**CS731 Spring 2011 Advanced Artificial Intelligence**

# Markov Chain Monte Carlo

*Lecturer: Xiaojin Zhu* *jerryzhu@cs.wisc.edu*

A fundamental problem in machine learning is to generate samples from a distribution:

$$\mathbf{x} \sim p(\mathbf{x}). \tag{1}$$

This problem has many important applications. For example, one can approximate the expectation of a function $\phi(\mathbf{x})$

$$\mu \equiv \mathbb{E}_p[\phi(\mathbf{x})] = \int \phi(\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{2}$$

by the sample average over $\mathbf{x}_1 \dots \mathbf{x}_n \sim p(\mathbf{x})$:

$$\hat{\mu} \equiv \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x}_i) \tag{3}$$

This is known as the Monte Carlo method. A concrete application is in Bayesian predictive modeling where $\int p(x \mid \theta)p(\theta \mid Data)d\theta$.

Clearly, $\hat{\mu}$ is an unbiased estimator of $\mu$. Furthermore, the variance of $\hat{\mu}$ decreases as

$$\mathbb{V}(\phi)/n \tag{4}$$

where $\mathbb{V}(\phi) = \int(\phi(\mathbf{x}) - \mu)^2 p(\mathbf{x})d\mathbf{x}$. Thus Monte Carlo methods depend on the sample size $n$, not on the dimensionality of $\mathbf{x}$.

Often, $p$ is only known up to a constant. That is, we assume $p(\mathbf{x}) = \frac{1}{Z}\tilde{p}(\mathbf{x})$ and we are able to evaluate $\tilde{p}(\mathbf{x})$ at any point $\mathbf{x}$. However, we cannot compute the normalization constant $Z$. For instance, in the Bayesian modeling example, the posterior distribution is $p(\theta \mid Data) = \frac{1}{Z}p(\theta)p(Data \mid \theta)$. This makes sampling harder. All methods below work on $\tilde{p}$.

## 1 Importance Sampling

Unlike other methods discussed later, importance sampling is *not* a method to sample from $p$. Rather, it is a method to compute (3). Assume we know the unnormalized $\tilde{p}$. It turns out that we (or Matlab) only know how to directly sample from very few distributions such as uniform, Gaussian, etc.; $p$ may not be one of them in general. Importance sampling takes an easy-to-sample distribution $q(\mathbf{x})$ where we can generate samples. In addition, it assumes that we can evaluate the unnormalized version $\tilde{q}$ (If one can compute $q$ that is fine but does not have any advantage). The procedure is simple. Sample $n$ items $\mathbf{x}_1 \dots \mathbf{x}_n \sim q$. For each item, define

$$w_i = \frac{\tilde{p}(\mathbf{x}_i)}{\tilde{q}(\mathbf{x}_i)}. \tag{5}$$

Then, the importance weighted estimator is

$$\hat{\mu} \equiv \frac{\sum_{i=1}^{n} w_i\phi(\mathbf{x}_i)}{\sum_{i=1}^{n} w_i}. \tag{6}$$

The weights correct over-generation and under-generation from $q$, compared to $p$. Importance sampling does not work very well in high dimensions.

## 2 Rejection Sampling

From here on, we discuss methods that actually generate samples from $p$. Again, assume we know $\tilde{p}$ only, and there is an easy-to-sample distribution $q$, and that we can evaluate $\tilde{q}$. *Further assume that we know a constant $c$ such that $c\tilde{q}$ dominates $\tilde{p}$:*

$$c\tilde{q}(\mathbf{x}) \geq \tilde{p}(\mathbf{x}), \ \forall \mathbf{x}. \tag{7}$$

Rejection sampling (potentially) generates a sample in two steps:

1. Sample $\mathbf{x} \sim q$;
2. Sample $a \sim uniform[0, c\tilde{q}(\mathbf{x})]$.
   - If $a \leq \tilde{p}(\mathbf{x})$ then $\mathbf{x}$ is accepted and becomes a sample;
   - otherwise $\mathbf{x}$ is rejected; no sample is generated.

★ *Consider the case where $q$ is uniform.*

The distribution $q$ is known as the proposal distribution. If $q$ is very different from $p$, $c$ will need to be large, and many proposed samples will be rejected, leading to poor efficiency. Rejection sampling does not work very well in high dimensions.

