MARKOV CHAIN MONTE CARLO: EXPOSITION AND APPLICATIONS

FINAL PROJECT REPORT FOR MATH 697AM

Long Le
Department of Mathematics and Statistics
University of Massachusetts Amherst

May 3, 2019

ABSTRACT

This is a final project report for *Math 697AM: Applied Math and Modeling* at UMass Amherst with Prof. *Yao Li*. In this report, we introduce the necessary mathematical background needed for and the property of Markov Chain Monte Carlo (MCMC). We will also introduce two MCMC methods: **Metropolis-Hastings** algorithm and **Gibbs Sampler**. We then focus on the **Ising problem** and its two applications in physics and image reconstruction.

***K*eywords** MCMC · Metropolis-Hastings · Gibbs Sampler · Ising model · Image denoising.

1 Preliminary

In this section, we will introduce the basic theory of a generic Markov chain in discrete time and discrete space needed for subsequent discussion. Readers who are familiar with the topic can freely skip this section.

First, we introduce the most important property (as far as we are concerned) assumed of a stochastic process.

Definition 1 (Markov property)**.** *A sequence of r.vs $\{X_t\}_{t\in\mathbb{N}}$ taking values in a common state space E is said to have Markov property if and only if*

$$\mathbb{P}[X_{n+1}|X_1, ..., X_n] = \mathbb{P}[X_{n+1}|X_n]$$

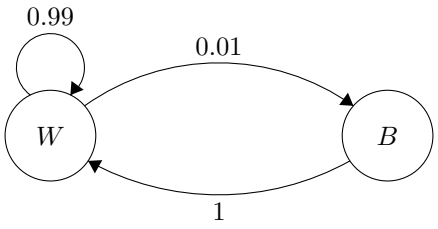
for all $n \in \mathbb{N}$. [1] *In addition, if we have*

$$\mathbb{P}[X_{n+1} = j|X_n = i] = p_{ij}$$

*for all $n \in \mathbb{N}$ then we say that the chain is **time-homogeneous**.*

If the Markov chain $\{X_t\}$ is time homogeneous, then we usually put each probability $\mathbb{P}[X_{n+1} = j|X_n = i] = p_{ij}$ as an entry into a transition matrix P and write $X_t \sim Markov(\lambda, P)$ with some initial distribution vector λ.

Example 1 (Machine failure,[Bertsekas and Tsitsiklis(2002)])**.** *A machine can either be working or broken down in any given day. If it is working, it can break down with probability* 0.01*. If it is broken down, it will be repaired and work on the next day with probability* 1*.*



[1] Here we use $\mathbb{P}[X_n]$ to denote $\mathbb{P}[X_n = x]$ for some $x \in E$, whose value we are not particularly interested in.

Assume that we start with a working machine, then the initial distribution is $\lambda = (1, 0)$. The transition matrix is

$$P = \begin{matrix} \\ W \\ B \end{matrix} \begin{matrix} \begin{matrix} W & B \end{matrix} \\ \begin{pmatrix} 0.99 & 0.01 \\ 1 & 0 \end{pmatrix} \end{matrix}$$

In sampling and other applications, we are usually concerned about the long-term dynamics of a chain. Thus, we will introduce this rather natural notion.

Definition 2 (Stationary distribution)**.** *A probability distribution π (row vector) is stationary with respect to P if and only if*

$$\pi P = \pi$$

The readers should note these two important fact

1. If our chain starts in stationarity (*i.e.* $\lambda \sim \pi$) then $X_t \sim \pi$ for all $t \in \mathbb{N}$.
2. Under some technical conditions (positively recurrent and aperiodic for our discrete case), the chain has unique stationary π and will converge to π.

Fact 1 is rather by matter of definition. Fact 2, however, is much richer and somewhat surprising. Indeed, this fact would allow us to sample from a chain X_t with stationary π and pretend that we are sampling from π itself.

Finally, we will introduce the last property that would allow us to construct a chain with certain desired stationary π.

Definition 3 (Detailed balance)**.** $X_t \sim Markov(\lambda, P)$ *is said to be in detailed balance with distribution μ iff*

$$\mu_i P_{i,j} = \mu_j P_{j,i}.$$

If X_t starts in μ, then we can " run time backwards".

$$\mathbb{P}[X_0 = x_0, ..., X_n = x_n] = \mathbb{P}[X_0 = x_n, ..., X_n = x_0] \quad \textit{(time reversibility)}$$

Detailed balance in the context of chemical reaction says that the forward reaction is balanced by the backward reaction so there is no net chemical concentration change. The intuition here is pretty much the same. We can think of μ_i as the concentration of the chemical complex i and P_{ij} as the reaction propensity k going from i to j. Then, the product $\mu_i P_{i,j}$ is just the reaction rate for the forward reaction. Surprisingly, detailed balance also means equilibrium as it does in chemical context.

