CS688: Graphical Models - Spring 2020

Assignment 4

Submitted by: Pulkit Sharma

Introduction: In this part, you will experiment with Monte Carlo Inference. More specifically, you will experiment with Monte Carlo image denoising using grid-structured conditional random field models.

Binary Model: Consider a probability distribution over a vector $y \in \{-1, +1\}^d$ model of the form

$$p(y|b, w) = \frac{1}{Z} \prod_{i=1}^{d} \exp(b_i y_i) \prod_{(i,j)\in \text{pairs}} \exp(w_{ij} y_i y_j).$$

Here, each component y_i of y is in $\{-1, +1\}$, we can think of the "bias" b_i as "how much y_i wants to be $+1$", and we can think of the weights w_{ij} as "how much y_i and y_j want to have the same value".

Question 1. (*5 points*) **Derivation of conditional distribution** Mathematically derive a formula for $p(y_i = 1|y_{-i}, b, w)$, the conditional probability that $y_i = 1$ given the state of all other variables. Write your solution using the function $\sigma(s) := 1/(1 + \exp(-s))$. You will need to reference the set of all neighbors of i, $\text{nb}(i)$.

Solution:

$$\begin{aligned}
p(y_i = 1|y_{-i}, b, w) &= \frac{p(y_i = 1, y_{-i}|b, w)}{p(y_{-i}|b, w)} && (1)\\
&= \frac{p(y_i = 1, y_{-i}|b, w)}{p(y_i = -1, y_{-i}|b, w) + p(y_i = 1, y_{-i}|b, w)} && (2)\\
&= \frac{1}{1 + \frac{p(y_i=-1, y_{-i}|b,w)}{p(y_i=1, y_{-i}|b,w)}} && (3)\\
&= \frac{1}{1 + \frac{\exp(-b_i)\prod_{j\in \text{nb(i)}} \exp(-w_{ij}y_j)}{\exp(b_i)\prod_{j\in \text{nb(i)}} \exp(w_{ij}y_j)}} && (4)\\
&= \frac{1}{1 + \exp(-2b_i)\prod_{j\in \text{nb(i)}} \exp(-2w_{ij}y_j)} && (5)\\
&= \frac{1}{1 + \exp(-2(b_i + \sum_{j\in \text{nb(i)}} w_{ij}y_j))} && (6)\\
&= \sigma\left(2(b_i + \sum_{j\in \text{nb(i)}} w_{ij}y_j)\right) && (7)
\end{aligned}$$

Question 2. (*5 points*) **Pseudo-Code for Gibbs Sampling** Write pseudo-code for the Gibbs sampling algorithm, where in each iteration, you update for $i = 1, i = 2, ..., i = d$. (This is slightly different from in

class where each iteration used a single randomly chosen i.) You should take as input (1) the total number of iterations, and (2) the model parameters $\{b, w\}$. You should return a list of the samples of y that you get after each iteration.

Solution:

For $k = \{-1, +1\}$,

$$p(y_i = k|y_{-i}, b, w) = \sigma\left(-2k(b_i + \sum_{j \in \text{nb(i)}} w_{ij} y_j)\right) \tag{8}$$

y_i can be sampled using Uniform distribution and then using sigma function as a cut-off.

Algorithm 1 Samples using Gibbs Sampling

function GIBBSSAMPLING(N, b, w, D)
  Initialize $\mathbf{y}^{(0)}$ ▷ $y \in \mathcal{R}^D$
  for t = 1, 2, ..., N **do**
    $\mathbf{y} \leftarrow \mathbf{y}^{(t-1)}$
    for d = 1, 2, ..., D **do**
      Sample $r \sim U(0, 1)$
      if $r < \sigma\left(2(b_d + \sum_{j \in \text{nb(d)}} w_{dj} y_j)\right)$ **then**
        $y_d = 1$
      else
        $y_d q10_a rgs_l ist = -1$
      end if
    end for
    Record $\mathbf{y}^{(t)} \leftarrow \mathbf{y}$
  end for
  return $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(N)}$
end function

Question 3. (*5 points*) **Samples** Consider an 30×30 4-connected grid structured model where we have a zero bias of $b_i = 0$, and a constant interaction strength of $w_{ij} = \bar{w}$. Now, implement the Gibbs sampling algorithm. For each value of $\bar{w} \in \{0, .1, .2, .3, .4, .5\}$, run a single chain Markov chain for 100 iterations. (Where, again, one iteration is a full pass over the entire grid.) Plot the final samples for each $\bar{w}$ value. Show each sample as an image with black pixels for variables with $y_i = -1$ and white pixels for variables with for $y_i = +1$. Make sure to label each image with the value of $\bar{w}$ it corresponds to.

