CS228 Homework 4 Solutions

Instructor: Stefano Ermon – ermon@stanford.edu

Available: 02/17/2017; Due: 03/3/2017

1. **[20 points]** We have a data association problem where there are K objects and we are given K observations. Each observation corresponds to a single object, and we are given one observation for each object. However, we don't know which observation corresponds to which object, and we would like to infer that using a probabilistic model relating observations to objects. Specifically we have

   - K objects $u_1, \ldots, u_K$
   - observations $v_1, \ldots, v_K$, where $\mathrm{Val}(v_i) = \{a_1, \ldots, a_L\}$ (so v_i is a discrete random variable), where *each observation corresponds to the appearance of one object and there is exactly one observation of each object*
   - correspondence variables $C_1, \ldots C_K$, where $\mathrm{Val}(C_i) = \{1, \ldots, K\}$; $C_i = k$ denotes that measurement v_i is derived from object u_k
   - a *known* appearance model for each object u_k, $P_k(v_i = a_l | C_i = k)$.

   Note that because of the mutex constraints, the correspondence variables $C_1, \ldots, C_K$ will be a permutation over $1, \ldots, K$. We also assume for simplicity that all permutations are equally likely *a priori*.

   We wish to compute the marginals $P(C_i | v_1, \ldots, v_K)$, for $i = 1, \ldots, K$, using Metropolis-Hastings (MH) to sample the correspondence variables. We will start with an arbitrary assignment to $C_1, \ldots C_K$, and take MH-steps. The proposal distribution that we will use randomly picks two correspondence variables C_i, C_j from a uniform distribution over all pairs of correspondence variables, and swaps their assignments.

   (a) **[10 points]** Compute the acceptance probability for each MH step.

   Answers The stationary distribution of our chain is $\pi(C) = P(C_1, \ldots, C_K | v_1, \ldots, v_K) \propto P(C) \prod_{k=1}^{K} P_{C_k}(v_k | C_k)$, so each sample consists of a joint assignment to all the correspondence variables.

   Let c, c' be complete assignments to the correspondence variables, so $c = (c_1, \ldots, c_K)$, and $c' = (c'_1, \ldots, c'_K)$. The MH acceptance probability, for a given transition $c \to c'$ where we swapped the assignments for C_i and C_j is given by

   $$\mathcal{A}(c, c') = \min\left[1, \frac{\pi(c')Q(c' \to c)}{\pi(c)Q(c \to c')}\right] = \min\left[1, \frac{\prod_{k=1}^{K} P(v_k | c'_k)}{\prod_{k=1}^{K} P(v_k | c_k)}\right] = \min\left[1, \frac{P(v_i | c'_i)P(v_j | c'_j)}{P(v_i | c_i)P(v_j | c_j)}\right]$$

   where the second equality holds because the proposal is symmetric, and that the $P(C)$ terms cancel, and the third equality holds because only the assignments to C_i and C_j were changed; the rest of the assignments are the same and the terms cancel.

   In general, if your final expression was correct, you were given full credit; otherwise, partial credit was given based on how far along you got. Many did not realise what the stationary distribution was.

   (b) **[5 points]** Suppose we have run the MH sampler for a long time and collected M samples $(C_1[m], \ldots, C_K[m])$ for $m = 1, \ldots, M$ after the chain has mixed. Give an explicit expression for estimating the marginal $P(C_i | v_1, \ldots, v_K)$.

   Answers $P(C_i = k | v_1, \ldots, v_K) \approx \frac{1}{M} \sum_{m=1}^{M} \mathbf{1}\{C_i[m] = k\}$

(c) **[5 points]** Your friend Geoff Gibbs hears about your MH algorithm and suggests that you can also consider using Gibbs sampling to compute your marginals. Briefly explain why this will or will not work.

Answers This will not work: given all the other correspondence variables, due to the mutex constraint, the variable being sampled cannot be changed. Hence, the state space is not connected (the chain is not regular) and Gibbs does not work.