Lemma 1 (Reversibility implies stationarity)**.** *If P is in detailed balance with μ then μ is the stationary distribution for $X_t \sim Markov(\lambda, P)$.*

Proof.

$$(\lambda P)_i := \sum_{j \in \Omega} \lambda_j P_{ji} = \sum_{j \in \Omega} \lambda_i P_{ij} = \lambda_i (\sum_{j \in \Omega} P_{ij}) = \lambda_i.$$

□

For a more detail treatment of Markov chain, we refer the reader to [Norris(1997)].

2 Markov Chain Monte Carlo (MCMC)

2.1 Problem

Suppose that we want to estimate

$$\int_{\mathbb{R}^d} f(\mathbf{x}) \pi(\mathbf{x}) \, d\mathbf{x}$$

for some function f and density π. If the integral is too hard to solve analytically, which is often the case, then the two general approaches to tackle the problem are

1. Quadrature
2. Monte-Carlo

Unfortunately, numerical integration does not tend to work on high dimensions and it also requires f, π to be smooth. The alternative, Monte Carlo, is the main topic of this report.

In Monte Carlo simulation, we would use the empirical distribution to estimate the underlying density π. Specifically, we Sample $X_1, ..., X_N \overset{i.i.d}{\sim} \pi$ and take the estimate

$$\overline{f}_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Although this estimator is not necessarily unbiased, it is consistent. By the law of large number, we have the following almost surely convergence

$$\mathbb{P}[\lim_{N\to\infty} \overline{f}_N = \mathbb{E}[f(X_i)] = \int f(x)p(x)\,dx] = 1$$

One caveat is that we will often sample from some distribution other than π as we usually only know π up to some normalizing constant Z that is expensive to compute. Indeed, the two algorithms that will be introduced shortly do not compute Z.

There are many Monte Carlo algorithms, including rejection sampling and importance sampling. Markov chain Monte Carlo is another class of algorithms. The idea is to devise a chain with stationary distribution π and then take $X_1, ..., X_n$ samples obtained from simulating the chain. We have two requirement

1. **Correctness**: Chain has to be ergodic with π. That is π has to be the stationary distribution and the chain has to converge to π.
2. **Efficient**: Compute the next state of the chain has to be efficient.

2.2 Error bound of MCMC

In this section, we quickly estimate the variance of the estimator $\overline{f}_N$ obtained from $X_1, ..., X_N$ samples from the chain $X_t \sim Markov(\pi, P)$ starting in stationarity. Recall from basic probability that

$$Cov(X_1 + X_2 + ... + X_n) = \sum_{i,j} Cov(X_i, X_j).$$

where $Cov(X_i, X_i) = X_i$. So

$$Cov(X_1 + X_2 + ... + X_n) = \sum_{i} Var(X_i) + 2\sum_{i\neq j} Cov(X_i, X_j).$$

We can greatly simplify this expression. But before we proceed, let us introduce some more terminology.

Definition 4 (Weakly (covariance) stationary process)**.** *X_t is weakly stationary (WWS) process if and only if*

1. $\mathbb{E}(x_t) = \mu$ *for all* t
2. $Cov(x_t, x_s) = \sigma(t - s)$
3. $Var(x_t) = \sigma(0) < \infty$

For comparison of nomenclature, because $\{X_t\} \sim Markov(\pi, P)$, we know that $X_n \sim \pi$ for all $n \in \mathbb{N}$. This is known as strict stationarity.

Lemma 2. *Let $\{X_t\} \sim Markov(\pi, P)$ in the state space E. We claim that*

$$Cov(X_t, X_{t+h}) = Cov(X_0, X_h)$$

for all $t, h \in \mathbb{N}$.

