BIN 504
Probabilistic and Statistical Modeling for Bioinformatics
Fall 2010-2011

Week 11
Monte Carlo Methods and MCMC
(slides from Statistical Models and Methods in CS CS 290I, 2004, UCSB)
Monte Carlo methods

- Definitions:
  - A mathematical technique that uses repeated calculations and random numbers to find an approximate solution to a complex problem.
  - Any method which solves a problem by generating suitable random numbers and observing that fraction of the numbers obeying some property or properties. The method is useful for obtaining numerical solutions to problems which are too complicated to solve analytically.
  - Methods to provide approximate solutions to a variety of mathematical problems by performing statistical sampling experiments on a computer. The method applies to problems with no probabilistic content as well as to those with inherent probabilistic structure.
  - Methods for solving various kinds of computational problems by using random numbers (or more often pseudo-random numbers), as opposed to deterministic algorithms.
  - Any method that utilizes sequences of random numbers to perform a statistical simulation
  - Statistical simulation methods
Goals of Monte Carlo methods

1. To generate random samples \( \{x'\} \) from a given probability distribution \( P(x) \)
   - “Sampling a distribution”

2. To estimate the expected value of a function under this distribution:

\[
\bar{g} = \langle g(x) \rangle = E[g(x)] = \int_{-\infty}^{\infty} g(x) p_x(x) dx \\
= \sum_{i=1}^{N} g(x_i) p_x(x_i)
\]

Assume we have a random number generator with uniform distribution \([0, 1]\)
Example of #2: Monte Carlo integration

- Wish to integrate an unknown function $g(x)$ by sampling it

\[
I = \int_{a}^{b} g(x) \, dx
\]

\[
\overline{g}_{ab} = E[g(x)] = \int_{-\infty}^{\infty} g(x) p_x(x) \, dx
\]

\[
= \int_{a}^{b} g(x) \frac{1}{b-a} \, dx
\]

\[
= \frac{1}{b-a} \int_{a}^{b} g(x) \, dx = \frac{1}{b-a} I
\]

So...

\[
I = (b-a) \overline{g}_{ab}
\]

\[
\approx (b-a) \frac{1}{N} \sum_{i=1}^{N} g(x_i)
\]
Transforming pdfs

- Given density function $p_x(x)$ and a one-to-one function that maps $x$ to $y$,

$$y = f(x)$$

we can determine $p_y(y)$, the pdf of $y$

$$p_y(y) = \frac{p_x(x)}{|\frac{dy}{dx}|}$$
Example

\[ p_x(x) = x, \quad 0 \leq x \leq ? \]

\[ y = x^2 \]

What is \( p_y(y) \)?

\[ p_y(y) = \frac{p_x(x)}{\left| \frac{dy}{dx} \right|} = \frac{x}{2x} = 0.5 \quad 0 \leq y \leq ? \]

2
What if $y = F_x(x)$?

- If the function $y$ happens to be the cumulative distribution function of $p_x(x)$, what will $p_y(y)$ be?

$$p_y(y) = \frac{p_x(x)}{\frac{dy}{dx}} = \frac{p_x(x)}{dF_x} = \frac{p_x(x)}{p_x(x)} = U(y)$$

- So this gives us a way to sample from any $p_x(x)$
  - As long as we can compute and invert $F_x(x)$
Sampling via inversion of the cdf

1. Sample a random number $y'$ from $U[0,1]$

2. Set $y' = F_x(x)$, where $F_x(x)$ is the cdf of $p_x(x)$

3. Solve for $x'$: $x' = F_x^{-1}(y')$

Result: $x'$ is sampled from $p_x(x)$
So is sampling solved?

- Why not?
  - Can’t always determine $F_x(x)$
  - Can’t always invert $F_x(x)$

- For discrete $P(x)$, $F(x)$, may be computationally infeasible
Uniform sampling

- **Estimate** $E[g(x)]$ by drawing random samples $x_i$ from the uniform distribution $U(x)$ and evaluating $g(x_i)$ and $p_x(x_i)$

$$E[g(x)] \approx \sum_{i=1}^{N} g(x_i) p_x(x_i)$$

- Assumption: We can evaluate $p_x(x_i)$, but we don’t know how to sample from it

- Why doesn’t uniform sampling work in general?
  - We are likely to miss (undersample) the “typical set,” causing the estimate to be wrong
Uniform sampling (cont.)

\[ E[g(x)] \approx \sum_{i=1}^{N} g(x_i) p_x(x_i) \]

- We would prefer to sample from \( p_x(x) \) rather than from \( U(x) \)
- But we don’t know how to!
Importance sampling

- **Estimate** $\mathbb{E}[g(x)]$ by drawing random samples $x_i$ from the “sampler density” $q_x(x)$ which we know how to evaluate
  - E.g., $q_x(x)$ could be a Gaussian

More likely to sample from the typical set of $p_x$

- **Weight each sample by its importance:** $w_i = \frac{p_x(x_i)}{q_x(x_i)}$
Importance sampling (cont.)