Solution: I initialize the image y with all ones and obtain the following images by running Gibbs Sampling for 100 iterations for different settings of w.

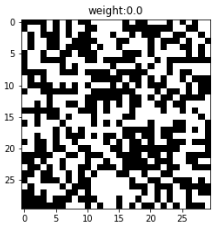


Figure 1: $\bar{w} = 0.0$

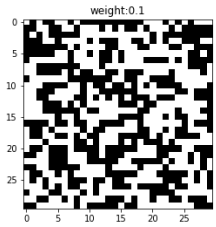


Figure 2: $\bar{w} = 0.1$

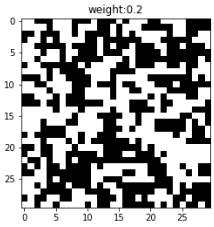


Figure 3: $\bar{w} = 0.2$

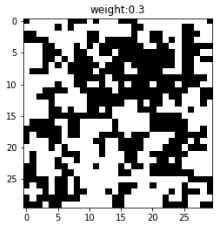


Figure 4: $\bar{w} = 0.3$

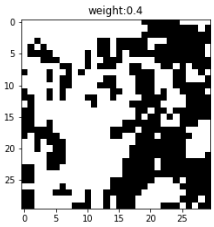


Figure 5: $\bar{w} = 0.4$

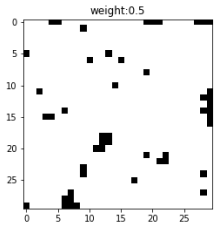


Figure 6: $\bar{w} = 0.5$

Question 4. (*5 points*) **Discussion** Explain briefly (no more than a single paragraph of 4 sentences) why the images from the previous question look like they do, and why they change for various values of $\bar{w}$.

Solution:

From the above images, the number of times $+1$ occurs in the final sampled image becomes higher as we increase the parameter, $\bar{w}$. Higher values of w imply pixel value depends a lot on its neighbors and the distribution becomes skewed towards all +1, or all -1. Since the image is initialized with all ones, most of the samples are drawn near all ones. Mathematically, distributions scales exponentially with $\bar{w}$

Question 5. (*5 points*) **Mixing Times** For the model in the previous problem, it is easy to see by symmetry that the true mean value of y is $\mathbb{E}[y] = 0$. To get an idea of average performance, for each value of $\bar{w} \in \{0, .1, .2, .3, .4, .5\}$ run *100 independent* Markov chains each for 100 iterations. (Where, again, one iteration is a full pass over the entire grid). For each iteration, compute the mean value of y (averaged over the 100 independent repetitions). Plot the 6 curves together on one graph, with the number of iterations on

the x-axis and the mean value of y on the y-axis. Make sure to label the axes and the value of $\bar{w}$ for each curve.

Solution:

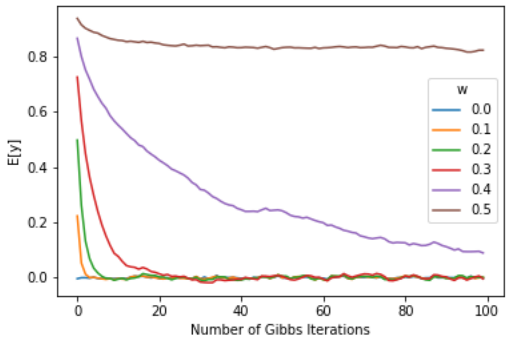


Figure 7: Expected value of y vs number of Sampling iterations

Question 6. (*5 points*) **Discussion** Do your samples give the correct value of $\mathbb{E}[y]$? When do the samples look better or worse? Explain in a single paragraph of no more than four sentences why the curves in the previous problem look as they do.

Solution:

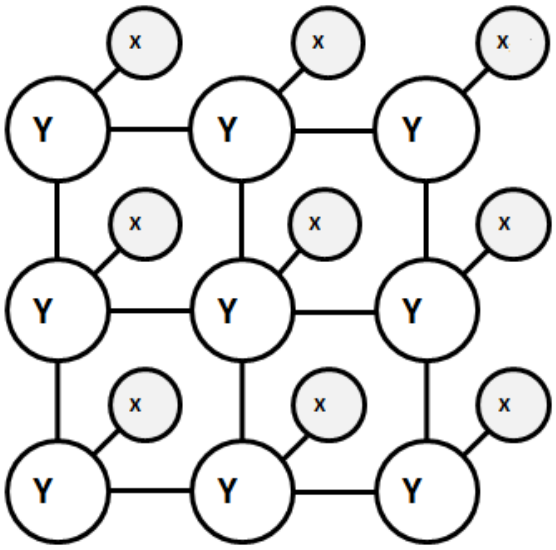
For higher $\bar{w}$ most of the samples are drawn just around all ones or all negative ones. Higher values of $\bar{w}$ will take longer to converge to the true mean value and 100 iterations are not sufficient. For lower values of $\bar{w}$, we start getting the correct expected value (=0) within a few iterations. However, the initial samples are still away from the true distribution because it is initialized at all ones.