Proof. Observe that

$$\mathbb{E}[X_t X_{t+h}] = \sum_{i\in E} \mathbb{P}[X_t = i]\mathbb{E}[X_t X_{t+h}|X_t = i]$$

$$= \sum_{i\in E} \pi_i \times \mathbb{E}[iX_{t+h}|X_t = i] = \sum_{i\in E} \pi_i \times (\sum_{j\in E} i \times j \times \mathbb{P}[X_{t+h} = j|X_t = i])$$

Now by the Markov property, we have

$$\mathbb{E}[X_t X_{t+h}] = \sum_{i \in E} \pi_i \times \Big(\sum_{j \in E} i \times j \times \mathbb{P}[X_h = j | X_0 = i] \Big) = \mathbb{E}[X_0 X_h].$$

Also, by strict stationarity we have $X_t \sim \pi$ for all $t \in \mathbb{N}$. Then $\mathbb{E}[X_t] = \mathbb{E}[X_{t+h}] = \mu$ for some μ. Note that

$$\begin{aligned} Cov(X_t, X_{t+h}) &= \mathbb{E}[X_t X_{t+h}] - \mathbb{E}[X_t]\mathbb{E}[X_{t+h}] \\ &= \mathbb{E}[X_0 X_h] - \mu^2 = Cov(X_0, X_h). \end{aligned}$$

□

With lemma 2, we now also assume condition 1 and 3 in definition 4 for our chain X_t so that it is weakly stationary. Then, we can give an estimate for the variance of the estimator as follows

$$\begin{aligned} Var(X_1 + X_2 + ... + X_n) &= nVar(X_1) + 2 \sum_{i \neq j} Cov(X_i, X_j) \\ &= nVar(X_1) + 2 \sum_{i=1}^{n-1} (n-i) Cov(X_1, X_i) \\ &= nVar(X_1) + 2 \sum_{i=1}^{n-1} (n-i) \sigma(i). \end{aligned} \tag{1}$$

The right-most term of equation (1) is summing over the off-main-diagonal entries of the covariance matrix.

$$\Sigma = \begin{pmatrix} ☺ & ☿ & ☻ \\ ☿ & ☺ & ☿ \\ ☻ & ☿ & ☺ \end{pmatrix}$$

Now note that $\sum_{i=1}^{n-1} (n-i)\sigma(i)$ is a Cesaro sum so if we assume absolute convergence

$$\sum_{i=1}^{\infty} |\sigma(i)| < \infty$$

then

$$\lim_{n \to \infty} \sum_{i=1}^{n-1} \frac{n-i}{n} \sigma(i) = \sum_{i=1}^{\infty} \sigma(i).$$

It follows that

$$Var\left(\frac{X_1 + ... + X_n}{n^2} \right) = \frac{1}{n} \left(Var(X_1) + \sum_{i=1}^{n-1} \frac{n-i}{n} \sigma(i) \right) \to \frac{1}{n} \sigma_\infty^2.$$

σ_∞^2 is known as the asymptotic variance. The important thing to note here is that the error decays with sample size like $1/n$. Note that our analysis in this section relies on the assumption that X_t starts in stationarity. This will generally not be the case for our algorithms so in practice it is common to discard some initial portion of the MCMC samples (burn-in).

2.3 Mixing time

Let us recall that a generic MCMC would aim to construct a Markov chain X_t with some desired stationary π. In general, there are many different X_t's that would all produce the desired π. However, most of these chains will not start in stationarity. This is bad because all we care about is being able to draw random samples from π. Thus, the sooner a chain converges to π the better. This section would allow us to formulate these ideas more precisely.

In order to talk about "convergence", we explain what it means for two distributions to be close to each other.

Definition 5 (Total variation)**.** *The total variation distance between two distributions μ and ν is*

$$||\mu - \nu||_{TV} = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)|$$

where $\mathcal{F}$ is a sigma-algebra, representing all possible events that can happen.

Thus, μ is "close" to ν if $||\mu - \nu||_{TV}$ is small. Intuitively, $||\mu - \nu||_{TV} = Area(I) = Area(II)$.

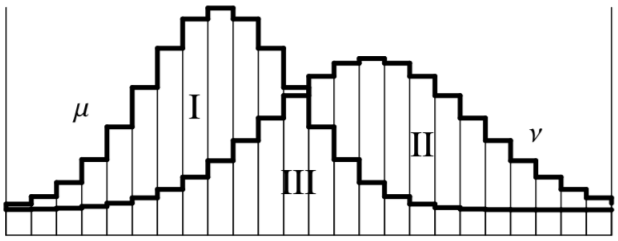


Figure 1: (the picture is borrowed from [Levin et al.(2006)Levin, Peres, and Wilmer])

Let $X_t \sim Markov(\lambda, P)$ with stationary π. At each time step n in our simulation, we can ask what is the "maximal distance" from our current marginal distribution X_n from the stationary distribution π.

