Homework 5

Deadline: December 12, 15:15

Exercise 1: Gibbs Sampling (50%)

1. Derive the Conditional Distributions

1. For the Gaussian model:

$$y_i \sim \mathcal{N}(\mu, \tau^{-1}),$$

where $\mu \sim \mathcal{N}(0, \omega^{-1})$ and $\tau \sim \text{Gamma}(\alpha, \beta)$. Prove the following:

$$\mu | y, \tau \sim \mathcal{N}\left(\frac{\tau}{n\tau + \omega} \sum y_i, \frac{1}{n\tau + \omega}\right)$$

$$\tau | y, \mu \sim \text{Gamma}\left(\alpha + \frac{n}{2}, \beta + \frac{1}{2}\sum (y_i - \mu)^2\right).$$

2. For the Poisson model:

$$y_i \sim \text{Poisson}(\mu),$$

where $\mu \sim \text{Gamma}(2, \beta)$ and $\beta \sim \text{Exponential}(1)$. Prove the following:

$$\mu | y, \beta \sim \text{Gamma}(2 + \sum_i y_i, n + \beta)$$

$$\beta | y, \mu \sim \text{Gamma}(3, 1 + \mu).$$

2. Implement Gibbs Sampler

Q1 Gaussian Model:

- Use the Python code to implement Gibbs sampling for the Gaussian model.
- Simulate $n = 100$ observations from $y_i \sim \mathcal{N}(5, 2^{-1})$.
- Plot traceplots and histograms for μ and τ^{-1} (variance). Compare with true values.

Q2 Poisson Model:

- Extend the Gibbs sampler to the Poisson model with $n = 100$ observations from $y_i \sim \text{Poisson}(5)$.
- Plot traceplots and histograms for μ and β. Compare with theoretical expectations.

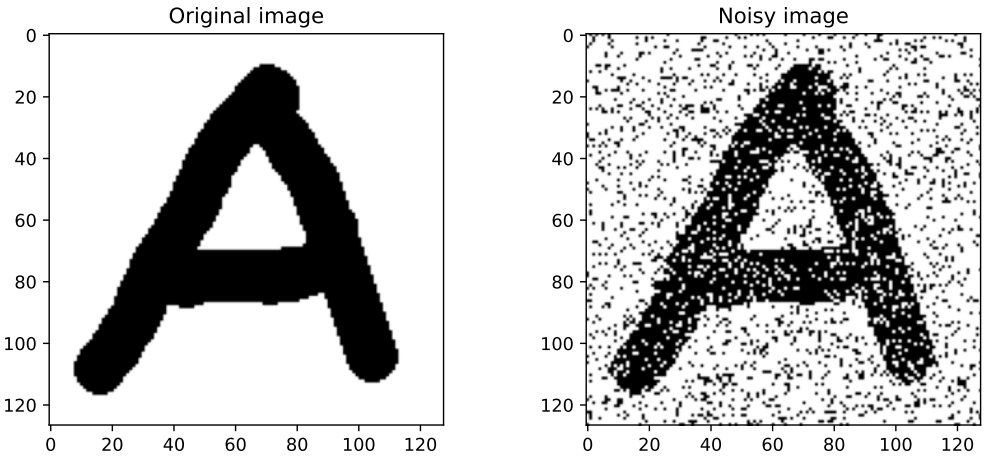

Figure 1: Original vs noisy image

Exercise 2

One application of Markov Random Fields (MRFs) is to denoise noisy images. In this task, we aim to minimize the following energy function using **Gibbs Sampling** and **Mean Field Approximation**:

$$E(x, y) = h \sum_i x_i - \beta \sum_{i,j} x_i x_j - \eta \sum_i x_i y_i,$$

where:

- $x_i \in \{-1, +1\}$: Binary state of pixel i in the denoised image,
- $y_i \in \{-1, +1\}$: Binary state of pixel i in the noisy observed image,
- h, β, and η: Scalar parameters controlling the trade-offs between the terms.

You can download the noisy image for this exercise from the following link. The image is stored in a CSV file format. You can read it into Python as follows:

```
import pandas as pd
img = pd.read_csv('letterA.csv').to_numpy()
```

Algorithms

Gibbs Sampling

1. **Initialization**: Set $x_i = y_i$ for all i.
2. For each pixel $j = 1, \ldots, N$:

- Compute the conditional probability:

$$P(x_j = +1 \mid x_{\setminus j}, y) \propto \exp\left(-E(x, y)\right),$$

  where $x_{\setminus j}$ denotes all pixel states except x_j.