2. **[20 points] Multi-conditional Parameter Learning, Markov Networks**

In this problem, we will consider the problem of learning parameters for a Markov network using a specific objective function. In particular assume that we have two sets of variables $\boldsymbol{Y}$ and $\boldsymbol{X}$, and a dataset $\mathcal{D} = \{(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^M, \boldsymbol{y}^M)\}$. We will estimate the model parameters $\boldsymbol{\theta} = [\theta_1 \ldots \theta_n]$ by maximizing the following objective function:

$$g(\boldsymbol{\theta}; \mathcal{D}) = (1 - \alpha)\ell_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{\theta}; \mathcal{D}) + \alpha \ell_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{\theta}; \mathcal{D})$$

where $\ell_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{\theta}; \mathcal{D})$ means the conditional log-likelihood of the dataset $\mathcal{D}$ using the distribution $P_{\boldsymbol{\theta}}(\boldsymbol{X} \mid \boldsymbol{Y})$ defined by the Markov network with parameters $\boldsymbol{\theta}$ (similarly for $\ell_{\boldsymbol{Y}|\boldsymbol{X}}$). Thus, our objective is a mixture of two conditional log-likelihoods ($0 < \alpha < 1$). As usual, we consider a log-linear parameterization of a Markov network, using a set of n features $f_i(\boldsymbol{X}_i, \boldsymbol{Y}_i)$ where $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ are some (possibly empty) subsets of the variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively.

(a) **[10 points]** Write down the full objective function $g(\boldsymbol{\theta}; \mathcal{D})$ in terms of the features f_i and weights θ_i

Answers

$$\begin{aligned} g(\boldsymbol{\theta}; \mathcal{D}) \quad = \quad & \sum_{m=1}^{M} \left(\sum_i \theta_i f_i(\boldsymbol{X}_i^m, \boldsymbol{Y}_i^m) \right) - (1 - \alpha) \log \sum_{\boldsymbol{Y}} \exp \sum_i \theta_i f_i(\boldsymbol{X}_i^m, \boldsymbol{Y}_i) - \\ & \alpha \log \sum_{\boldsymbol{X}} \exp \sum_i \theta_i f_i(\boldsymbol{X}_i, \boldsymbol{Y}_i^m) \end{aligned}$$

(b) **[10 points]** Derive $\frac{\partial}{\partial \theta_i} g(\theta; \mathcal{D})$: the derivative of the objective with respect to a weight θ_i. Write your final answer in terms of feature expectations $\boldsymbol{E}_Q[f_i]$, where Q is either: the empirical distribution of our dataset $\hat{P}$; or a conditional distribution of the form $P_{\boldsymbol{\theta}}(\boldsymbol{W} \mid \boldsymbol{Z} = \boldsymbol{z})$ (for some sets of variables $\boldsymbol{W}, \boldsymbol{Z}$, and assignment $\boldsymbol{z}$.)

Answers

$$\frac{\partial}{\partial \theta_i} g(\theta; \mathcal{D}) = M \boldsymbol{E}_{\hat{P}}[f_i] + \sum_{m=1}^{M} \left(-(1 - \alpha) \boldsymbol{E}_{P_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}^m)}[f_i] - \alpha \boldsymbol{E}_{P_{\boldsymbol{\theta}}(\boldsymbol{X}|\boldsymbol{Y}=\boldsymbol{y}^m)}[f_i] \right)$$

3. **[10 points] Expectation Maximization in a Naive Bayes Model**

Consider the Naive Bayes model with class variable C and discrete evidence variables $X_1, \ldots, X_n$. The CPDs for the model are parameterized by $P(C = c) = \theta_c$ and $P(X_i = x \mid C = c) = \theta_{x_i|c}$ for $i = 1, \ldots, n$, and for all assignments $x_i \in \mathit{Val}(X_i)$ and classes $c \in \mathit{Val}(C)$.

Now given a data set $\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$, where each $\boldsymbol{x}[m]$ is a complete assignment to the evidence variables, $X_1, \ldots, X_n$, we can use EM to learn the parameters of our model. Note that the class variable, C, is never observed.