Definition 6 (Maximal distance)**.** *The maximal distance at time t is defined as*

$$d(t) := \max_{x \in \Omega} ||\delta_x P^t - \pi||_{TV}$$

It can be shown that an equivalent formulation to the definition above involves considering the maximal total variance at time t regardless of initial distributions. To be precise, we claim that

$$\max_{x \in E} ||\delta_x P^t - \pi||_{TV} = \sup_{\lambda \in \Lambda} ||\lambda P^t - \pi||_{TV}.$$

with Λ being the space of all possible initial distributions on the state space E. So $d(t)$ is a conservative (pessimistic) estimate for all possible initial λ. This makes sense in the context of MCMC when we usually just choose λ randomly.

Now, we will define the metrics for MCMC performance.

Definition 7 (Mixing time)**.**

$$t_{mix}(\epsilon) := \inf\{t : d(t) \leq \epsilon\}.$$

In other words, mixing time is the time required (or the first time) that the distance to stationarity drops below ϵ. By convention,

$$t_{mix} = t_{mix}(1/4).$$

This theoretical metrics can be upper-bounded by various methods, including using bottle-neck ratio. See for example 7.2 of [Levin et al.(2006)Levin, Peres, and Wilmer].

3 Metropolis-Hastings Algorithm

Now, we finally come to the central question: how do we actually generate a chain with a desired stationary distribution π?

We want to find matrix P that admits a given π as its stationary distribution. We do not know P so we just pick a random transition matrix Ψ. Now, we just sample from Ψ. But the idea is that instead of just blindly following Ψ we design some acceptance probability a, where we only follow the policy from Ψ with probability a, such that we get the desired stationarity. The generic pseudocode for Metropolis algorithm is

Pick a random initial λ and pick $X_0 \sim \lambda$;
Given current $X_t = x$, pick a *proposal* $Y_{t+1} = y \sim \Psi$;
Given $a(x, y)$, and Bernoulli $B(p = a(x, y))$
if B *lands head* **then**
    Accept the proposal ;
    Set $X_{t+1} = y$.;
else
    Reject the proposal ;
    Stay at the same state. Set $X_{t+1} = x$.;
end

Algorithm 1: Metropolis algorithm

The effective probability is

$$P(x, y) = \Psi(x, y)a(x, y)$$

pick y by Ψ and then actually going to y with probability $a(x, y)$. The straight-forward way to ensure that P has stationary π is to demand a stronger condition: **detailed balance**.

$$\pi(x)\Psi(x, y)a(x, y) = \pi(y)\Psi(y, x)a(y, x).$$

Imposing the condition that Ψ is symmetry simplifies the equation to

$$\pi(x)a(x, y) = \pi(y)a(y, x). \tag{2}$$

We pick

$$a(x, y) = \min\left\{\frac{\pi(y)}{\pi(x)}, 1\right\}$$

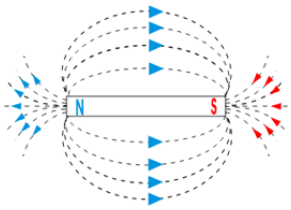
satisfying (2). Note that $a(x, y)$ is a ratio, which is good because we often can only compute π up to a normalizing constant.

4 Ising Model and Gibbs Sampler

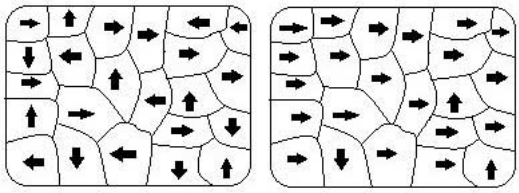
Here, we will first-hand witness MCMC in application. In fact, the early models of MCMC were introduced to simulate a famous physics problem in magnetic known as the Ising model. Before we move to introduce the model, we would like to offer an apology to the physicists for the abhor able treatment of the subject that we are about to commit.

4.1 Ferromagnetic

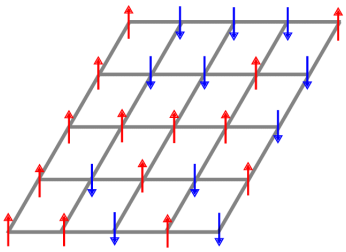
For lack of training in physics, instead of trying to explain the signifcance of ferromagnetic, we will instead throw in a picture of a magnet.



We can partition a piece of ferromagnetic material (like iron) into small regions known as magnetic domains, where the magnetic dipoles of each atom have roughly the same orientation.



Now, let talk about mathematics. For the purpose of modeling, we would consider a piece of materials as some adjacent vertices in the integer lattice $\mathbb{Z}^2$. For each vertex $v \in \mathbb{Z}^2$, we only allow the magnetic dipole to be pointing up or down, corresponding to one of the labels $\{-1, 1\}$.



So generally, this is a coloring problem.

