CS 5135/6035 Learning Probabilistic Models

Lecture 18: Random Sampling

Gowtham Atluri

November 4, 2018

Reading Material

- Chapter 2. Random Variable Generation
  Christian Robert and George Casella. Introducing Monte Carlo Methods with R

Topics

- Point-estimation from posterior
- Random Number Generation
  - True vs. Pseudo random numbers
- Inverse transform method
  - Continuous distributions
  - Discrete distributions

Posterior point-estimation: Standard form

When the posterior has a standard functional form (due to conjugacy):

- we can compute a summary of the distribution analytically
  - mean of a $Beta(a, b)$ is $\frac{a}{a+b}$
- we can simulate data from the posterior and summarize
  - $\theta \sim Beta(a, b)$

Coin Toss Example:

- Likelihood
  $p(y|\theta) = \theta^{N_H}(1-\theta)^{N_T}$
- Prior $p(\theta) = Beta(a, b)$
- Posterior
  $p(\theta|y) = Beta(a + N_H, b + N_T)$

```
Posterior = Beta(7,13);
x = 0:0.01:1;
y = pdf.(Posterior,x);
Gadfly.plot(x=x,y=y,Geom.line)
# Computing Posterior summary
mean_val = mean(Posterior);
median_val = median(Posterior);
mode_val = mode(Posterior);

sample = rand(Posterior,1000);
mean_val = mean(sample);
median_val = median(sample);
```


Posterior point-estimation: non-standard form

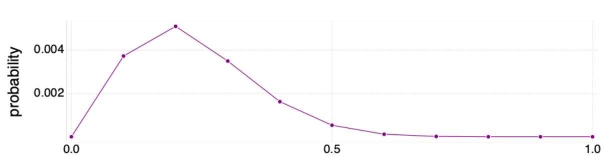
When posterior does not have a standard form

- compute values of the posterior on a grid of points
- we can approximate the posterior by a discrete posterior

Coin Toss Example:

- Likelihood
  $p(y|\theta) = \theta^{N_H}(1-\theta)^{N_T}$
- Prior $p(\theta) \propto 1/(1+\theta)^{3/2}$
- Posterior $p(\theta|y) \propto$
  $\theta^{N_H}(1-\theta)^{N_T} \times 1/(1+\theta)^{3/2}$

```
x = 0:0.1:1;
prior =  1./(1.+x).^1.5;
posterior = (x.^2).*((1.-x).^8).*prior;
Gadfly.plot(x=x,y=posterior,
            Geom.point,Geom.line)
```



Posterior point-estimation

- When we have posterior in a standard functional form
  - we can compute a summary of the distribution analytically
  - we can simulate data from the posterior and summarize
- When posterior does not have a standard form
  - compute values of the posterior on a grid of points
  - we can approximate the posterior by a discrete posterior
    - more points on the grid results in better approximation
  - How to do point estimation?
    - cannot directly use 'rand()' in Julia!
  - high-dimensions (i.e., dimesionality of parameters $\boldsymbol{\theta}$)
    - combinatorially evaluating at grid points on all dimensions
    - computationally prohibitive

Posterior point-estimation: Non-standard form

- Random numbers
- Uniform random numbers
- Drawing random samples from standard distributions
  - Continuous distributions
  - Inverse-transform method
- Discrete distributions
- Mixture representations

Random numbers

- Uniform random variable is very important
  - many other random variables can be derived and transformed from it

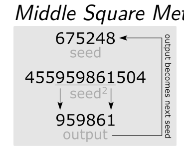
True random numbers:

- based on physical phenomenon (e.g. atmospheric noise, thermal noise, cosmic background radiation) that is known to be random
- very slow

Pseudo random numbers:

- Generated by computational algorithms
- these algorithms produce a long sequence of apparently random results
- they begin with a 'seed'
- the entire random sequence can be reproduced if 'seed' is known

von Neumann's *Middle Square Method*



Pseudo random number generator in Julia

- Mersenne Twister[1]
  - used by every widely distributed mathematical software package

```
rand()
```

```
## 0.6663732297947118
```

- We are not concerned with the mechanics of MT
- We will test this generator to make sure that it produces uniform variables
  - by plotting histogram of x_i's
  - by plotting pairs (x_i, x_{i+1})
  - by estimating autocorrelation function
- Relying on MT, we will see how to generate random numbers for well-known and new probability distributions.

