STAT 576 Bayesian Analysis

Lecture 7: Bayesian Computation

Chenceng Cai

Washington State University

Monte Carlo Methods

- Suppose we have $x_1, \ldots, x_n$ i.i.d. from a distribution $p(x)$.
- Let $f(x)$ be a measurable function with finite expectation under p.
- By law of large numbers, we have

$$\bar{f}_n = \frac{1}{n}\left(f(x_1) + f(x_2) + \cdots + f(x_n)\right) \xrightarrow{P} \mathbb{E}[f(x)] = \int f(x)p(x)d\mu(x)$$

- By central limit theorem, we have

$$\sqrt{n}\left(\bar{f}_n - \mathbb{E}[f(x)]\right) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \sigma^2),$$

where

$$\sigma^2 = \mathrm{Var}[f(x)] = \int (f(x) - \mathbb{E}[f(x)])^2 p(x)d\mu(x)$$

Monte Carlo Methods

Now we consider the reverse.

- If we want to compute the following integral:

$$I = \int_D f(x) d\mu(x)$$

- Method 1:
  - Generate $x^{(1)}, \ldots, x^{(n)}$ i.i.d. and uniformly from D.
  - Estimate the integral by the sample mean:

$$\hat{I}_n = |D| \frac{f(x^{(1)}) + f(x^{(2)}) + \cdots + f(x^{(n)})}{n}$$

  - Variance:

$$\mathrm{var}[\hat{I}_n] = \frac{|D|^2}{n} \mathrm{Var}_{\mathsf{unif}}[f(x)] = \frac{|D|^2}{n} \int_D \left(f(x) - \frac{I}{|D|} \right)^2 \frac{1}{|D|} d\mu(x)$$

Monte Carlo Methods

- Method 2:
  - Generate $x^{(1)}, \ldots, x^{(n)}$ i.i.d. from a non-uniform distribution $p(x)$ on D.
  - Estimate the integral by the sample mean:

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^{n} \frac{f(x^{(i)})}{p(x^{(i)})}$$

  - Variance:

$$\mathrm{Var}[\hat{I}_n] = \frac{1}{n} \mathrm{Var}_p \left[\frac{f(x)}{p(x)} \right] = \frac{1}{n} \int_D \left(\frac{f(x)}{p(x)} - I \right)^2 p(x) d\mu(x)$$

  - $p(x)$ is known as the **sampling** distribution.
  - The sampling distribution that minimizes the variance of $\hat{I}_n$ is

$$p(x) \propto f(x)$$

Monte Carlo Methods

$$I = \int_D f(x) d\mu(x)$$

- The optimal sampling distribution is

$$q(x) = \frac{f(x)}{I}$$

- For any sampling distribution $p(x)$, we have

$$\mathrm{Var}[\hat{I}_n] = \frac{I^2}{n} \underbrace{\int_D \left(\frac{q(x)}{p(x)} - 1 \right)^2 p(x) d\mu(x)}_{\chi^2\text{-divergence: } \chi^2(q||p)}$$

- The variance of the Monte Carlo estimator depends on the χ^2 divergence between the sampling distribution and the optimal one.
- In practice, $q(x)$ is not always tractable. We should choose tractable $p(x)$ that is close to $q(x)$.

Example 1

We want to compute the following integral

$$\int_0^1 (1 - 2|x - 0.5|)\, dx$$

Method 1: draw samples from $\mathrm{unif}[0, 1]$.

```
f <- function(x){1 - 2*abs(x-0.5)}

n = 20
r = 100

Ihat_unif = rep(0, r)
for(i in 1:r){
  x = runif(n)
  Ihat_unif[i] = mean(f(x))
}
```

