, the model negatively correlates being close to the noisy image and hence inverts the pixel values. This can be observed with the flip in the light and dark regions in the image (Figure d).

- When both are negative(Figure e), a combination of effects from last 2 conditions is observed.

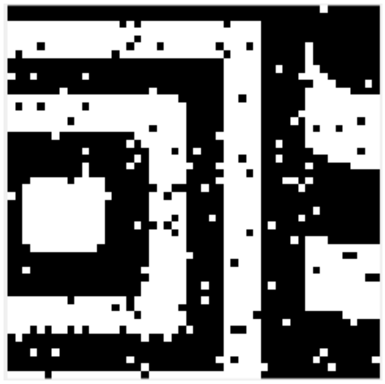
(b) Posterior marginal decoding: The best mean absolute error for the posterior mean image $\bar{y}$ using the $T = 50$ samples from the Gibbs sampler is 0.0024 (0.24 %) which is significant improvement as compared to the baseline of 0.0484(4.84 %). This was obtained for $W^P = 11$ and $W^L = 14$. The best posterior mean image $\bar{y}$ along with posterior mean images with over ($W^P = 130$ and $W^L = 1$) and under smoothing ($W^P = 1$ and $W^L = 120$) is shown in Figure f, g, h.



(f) best WP = 11, WL = 14

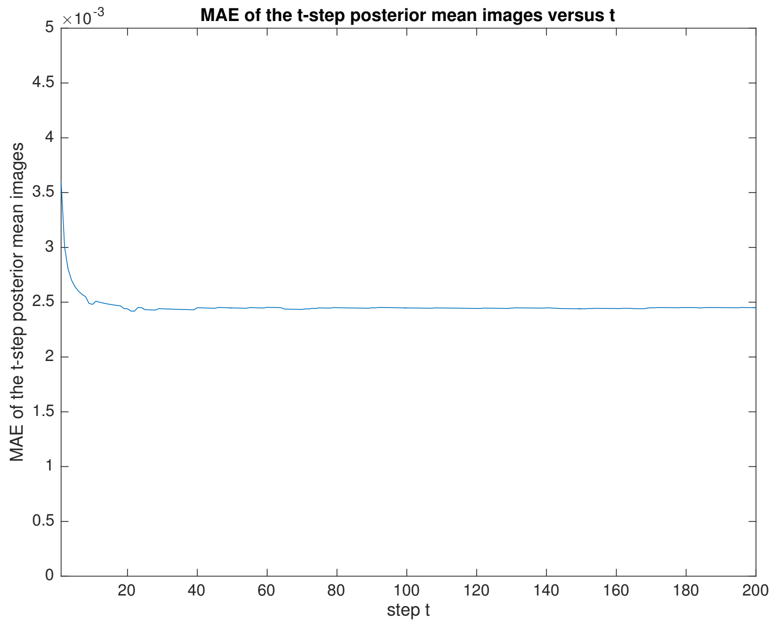


(g) over smoothing WP = 130 WL = 1