[1] M. Matsumoto and T. Nishimura. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator." ACM Transactions on Modeling and Computer Simulation, 8(1):3-30. 1998

Testing MT in Julia

```
rand(MersenneTwister(12),5)
```

```
## 5-element Array{Float64,1}:
##  0.25851
##  0.969254
##  0.474177
##  0.434506
##  0.96579
```

```
rand(5)
```

```
## 5-element Array{Float64,1}:
##  0.941746
##  0.633589
##  0.780107
##  0.365629
##  0.546959
```

```
rand(MersenneTwister(12),5)
```

```
## 5-element Array{Float64,1}:
##  0.25851
##  0.969254
##  0.474177
##  0.434506
##  0.96579
```

Same seed, same sequence!

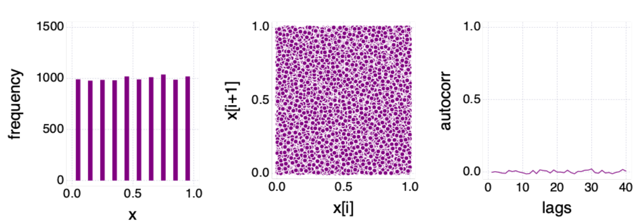
```
rand(5)
```

```
## 5-element Array{Float64,1}:
##  0.807338
##  0.794705
##  0.543888
##  0.995793
##  0.370127
```

Different seed, diff. sequence!

Testing uniform random numbers

```
x = rand(10000);
myplot1 =  plot(x=x, Geom.histogram(bincount=10));
myplot2 =  plot(x=x[1:end-1],y=x[2:end], Geom.point);
myplot3 =  plot(x=1:40,y=autocov(x,1:40), Geom.line);
myplot = hstack(myplot1,myplot2,myplot3);
draw(PNG("./figs/mt_rv_test.png", 7inch, 2inch), myplot);
```

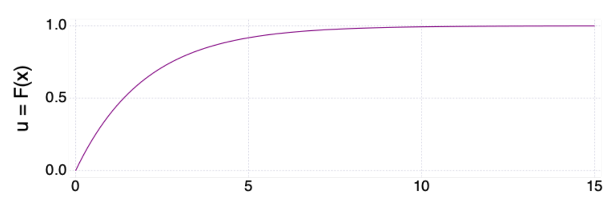


The inverse transform

- **Probability integral transform**: Any random variable can be transformed into a uniform random variable, and vice versa.
- If x has density $f(x)$ and cumulative dist. function $F(x)$, then we have

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

- If we set $u = F(x)$, then $u \sim \mathcal{U}(0, 1)$
  $p(u \leq u_a) = p[F(x) \leq F(x_a)] = p[F^{-1}(F(x)) \leq F^{-1}(F(x_a))] = P(x \leq x_a)$



The inverse transform

For an arbitrary random variable x with cdf F, define the generalized inverse of F by

$$F^{-1}(u) = \inf\{x; F(x) \geq u\}$$

If $u \sim \mathcal{U}(0,1)$, then $F^{-1}(u)$ is distributed like x.

So, using a uniform random number generator, we can draw samples that follow a given cdf F

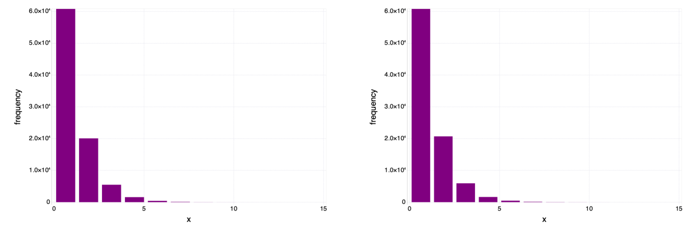
Example: Develop a procedure to draw samples for $x \sim Exp(\lambda = 1)$ with density $f(x) = \lambda e^{-\lambda x} = e^{-x}$, using a uniform random number generator?

