Image denoising by MCMC

Using Markov Chain Monte Carlo in conjunction with an Ising model to clean up a noisy binary image

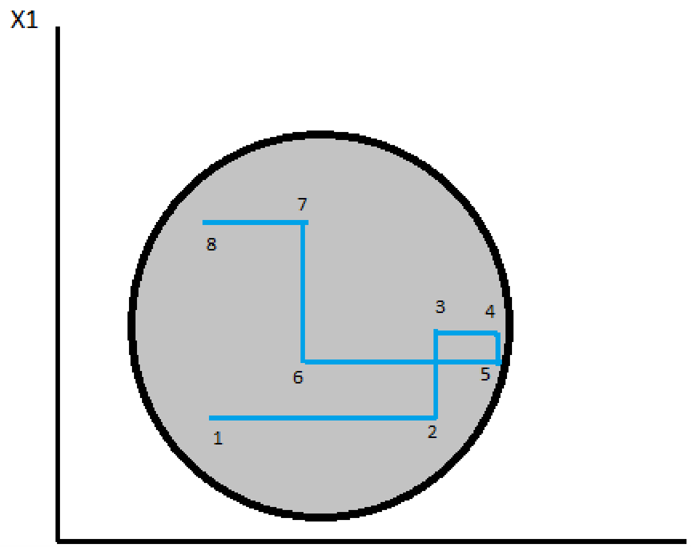
Pranav Prathvikumar Follow
Sep 30 · 7 min read ★

In this article, I will be demonstrating the use of Markov Chain Monte Carlo to denoise a binary image.

various techniques comprising MCMC are differentiated from each other based on the method used for drawing samples. Some of the more well known MCMC techniques are Metropolis — Hastings, Gibbs sampling and Hamiltonian Monte Carlo. The technique that I will be using is Gibbs sampling.

Gibbs sampling is a method for sampling from a multivariate distribution given all other variable are kept the same.

For example, if there is distribution with only two variables x1 and x2, the sampling is as follows:



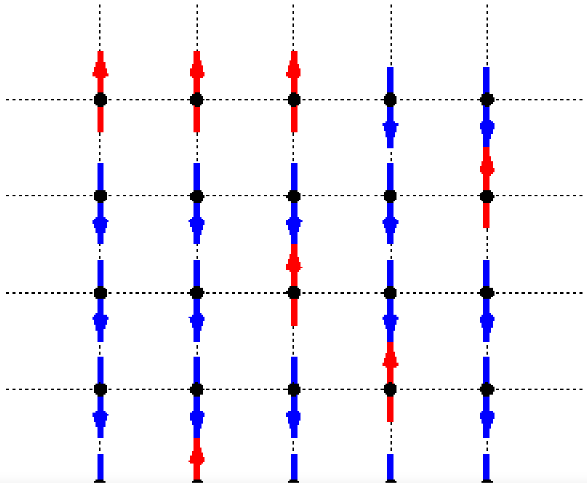
It starts at point 1 and then when the second point is to be sampled, it goes as:

P(x2|x1), it looks for the next sample on the same line as x1 (keeping it constant)

Following which the third point is sampled as follows:

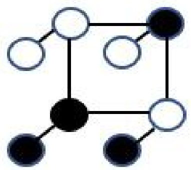
P(x1|x2) , it looks for the next sample on the same line as x2 (keeping it constant)

And so on, the sampling process continues for a set number of points allowing it to traverse the entire space.



The Ising model is a mathematical model corresponding to a square lattice used for modelling phase transitions. Each element in the lattice can exist in two discrete states and can be represented by +1 and -1. Each element exerts an influence on all its neighbouring elements and tries to reach a state of equilibrium where all elements exist in the same state.

Application:



Conceptualized model of image

The binary image can be thought of in the form of a lattice where each pixel represents one element. The pixel's state can be represented as 1 or -1 depending on the colour of the pixel. The image can be imagined consisting of two layers, the underlying layer which represents the true image and the above layer representing the noise. The Gaussian noise is said to be superimposed upon the image with it matching in some places to the actual image and in some places taking the opposite values.

tied to each other, represented by the edge potential. The closer they are tied to each other, the more they try to be in the same state. The formula for the edge potential is given by:

$$\exp(J, X_a, X_n)$$

Here J is the coupling constant which denotes how closely neighbours are tied to each other. X_a represents the pixel under consideration and X_n represents the observed values of its neighbours.