## 3 Markov Chain Monte Carlo

The methods discussed so far are Monte Carlo methods. They work with fixed distributions in each trial. We now introduce the widely successful Markov Chain Monte Carlo (MCMC) methods.

We discuss the finite case for its simplicity. A *Markov chain* is defined by a transition matrix

$$T(\mathbf{x}' \mid \mathbf{x}) \equiv Pr(\mathbf{x} \to \mathbf{x}') \tag{8}$$

with $T(\mathbf{x}' \mid \mathbf{x}) \geq 0, \sum_{\mathbf{x}'} T(\mathbf{x}' \mid \mathbf{x}) = 1$ for each $\mathbf{x}$. Let $p^{(0)}$ be an initial distribution on $\mathbf{x}$, and $p^{(t)}$ the distribution at iteration $t$.

★ *If we run infinite number of independent chains in parallel, the distribution of states at time $t$ will be $p^{(t)}$.*

$p^{(t)}$ is given by

$$p^{(t)}(\mathbf{x}') = \sum_{\mathbf{x}} p^{(t-1)}(\mathbf{x}) T(\mathbf{x}' \mid \mathbf{x}). \tag{9}$$

In matrix form, let $T$ be the matrix with the element at row $\mathbf{x}'$ and column $\mathbf{x}$ be $T(\mathbf{x}' \mid \mathbf{x})$, then

$$p^{(t)} = T^t p^{(0)}. \tag{10}$$

We call the outcomes $X_0 = \mathbf{x}_0, X_1 = \mathbf{x}_1, X_2 = \mathbf{x}_2, \ldots$ as a run of the chain starting at $\mathbf{x}_0$.

A *stationary distribution* $\pi$ of a Markov chain satisfies

$$\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) T(\mathbf{x}' \mid \mathbf{x}), \tag{11}$$

or, equivalently

$$\pi = T\pi. \tag{12}$$

Note this implies that $\pi$ is a right eigenvector of $T$ with eigenvalue 1. The goal of MCMC is to design $T$ such that $\pi$ is the desired distribution $p$.

A Markov chain is *irreducible* if it is possible to get to any state from any state. A state $\mathbf{x}$ is *transient* if, given that we start from $\mathbf{x}$ there is a non-zero probability of never returning to $\mathbf{x}$. Formally, let $fp(\mathbf{x}, \mathbf{x}', t)$ be the *first-passage time probability*, i.e. the probability that the chain goes from $\mathbf{x}$ to $\mathbf{x}'$ in exactly $t$-steps:

$$fp(\mathbf{x}, \mathbf{x}', t) = P(X_t = \mathbf{x}', X_k \neq \mathbf{x}' \text{ for } k = 1 \ldots t-1 \mid X_0 = \mathbf{x}). \tag{13}$$

A state $\mathbf{x}$ is transient if

$$\sum_{t=1}^{\infty} fp(\mathbf{x}, \mathbf{x}, t) < 1 \tag{14}$$

and *recurrent* if

$$\sum_{t=1}^{\infty} fp(\mathbf{x}, \mathbf{x}, t) = 1. \tag{15}$$

The expected return time for state $\mathbf{x}$ is

$$r(\mathbf{x}) = \sum_{t=1}^{\infty} t \cdot fp(\mathbf{x}, \mathbf{x}, t). \tag{16}$$

Recurrent states may or may not have finite expected return time. A state is *positive recurrent* if $r(\mathbf{x}) < \infty$; otherwise it is *null recurrent*.

An irreducible chain has a stationary distribution $\pi$ if and only if all of its states are positive recurrent. Furthermore, we have

$$\pi(\mathbf{x}) = \frac{1}{r(\mathbf{x})}. \tag{17}$$

A state $\mathbf{x}$ has *period* $k$ if any return to $\mathbf{x}$ must occur in multiples of $k$ steps. Formally,

$$k = gcd\{n : Pr(X_n = \mathbf{x} \mid X_0 = \mathbf{x}) > 0\} \tag{18}$$

where *gcd* is the greatest common divisor. For example, suppose it is possible to return to $\mathbf{x}$ in 6,8,10,12,... steps; then $k = 2$ although it is not possible to return in 2 steps. A chain is *aperiodic* if all states have period $k = 1$, i.e., returns can occur at irregular steps. Critically, if the chain is also aperiodic, then

$$\lim_{t \to \infty} (T^t)_{\mathbf{x}'\mathbf{x}} = \pi(\mathbf{x}'). \tag{19}$$

That is, the matrix $T^{\infty}$ consists of identical columns, each column is $\pi$. Consequently, $p^{(\infty)} = \pi$ regardless of the initial distribution $p^{(0)}$.