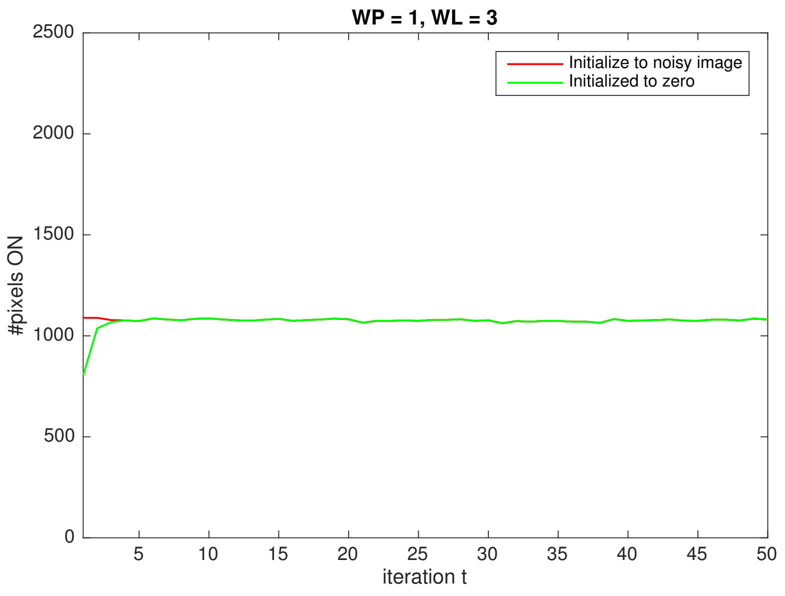


(h) under smoothing WP = 1, WL = 120

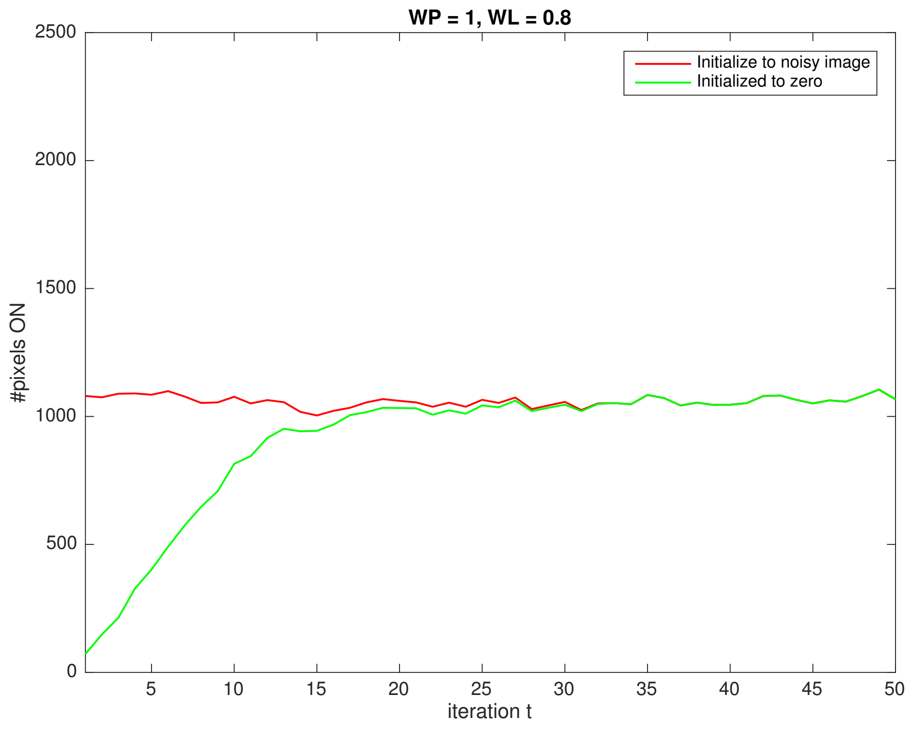
(c) Running the Gibbs sampler with $W^P = 11$ and $W^L = 14$ for 200 iterations, gives the plot below. MAE seems to converge to 0.002418 at around 30th iteration. X axis is the number of iterations and Y axis is the MAE value.



(d) Initialization and traceplots: Plotting the fraction of pixels that are turned on at every iteration for parameters $W^P = 1$ and $W^L = 3$, we get the below plot comparing the one initialized with the noisy image and another initialized with all zeros. These two converge very early i.e, in around 4 iterations. X axis gives the iteration count and Y axis is the number of pixels turned on.

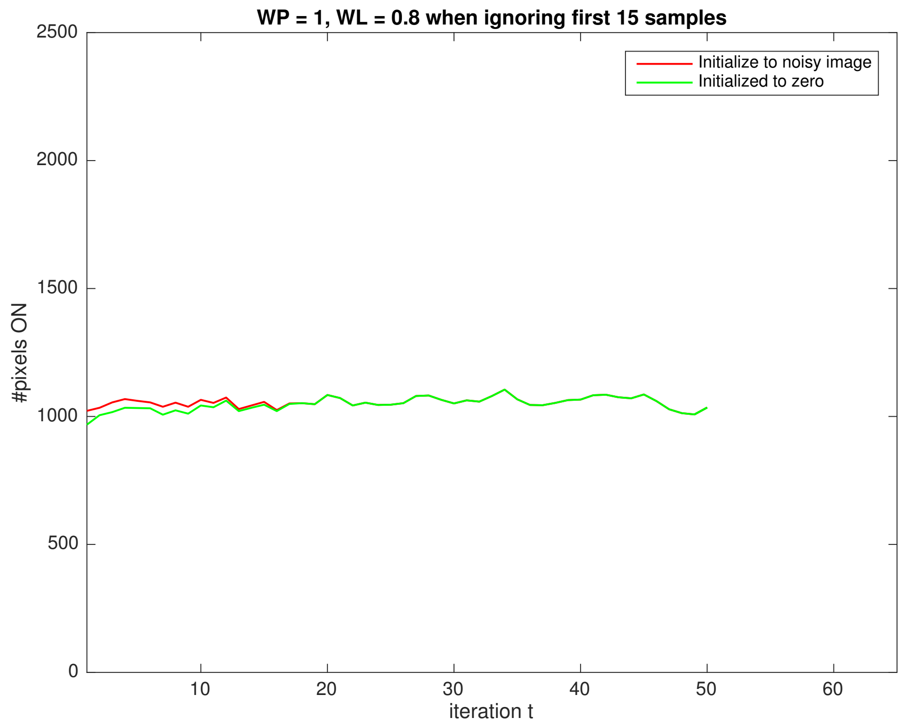


A similar plot for the parameter setting $W^P = 1$ and $W^L = 0.8$, makes the initialization a significant factor for convergence. It takes around 31 iterations to converge.