Show that if we initialize the parameters uniformly,

$$\theta_c^0 = \frac{1}{|\mathit{Val}(C)|} \qquad \text{and} \qquad \theta_{x_i|c}^0 = \frac{1}{|\mathit{Val}(X_i)|},$$

for all x_i, c, then the EM algorithm converges in one iteration, and give a closed form expression for the parameter values at this convergence point.

Answer:

Computing the first iteration of EM (with initialization given above) we get:

$$\begin{aligned}
\theta^1_{C=c} &= \frac{\bar{M}_{\theta^t}[c]}{M} = \frac{1}{M}\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^0) \\
&= \frac{1}{M}\sum_m \frac{P(\boldsymbol{x}[m] \mid c, \boldsymbol{\theta}^0)P(c \mid \boldsymbol{\theta}^0)}{P(\boldsymbol{x}[m] \mid \boldsymbol{\theta}^0)} \\
&= \frac{1}{M}\sum_m \frac{P(\boldsymbol{x}[m] \mid c, \boldsymbol{\theta}^0)P(c \mid \boldsymbol{\theta}^0)}{\sum_{c' \in \mathit{Val}(C)} P(\boldsymbol{x}[m] \mid c', \boldsymbol{\theta}^0)P(c' \mid \boldsymbol{\theta}^0)} \\
&= \frac{1}{M}\sum_m \frac{P(c \mid \boldsymbol{\theta}^0)\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^0)}{\sum_{c' \in \mathit{Val}(C)} P(c' \mid \boldsymbol{\theta}^0)\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c', \boldsymbol{\theta}^0)} \\
&= \frac{1}{M}\sum_m \frac{\frac{1}{|\mathit{Val}(C)|}\prod_{i=1}^n \frac{1}{|\mathit{Val}(X_i)|}}{\sum_{c' \in \mathit{Val}(C)} \frac{1}{|\mathit{Val}(C)|}\prod_{i=1}^n \frac{1}{|\mathit{Val}(X_i)|}} \\
&= \frac{1}{|\mathit{Val}(C)|}
\end{aligned}$$

and

$$\begin{aligned}
\theta^1_{X_i=x|C=c} &= \frac{\bar{M}_{\theta^0}[x,c]}{\bar{M}_{\theta^0}[c]} = \frac{\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^0)\boldsymbol{1}\{x_i[m]=x\}}{\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^0)} \\
&= \frac{\sum_m \frac{1}{|\mathit{Val}(C)|}\boldsymbol{1}\{x_i[m]=x\}}{\frac{M}{|\mathit{Val}(C)|}} \qquad \text{from above} \\
&= \frac{1}{M}\sum_m \boldsymbol{1}\{x_i[m]=x\} = \frac{M[x]}{M}
\end{aligned}$$

where $M[x]$ is the number of times $X_i = x$ appears in the data, $\mathcal{D}$.

We will now show by induction that for all $t \geq 1$, $\theta^t_{C=c} = \frac{1}{|\mathit{Val}(C)|}$ and $\theta^t_{X_i=x|C=c} = \frac{M[x]}{M}$. The second parameter implies that $P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^t) = P(\boldsymbol{x}_i[m] \mid c', \boldsymbol{\theta}^t)$ for all $c, c' \in \mathit{Val}(C)$. We have just shown this for $t = 1$, so assuming true for some t, we have for $t+1$,