A Gaussian observation model is used to model the relation between the noise and the pre-existing pixels. The noise is said to be a function of the actual underlying pixel value and of the standard deviation from it. It can be represented by:

$$N(Y_a \mid X_a, \sigma^2)$$

Here Y_a represents the observed value of the pixel, X_a the pixel under consideration and σ^2 the standard deviation.

the actual colour of the pixel to the observed colour(noise). The influence of these two forces can be manipulated by changing the values of J and that of σ^2.

Gibbs sampling applied would sample each pixel conditioned on all its neighbours and the underlying true value. It would then fix this value and proceed to the next element and repeat the same operation. One iteration would be completed when it has finished traversing through the entire lattice. Based on the number of iterations, the final image quality can differ.

$$\frac{N(Y_a \mid X_a, \sigma^2) \prod_{n \in nbr(a)} exp(J, X_a, X_n)}{\sum_{X_a} N(Y_a \mid X_a, \sigma^2) \prod_{n \in nbr(a)} exp(J, X_a, X_n)}$$

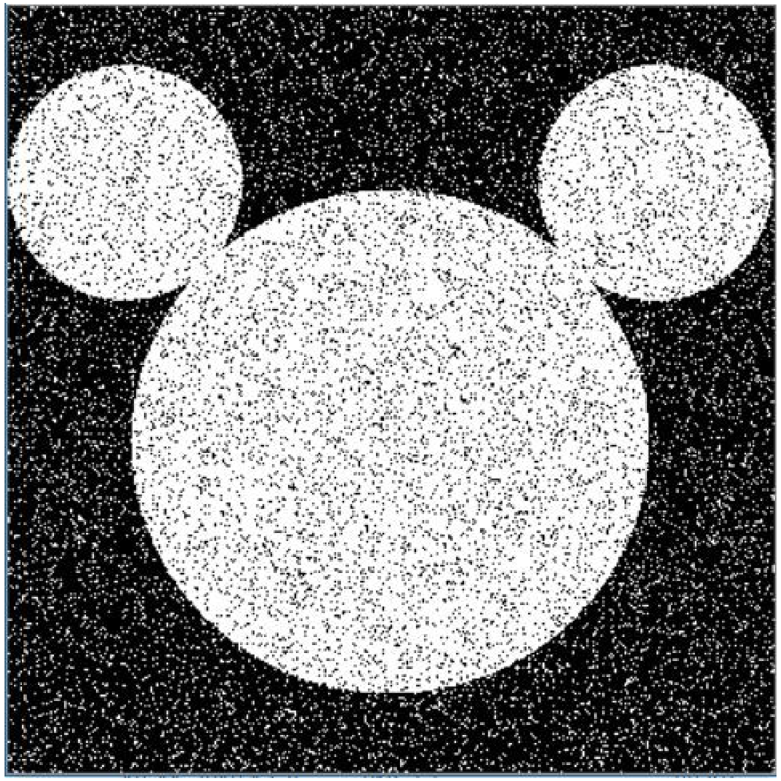
X_a the pixel under consideration can take either the values +1 or -1. The above expression gives the likelihood of X_a taking the values +1 or -1.

The numerator gives the likelihood of X_a to be, for instance, +1, and checking its relationship with the observed value of the pixel Y_a, and with the observed values of its neighbours X_n.

It is divided by the sum of the likelihood of it taking the values +1 and -1 each to give the probability of the actual value of X_a to be +1.

Code:

The purpose of the code is to recover the original image from the corrupted image.



Corrupted image

```
import pandas as pd
from PIL import Image

data = Image.open('noise_img.png')
image = np.asarray(data).astype(np.float32)
```

The image was imported and saved in the form of a 2-D array with the grayscale values of the pixels. This image was converted into an array for ease of operating upon.