Example

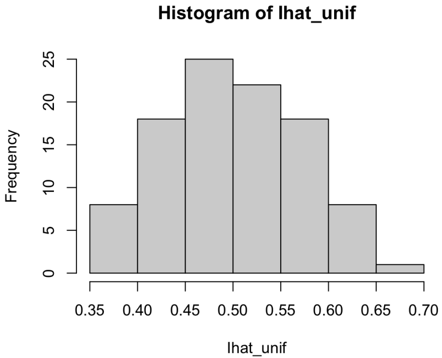
$$\int_0^1 (1 - 2|x - 0.5|)\,dx$$

Method 1 + vectorization: draw samples from $\mathrm{unif}[0, 1]$.

```
n = 20
r = 100

x = matrix(runif(n*r), ncol = r)
Ihat_unif = colMeans(f(x))
hist(Ihat_unif)
```

- Runtime without vectorization: 0.346 ms
- Runtime with vectorization: 0.025 ms

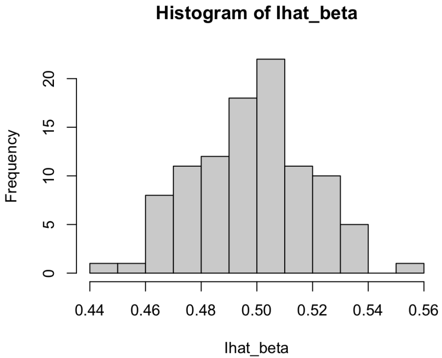


Example

$$\int_0^1 (1 - 2|x - 0.5|)\, dx$$

Method 2: draw samples from $\mathrm{Beta}(2, 2)$.

```
x = matrix(rbeta(n*r, 2, 2), ncol=r)
Ihat_beta = colMeans(f(x) / dbeta(x,
    2, 2))
hist(Ihat_beta)
```

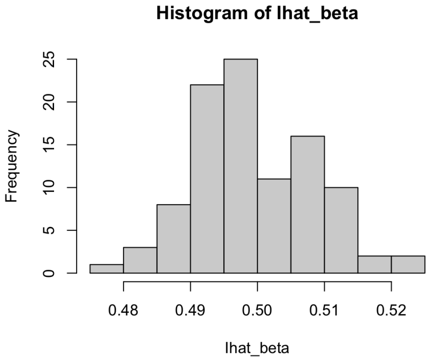


Example

$$\int_0^1 (1 - 2|x - 0.5|)\, dx$$

Method 3: draw samples from $\mathrm{Beta}(2, 2)$ with more MC samples.

```
n = 100
x = matrix(rbeta(n*r, 2, 2), ncol=r)
Ihat_beta = colMeans(f(x) / dbeta(x,
     2, 2))
hist(Ihat_beta)
```



Quasi Monte Carlo Methods

- Monte Carlo method: draw $x^{(1)}, \ldots, x^{(n)}$ i.i.d. from a sampling distribution.
- Quasi Monte Carlo method: pick $x^{(1)}, \ldots, x^{(n)}$ to represent the sampling distribution.
- The samples in the quasi Monte Carlo method are deterministic and are assume to be "uniform" in the whole space.
- The sample sequence $x^{(1)}, x^{(2)}, \ldots$ is called **low discrepancy sequence** (e.g. Sobel sequence).

Example

$$\int_0^1 (1 - 2|x - 0.5|)\, dx$$

Method 4: QMC samples from $\mathrm{unif}[0, 1]$.

```
x = (seq(n)-0.5)/n
Ihat_unif_qmc = mean(f(x))
print(Ihat_unif_qmc)
```

The outcome is 0.5.

Example

$$\int_0^1 (1 - 2|x - 0.5|)\, dx$$

Method 5: QMC samples from $\mathrm{Beta}(2, 2)$.

```
x = (seq(n)-0.5)/n
y = qbeta(x, 2, 2)
Ihat_beta_qmc = mean(f(y)/dbeta(y, 2, 2))
print(Ihat_beta_qmc)
```

The outcome is 0.50002.

Random Number Generator

- Most currently used random number generators on modern computers are **pseudo random number generators (PRNG)**.
- PRNG is a deterministic sequence that requires a starting value (known as **seed**).
- The sequence generated by PRNG behaves like independent random numbers.
- The sequence generated by PRNG will finally repeat.
- Two sequences generated by the same PRNG and the same seed should be identical.
- Common practices:
  - Set the seed at the beginning of your program for easy replication of the results.