General framework: Let V be a set of vertices and S be the set of colors, then a configuration of the system is a map $\sigma : V \mapsto S$. The state space of all possible configurations σ is $\Sigma \subset S^V$. By physics, each configuration σ has certain probability $\pi(\sigma)$ so we can define the probability distribution on Σ

$$\pi : \Sigma \mapsto [0, 1]$$

The Gibbs sampler for this problem is

$i \leftarrow 0$;
Initialize configuration σ randomly and say $\sigma \leftarrow \sigma_0$;
while $i <$ *number of iterations* **do**
  Pick $v \in V$ uniformly ;
  Consider the set $S = \{\sigma' \in \Sigma : \sigma'[v] = \sigma[v]\}$;
  Define the conditional multinomial $Multi_i$ distribution with $\pi(s_i|\sigma_{i-1}) = {}^{\pi(s_i)}/_{\pi(S)}$;
  Sample σ_i from $Multi_i$;
  $\sigma \leftarrow \sigma_i$;
  $i \leftarrow i + 1$;
end

Algorithm 2: Gibbs Sampler

Here, $\sigma'[v] = \sigma[v]$ means that the two configurations σ and σ' are exactly identical expect at vertex v where their coloring might differ. The idea is that at each iteration, instead of considering all possible σ's we only consider those where we perturb a system at one vertex. The conditional distribution $\pi(.|\sigma_i)$is simpler and only requires knowing the marginal distribution π up to some constant. We will now show that our construction is valid.

Theorem 1 (Ergodicity of Gibbs Sampler)**.** *The Markov chain $X_t \sim Markov(., P)$ generated above is ergodic with π. We will show that*

$$\pi P = \pi.$$

Proof. The algorithm described above where we randomly choose v at each step is known as **random scan Gibbs sampler**. We consider a more tamed version of it. This time, we always pick a specific vertex x to flip. The induced transition graph is P_x. We have

$$(\pi P_x)_\sigma = \sum_{\sigma' \in \Omega} \pi(\sigma') P_{\sigma',\sigma} = \sum_{\sigma' \in \Omega; \sigma'[x] = \sigma[x]} \pi(\sigma') P_{\sigma',\sigma} + 0$$

$$= \sum_{\sigma' \in \Omega; \sigma[x] = \sigma'[x]} \pi(\sigma') \frac{\pi(\sigma)}{\sum_{\rho \in \Omega; \rho[x] = \sigma'[x]} \pi(\rho)} = \pi(\sigma) \sum_{\sigma' \in \Omega; \sigma'[x] = \sigma[x]} \frac{1}{\sum_{\rho \in \Omega; \rho[x] = \sigma'[x]} \pi(\rho)}$$

$$= \pi(\sigma).$$

Note that the transition matrix for the random scan is

$$P = \frac{1}{|V|} \sum_{x \in V} P_x.$$

Because π is ergodic with each P_x, we know that π is also ergodic with P at least for finite $|V|$. □

4.2 Distribution for Ising Model

In this section, we will introduce the mysterious π for our Ising model. π is known as the Boltzmann distribution. First, in physics, we can assign certain energy to each configuration. Systems with high energy are unstable and thus are less likely. In Ising, configurations where neighboring vertices agree in terms of spin are more likely. The energy (Hamiltonian) of a configuration σ is

$$\mathcal{H}(\sigma) = - \sum_{v,w \in V, v \sim w} \sigma(v)\sigma(w)$$

where $v \sim w$ means that v and w are neighbors in the graph. An interior point (x, y) in $\mathbb{Z}^2$ will have 4 neighbors at coordinates: $(x, y-1), (x-1, y), (x+1, y), (x, y+1)$. Also, recall that a configuration σ is a coloring assignment of the graph *i.e.* $\sigma : V \mapsto S$ so $\sigma(v) \in \{1, -1\}$. The probability density is

$$\mu(\sigma) = \frac{1}{Z(\beta)} \exp(-\beta \mathcal{H}(\sigma))$$

where the normalizing constant is

$$Z(\beta) = \sum_{\sigma \in \Sigma} \exp(-\beta \mathcal{H}(\sigma)).$$

The constant $\beta \propto T$ captures the effect of temperature on the system. As we will see in the following simulation, as T increases all systems are approximately uniformly distributed. With low T however, systems with agreeing neighbor vertices have higher probability so we will see patches of the same color.