- Sample x_j from the distribution:

$$P(x_j \mid x_{\setminus j}, y).$$

3. Repeat for multiple iterations or until convergence.

Mean Field Approximation

Now, we try to apply mean-field approximation to denoise the image. The posterior for the Ising model has the following form:

$$P(x \mid y) = \frac{1}{Z} \exp(-E(x)),$$

where:

$$E(x) = E_0(x) - \sum_i L_i(x_i), \quad E_0(x) = -\beta \sum_{i,j} x_i x_j, \quad L_i(x_i) = \eta \sum_i x_i y_i.$$

In mean-field approximation, we try to approximate this posterior by a fully factored approximation:

$$q(x) = \prod_i q(x_i; \mu_i),$$

where μ_i is the mean value of node i. To derive the update for the variational parameter μ_i, we first write out:

$$\log \tilde{p}(x) = -E(x),$$

dropping terms that do not involve x_i:

$$\log \tilde{p}(x) = \beta x_i \sum_j x_j + L_i(x_i) + \text{const.}$$

This only depends on the states of the neighboring nodes. Now we take expectations of this with respect to $\prod_{j \neq i} q(x_j)$ to get:

$$q(x_i) \propto \exp\left(x_i \sum_j \beta \mu_j + L_i(x_i) \right).$$

Thus, we replace the states of the neighbors by their average values. Let:

$$m_i = \sum_j \beta \mu_j,$$

be the mean field influence on node i. Also, let $L_i^+ = L_i(1)$ and $L_i^- = L_i(-1)$. The approximate marginal posterior is given by:

$$q_i(x_i = 1) = \frac{\exp(m_i + L_i^+)}{\exp(m_i + L_i^+) + \exp(-m_i + L_i^-)} = \sigma(2a_i),$$

where:

$$a_i = m_i + 0.5(L_i^+ + L_i^-).$$

Similarly, we have $q_i(x_i = -1) = \sigma(-2a_i)$. From this, we can compute the new mean for site i:

$$\mu_i = \mathbb{E}_{q_i}[x_i] = \tanh(a_i).$$

Hence, the update equation becomes:

$$\mu_i = \tanh\left(\sum_j \beta\mu_j + 0.5(L_i^+ - L_i^-)\right).$$

It is usually better to use damped updates of the form:

$$\mu_i^t = (1-\lambda)\mu_i^{t-1} + \lambda \tanh\left(\sum_j \beta\mu_j^{t-1} + 0.5(L_i^+ - L_i^-)\right), \quad 0 < \lambda < 1.$$

Tasks

Q1 Implement Gibbs Sampling for Image Denoising:

- Simulate noisy images with varying noise levels (e.g., Gaussian noise or salt-and-pepper noise).
- Implement Gibbs Sampling to minimize the energy function.
- Apply the algorithm to the noisy image and produce a denoised image.
- Plot the denoised image after iterations from the following list: $[0, 1, 5, 15, 50]$.

Q2 Implement Mean Field Approximation for Image Denoising:

- Implement the mean-field approximation using the update equations provided.
- Apply the algorithm to the noisy image and produce a denoised image.
- Plot the denoised image after iterations from the following list: $[0, 1, 5, 15, 50]$.

Q3 Noise and Performance Analysis:

- Compute and plot the **Normalized Mean Squared Error (NMSE)** between the denoised image (D) and the original ground truth image (G) for each noise level:

$$\text{NMSE} = \frac{\|D - G\|^2}{\|G\|^2}.$$

- Compare the performance of Gibbs Sampling and Mean Field Approximation as the noise level increases.

Q4 Comparison of Results:

- Evaluate the trade-offs between the methods, considering:
  - Convergence speed,
  - Final NMSE values,
  - Visual quality of the denoised image.

Q5 Parameter Sensitivity Analysis:

- Vary the parameters β, and η in the energy function and evaluate their impact on denoising performance for both algorithms.
- For each parameter, plot:
  - NMSE vs β,
  - NMSE vs η.
- Discuss and justify how each parameter influences the trade-offs between pixel similarity, smoothness, and adherence to the noisy image.