    ```
    set.seed(0)
    ```

  - Do not abuse it! Use a predetermined seed instead of optimizing it.

Generating Random Numbers

- The default random numbers generated by PRNG are i.i.d. $\text{unif}[0, 1]$.
- How do we generate random numbers from an arbitrary univariate distribution F?
  - Transformation.
  - Inverse C.D.F.
  - Accept-reject sampling.

Generating Random Numbers — Transformation

Let $u_1, u_2, \ldots$ be a sequence of i.i.d. $\mathrm{unif}[0,1]$ random variables.

- Let $z_i = \mathbb{I}\{u_i > 0.5\}$.
  Then $z_1, z_2, \ldots$ is an i.i.d. sequence of $\mathrm{Bernoulli}(0.5)$ random variables.
- Let $y_i = \sum_{j=1}^{n} z_{n(i-1)+j}$.
  Then $y_1, y_2, \ldots$ is an i.i.d. sequence of $\mathrm{Binomial}(n, 0.5)$ random variables.
- Let $d_i^j = \lfloor 2^j u_i \rfloor \mod 2$. That is $u_i = 0.d_i^1 d_i^2 d_i^3 \ldots$ is a base-2 representation.
  Then d_i^j's are i.i.d. $\mathrm{Bernoulli}(0.5)$.
- Let $x_i = \sum_{j=1}^{n} d_i^j$.
  Then $x_1, x_2, \ldots$ is an i.i.d. sequence of $\mathrm{Binomial}(n, 0.5)$ random variables.
- Let $w_i = 2u_i$.
  Then $w_1, w_2, \ldots$ is an i.i.d. sequence of $\mathrm{unif}[0,2]$ random variables.
- Let $r_i = -\log u_i$.
  Then $r_1, r_2, \ldots$ is an i.i.d. sequence of $\mathrm{Exp}(1)$ random variables.

Generating Random Numbers — Inverse C.D.F.

A special type of transformation is using the inverse c.d.f. function.

- The c.d.f. of a distribution F is given by

$$F(x_0) = \mathbb{P}[x \leq x_0]$$

- The inverse c.d.f. is given by

$$F^{-1}(q) = \inf\ \{x : F(x) \geq q\}$$

- If $u_1, u_2, \ldots$ is an i.i.d. sequence of $\mathrm{unif}[0,1]$ random variables, then $F^{-1}(u_1), F^{-1}(u_2), \ldots$ is an i.i.d. sequence of F random variables.
- **Justification:**

$$\mathbb{P}[F^{-1}(u_1) \leq x_0] = \mathbb{P}[u_1 \leq F(x_0)] = F(x_0)$$

Example: Generating Standard Normal Random Variables

Method 1: approximated inverse c.d.f.
We approximate the inverse c.d.f. of a standard normal by (for $0 < q < 1/2$)

$$\Phi^{-1}(q) \approx t - \frac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2 + d_3 t^3}$$

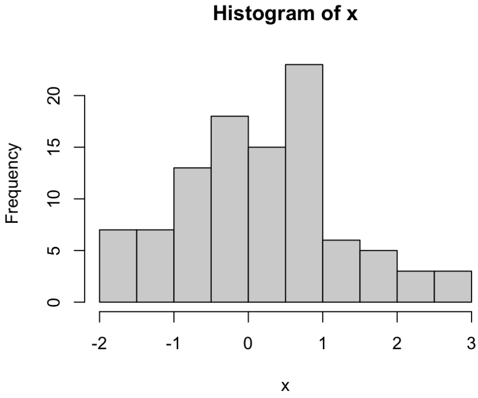
for $t = \sqrt{-2 \log q}$ and

$$c_0 = 2.515517 \qquad d_1 = 1.432788$$
$$c_1 = 0.802853 \qquad d_2 = 0.189269$$
$$c_2 = 0.010328 \qquad d_3 = 0.001308$$

Example: Generating Standard Normal Random Variables

```
c0 = 2.515517
c1 = 0.802853
c2 = 0.010328
d1 = 1.432788
d2 = 0.189269
d3 = 0.001308

u = runif(100)
t = sqrt(-2*log(abs(u-0.5)))
denum = c0 + c1*t + c2*t**2
num = 1 + d1*t + d2*t**2 + d3*t**3
x = t - denum/num
x = x * sign(u - 0.5)
hist(x)
```



Example: Generating Standard Normal Random Variables

Method 2: Box-Muller transformation.