4.3 2-D Ising Model in Python

The following Python code simulates a Ising model on 64×64 grid in $\mathbb{Z}^2$. We use Gibbs sampler to record the evolution of an initially random configuration over 9 time spans.

```
import numpy as np
from numpy.random import rand
from random import choices
import matplotlib.pyplot as plt

'''
Configuration is represented as numpy 2-D NxN array.
'''
def configPlot(f, config, i, N, n_):
    X, Y = np.meshgrid(range(N), range(N))
    sp =  f.add_subplot(3, 3, n_ )
    plt.setp(sp.get_yticklabels(), visible=False)
    plt.setp(sp.get_xticklabels(), visible=False)
    plt.pcolormesh(X, Y, config, cmap=plt.cm.RdBu);
    plt.title('Time=%d'%i); plt.axis('tight')


# gibbs sampler
def gibbs_move(config, beta):
    N = len(config)

    x = np.random.randint(0, N)
    y = np.random.randint(0, N)

    s1 = np.array(config, copy=True)
    s2 = np.array(config, copy=True)
    s1[x,y] = 1
    s2[x,y] = -1

    e1 = calcEnergy(s1)
    e2 = calcEnergy(s2)

    p1 = np.exp(-e1*beta)
    p2 = np.exp(-e2*beta)

    if p1 == float('inf'):
        p1 = 1
        p2 = 0
    elif p2 == float('inf'):
        p2 = 1
        p1 = 0

    else:
        tot = p1 + p2
        p1 = p1/tot
        p2 = p2/tot

    more_prob = np.random.choice([1, 2], p=[p1, p2])

    if more_prob == 1:
        return s1
    else:
```

```
53          return s2
54
55  def calcEnergy(config):
56      '''Energy of a given configuration'''
57      N = len(config)
58
59      energy = 0
60      for i in range(len(config)):
61          for j in range(len(config)):
62              S = config[i,j]
63              nb = config[(i+1)%N, j] + config[i,(j+1)%N] + config[(i-1)%N, j] + config[i,(j-1)%N]
64              energy += -nb*S
65      return energy/4.
66
67  def simulate(N, beta):
68      f = plt.figure(figsize=(15, 15), dpi=80);
69
70      # 2*(RandInt[0,1]) - 1 --> RandInt[-1, 1]
71      config = 2*np.random.randint(2, size=(N,N))-1
72
73      n_iters = 3001
74      configPlot(f, config, 0, N, 1);
75      for i in range(n_iters):
76          config = gibbs_move(config, beta)
77          if i == 1:        configPlot(f, config, i, N, 2);
78          if i == 4:        configPlot(f, config, i, N, 3);
79          if i == 32:       configPlot(f, config, i, N, 4);
80          if i == 100:      configPlot(f, config, i, N, 5);
81          if i == 1000:     configPlot(f, config, i, N, 6);
82          if i == 1500:     configPlot(f, config, i, N, 7);
83          if i == 2000:     configPlot(f, config, i, N, 8);
84          if i == 3000:     configPlot(f, config, i, N, 9);
85
86  beta = (1/(100))
87  N = 64
88
89  simulate(N, beta)
90  plt.show()
```

Here are the outputs for $T \propto 0.2, 10, 200$.