Data Set: The data for the remaining questions consists of a single 200×154 pixel binary image. The original image is `im_clean.png`, while the copy with noise is `im_noisy.png`. The two images are shown below. We will refer to the images on the left as the "true" image and the image with noise as the "observed" or "noisy" image.

(a) im_clean (b) im_noisy

Image Denoising Model: Intuitively, given a "noisy image" x, we should think of this defining a probabilistic model over the "clean image" y, with the structure shown in the figure on the next page.

We can think of the model as being defined exactly as in the previous problem, except that $b(x, \theta)$ and $w(x, \theta)$ are now functions of the input x as well as some parameters θ



$$p(y|x,\theta) = \frac{1}{Z(x)} \prod_{i=1}^{d} \exp(b_i(x,\theta)y_i) \prod_{(i,j)\in \text{pairs}} \exp(w_{ij}(x,\theta)y_i y_j).$$

Question 7. (*10 points*) **Fixed Parameter Denoising** Make sure you scale the input x so that each value is either -1 or +1. Then, determine the parameters by the simple mapping of $b_i = .5x_i$ and $w_{ij} = 0.3$. Run 100 iterations of Gibbs sampling. Show an image of μ, the mean value of y you obtain over those samples, and report the mean per-pixel absolute difference of μ from the true output. (Make sure to scale your image appropriately so that all values between -1 and $+1$ are shown, and make sure that both μ and y are in the original space between -1 and $+1$ before computing a difference.)

Solution: Using the given values of b and w, following image is the mean image obtained using denoising CRF model

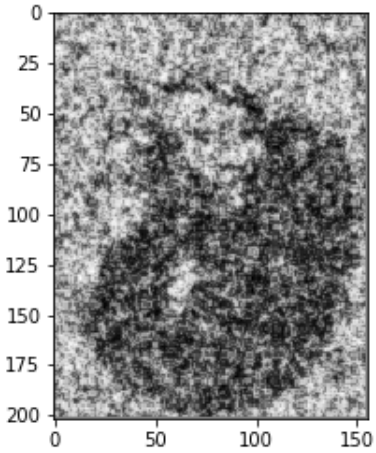


Figure 8: Denoised image using the given parameters

Mean per-pixel Absolute Difference of the denoised image and clean image: 0.6525

Question 8. (*10 points*) **Varying Parameters** Consider setting $b_i = \theta_1 x_i$ and $w_{ij} = \theta_2$ where θ_1 and θ_2 are 2 unknown parameters. (In the previous problem we essentially used $\theta_1 = .5$ and $\theta_2 = 0.3$) Can you find a pair of θ_1 and θ_2 to give you lower error than the values from the previous problem? Limit the total number of iterations you do for any parameter setting to 100 iterations. Describe (a) how you tried to find better parameters (there is no single correct answer here) (b) show an image of your final denoised image and (c) report the final error.

Solution:

(a) I first run a hyperparameter grid search over a range of values of θ_1 and θ_2.
$\theta_1 \in 0, 0.1, ..., 1.0$
$\theta_2 \in 0, 0.1, ..., 1.0$
This would generate 100 different images. Now to choose one setting of parameters that works good we need a metric that compares similarity of two images. I used following three metrics:

(i) MSE: Mean Squared Error

(ii) MAE: Mean Absolute Error

(iii) Structural Similarity Index

Using all the above metrics, I choose the parameters θ which are performing the best. I print the image obtained for best setting of these parameters and compare the images qualitatively. The I run a finer grid search for parameters θ around the best parameters found from above in $[-0.5, +0.5]$ range of that value with steps of 0.01.
(b) Optimal parameters found on grid search: $\theta_1 = 0.52$, $\theta_2 = 0.79$. Measure of similarity I used is MSE as perceptually it gave the most satisfying result. MSE for this image w.r.t. the original image is 0.0022. SSIM of original image and noisy image is 0.007. (lower the better). Mean Absolute Error for this obtained image is 0.13

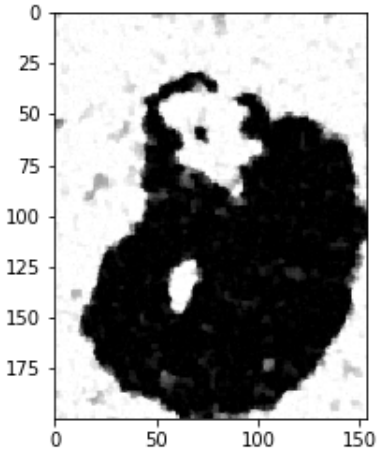


Figure 9: Denoised Image

Extra Credit Finally, here are some extra-credit problems. These are *far* more difficult than the above problems and have very small point values. *These problems are a terrible "value" in terms of the points you will earn for spending your time*. To maximize your score with limited time, you should make sure the above problems are done thoroughly and ignore these. We will be very stingy in giving credit for these problems- do them only for the glory, and at your own risk! These problems are more open-ended. As a result, you will need to carefully describe what you did for each problem.

