The Ising model and Markov chain Monte Carlo

Ramesh Sridharan*

These notes give a short description of the Ising model for images and an introduction to Metropolis-Hastings and Gibbs Markov Chain Monte Carlo (MCMC). These notes assume you're familiar with basic probability and graphical models.

The notation here is borrowed from *Introduction to Probability* by Bertsekas & Tsitsiklis: random variables are represented with capital letters, values they take are represented with lowercase letters, p_X represents a probability distribution for random variable X, and $p_X(x)$ represents the probability of value x (according to p_X).

1 The Ising model

In many cases where we're interested in doing inference, we can't do it exactly. With such cases, we can approximate the true distribution using samples from it. Let's look at a model where we need to use techniques like this: the *Ising model* [1].

The Ising model isn't the only one where sampling techniques like the ones we'll discuss are useful, and these techniques aren't the only way to do approximate inference here, but they provide a convenient story for illustrating both ideas.

1.1 Ising model for images

Suppose we have a binary image: that is, each pixel can either be 0 or 1. Let X_i be a random variable with the value of pixel i, and let's say we have n pixels. To save ourselves some writing, we'll use $\underline{X}$ to represent the set $\{X_1, X_2, \ldots, X_n\}$. The Ising model says that the distribution for $\underline{X}$ is:

$$p_{\underline{X}}(\underline{x}) = \frac{1}{Z} \prod_{(i,j) \in E} \psi(x_i, x_j), \tag{1}$$

where

- $\psi(\cdot, \cdot)$ is an *edge potential* that's usually chosen to encourage neighboring pixels to have the same values,

*Contact: rameshvs@csail.mit.edu

[1] Technically, the Ising model refers to a model like the one described, but where each X_i takes on values in $\{-1, +1\}$ instead of $\{0, 1\}$. The model here is also frequently referred to as a Markov Random Field, or MRF, even though the term MRF is in fact more general. The standard Ising model (as described in physics) also doesn't have associated observations, but this usage is common enough in machine learning.

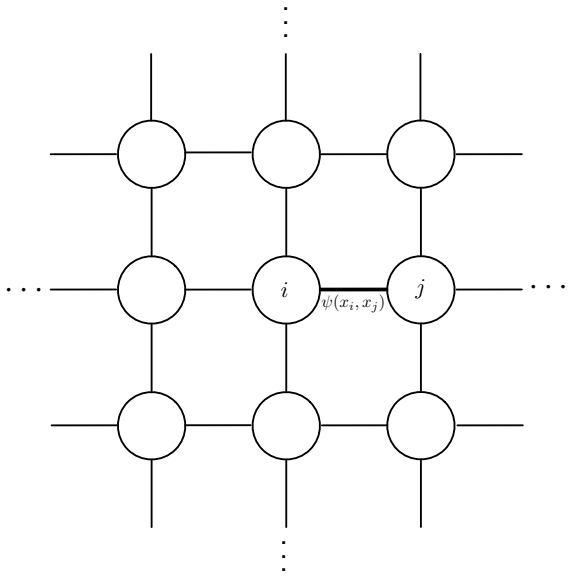

Figure 1: Graphical model for the Ising model described in Equation 1. Since this isn't a tree, we can't use the sum-pushing tricks of the belief propagation algorithm.

- Z is a normalization constant,
- and the notation "$(i, j) \in E$" refers to all pairs of pixels i and j such that i and j are adjacent.

The graphical model for this problem is shown in Figure 1. Now suppose we also have a set of observations $\underline{Y} = \{Y_1, Y_2, \ldots, Y_n\}$, which we model as being conditionally independent given $\underline{X}$:

$$p_{\underline{Y}|\underline{X}}(\underline{y}|\underline{x}) = \prod_i p_{Y_i|X_i}(y_i|x_i) = \prod_i \phi(y_i, x_i), \tag{2}$$

where the conditional distribution ϕ is the same at every pixel. So, the full joint distribution over $\underline{X}$ and $\underline{Y}$ is:

$$p_{\underline{X},\underline{Y}}(\underline{x}, \underline{y}) = \frac{1}{Z} \cdot \prod_{(i,j)\in E} \psi(x_i, x_j) \cdot \prod_i \phi(y_i, x_i) \tag{3}$$

This is similar to a 2-D version of the HMM: we have random variables $\underline{X}$ which are hidden and locally dependent, and observations $\underline{Y}$ which are conditionally independent given $\underline{X}$. Unlike the HMM, however, since the graphical model corresponding to this distribution is not a tree, we can't use techniques like the sum-product algorithm or the forward-backward algorithm like we can for trees or HMMs, respectively.