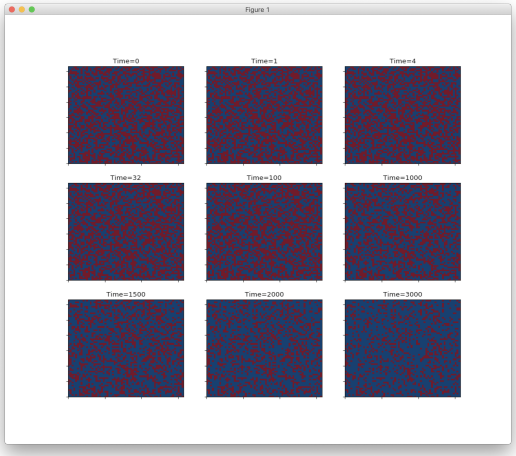


Figure 2: Low temperature

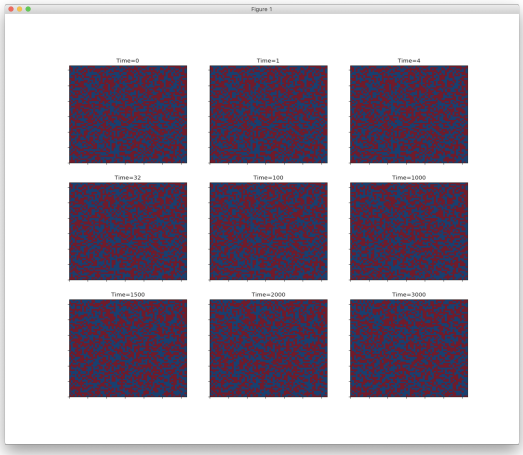


Figure 3: Temperature temperature

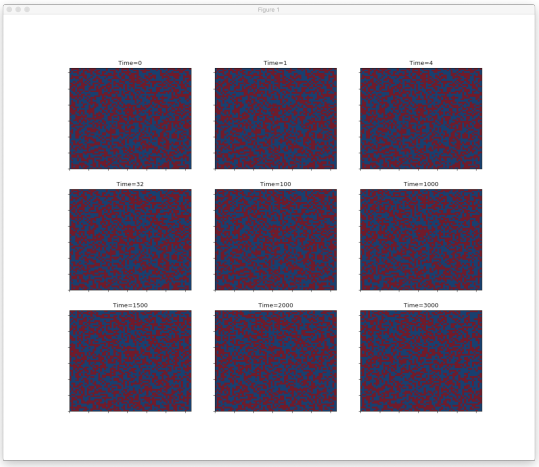


Figure 4: High temperature

The patterns produced are expected. As temperature increases, the color distribution becomes more and more uniform.

5 Image Reconstruction

In the last section, we have already seen an application of Ising model in physics. Now, we will tackle a problem in computer vision discussed at length in [Murphy(2013)].

Problem: Suppose that we have a binary (red/black pixels) $N \times N$ image $Y = (y_i)_{i \in N \times N}$ where $y_i \in \{-1, 1\}$. Y is dirtied with some Gaussian noise so that each pixel in the "evidence" image $X = \{x_i\}$ is drawn from $\mathcal{N}(y_i, \sigma)$ where the fixed variation σ would determine the level of noise. Given X, we would like to reconstruct Y.

This problem can be tackled by Markov random field (MRF) discussed in [Murphy(2013)]. Here, we will assume that the prior Y follows the probability defined in Ising Model and use Gibbs sampling algorithm to reconstruct the

image. Recall that the Ising model assigns more probability to configuration with agreeing neighboring pixels. This is reasonable assumption for a clean image. Then, using MRF terminology, given a configuration σ we define the clique factor as

$$\psi(i,j) = \exp(J\sigma(i)\sigma(j))$$

for $i, j \in V$ and J is the connection strength between the two vertices. For simulation, we set $J = 1$ for neighboring pairs and $J = 0$ otherwise. Then assuming the Markov property that the pixel of a given vertex only depends on its neighboring nodes, the *prior* density over a configuration factors as follows

$$\mathbb{P}(Y = \sigma) = \mathbb{P}(y_1, ..., y_n) = \prod_{C \in Cl(G)} \psi_C(\sigma(C))$$

$$= \frac{1}{Z} \prod_{(i,j)\in V} \psi(i,j) = \frac{1}{Z} \prod_{(i,j)\in V} \exp(J\sigma(i)\sigma(j)) = \frac{1}{Z} \exp(J \sum_{i\sim j} \sigma(i)\sigma(j)).$$

for some normalizing constant Z. The data likelihood function, as assumed, is Gaussian

$$f_{y_i}(x_i) \propto \exp(-(y_i - x_i)^2/(2\sigma^2)).$$

At every step of the Gibbs sampling algorithm, we fix all the prior configuration all but one node i and consider the posterior probability of y_i given the current iterative configuration σ. Let Y_{-i} denote σ assignment everywhere in V but i. We have

$$\mathbb{P}(y_i = +1 | Y_{-1}, X, J) = \frac{\mathbb{P}(y_i = +1, Y_{-1}, X, J)}{\mathbb{P}(y_i = -1, Y_{-1}, X, J) + \mathbb{P}(y_i = -1, Y_{-1}, X, J)}.$$

The joint probability is calculated as follows

$$\mathbb{P}(y_i = \alpha, Y_{-1}, X, J) = \mathbb{P}(y_i = \alpha, Y_{-1}|J)\mathbb{P}(x_i|y_i = \alpha)$$

$$\propto \exp(J \sum_{j\sim i} \sigma(i)\sigma(j)) \exp(-(\alpha - x_i)^2/(2\sigma^2)) = \exp(J\alpha \sum_{j\sim i} \sigma(j)) \exp(-(\alpha - x_i)^2/(2\sigma^2)).$$

For nicer visual effect, we usually denote $\exp(-(y_i - x_i)^2/(2\sigma^2)) = \psi_i(\alpha)$. It follows that

$$\mathbb{P}(y_i = +1 | Y_{-1}, X, J) = \frac{\exp(J \sum_{j\sim i} \sigma(j))\psi_i(+1)}{\exp(J \sum_{j\sim i} \sigma(j))\psi_i(+1) + \exp(-J \sum_{j\sim i} \sigma(j))\psi_i(-1)}$$

$$= sigmoid(2J \sum_{j\sim i} \sigma(j) - \log {}^{\psi_i(-1)}/_{\psi_i(+1)}) = sigmoid(2J\eta_i - \log {}^{\psi_i(-1)}/_{\psi_i(+1)}). \quad (3)$$