It was then converted into an ising model by replacing all 0s(corresponding to black)with -1 and all 255s(corresponding to white) with +1. The array was padded with 0s on all sides to make the task of iterating over the lattice easier.

```
#Convert image values to ising model
for i in range(len(image)):
    for j in range(len(image[0])):
        if image[i,j,:] == 255:
            image[i,j,:] = 1
        else:
            image[i,j,:] = -1

#Create array to perform operations on
ising = np.zeros((len(image)+2,len(image[0])+2))

for i in range(len(image)):
    for j in range(len(image[0])):
        ising[i+1,j+1] = image[i,j,:]
```

2	0	-1	1	-1	-1	-1	-1	-1
3	0	-1	-1	1	-1	-1	-1	-1
4	0	-1	-1	-1	1	-1	-1	-1
5	0	-1	-1	-1	-1	1	-1	-1
6	0	-1	-1	-1	-1	-1	-1	-1
7	0	-1	1	-1	-1	-1	-1	1
8	0	-1	-1	1	-1	-1	-1	-1
9	0	-1	-1	-1	-1	-1	-1	-1
10	0	-1	1	-1	-1	-1	-1	-1
11	0	-1	-1	-1	-1	-1	-1	-1
12	0	-1	1	-1	-1	1	1	-1

Image array

The first row and column are the padding. Rest of the rows and column correspond to colour of one pixel each. The row number and column number correspond to the position of the pixel and the cell value corresponds to the colour.

A value was set for the coupling strength before beginning Gibbs sampling.

```
#Coupling strength
J=4

#Gibbs sampling
for n in range(3):
    for i in range(1,len(ising[0])-1):
        for j in range(1,len(ising)-1):
            pot = []
            for x in [-1, 1]:
```

```
1, cov = 1)*pot[1] + multivariate_normal.pdf(image[j-1,i-1,:], mean =
-1, cov = 1)*pot[0])
            if np.random.uniform() <= prob1:
                ising[j,i] = 1
            else:
                ising[j,i] = -1
```

All the values in the original array excluding the zeros are iterated over. Each point is sampled by checking for the likelihood of the original colour being +1 or -1. The edge potential with its neighbours is calculated for both these cases. Along with the edge potential, the likelihood of the value observed is calculated with respect to the true value being either +1 or -1.

Based on the above two values, the likelihood of the pixel being +1 is calculated by dividing it by the likelihoods of the pixel being +1 and -1.

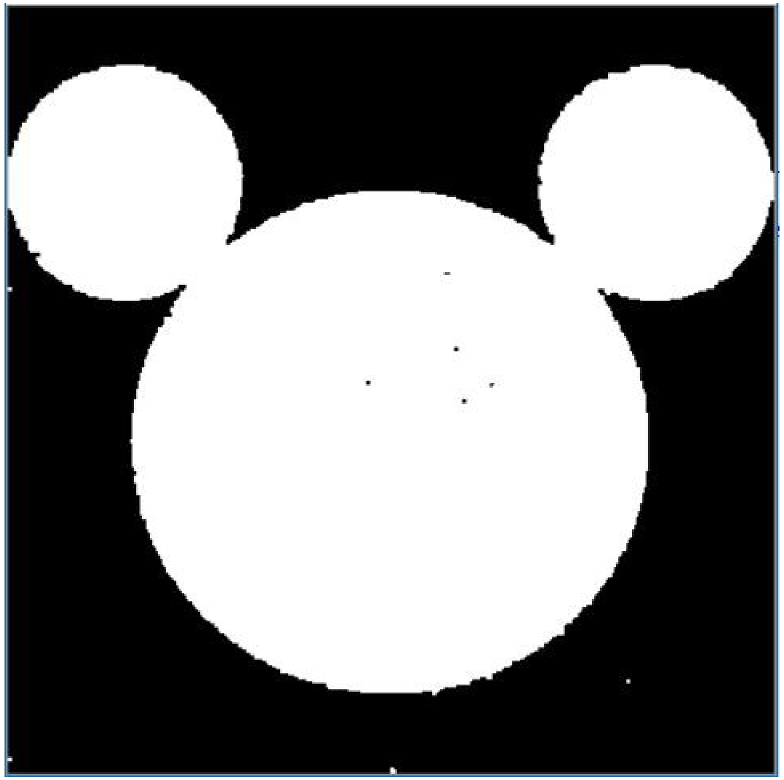
This value is then compared to a value randomly drawn from a standard normal distribution. If the probability of the original value being +1 is higher, then the value is set to +1 otherwise it is set to -1.

When the next element is sampled, it takes the new value of the above element while calculating the edge potential. This code goes through every element of the entire lattice for three times.

```
#Retrieving the final array
```

```
if final[j,i] == 1:
    final[j,i] = 255
else:
    final[j,i] = 0
```

The padding is then removed and then the values are converted back to grayscale values and then visualised.



Cleaned image

You can connect with me on LinkedIn as well.

Machine Learning Artificial Intelligence Data Science Towards Data Science