$$\begin{aligned}
\theta^{t+1}_{C=c} &= \frac{\bar{M}_{\theta^t}[c]}{M} = \frac{1}{M}\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^t) \\
&= \frac{1}{M}\sum_m \frac{P(\boldsymbol{x}[m] \mid c, \boldsymbol{\theta}^t)P(c \mid \boldsymbol{\theta}^t)}{P(\boldsymbol{x}[m] \mid \boldsymbol{\theta}^t)} \\
&= \frac{1}{M}\sum_m \frac{P(\boldsymbol{x}[m] \mid c, \boldsymbol{\theta}^t)P(c \mid \boldsymbol{\theta}^t)}{\sum_{c' \in \mathit{Val}(C)} P(\boldsymbol{x}[m] \mid c', \boldsymbol{\theta}^t)P(c' \mid \boldsymbol{\theta}^t)} \\
&= \frac{1}{M}\sum_m \frac{P(c \mid \boldsymbol{\theta}^t)\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^t)}{\sum_{c' \in \mathit{Val}(C)} P(c' \mid \boldsymbol{\theta}^t)\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c', \boldsymbol{\theta}^t)} \\
&= \frac{1}{M}\sum_m \frac{\frac{1}{|\mathit{Val}(C)|}\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^t)}{\prod_{i=1}^n P(\boldsymbol{x}_i[m] \mid c, \boldsymbol{\theta}^t) \cdot \sum_{c' \in \mathit{Val}(C)} \frac{1}{|\mathit{Val}(C)|}} \\
&= \frac{1}{M}\sum_m \frac{1}{|\mathit{Val}(C)|} = \frac{1}{|\mathit{Val}(C)|}
\end{aligned}$$

and

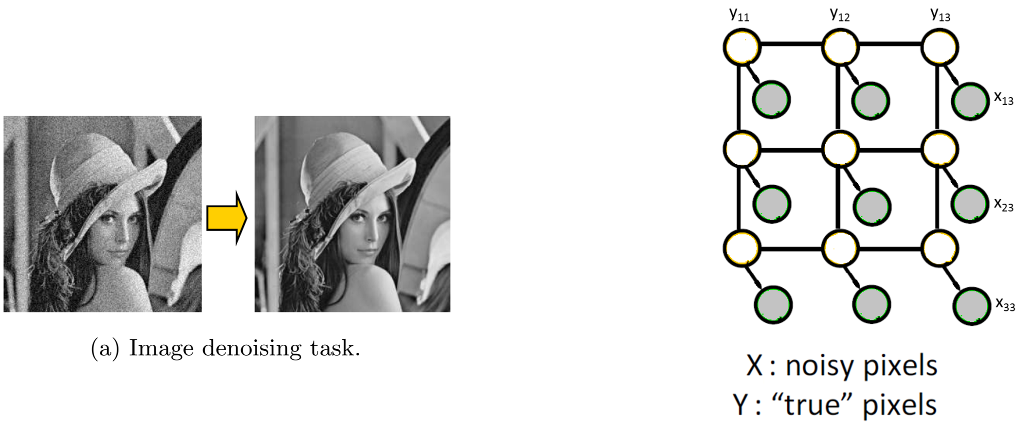
$$
\begin{aligned}
\theta^{t+1}_{X_i=x|C=c} &= \frac{\bar{M}_{\boldsymbol{\theta}^t}[x,c]}{\bar{M}_{\boldsymbol{\theta}^t}[c]} = \frac{\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^t)\boldsymbol{1}\{x_i[m]=x\}}{\sum_m P(c \mid \boldsymbol{x}[m], \boldsymbol{\theta}^t)} \\
&= \frac{\sum_m \frac{1}{|\mathit{Val}(C)|}\boldsymbol{1}\{x_i[m]=x\}}{\frac{M}{|\mathit{Val}(C)|}} \qquad \text{from above} \\
&= \frac{1}{M}\sum_m \boldsymbol{1}\{x_i[m]=x\} = \frac{M[x]}{M}
\end{aligned}
$$

Thus we have shown that initializing the parameters uniformly the EM algorithm converges in one iteration to

$$
\theta^t_{C=c} = \frac{1}{|\mathit{Val}(C)|} \qquad \text{and} \qquad \theta^t_{X_i=x|C=c} = \frac{M[x]}{M}.
$$

[60 points] Programming Assignment [1]

In this homework, you will apply Gibbs sampling to a simple Markov random field model for image restoration. In an image restoration problem, you are given an image corrupted by noise X and you want to recover the original image Y (see Figure 1 and Lecture 4).



(a) Image denoising task.

(b) Graphical model structure, for $N = M = 3$.

Figure 1: Image denoising with graphical models.