Question 9. (*5 points*) **MCMC: Varying w_{ij} parameters** Come up with a scheme to have w_{ij} vary as a function of x, rather than being constant. (1) Describe the informal motivation for the scheme (2) describe formally what you have done (3) Give an image of the final results (4) list your final error, and how much this improves on the previous problem.

Solution: Current modelling of w_{ij}, states how similar should a pixel be to its neighboring pixels. This enforces a pixel to be more similar to its neighbors unless the component from bias term is significant. We can also exploit this neighborhood information from the original noisy image.

1. One way to do this could be by increasing the similarity weighing to its neighbors for a pixel if in the noisy image, the region around the pixel is smooth. So regions with low gradient should be more similar to its neighbors. Assumption here is that the noise is generally non-continuous.

2. $w = \theta_2 \times sobel(noisyImage)$. This w is used to represent the new distribution of $y|x$.

3. below is the denoised output:

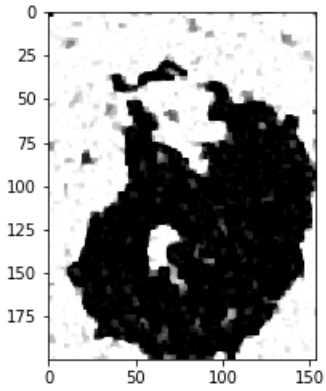


Figure 10: Denoised Image

4. Final MAE: 0.0158. The MAE of previous approach was better than this one.

Question 10. (*5 points*) **MCMC: Faster mixing** Above, we used a fixed update order of $s = 1, 2, ..., d$. Can you come up with a different update order that leads to faster mixing? (Meaning that you get closer to the true mean value of 0 during the 100 iterations. (1) Describe what your improved update order is and (2) make another version of the same graph from Question 5, showing your improvement. (3) Discuss why you think this mixes faster.

Solution:

1. From the plots in Q5, the initial points are drawn away from the true mean of the distribution which results in a bad estimate of the true mean. The true distribution is symmetric and becomes more concentrated at extremes as $\bar{w}$ increases. That is the reason why we get poorer estimates for $\mathbb{E}[y]$. To fix this we gave make updates to the dimensions which take us closer to the true mean quickly. Mathematically this would mean the dimensions which have negative gradient towards zero.

2. Plot:

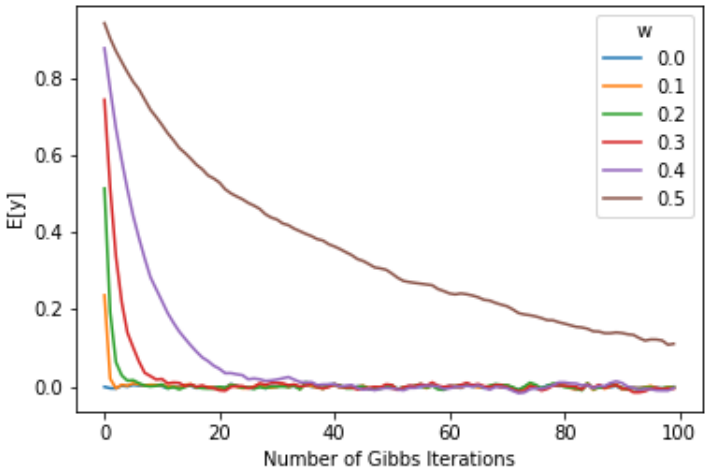


Figure 11: Faster Mixing

3. Gradient of $logP$ can be written as:

$$logP(\mathbf{Y}) = b\sum_{i=1}^{D} Y_i + \bar{w} \sum_{pairs(i,j)} Y_i Y_j \tag{9}$$

$$\frac{\partial logP}{\partial y_i} = b + \bar{w} \sum_{j \in nb(i)} Y_j \tag{10}$$

The new chosen order picks the dimensions where value of gradient of the probability model is lowest if Y_i is +1, and high if Y_i is -1.

$$m_d = -(b + \bar{w} \sum_{j \in nb(i)} Y_j) Y_i$$

More Precisely, higher the value of m_d, higher the preference and vica versa.
Since the updates are such that the directions that take us towards zero are preferred, thus we get more samples near the true mean more quickly and Monte-Carlo estimate on these samples produces better results.