- Assume x_1 and x_2 are independent standard normal random variables.
- The joint density is

$$p(x_1, x_2) \propto e^{-\frac{x_1^2 + x_2^2}{2}}$$

- Consider the following transformation

$$r = \sqrt{x_1^2 + x_2^2} \qquad x_1 = r\cos\theta$$

$$\theta = \arctan\frac{x_2}{x_1} \qquad x_2 = r\sin\theta$$

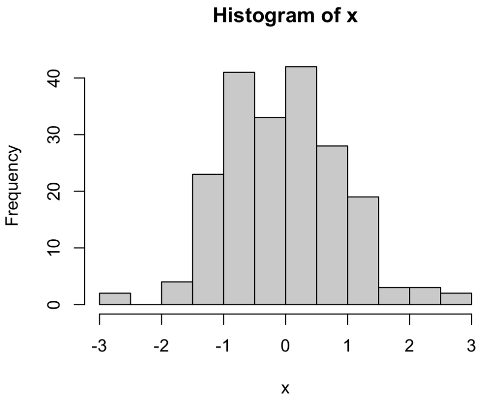
- The density for (r, θ) is

$$p(r, \theta) = p(x_1, x_2)\left|\frac{\partial(x_1, x_2)}{\partial(r, \theta)}\right| \propto re^{-r^2/2}$$

- $\theta \sim \mathrm{unif}[0, 2\pi)$ and $p(r) \propto re^{-r^2/2}$ with c.d.f. $1 - e^{-r^2/2}$ (i.e. $r^2 \sim \mathrm{Exp}(1/2)$)

Example: Generating Standard Normal Random Variables

```
u = runif(100)
theta = runif(100) * 2 * pi
r = sqrt(-2*log(u))
x1 = r * sin(theta)
x2 = r * cos(theta)
x = c(x1, x2)
hist(x)
```



Accept-Reject Sampling

Generate random variables that are uniform in a unit circle.

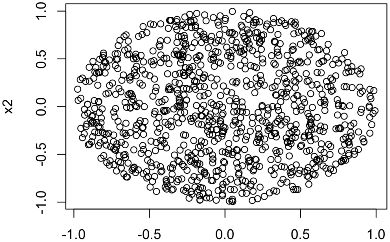
Method 1: Transformation.

We use the polar coordinate (r, θ) instead of (x_1, x_2).

$$p(r, \theta) = p(x_1, x_2) \left| \frac{\partial(x_1, x_2)}{\partial(r, \theta)} \right| \propto r$$

- generate $\theta \sim \mathrm{unif}[0, 2\pi)$.
- generate $p(r) \propto r$ (use inverse c.d.f.)

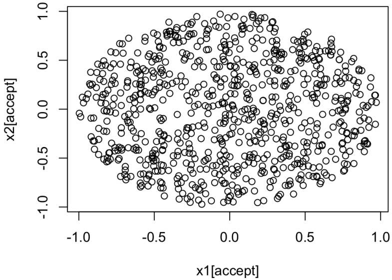
```
n = 1000
r = sqrt(runif(n))
theta = runif(n, 0, 2*pi)
x1 = r*cos(theta)
x2 = r*sin(theta)
plot(x1, x2)
```



Accept-Reject Sampling

Method 2: Accept-Reject Sampling (naive version).
We can generate (x_1, x_2) uniformly from $[-1, 1] \times [-1, 1]$
and **only keep** the samples that are in the unit circle.

```
x1 = runif(n, -1, 1)
x2 = runif(n, -1, 1)
accept = (x1**2 + x2**2) <= 1
plot(x1[accept], x2[accept])
```



Accept-Reject Sampling

In general, we call such an algorithm **Accept-Reject Algorithm** (**Rejection Sampling**) that generate a set of samples and then take a subset of them.