This is the basis of MCMC methods. What remains is to design $T$ such that $\pi$ is any desired distribution $p$. A typical approach is to design $T$ such that it satisfies detailed balance w.r.t. $\pi$, which is a sufficient (but not necessary) condition of $T$ to have a stationary distribution $\pi$. The *detailed balance* property is

$$\pi(\mathbf{x})T(\mathbf{x}' \mid \mathbf{x}) = \pi(\mathbf{x}')T(\mathbf{x} \mid \mathbf{x}'). \tag{20}$$

A Markov chain that satisfies detailed balance is also called *reversible*.

# 4 The Metropolis-Hastings Algorithm

Again, we assume we know $\tilde{p}$ only. In the Metropolis-Hastings algorithm, we require a proposal distribution $q(\mathbf{x}'|\mathbf{x})$. Unlike importance or rejection sampling, $q$ can be quite different from $p$. Like rejection sample, the Metropolis-Hastings algorithm is a two-step procedure. Unlike rejection sample, the Metropolis-Hastings algorithm *always* generates a sample. It needs the previously generated sample $\mathbf{x}^{(t-1)}$:

1. Propose a sample $\mathbf{x}' \sim q(\mathbf{x}' \mid \mathbf{x}^{(t-1)})$.

2. With probability

$$a = \min\left(1, \frac{\tilde{p}(\mathbf{x}')q(\mathbf{x}^{(t-1)} \mid \mathbf{x}')}{\tilde{p}(\mathbf{x}^{(t-1)})q(\mathbf{x}' \mid \mathbf{x}^{(t-1)})}\right), \tag{21}$$

accept the proposal: $\mathbf{x}^{(t)} = \mathbf{x}'$. Otherwise reject the proposal but *retain* the old point: $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}$.

To understand this algorithm, let us identify the transition matrix $T$. Define

$$A(\mathbf{x} \to \mathbf{x}') \equiv \frac{\tilde{p}(\mathbf{x}')q(\mathbf{x}^{(t-1)} \mid \mathbf{x}')}{\tilde{p}(\mathbf{x}^{(t-1)})q(\mathbf{x}' \mid \mathbf{x}^{(t-1)})}. \tag{22}$$

Metropolis-Hastings can be viewed as constructing $T$ satisfying detailed balance w.r.t. $p$, using a proposal distribution $q$ that might be irrelevant to $p$. In particular,

$$T(\mathbf{x}' \mid \mathbf{x}) = \begin{cases} q(\mathbf{x}' \mid \mathbf{x}) & \mathbf{x} \neq \mathbf{x}', A(\mathbf{x} \to \mathbf{x}') \geq 1 \\ q(\mathbf{x}' \mid \mathbf{x})A(\mathbf{x} \to \mathbf{x}') & \mathbf{x} \neq \mathbf{x}', A(\mathbf{x} \to \mathbf{x}') < 1 \\ q(\mathbf{x} \mid \mathbf{x}) + \sum_{\mathbf{x}: A(\mathbf{x} \to \mathbf{z}) < 1} q(\mathbf{z} \mid \mathbf{x})(1 - A(\mathbf{x} \to \mathbf{z})) & \mathbf{x} = \mathbf{x}' \end{cases} \tag{23}$$

Then, detailed balance is satisfied

$$p(\mathbf{x})T(\mathbf{x}' \mid \mathbf{x}) = p(\mathbf{x}')T(\mathbf{x} \mid \mathbf{x}') \tag{24}$$

and $p$ is the stationary distribution of $T$.

Note the samples $\mathbf{x}^{(1)} \dots \mathbf{x}^{(t)} \dots$ are dependent (imagine $q$ is a narrow Gaussian centered on the previous point). The distribution of $\mathbf{x}^{(t)}$ will approach $p(\mathbf{x})$ as $t \to \infty$. The initialization $\mathbf{x}^{(0)}$ does not matter. However, it is common practice to discard some number $T$ of initial samples (known as "burn-in") and estimate (3) using $\mathbf{x}^{(T)} \dots$. All samples after burn-in should be used to estimate (3). "Thinning," the practice of taking out a sample after some number of iterations, is not helpful in this regard. It is also better to run a single deep chain ($t \to \infty$), or when parallelism is available, several somewhat deep chains, than to run a large number of short chains.