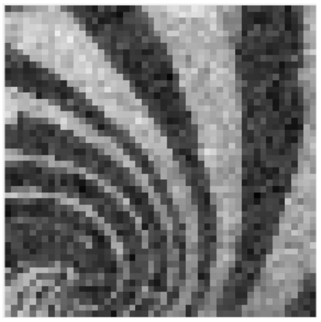
Reason for different sensitivity - Initialization to noisy image gives a quick start to the model to reach the optimum sooner than an initialization to zeros. The quality of noisy image matters when initialized to noisy image data especially when W^L is high.
A difference in values of W^L and W^P defines the importance the model gives to the observed noisy data (W^L) versus the neighbor pixel values (W^P).
A relatively higher value of W^L as compared to W^P, makes the model's search give more importance to the noisy image data. With an initialization to noisy data, this setting prefers trusting the noisy image data more. The quality of the noisy image has an impact on reaching the optimum.
When W^P dominates, the neighboring pixel tries to make a pixel as close to them as possible. With an initialization of zero, this setting forces most values to be to zero. With a suppressed W^L, the noisy input image has a very less say in directing the model close to truth values.

(e) Burn in: If we do an initialization with the noisy image data, we would already have a headstart and burn-in doesn't contribute much. But if we start with an initialization of all zeros, it could be beneficial to throw away first few samples to ignore the noisy samples. Below is the plot for the cold start when ignoring first 15 samples (value chosen by referring to the graph in the last question). The convergence is faster here.



3. Monte Carlo De-noising for Real-Valued Images:

(a) The baseline MAE with the given noisy data is 0.07930. The weight values that gives a reasonably good de-noising performance are W^P= 9 and W^L=57 giving a MAE of 0.06331. With some manual trail and error, I could find a parameter combination which performed the best amongst the values I searched giving an MAE of 0.06228. This corresponds to the parameters W^P= 193 and W^L=1100. And the corresponding posterior mean image is shown below. *Note* - This looks to me like overfitting. But since the question asked for a better performing model just on this image, I tried to explore W^P and W^L to decrease MAE as the only criteria. This might not have a good generalization performance.



(b) Starting with the given basic equation we have the conditional distribution of Y_{ij} given $\mathbf{y}_{\mathbf{A}_{ij}}$ and x_{ij} with the general case of a different pairwise weight W^P_{ijkl}as,

$$P(Y_{ij} = y|\mathbf{y}_{\mathbf{A}_{ij}}, x_{ij}) \propto \exp\left(- \sum_{(k,l)\in A_{ij}} W^P_{ijkl}(y_{kl} - y)^2 - W^L(x_{ij} - y)^2 \right) \qquad (4)$$

Simplifying and dropping additional terms that don't depend on y, we get-

$$P(Y_{ij} = y | \mathbf{y}_{\mathbf{A}_{ij}}, x_{ij}) \propto \exp\left(-y^2(\sum_{(k,l)\in A_{ij}} W^P_{ijkl} + W^L) + 2y\left(W^L x_{ij} + \sum_{(k,l)\in A_{ij}} W^P_{ijkl} y_{kl}\right)\right) \tag{5}$$

In a normal distribution with mean μ_{ij} and standard deviation σ_{ij}, we have that:

$$\mathcal{N}(y|\mu_{ij}, \sigma_{ij}) \propto \exp\left(-\frac{1}{2\sigma^2_{ij}}(y-\mu_{ij})^2\right) \propto \exp\left(-y^2\frac{1}{2\sigma^2_{ij}} + y\frac{1}{\sigma^2_{ij}}\mu_{ij}\right) \tag{6}$$