- Estimate $E[g(x)]$ by the importance weighted sum of $g(x_i)$
  \[
  E[g(x)] \approx \frac{\sum_{i=1}^{N} w_i g(x_i)}{\sum_{i=1}^{N} w_i}
  \]

- The estimate converges to $E[g(x)]$ as $N \to \infty$
  - But $N$ may have to be very large for a good estimate

- What if $q_x(x) = U(x)$?
  - Same as uniform sampling
Rejection sampling

- Generate random samples from $p_x(x)$ by first drawing random samples $x_i$ from the “proposal density” $q_x(x)$ which we know how to evaluate, such that for some constant $c$

$$cq_x(x) > p_x(x) \quad \text{for all } x$$

Then:
- Sample $x'$ from $q_x(x)$
- Sample $u$ from $U(x)$, $0 \leq x \leq cq_x(x')$
- If $u > p_x(x')$, reject $x'$
- Else, keep $x'$
Rejection sampling (cont.)

- Rejection sampling works best if $q_x(x)$ is a good approximation to $p_x(x)$
  - What does “works best” mean?

- Overall acceptance rate is $1/c$
  - Equals the area under $p_x(x)$ divided by the area under $q_x(x)$
  - For high dimensional problems, $c$ can be very large

- What if $q_x(x) = U(x)$?
  - I.e., $q_x(x) = \max_x (p_x(x))$?
Note

- Monte Carlo methods generally address two problems:
  - Sampling (generating samples from $p_x(x)$)
  - Estimating expectations of functions $E[g(x)]$ (where RV $x$ has a distribution $p_x(x)$)

- If we solve the first, can we directly solve the second?
  - Yes, by computing the sample mean of the function

$$\hat{g} = \frac{1}{N} \sum_{i} g(x_i) \quad x_i \text{ generated from } p_x(x)$$

  - If $g(x)$ has a variance of $\sigma^2$, what is the variance of $\hat{g}$?
    $$\sigma^2/N$$
  - What does this mean?
Note (cont.)

- What if we can’t solve the first, but we can come reasonably close to it?
- Then we can do importance sampling, where

\[
\hat{g} = \frac{1}{N} \sum_i g(x_i) \quad \text{becomes} \quad \hat{g} = \frac{\sum_i w_i g(x_i)}{\sum_i w_i} \quad w_i = \frac{p_x(x_i)}{q_x(x_i)}
\]

These are the same when \( w_i = 1 \)

Which implies that the sampler density \( q_x(x) \) equals the desired density \( p_x(x) \)
Markov chains

- Let $x^{(i)}$ be a stochastic process
  - A collection of random variables $\{ x^{(1)}, x^{(2)}, x^{(3)}, \ldots \}$
    - Typically, the index $i$ represents time, and $x^{(i)}$ represents the state of the process at time $i$
  - $x^{(i)}$ can take $s$ discrete values (states) $\{ s_1, s_2, \ldots, s_s \}$

- The stochastic process $x^{(i)}$ is called a Markov chain if
  $$ p(x^{(i+1)} | x^{(i)}, x^{(i-1)}, \ldots, x^{(1)}) = p(x^{(i+1)} | x^{(i)}) $$
  - I.e., the next state depends only on the current state (not the whole history)
  - The Markov chain is homogeneous if $p(x^{(i+1)} | x^{(i)})$ is identical for all $i$
    - Then the evolution of the chain depends solely on the current state of the chain and a fixed transition matrix $T$
Markov process

- Which of these can be modeled as Markov processes?
  - The weather in Santa Barbara
  - A random walk
  - A random white noise process
  - Words in English
  - The notes of a melody
  - The Dow Jones Industrial Average
  - The location of a sprinter
  - The \{location, velocity\} of a sprinter
  - Moves in a game of chess
  - Web browsing behavior