Let $\boldsymbol{x} = \{x_{ij}\}$ denote the observed image, with $x_{ij} \in \{-1, +1\}$ representing the pixel at row i and column j. Assume a black-and-white image, with -1 corresponding to white and +1 to black. The image has dimensions $N \times M$, so that $1 \leq i \leq N$ and $1 \leq j \leq M$. Assume a set of (unobserved) variables $\boldsymbol{y} = \{y_{ij}\}$ representing the true (unknown) image, with $y_{ij} \in \{-1, +1\}$ indicating the value of x_{ij} before noise was added. Each (internal) y_{ij} is linked with four immediate neighbors, $y_{i-1,j}$, $y_{i+1,j}$, $y_{i,j-1}$, and $y_{i,j+1}$, which together are denoted $y_{N(i,j)}$. Pixels at the borders of the image (with $i \in \{1, N\}$ or $j \in \{1, M\}$) also have neighbors denoted $y_{N(i,j)}$, but these sets are reduced in the obvious way. We denote E the corresponding set of edges. For example, the pair $((1,1),(1,2)) \in E$, but the pair $((1,1),(2,2)) \notin E$. The joint probability of $\boldsymbol{y}$ and $\boldsymbol{x}$ can be written (with no prior preference for black or white):

$$p(\boldsymbol{y}, \boldsymbol{x}) = \frac{1}{Z}\left\{\prod_{i=1}^{N}\prod_{j=1}^{M} \exp^{\eta y_{ij} x_{ij}}\right\} \times \left\{\prod_{((i,j),(i',j')) \in E} \exp^{\beta y_{ij} y_{i'j'}}\right\} \tag{1}$$

$$= \frac{1}{Z}\exp\left\{\eta \sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}x_{ij} + \beta \sum_{((i,j),(i',j')) \in E} y_{ij}y_{i'j'}\right\} \tag{2}$$

where

$$Z = \sum_{\boldsymbol{y},\boldsymbol{x}} \exp\left\{\eta \sum_{i,j} y_{ij}x_{ij} + \beta \sum_{((i,j),(i',j')) \in E} y_{ij}y_{i'j'}\right\} \tag{3}$$

(Notice in particular that each pair of neighbors, y_{ij} and $y_{i'j'}$, factors into the formula only once, despite that each variable is a neighbor of the other. Failing to account for this will lead to double counting of β values.) This is equivalent to a Boltzmann (sometimes called Gibbs) distribution with "energy":

$$E(\boldsymbol{y}, \boldsymbol{x}) = -\eta \sum_{i,j} y_{ij}x_{ij} - \beta \sum_{((i,j),(i',j')) \in E} y_{ij}y_{i'j'} \tag{4}$$

[1] Assignment adapted from Cornell's BTRY 6790, instructed by Adam Siepel

The system will have lower energy, and hence higher probability, in states in which neighboring y_{ij} variables, and neighboring y_{ij} and x_{ij} variables, tend to have the same value (assuming η and β are positive). This captures the fact that each noisy pixel x_{ij} is likely to be similar to the corresponding "true" pixel y_{ij}, and that images tend to be "smooth".

There are algorithms for deterministically estimating $\boldsymbol{y}$ given an image $\boldsymbol{x}$ but we will here use the alternative approach: we will devise a Markov Chain Monte Carlo (MCMC) algorithm to sample values of $\boldsymbol{y}$ conditional on $\boldsymbol{x}$. Here are some advantages over the deterministic algorithms:

i. It is very general, and can easily be extended to more complex graphs.

ii. It provides great flexibility for quantifying the uncertainty of $\boldsymbol{y}$ (and, potentially, for the parameters η and β).

iii. It is relatively straightforward in this setting to derive the exact conditional distributions for nodes given the Markov blanket, so Gibbs sampling is possible, and one need not worry about the acceptance rate for proposed samples.

You will apply your methods to two small, black-and-white images that have been made available with the problem set. These two noisy images, and the original, undistorted image from which they derive, are available both in PNG format and in a simple text format that lists each coordinate pair (i, j) and the corresponding value of x_{ij}. You may find it useful to convert between this text representation and a viewable image format.