Here is the Gibbs sampling algorithm for reconstruction over 100 iterations. At every iteration, we consider each pixel $(x, y) \in N \times N$ and do a Gibbs move on it so this is not a random scan algorithm. The multinomial distribution in algorithm 2 of section 4.1 is distributed according to equation (3) just derived.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread
import copy

def gauss_density(x, loc, scale):
    norm = 1 / np.sqrt(2 * np.pi * scale)
    return norm * np.exp(-((x-loc) ** 2)/(2*scale))

def neighbors(x, y):
    return np.array([(x-1, y), (x, y-1), (x+1, y), (x, y+1)])

def sign(x):
    if x > 0: return 1
    elif x <= 0: return -1

def update_posterior(posterior, evidence, sigma, J, x, y):
    cur_state = posterior[x, y]
    ev = evidence[x,y]
    neighb = neighbors(x, y)

    n,m = posterior.shape

```

```
    neigh_val = [posterior[i, j] for i,j in neighb if (i >= 0 and j >= 0 \
                                                       and i < n and j < m)]
    eta = sum(neigh_val)
    psi_plus = gauss_density(ev, 1, sigma)
    psi_minus = gauss_density(ev, -1, sigma)

    p1 = np.exp(J*eta)*psi_plus
    p0 = np.exp(-J*eta)*psi_minus

    prob = p1/(p0+p1)

    ind = np.random.binomial(1, prob)

    posterior[x,y] = sign(ind)

def gibbs_sampler(evidence, sigma, J, n_iter):
    posterior = copy.deepcopy(evidence)

    for i in range(n_iter):
        for x in range(evidence.shape[0]):
            for y in range(evidence.shape[1]):
                update_posterior(posterior, evidence, sigma, J, x, y)

    return posterior

Y = imread('letterA.bmp')
J = 1
sigma = 2
n_iters = 100

mean = np.mean(Y)

X = +1.0*(Y>mean) + -1.0*(Y<mean)

gaussian_noise = np.random.normal(0, sigma, img2.shape)
X += gaussian_noise
posterior = gibbs_sampler(Y,  sigma, J, n_iters)

f = plt.figure(figsize=(10, 10), dpi=50);

f.add_subplot(1, 3, 1)
plt.title('Original image (prior)')

plt.imshow(img)
f.add_subplot(1, 3, 2)
plt.title('Noisy image (evidence)')

plt.imshow(img2)

f.add_subplot(1, 3, 3)
plt.title('Denoised image (posterior)')

plt.imshow(posterior)

plt.show()
```

Here is the result.

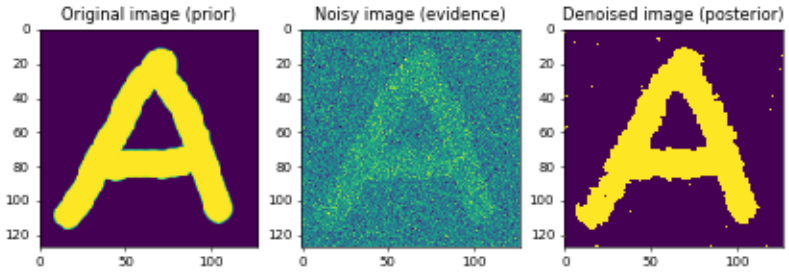

Original image (prior)
Noisy image (evidence)
Denoised image (posterior)
0
20
40
60
80
100
120
0
50
100

Figure 5:

References

[Bertsekas and Tsitsiklis(2002)] D.P. Bertsekas and J.N. Tsitsiklis. *Introduction to Probability*. Athena Scientific books. Athena Scientific, 2002. ISBN 9781886529403. URL https://books.google.com/books?id=bcHaAAAAMAAJ.

[Levin et al.(2006)Levin, Peres, and Wilmer] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006. URL http://scholar.google.com/scholar.bib?q=info:3wf9IU94tyMJ:scholar.google.com/&output=citation&hl=en&as_sdt=2000&ct=citation&cd=0.

[Murphy(2013)] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013. ISBN 9780262018029 0262018020. URL https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2.

[Norris(1997)] J. R. Norris. *Markov Chains*. Cambridge University Press, Cambridge, UK, 1997.