- Well, all of them *can* be modeled as Markov processes, but some may not be very accurate models!
Markov chain example

- Three states
  - \( \{ s_1, s_2, s_3 \} \)

- Transition matrix

\[
T = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0.1 & 0.9 \\
0.6 & 0.4 & 0 
\end{bmatrix}
\]

\[p(s_3 \mid s_2, s_1) = ?\]

\[
p(x^{(i+1)} = s_3 \mid x^{(i)} = s_2, x^{(i-1)} = s_1) = ?
= p(x^{(i+1)} = s_3 \mid x^{(i)} = s_2) = 0.9
\]
Markov process example: Weather

- Three states: \{ sunny, cloudy, rainy \}
- Transition matrix:

\[
T = \begin{bmatrix}
0.7 & 0.3 & 0.4 \\
0.3 & 0.4 & 0.4 \\
0.0 & 0.3 & 0.2 \\
\end{bmatrix}
\]

- Day 1 is sunny

\[
x^{(1)} = \begin{bmatrix}
1 \\
0 \\
0 \\
\end{bmatrix}
\]

- What can we say about day 2?

\[
x^{(2)} = Tx^{(1)} = \begin{bmatrix}
0.7 & 0.3 & 0.4 \\
0.3 & 0.4 & 0.4 \\
0.0 & 0.3 & 0.2 \\
\end{bmatrix} \begin{bmatrix}
1 \\
0 \\
0 \\
\end{bmatrix} = \begin{bmatrix}
0.7 \\
0.3 \\
0 \\
\end{bmatrix}
\]
Notation note

- I’m using different notation from the MCMC paper (Andrieu et al.)

- My $T$ matrix is the transpose of their $T$ matrix
  - Each column sums to 1.0 (rather than each row)
  - Transition is modeled as $x^{(i+1)} = T x^{(i)}$ (rather than $x^{(i+1)} = x^{(i)} T$)
Markov process example: Weather (cont.)

- What can we say about day 3?

\[
x^{(3)} = T x^{(2)} = \begin{bmatrix} 0.7 & 0.3 & 0.4 \\ 0.3 & 0.4 & 0.4 \\ 0.0 & 0.3 & 0.2 \end{bmatrix} \begin{bmatrix} 0.7 \\ 0.3 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.58 \\ 0.33 \\ 0.09 \end{bmatrix}
\]

- What’s the long range weather forecast? (I.e., the “steady state” weather)

\[
x^{(4)} = T x^{(3)} = T T x^{(2)} = T T T x^{(1)} = T^3 x^{(1)}
\]

\[
x^{(N+1)} = \begin{bmatrix} 0.7 & 0.3 & 0.4 \\ 0.3 & 0.4 & 0.4 \\ 0.0 & 0.3 & 0.2 \end{bmatrix}^N \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \ ?
\]

\[
\lim_{n \to \infty} x^{(n)} = \begin{bmatrix} 0.52 \\ 0.35 \\ 0.13 \end{bmatrix}
\]
Markov Chain Monte Carlo Methods

• For estimating a complex distribution by iteratively sampling from it
• The probability of drawing a sample depends only on the previous sample
• MCMC methods which include the Gibbs sampler and Metropolis–Hastings algorithms, provide a general process for iteratively sampling values of the parameters from approximate posterior distributions and then continually updating the sampling to better approximate the true posterior distribution
The Metropolis method

- Importance sampling (to estimate $E[g(x)]$) and rejection sampling (to sample $p_x(x)$) only work well if the proposal density $q_x(x)$ is similar to $p_x(x)$
  - In complex and high dimensional problems, it is hard to find a suitable proposal density $q_x(x)$

- The Metropolis method produces samples from $p_x(x)$ via a state-based process using a “Markov chain”
  - An example of a Markov Chain Monte Carlo (MCMC) method
  - Produces sample points \{ $x_1, x_2, x_3, \ldots, x_T$ \}
  - The sampled points $x_i$ are not independent – $x_i$ depends on the previous value $x_{i-1}$
  - The Markov chain may have to “run” for a long time before it generates enough samples to represent $p_x(x)$
The Metropolis method (cont.)