The main issue of MCMC efficiency is mixing rate, to be defined later. Intuitively, $p$ can have multiple modes separately by large low density regions. Good mixing means the samples can hop among modes easily. The proposal distribution $q(\mathbf{x}'|\mathbf{x})$ plays a critical role in this regard. Three types of commonly used proposal distributions are:

- Local proposal such as $q(\mathbf{x}'|\mathbf{x}) = N(\mathbf{x}, \Sigma)$ which centers on $\mathbf{x}$ and (usually) makes small moves. This produces a random walk behavior. If the length scale of high probability region is $L$, a random walk with average step size $w$ will need at least $(L/w)^2$ steps to cover the region. Furthermore, such local proposal usually does not encourage mode-hopping.
- Independent proposal $q(\mathbf{x}'|\mathbf{x}) = q(\mathbf{x}')$ which does not depend on the previous sample $\mathbf{x}$. This can be a hit or miss, depending on how close $q$ is to $p$. It can encourage mode-hopping if it knows where the modes are.
- Hybrid proposal which uses a mixture of the above, or interleave the above two in different trials.

Also note that when $q$ is symmetric

$$q(\mathbf{x}'|\mathbf{x}) = q(\mathbf{x}|\mathbf{x}') \tag{25}$$

such as when $q(\mathbf{x}'|\mathbf{x}) = N(\mathbf{x}'; \mathbf{x}, \Sigma)$, the acceptance probability simplifies to

$$a = \min\left(1, \frac{\tilde{p}(\mathbf{x}')}{\tilde{p}(\mathbf{x}^{(t-1)})}\right). \tag{26}$$

# 5 Gibbs Sampling

The Gibbs sampler is a special case of Metropolis-Hastings where the proposal distributions loops over the following conditional distribution:

$$q_i(\mathbf{x}'|\mathbf{x}) = \begin{cases} p(x_i \mid \mathbf{x}_{\neg i}) & \mathbf{x}'_{\neg i} = \mathbf{x}_{\neg i} \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

where $i = t \mod d$ for iteration $t$, and $d$ is the dimensionality of $\mathbf{x}$. It is easy to see that the acceptance probability is $a = 1$, thus all proposed samples are accepted. Gibbs sampling is advantageous when $p(x_i \mid \mathbf{x}_{\neg i})$ is easy to compute, which is true if the Markov blanket of any variable is relatively small. However, Gibbs sampling is not particularly good at mode-hopping.

# 6 Slice Sampling

The idea of slice sampling is to augment $p(\mathbf{x})$ with an auxiliary variable $u$ to produce a joint distribution $p(\mathbf{x}, u)$. It might be easier to sample from the latter to produce samples $(\mathbf{x}_1, u_1) \dots (\mathbf{x}_n, u_n)$. One then discard the $u$ component to get samples $\mathbf{x}_1 \dots \mathbf{x}_n$, which follows the marginal distribution $p(\mathbf{x})$. It might seem counter-intuitive to go against the curse of dimensionality by working in higher dimensions, but slice sampling may encourage large and self-adaptive movements.

An (over)-simplification of slice sampling works as follows. Suppose we have sampled $\mathbf{x}$ in the previous iteration. Draw $u \sim uniform(0, \tilde{p}(\mathbf{x}))$, which is a random height beneath the point $(\mathbf{x}, \tilde{p}(\mathbf{x}))$. Think of $\tilde{p}$ as defining the landscape. Now flood the $\mathcal{X}$ space to height $u$. There should be land higher than this water level (perhaps in the form of many isolated islands). Slice sampling draws the next sample $\mathbf{x}'$ uniformly from the union of such land areas (the "slice"). The process then repeats.

The difficulty in the above procedure is the last step of drawing a uniform sample from the slice. A more practical slice sampling algorithm is given below, which does not consider the whole slice but a somewhat local version of it. This procedure is for 1D.