(a) **[5 points]** Derive an expression for the conditional probability that pixel (i, j) is black given its Markov blanket, i.e. $p(y_{ij} = 1|y_{M(i,j)})$, where $y_{M(i,j)}$ denotes the variables in the Markov blanket of y_{ij} (but you should be explicit about which variables are included). Your expression should take the form of a logistic function and should depend only on η, β,and $y_{M(i,j)}$.

Answers:

$$\frac{1}{1 + e^{-2\eta x_{ij} - 2\beta \sum_{y \in y_{M(i,j)} \setminus x_{ij}} y}}$$

(b) **[10 points]** Outline a Gibbs sampling algorithm (in pseudocode) that iterates over the pixels in the image and samples each y_{ij} given its Markov blanket. Use the simple approach of sweeping across the image in row-major fashion on every iteration of the algorithm. Thus, an "iteration" will generate a complete new sample of $\boldsymbol{y}$. Allow for a burn-in of B iterations, followed by draws of S samples. You may assume η and β are fixed constants. How can we show in our case that the equilibrium distribution is in fact the posterior distribution $p(\boldsymbol{y}|\boldsymbol{x})$?

Answers:

```
  Given y
 \\ Burns
for b in 1...#burns
     for i in 1...rows
          for j in 1...cols
             yij <- sample(yij|Markov Blanket)
     iteration <- iteration + 1

 \\ Sampling
 for s in 1...#samples
     for i in 1...rows
          for j in 1...cols
             yij <- sample(yij|Markov Blanket)
     iteration <- iteration + 1
     collect y
```

(c) **[15 points]** Implement your algorithm and apply it to the image with 20% noise (noisy 20.png,txt). Use values of $\eta = 1$, $\beta = 1$, $B = 100$, and $S = 1000$. On each iteration of your algorithm, compute the energy $E(\boldsymbol{y}, \boldsymbol{x})$ for the current sample of $\boldsymbol{y}$ and output it to a log file, keeping track of which values correspond to the burn-in. Run your algorithm with three different initializations - one in which each y_{ij} is initialized to x_{ij} , one in which each y_{ij} is initialized to $-x_{ij}$, and one in which the y_{ij} are set to -1 or $+1$ at random. Plot the energy of the model as a function of the iteration number for all three chains and visually inspect these traces for signs of convergence. Do all three seem to be converging to the same general region of the posterior, or are some obviously suboptimal? Does the burn-in seem to be adequate in length? Is there substantial fluctuation from iteration to iteration, indicating that the chain is mixing well, or does it become stuck at particular energies for several iterations at a time?

Answers:

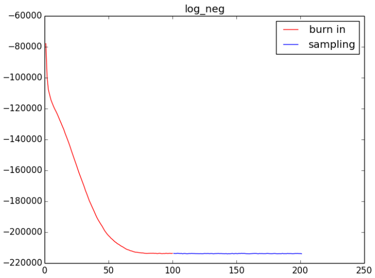


Figure 2: Initialize with $\boldsymbol{y} = -\boldsymbol{x}$

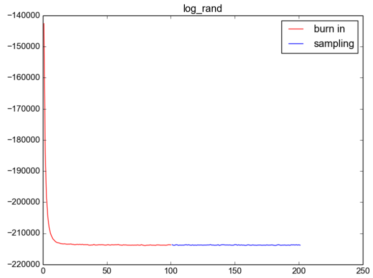


Figure 3: Initialize with random values

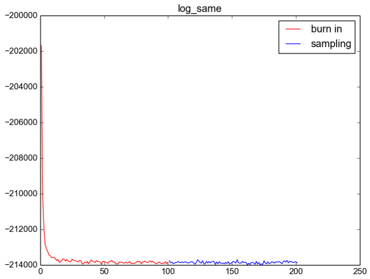


Figure 4: Initialize with $\boldsymbol{y} = \boldsymbol{x}$

It seems that the burn-in is adequate, that they are all converging to the same-ish posteriour value, and that there is good mixing.