- The proposal density $q_x(x; x_i)$ depends on the current state $x_i$
  - It can be any fixed density
  - It does not have to be similar to $p_x(x)$
The Metropolis-Hastings algorithm

- At the current value of $x_i$, a tentative new state $x'$ is generated from $q_x(x; x_i)$

$$a = \frac{p_x(x')}{p_x(x_i)} \frac{q_x(x_i; x')}{q_x(x'; x_i)} = 1 \text{ for a symmetric } q_x$$

- Accept $x'$ if $a \geq 1$, and set $x_{i+1} = x'$
- Otherwise, accept $x'$ with probability $a$ and set $x_{i+1} = x'$
- If $x'$ is rejected, set $x_{i+1} = x_i$ (repeat current state / sample value)
$x_1 = 10$

**Example**

$$q_x(x; x_i)$$

$$\frac{1}{21}$$

$$\begin{align*}
0 & \quad x_i & \quad 20 \\
p_x(x) & \quad & \\
q_x(x; x_i) & \quad & \\
q_x(x_i; x') & \quad & \\
\frac{p_x(x') q_x(x_i; x')}{p_x(x_i) q_x(x'; x_i)} & = 1
\end{align*}$$
For any positive $q_x(x)$, the distribution of $x$, approaches $p_x(x)$ as $T \to \infty$

- Widely used for high dimensional problems
  - Better behaved than importance sampling and rejection sampling

- Generally, we want $q_x(x)$ to be fairly narrow
  - Too wide: Many states rejected, thus slow progress
  - Too narrow: Random walk behavior takes a long time to sample the whole space, thus slow progress

- It is difficult to assess convergence – how many samples are enough to effectively sample $p_x(x)$?
  - If we want $N$ independent samples, now many Metropolis samples do we need to compute?
Metropolis samples

- Metropolis samples are correlated
  - How then to generate independent samples?

- The state space is explored essentially by a random walk

- A random walk ("Brownian motion") that takes $T$ steps of size $\varepsilon$ is likely to reach a distance of $\varepsilon (T)^{1/2}$
  - So for a distribution of "size" $L$, we have to generate at least $T \approx (L / \varepsilon)^2$ samples to obtain an additional independent sample
  - This assumes that every step is accepted. If only a fraction $f$ of the steps are accepted, then $T \approx (L / \varepsilon)^2 / f$

- The algorithm spends more iterations in the "most important" regions of $p_x(x)$
Example

- Independent samples
  - $L \approx 20$ ("size" of distribution)
  - $\varepsilon = 1$ (step size)
  - $T \approx (L / \varepsilon)^2 = 20^2 = 400$
  - Thus it takes about 400 iterations to generate a new, effectively independent sample

- Metropolis is similar to simulated annealing
  - Probabilistic steps; might go "backwards"
  - Simulated annealing essentially modifies the "width" of $q_x(x; x_i)$ over time
Metropolis and Metropolis-Hastings Algorithms

• Both algorithms generate random perturbations of the current parameter values and then employ an acceptance/rejection rule to converge to the true joint distribution

• Generate a new set of parameters from the current set of parameters using a jumping distribution or an approximate posterior distribution

• Accept/reject the new set of parameters by comparing them to the current set of parameters.
  – If the new set more probable, accept them
  – If the new set is less probable, accept it with a small probability

• The probability of a current set of parameters can be computed by simulation
An example Metropolis-Hastings Algorithm

B1 Initialize $\theta(0) = (w(0), w^0(0), T(0))$, $t = 0$.
B2 Sample $\bar{\theta}$ from a proposal distribution $q(\cdot | \theta(t))$.
B3 Simulate a dataset $S$ using the stochastic model described by Equations (1) and (2), with parameters $\bar{\theta}$.
B4 If $S \neq \mathcal{D}'$ (based only on the observed nodes in $\mathcal{D}'$), let $\theta(t+1) = \theta(t)$, increase $t$ and go to B2.
B5 Compute $\gamma = \min \left( 1, \frac{\pi(\bar{\theta})q(\theta(t)|\bar{\theta})}{\pi(\theta(t))q(\theta|\theta(t))} \right)$.
B6 Accept $\theta(t+1) = \bar{\theta}$ with probability $\gamma$, otherwise stay at the old point $\theta(t+1) = \theta(t)$.
B7 Increase $t$ and go to B2, until enough points are sampled.
Gibbs Sampling

• Allows sampling from a high-dimensional multi-variate joint distribution

• Algorithm:
  – Assume you have $k$ parameters to estimate
  – Assume an initial set of parameters
  – By marginalizing over $k-1$ parameters compute the conditional probability of one of the variables and sample from this distribution.
  – Repeat this for each variable in an iteration
  – Repeat this process until convergence
Gibbs sampling