1. Draw $u \sim uniform(0, \tilde{p}(x))$
2. Generate an interval $(x_l, x_r)$ such that $x \in (x_l, x_r)$
3. REPEAT
   (a) Draw $x' \sim uniform(x_l, x_r)$
   (b) IF $\tilde{p}(x') > u$ BREAK;
   (c) ELSE shorten the interval $(x_l, x_r)$

In step 2, the goal is to create an interval that contains a local slice. This means that both ends should be below water level (though the interval may not contain other distance islands). This can be done as follows, fixing a step size parameter $w$:

1. Draw $r \sim uniform(0, 1)$.
2. $x_l = x - rw, x_r = x + (1 - r)w$ % unit interval around $x$, randomly shifted
3. WHILE $(\tilde{p}(x_l) > u)$ let $x_l \leftarrow x_l - w$
4. WHILE $(\tilde{p}(x_r) > u)$ let $x_r \leftarrow x_r + w$

In step 3(b), one breaks out of the loop when $x'$ is in the local slice. Otherwise one shortens the interval in step 3(c). This shortened interval will still have both ends below water level, and contain a potentially smaller local slice. Specifically, step 3(c) can be implemented as follows:

1. IF $x' < x$ THEN $x_l \leftarrow x'$ ELSE $x_r \leftarrow x'$.

Slice sampling has a nice self-adaptive property: If $w$ is too small, for high probability region with length scale $L$ it takes $O(L/w)$ iterations to grow the interval to cover that slice – compare this to $(L/w)^2$ needed for Metropolis-Hastings with a local proposal. If $w$ is too large, it only takes $O(\log(w/L))$ iterations to shrink the interval to fit snugly. Note, however, that this version of slice sampling does not completely solve the mode-hopping problem (it might miss distant islands).

What about high dimensional $\mathcal{X}$? One can use the 1D slice sampler above as a building block to iteratively sample through each dimension in turn (i.e., axis-parallel directions), or use random directions.

# 7 Exact Sampling: Coupling from the Past

A fundamental problem with the MCMC methods is that they sample from the stationary distribution only asymptotically – how long is "eventually"? Coupling from the Past gives a certificate that a sample is indeed from the stationary distribution. Caveat: it may take a very long time to obtain the certificate, and it only works for certain Markov chains (monotonicity).

Consider the example with integer $x$ and target distribution $p(x) = 1/21$ if $x \in \{0, 1, \ldots, 20\}$, and 0 otherwise. Consider Metropolis-Hastings with proposal $q(x' \mid x) = 1/2$ if $x' = x \pm 1$, and 0 otherwise. Note a proposal is always accepted unless it goes to -1 or 21, in which case it is rejected. Therefore, the Markov chain behaves like a 1D random walk with walls on both sides.

Coupling from the past is built upon three ideas:

1. Coalescence of coupled Markov chains. Consider running 21 Markov chains in parallel, each starting from a different $x$. Critically, these chains are *coupled* by the same randomness (think of the same random number generator and seed). In particular, at a given iteration, the direction of the proposal is the same for all chains: all toward left or all toward right. Multiple chains could *coalesce* when they hit the same boundary: that is, they will reach the same state $x = 0$ (or $x = 20$). Once two chains coalesce, they will be identical in future iterations because they share the same randomness.

2. Coupling from the past. The moment when all chains (starting from each possible $x$) coalesce is not a valid sample. This is easy to see in our example where such coalesce can only happen when the chains are at $x = 0$ or 20. The idea is to start all chains from a time $T_0$ in the past, up to the present. If coalescence has happened, the present sample is an unbiased sample from the stationary distribution. All coupled Markov chains starting on any state from the infinity past must pass *some* (may not be all) of the states at $T_0$, and therefore coalesce at present – in effect we have run the Markov chains forever.

   If coalescence has not happened, we re-run all chains starting at $T_1 = 2T_0$ in the past. Critically, when these chains reach $T_0$ in the past (where some of them might have coalesced), we re-use the same randomness as what has been used when we started at $T_0$. This doubling-into-the-past repeats until all chains coalesce at present. We always re-use the appropriate randomness.

3. Monotonicity. The procedure so far is not practical for large $\mathcal{X}$ spaces because we need to run one Markov chain starting from *each* state. However, for certain Markov chains it is possible to detect that all chains have coalesced without actually running all of them. In our example, notice that no chains ever cross each other. Thus, we only need to run two chains starting at $x = 0$ and $x = 20$ respectively: when these two have coalesced, all chains must have, too.