(d) **[10 points]** Have your program output a restored image after completing its sampling iterations, by thresholding the estimated posterior probabilities for the y_{ij} variables at 0.5 - i.e., by estimating the "true" color of each pixel (i, j) as:

$$\hat{y}_{ij} = \begin{cases} +1 & \text{if } p(y_{ij} = 1|\boldsymbol{x}) > 0.5 \\ -1 & \text{otherwise} \end{cases}$$

To estimate the required posterior probabilities, store a running count c_{ij} of the number of (retained) samples for which each $y_{ij} = 1$, and then use the Monte Carlo estimate:

$$p(y_{ij} = 1|\boldsymbol{x}) \approx \frac{1}{S} \sum_t 1(y_{ij}^{(t)} = 1) = \frac{1}{S} c_{ij} \tag{5}$$

where $y_{ij}^{(t)}$ represents the t^{th} sample of y_{ij}. Restore both the 10% - and 20% -noise images in this way, using the same values of η, β, B, and S as above. Evaluate the quality of the restoration by computing the fraction of all pixels that differ between the restored images and the original image. Prepare a figure for each the the two images, showing the original, the noisy version, and the restoration side by side.

Answers:

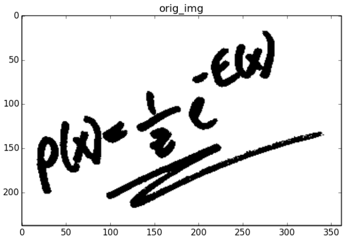


Figure 5: Original image

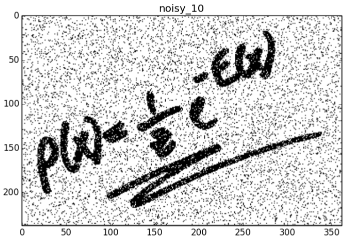


Figure 6: Noisy 10 image

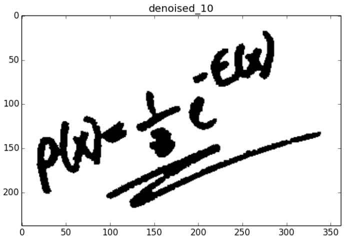


Figure 7: Denoised

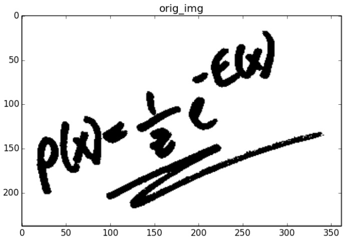


Figure 8: Original image

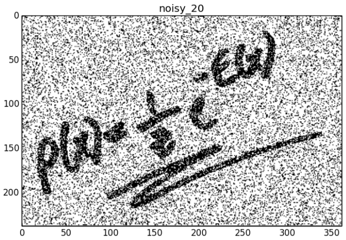


Figure 9: Noisy 20 image

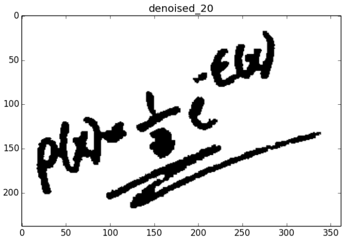


Figure 10: denoised image

(e) **[10 points]** If you have implemented your algorithm correctly, your restored images should be quite close to the original. But is this because you have a clever algorithm or just because the problem is easy? To examine this question, implement a trivial reconstruction algorithm that sets each y_{ij} equal to the consensus (majority) of its neighbors (including x_{ij}), and iterates a few times until convergence (use sequential rather than batch updates, as in Gibbs sampling. This algorithm need not converge in theory, but quickly do quite often in practice. To be safe, you can force it to terminate after, say, 30 iterations.) You should be able to get this program working quickly and easily by reusing code from your Gibbs sampler. However, note that in this case, you should not average over samples (there are no samples here) but instead should use the final value of the y_{ij} variables for your restored image. Run this program on both images and compute its restoration error. Include figures for the images restored in this way. Does the Gibbs sampler do better than the trivial algorithm? Why or why not?