- For sampling multidimensional distributions
  \[ p_x(x) = p_x(x_1, \ldots, x_k) \]
- Gibbs sampling is a Metropolis method where the sampling proposal distribution \( q_x \) is defined as conditional distributions of \( p_x(x) \)
  - Produce \( x^{(t+1)} \) from \( x^{(t)} \)
- Sample the conditional densities sequentially
  - \( x_1^{(t+1)} \sim p_x(x_1 \mid X_2=x_2^{(t)}, \ldots, X_k=x_k^{(t)}) \)
  - \( x_2^{(t+1)} \sim p_x(x_2 \mid X_1=x_1^{(t+1)}, X_3=x_3^{(t)}, \ldots, X_k=x_k^{(t)}) \)
  - \( x_3^{(t+1)} \sim p_x(x_3 \mid X_1=x_1^{(t+1)}, X_2=x_2^{(t+1)}, X_4=x_4^{(t)}, \ldots, X_k=x_k^{(t)} \)
  - \( \ldots \)
  - \( x_k^{(t+1)} \sim p_x(x_k \mid X_1=x_1^{(t+1)}, X_2=x_2^{(t+1)}, \ldots, X_{k-1}=x_{k-1}^{(t+1)}) \)
- All samples are accepted
Example: \( p_x(x) = p_x(x_1, x_2) \)
Gibbs sampling (cont.)

- As a Metropolis algorithm, the state space is explored by a random walk
  - It can take many iterations to produce a “new” independent random sample

- Gibbs sampling involves no adjustable parameters (like $q_x$ in regular Metropolis)
  - Easy to implement

- As with Metropolis, Gibbs sampling is a MCMC method
  - These methods are useful for high dimensional problems
  - Much effort goes into methods to speed them up….
Simulated Annealing

• Example task: Find the maximum of $p_x(x)$ or $g(x)$
  – Could take the *maximum sample value*; may miss the mode, though

• Similar to hill-climbing search
  – But includes a random element
  – Sometimes take “bad” steps to escape local maxima
  – Motivated by the roughly analogous physical process of annealing: heating and then slowly cooling a substance to obtain a strong crystalline structure

• Analogy with physical annealing:
  – $T$ is temperature, $E$ is energy
  ♦ A *schedule* determines the rate at which $T$ is lowered
Simulated Annealing (cont.)

- For the current state, evaluate the operators/actions
- Instead of taking the best action, choose an action at random
  - If that action results in a higher "goodness" value ($\Delta E > 0$), take it
  - Otherwise, take it with probability between 0 and 1

```
Delta E = 1

Delta E = 3

Delta E = -2

Etc...
```
Probabilistic action

- $\Delta E : \text{Value(action)} - \text{Value(current)}$
  - If $\Delta E > 0$, take the action
  - If $\Delta E < 0$, maybe take it, maybe not

- $P(\text{action}) = e^{\Delta E/T}$ (for negative $\Delta E$)
  - $T$: Determined by “schedule”
    - Starts large, decreases towards zero as the algorithm iterates
    - Large $T$ means ???
    - Small $T$ means ???
  - What happens when $T = 0$?
Probabilistic action

• What does it mean to take an action with probability $P(A) = e^{\Delta E/T}$?

• For action $A_1$
  – Let $\Delta E = -3$ and $T = 20$
  – $P(A_1) = e^{\Delta E/T} = 0.71$

• Choose a random number $r \in [0..1]$
  – If $r < 0.71$, take action $A_1$
  – Else do nothing

• For action $A_2$
  – $\Delta E = -9$ and $T = 20$
  – $P(A_2) = e^{\Delta E/T} = 0.35$

• Choose a random number $r \in [0..1]$
  – If $r < 0.35$, take action $A_2$
  – Else do nothing
$P(\text{action})$

Decreasing $T$

$e^{\Delta E/T}$

"Badness" of action
Intuition of SA

- Consider a ball bouncing around a vibrating surface, which you are holding with your eyes closed
  - Goal: Get the ball to the deepest point of the surface
  - What’s your strategy?
  - T : amount of vibration
  - E : depth of the ball
Example: Get the ball to the minimum
Large T
Smaller T
Even smaller T
Simulated Annealing (cont.)

• The trick is coming up with a proper schedule
  – Where to start (what T)?
  – How quickly to lower T?
    ♦ Too quickly?
    ♦ Too slowly?

• SA is an example of an MCMC method