1.2 Exact inference is hard

Suppose we observe $\underline{Y}$ and wish to compute the posterior distribution over $\underline{X}$:

$$p_{\underline{X}|\underline{Y}}(\underline{X}|\underline{Y}) = \frac{p_{\underline{X},\underline{Y}}(\underline{x},\underline{y})}{\sum_{\underline{x}} p_{\underline{X},\underline{Y}}(\underline{x},\underline{y})} \tag{4}$$

Unfortunately, the sum in the denominator is over all 2^n possible configurations for $\underline{x}$ (and because the graph isn't a tree, we can't use any of the sum-pushing tricks that are useful for the sum-product algorithm). So, computing the denominator is intractable. Computing the marginal distribution at any one pixel involves a similar summation in the numerator, and is also intractable. For example, a typical smartphone camera has 8 megapixels, or 8×10^6 pixels, forcing us to sum over $2^{8\times 10^6}$ possibilities. As a comparison, the number of atoms in the universe is estimated to be about 2^{265}!

What about MAP estimation?

$$\operatorname*{argmax}_{\underline{x}} p_{\underline{x}|\underline{Y}}(\underline{x}|\underline{Y}) = \operatorname*{argmax}_{\underline{x}} p_{\underline{X},\underline{Y}}(\underline{x},\underline{y}) \tag{5}$$

$$= \operatorname*{argmax}_{\underline{x}} \left[\prod_{(i,j)\in E} \psi(x_i, x_j) \cdot \prod_i \phi(y_i, x_i) \right] \tag{6}$$

Even though we eliminated the need to compute any intractable sums, we're still stuck: this is a maximization over 2^n discrete values, which we can't use any shortcuts for. So, doing exact inference requires us to search over every one of them, and so is intractable. In the next section, we'll look at a few ways for approximating this and other complex distributions.

As mentioned before, it's important to note that this is not the only distribution for which inference is intractable, and the approximate methods discussed here are not the only approximations that we can use.

2 Approximate Inference: Sampling

In *Monte Carlo* methods, we use randomly generated samples x to approximate a quantity or distribution of interest, which we'll call p_X. In *Markov Chain Monte Carlo (MCMC)* methods, these samples are generated "Markov-chain style": we start with a sample, which we use to generate the next sample, and so on. Each sample only depends on the one before it, and the transitions between samples are constructed so that in steady-state (i.e., after repeating many times), the samples we obtain come from p_X.

2.1 Metropolis-Hastings

The Metropolis-Hastings algorithm requires only two things:

1. The ability to compute unnormalized probabilities of samples $\tilde{p}_X(x)$: here, unnormalized is okay because we'll only be interested in ratios

2. A proposal distribution $V(x'|x)$, which tells us how to generate the next sample x' given the current sample x

The algorithm itself is simple:

Inputs: Unnormalized probability distribution $\widetilde{p}_X(\cdot)$, proposal distribution $V(\cdot|\cdot)$

1: **function** MetropolisHastings($\widetilde{p}_X, V$)
2: $x \leftarrow$any valid value
3: **loop**
4: $x' \leftarrow$Sample from $V(\cdot|x)$
5: $a \leftarrow \min\left\{1, \frac{\widetilde{p}_X(x')}{\widetilde{p}_X(x)} \cdot \frac{V(x|x')}{V(x'|x)}\right\}$
6: With probability a, $x \leftarrow x'$
7: Save x

Often, we'll choose proposal distributions that are symmetric (i.e., $V(x'|x) = V(x|x')$), and the ratio in line 5 reduces to $\widetilde{p}_X(x')/\widetilde{p}_X(x)$. That is, if the new sample is more probable, we definitely accept it (the min is there just to avoid probabilities that are greater than 1), and if it is less probable, we randomly accept it depending on how much less probable it is. In general, we need to weight that ratio to make sure our proposal distribution isn't unfairly pushing us one way or the other: that's what the $V(x|x')/V(x'|x)$ term is for.

Some important points to note:

(a) Choosing a good proposal distribution can be tricky, and is usually problem-dependent. Suppose X takes on values in between 0 and 1000. Let's look at three different proposal distributions:

- $V_1(\cdot|x)$ is uniform over the interval $x \pm 50000$.
- $V_2(\cdot|x)$ is uniform over the interval $x \pm 1$.
- $V_3(\cdot|x)$ is uniform over the interval $x \pm 300$.

Most of the time, proposals generated from V_1 will be outside the range $[0, 1000]$, and our acceptance probability will be 0. Proposals from V_2 will almost always be accepted, but it will take tens of thousands of iterations just to cover the entire space of possibilities. V_3 attempts to strike a balance in between these, with not too many rejections but a decent coverage of the sample space.

(b) Notice that we initialized completely arbitrarily: in many cases, this initialization could be in a particularly low-probability location. A common practice is to wait K iterations before collecting any samples, to avoid any artifacts from initialization. In this case, K is known as the **burn-in time**.

(c) Another somewhat common practice is to not save every single sample, but rather to wait a fixed number of iterations between each save. This prevents samples from being dependent on each other, but is not necessarily a problem for a well-chosen proposal distribution with enough samples.