By matching the coefficients in the two distributions, we can see that:

$$-y^2\frac{1}{2\sigma^2_{ij}} = -y^2(\sum_{(k,l)\in A_{ij}} W^P_{ijkl} + W^L) \tag{7}$$

$$\sigma^2_{ij} = \frac{1}{2(\sum_{(k,l)\in A_{ij}} W^P_{ijkl} + W^L)} \tag{8}$$

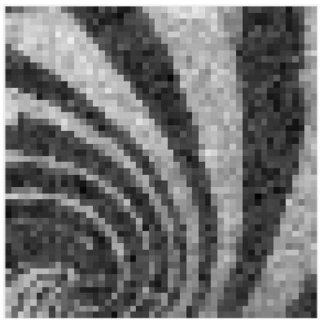
$$\tag{9}$$

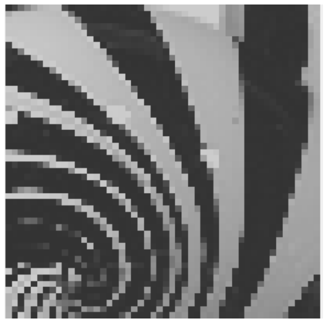
$$y\frac{1}{\sigma^2_{ij}}\mu_{ij} = 2y\left(W^L x_{ij} + \sum_{(k,l)\in A_{ij}} W^P_{ijkl} y_{kl}\right) \tag{10}$$

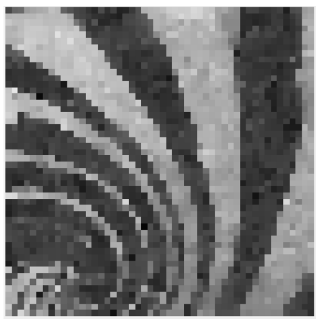
$$\mu_{ij} = 2\sigma^2_{ij}\left(W^L x_{ij} + \sum_{(k,l)\in A_{ij}} W^P_{ijkl} y_{kl}\right) \tag{11}$$

$$= \frac{1}{\sum_{(k,l)\in A_{ij}} W^P_{ijkl} + W^L}\left(W^L x_{ij} + \sum_{(k,l)\in A_{ij}} W^P_{ijkl} y_{kl}\right) \tag{12}$$

(c) The MAE I got is 0.04997, the values of the weights corresponding to this are W^P= 11 and W^L= 415 and the de-noised image is shown below in Figure k. This is a significant improvement in MAE from the previous model. Also, this model looks visually less noisy than before. There is a better contrast especially in the edges. To compare better, the output from the last model, original and the output from this model are displayed below. Another thing to note is, that this model performed badly for the parameter values from the previous model W^P= 193 and W^L=1100 giving an MAE of 0.08465 and made the output look very hazy.


(i) with global pairwise weights


(j) original image


(k) with general pairwise weights

4. Parameter learning pseudocode: MCMC helps sampling in high dimensional distributions by breaking them into univariate or simpler multivariate distribution. In the given example, Gibb's sampling becomes a simpler approach as we can use the conditional distribution (here it take known forms for the parameters) with the rest of the parameters fixed (in their current value). Hence, reducing the burden of calculating the full partition function / normalizer (Z).
In case of MLE, we get single point estimates of these parameters that maximizes the likelihood function for the observed data. For MLE to work on par, it needs a lot of data.

Pseudocode -
To illustrate the use of Gibb's sampling, I am simplifying the model. Consider the binary image case and assume that we had a single parameter W_{ij}^L . Using Bayesian Learning, this would be a counting over the training images (Bernoulli)-

For each pixel location,

$$W_{ij}^L = \frac{N_{ij}^1}{N_{ij}^1 + N_{ij}^0}$$

where N_{ij}^1 is the number of times the ij pixel (y_{ij}) is turned on and N_{ij}^0 is the number of times the ij pixel is turned off considering all the training images.

MCMC helps here by avoiding a point estimate of the parameters by counting over all training images and just considering samples from the training set.

Consider the Metropolis Hastings sampler algorithm -

```
Intialize $\mathbf{x_1}$
for t from 1 to T do
    Sample a new value $\mathbf{x}' \sim Q(\mathbf{x}'|\mathbf{x_t})$
    Compute the acceptance ration $\alpha(\mathbf{x}', \mathbf{x_t}) = \frac{P(\mathbf{x}')Q(\mathbf{x_t}|\mathbf{x}')}{P(\mathbf{x_t})Q(\mathbf{x}'|\mathbf{x_t})}$
    Draw a uniformly distributed random variable $u \sim U(0, 1)$
    if$(u <= \alpha)$ set $\mathbf{x_{t+1}} = \mathbf{x}'$ else set $\mathbf{x_{t+1}} = \mathbf{x_t}$
end for
Return set of samples $\mathbf{x_1}, ..., \mathbf{x_T}$
```

Gibb's sampling could be used here in the loop as -
$Q(\mathbf{x}'|\mathbf{x}) = P(x_i'|\mathbf{x'_{-i}})(\mathbf{x'_{-i}} = \mathbf{x_{-i}})$

In general, when there are multiple parameters and we have to consider sampling from this multi dimensional pixel conditional.