- $F(x) = \int_0^x e^{-t} dt = 1 - e^{-x}$
- Set $u = F(x) = 1 - e^{-x}$
- Solving for x, $x = F^{-1}(u) = -log(1-u)$
- Draw $u \sim \mathcal{U}(0,1)$, then compute $x = -\log(u) \sim Exp(\lambda = 1)$
  - as u and $1-u$ are both uniform

Testing uniform random numbers

```
u = rand(100000);
x = -log.(u);
myplot =  plot(x=x,
        Geom.histogram(bincount=10));
```

```
d = Exponential(1);
x = rand(d,100000);
myplot =  plot(x=x,
        Geom.histogram(bincount=10));
```



Histogram of samples drawn using inverse transform method and those drawn from the exponential distribution.

General transformation methods

- When a distribution with density f is linked in a relatively simple way to another distribution that is easy to simulate
  - this can be exploited to construct an algorithm to simulate variables from f.

Example: If the x_i's are i.i.d Exp($\lambda = 1$) random variables, then three standard distributions can be derived as

$$y = 2\sum_{j=1}^{\nu} x_j \sim \mathrm{X}^2_{2\nu}, \qquad \nu \in \mathbb{N}^*$$

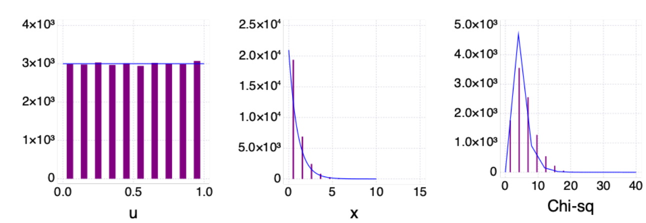
$$y = \beta\sum_{j=1}^{\alpha} x_j \sim Gamma(\alpha, \beta), \qquad \alpha \in \mathbb{N}^*$$

$$y = \frac{\sum_{j=1}^{\alpha} x_j}{\sum_{j=1}^{\alpha+\beta} x_j} \sim Beta(\alpha, \beta), \qquad \alpha, \beta \in \mathbb{N}^*$$

where $\mathbb{N}^* = \{1, 2, \ldots\}$

Julia example

```
u = rand(10000,3);
x = -log.(u);
chi_sq = 2*sum(x,2);
myplot1 =  plot(x=u[:], Geom.histogram(bincount=10));
myplot2 =  plot(x=x[:], Geom.histogram(bincount=10));
myplot3 =  plot(x=chi_sq, Geom.histogram(bincount=10));
myplot = hstack(myplot1,myplot2,myplot3);
```



Discrete Distributions

- Inverse transform principle can be used to construct a generic algorithm
  - works for any discrete distribution
- To generate $x \sim p$, where p is supported by integers
  1. Compute the probabilities

     $$p_0 = p(x \leq 0), \qquad p_1 = p(x \leq 1), \qquad p_2 = p(x \leq 2), \ldots$$

  2. Generate $u \sim \mathcal{U}(0,1)$
  3. Take $x = k$, if $p_{k-1} < u < p_k$
- Example: To generate $x \sim Binomial(10, 0.3)$, probability values are obtained using

```
cdf(Binomial(10,0.3),0:10)
```

```
## 1×11 RowVector{Float64,Array{Float64,1}}:
##  0.03  0.15  0.38  0.65  0.85  0.95  0.99  1.0  1.0  1.0  1
```


Mixture representations

- Probability distributions can be naturally represented as a mixture distribution

  $$f(x) = \int_y g(x|y)p(y)dy \qquad \text{or} \qquad f(x) = \sum_{i \in Y} p_i f_i(x)$$

  where g and p are standard distributions that can easily be simulated.
- To generate a variable x
  1. First generate y from $p(y)$
  2. Then generate x from the conditional $g(x|y)$
- For continuous x, if $y \sim p(y)$ and $x \sim g(x|y)$, then $x \sim f(x)$
- For discrete x, if $\gamma \sim p(\gamma = i) = p_i$ and $x \sim f_\gamma(x)$, then $x \sim f(x)$
- Same as the 'simulation approach' used for marginalizing 'nuisance' parameters.

Summary

- Point-estimation
  - Standard form
  - Non-standard form
- Random numbers
  - True vs. Pseudo random numbers
- Generating random numbers according to standard distributions
  - Inverse transform method
  - Continuous, discrete, mixtures