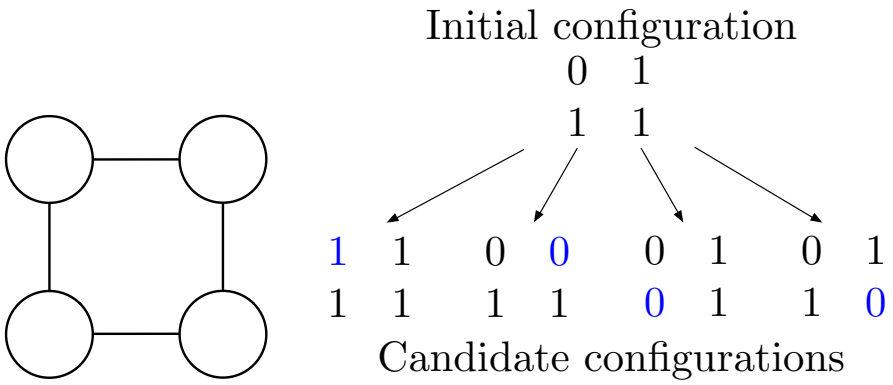

Figure 2: Example proposal distribution for a 4-pixel grid (left). The proposal distribution is uniform over the 4 candidates shown (right).

2.1.1 Metropolis-Hastings MCMC for the Ising model

Let's return to the Ising model. Suppose for our proposal distribution $V(\cdot|\underline{x})$, we flip one randomly chosen value from $\underline{x}$ (call it x_k), and use a distribution that is uniform over all such configurations. An example for a four-pixel image is shown in Figure 2.
Exercise: Compute the acceptance probability for flipping X_k from x_k to x'_k.
Solution: We simply compute the ratio from line 5 of the algorithm. We begin by noting that $V(\underline{x}'|\underline{x}) = V(\underline{x}|\underline{x}')$, since V flips x_k with probability $1/n$ from either $\underline{x}'$ or $\underline{x}$. So, we need only compute the ratio

$$\frac{\widetilde{p}_{\underline{X}}(\underline{x}')}{\widetilde{p}_{\underline{X}}(\underline{x})} = \prod_{j:(j,k)\in E} \frac{\psi(x_j, x'_k)}{\psi(x_j, x_k)} \cdot \frac{\phi(y_k, x'_k)}{\phi(y_k, x_k)},$$

since all terms not involving x_k stay the same and therefore cancel between the numerator and the denominator. Therefore, for any proposed sample, we need only evaluate about 10 numbers and perform a few divisions and multiplications.
Exercise: Why might this not be a good proposal distribution? Can you come up with a better one?

2.2 Gibbs Sampling

Like Metropolis-Hastings, Gibbs sampling is a flavor of MCMC, but it's conceptually simpler! If we want to sample from a distribution over several random variables, Gibbs sampling fixes all but one random variable, samples that one conditioned on the others, and then repeats the process for each random variable. So, all we need are the conditional distributions.

We'll use the notation $\underline{X}_{\neg i}$ to represent all of $\underline{X}$ except X_i: for example,

$$\underline{X}_{\neg 3} = \{X_1, X_2, X_4, X_5, X_6, \ldots, X_n\}.$$

Then our algorithm is as follows:

Inputs: Conditional distributions $p_{X_i|\underline{X}_{\neg i}}(x_i|\underline{x}_{\neg i})$

function GibbsSample($p_{X_i|\underline{X}_{\neg i}}(x_i|\underline{x}_{\neg i})$)
    $\underline{x} \leftarrow$any valid value
    loop
        for i in $\{1, \ldots, n\}$ **do**
            $x_i \leftarrow$Sample from $p_{X_i|\underline{X}_{\neg i}}(\cdot|\underline{x}_{\neg i})$
        Save $\underline{x}$

As with Metropolis-Hastings MCMC, we often only start saving after a burn-in period, and sometimes will wait for several iterations of the outer loop between saves.

2.2.1 Gibbs sampling for the Ising Model

Let's return once again to the Ising model and look at how to apply Gibbs sampling there.
Exercise: Compute the conditional distribution $p_{X_i|\underline{X}_{\neg i},\underline{Y}}(x_i|\underline{x}_{\neg i}, \underline{y})$ up to a constant of proportionality. Explain why we don't need to compute normalizing constants.
Solution: As always, we compute the conditional distribution as the joint distribution over all $\underline{X}$ (conditioned on $\underline{Y}$) divided by the marginal distribution of what we're conditioning on, $\underline{X}_{\neg i}$ (conditioned on $\underline{Y}$):

$$p_{X_i|\underline{X}_{\neg i},\underline{Y}}(x_i|\underline{x}_{\neg i}, \underline{y}) = \frac{p_{\underline{X}|\underline{Y}}(\underline{x}|\underline{y})}{p_{\underline{X}_{\neg i}|\underline{Y}}(\underline{x}_{\neg i}|\underline{y})} \tag{7}$$

$$\propto \prod_{j:(i,j)\in E} \psi(x_i, x_j)\phi(y_i, x_i) \cdot f(\underline{X}_{\neg i}), \tag{8}$$

where the second equality follows from the fact that most terms in the first expression do not depend on x_i, and are therefore strictly a normalizing constant. We can in fact show that most of these terms cancel, but that isn't even necessary here, since we can just compute the normalization constant by plugging in 0 and 1 and adding up the results. For example, if plugging in gives 10 and 20 respectively, then we know that the probabilities are 1/3 and 2/3, since they must add up to 1.