Answers:

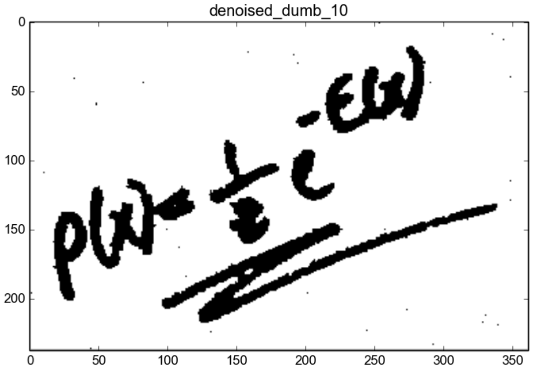


Figure 11: Naive denoising for noisy_10

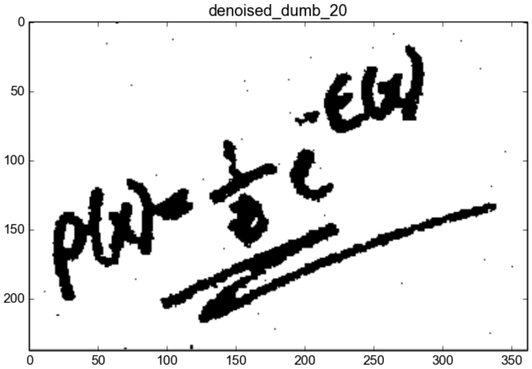


Figure 12: Naive denoising for noisy_20

The naive technique performs much more poorly. (And if not initiated to $X = Y$, it performs abominably).

(f) **[10 points]** While the Gibbs sampler is useful for obtaining marginal posterior probabilities of interest, much of its appeal derives from its flexibility in estimating posterior distributions for more complex features of the model. To get a sense for its flexibility, use your Gibbs sampler to estimate the posterior distribution over the number of pixels in the "Z" in the image, which approximately falls in the rectangle from $(i = 125, j = 143)$ to $(i = 162, j = 174)$. Using the same parameters as above, simply count the number of cases of $y_{ij} = +1$ within this rectangle for each retained sample, output one count per iteration as your sampler runs, then use their relative frequencies as an estimate of the posterior distribution of interest. Plot a histogram showing these relative frequencies for both images and comment on any differences between the two estimated posterior distributions.

Answers:

The distributions are very similar, which suggests that the denoising technique is pretty good at recovering the original distribution. However, noisy_20 indicates a higher probability of black pixels. (I.e. the mean of the PDF for black pixels is higher than it is for noisy_10.)

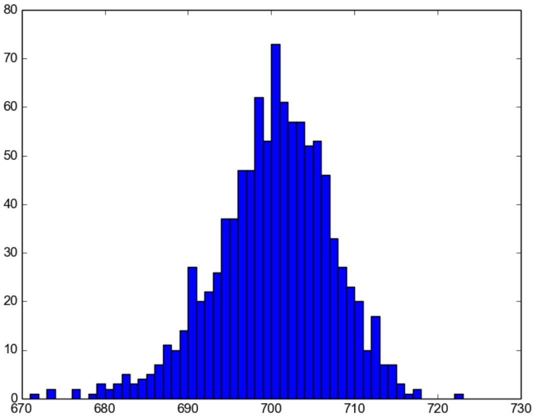


Figure 13: Frequency vs Counts for 10%

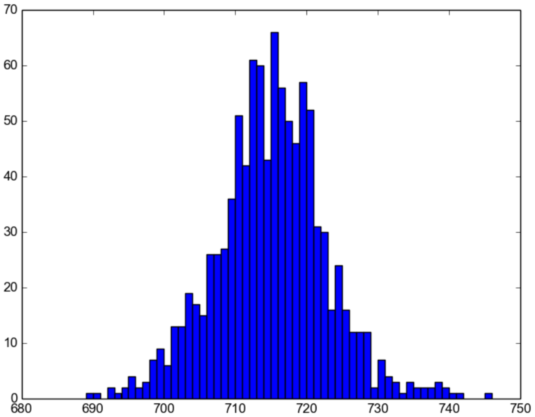


Figure 14: Frequency vs Counts for 20%