The general accept-reject sampling: (target distribution F supported on $\mathcal{X}$)

- Draw $x^{(1)}, \ldots, x^{(n)}$ i.i.d. from G
- For each $i = 1, \ldots, n$, accept $x^{(i)}$ with probability

$$\frac{f(x^{(i)})}{c \cdot g(x^{(i)})}$$

  for some constant $c > 0$.

Conditions:

- F is absolutely continous with respect to G: $\mathrm{supp}(G) \supseteq \mathrm{supp}(F)$
- The constant $c > 0$ satisfies

$$f(x) \leq c \cdot g(x) \; \forall x \in \mathcal{X}$$

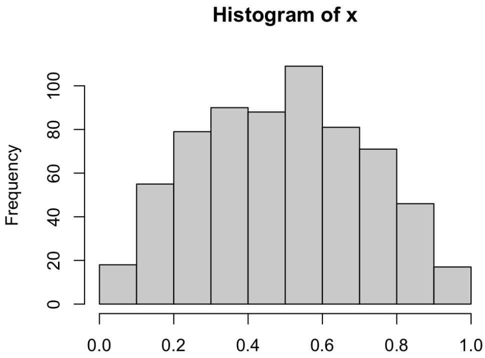
Example

Generate random variables from the $\mathrm{Beta}(2,2)$ distribution.

- Consider a sampling distribution using $\mathrm{unif}[0,1]$.
- The constant c should satisfy

$$c \geq \sup_x \frac{\mathrm{Beta}(x;2,2)}{\mathrm{unif}(x;0,1)} = \frac{\mathrm{Beta}(1/2;2,2)}{\mathrm{unif}(1/2;0,1)}$$

```
n = 1000
c = dbeta(0.5, 2, 2)
x = runif(n)
p_accept = dbeta(x, 2, 2)/c
x = x[runif(n) <= p_accept]
hist(x)
```



Accept-Reject Sampling

The probability of acceptance:

$$
\begin{aligned}
p[x^{(1)} \text{ is accepted}] &= \mathbb{E}_g\left[p[x^{(1)} \text{ is accepted} \mid x^{(1)} = x]\right] \\
&= \int_{\mathcal{X}} p[x^{(1)} \text{ is accepted} \mid x^{(1)} = x] g(x) d\mu(x) \\
&= \int_{\mathcal{X}} \frac{f(x)}{c \cdot g(x)} g(x) d\mu(x) = \frac{1}{c}
\end{aligned}
$$

Distribution density after acceptance:

$$
p[x^{(1)} = x \mid x^{(1)} \text{ is accepted}] = \frac{p[x^{(1)} = x \text{ and } x^{(1)} \text{ is accepted}]}{p[x^{(1)} \text{ is accepted}]} = \frac{g(x)\frac{f(x)}{c \cdot g(x)}}{1/c} = f(x)
$$

Accept-Reject Sampling

- We only need to know the densities f and g up to a constant (i.e. in proportional form). (The constants are absorbed into c.)
- We should choose c as small as possible to increase acceptance rate.
- c is lower bounded by $\sup \; f(x)/g(x)$.
- We should choose g to minimize the ratio.

- The major drawback of accept-reject sampling is that we have to discard some samples.
- To make full use of all samples, we should consider importance sampling.

Weighted Sample

Let $\{x^{(i)}\}_{i=1}^n$ be a sample. If we equip each value $x^{(i)}$ with a **nonnegative weight** $w^{(i)}$, then $\{(x^{(i)}, w^{(i)})\}_{i=1}^n$ is called a (unnormalized) **weighted sample**.

- (weighted) sample mean of $f(x)$:

$$\bar{f} = \frac{\sum_{i=1}^n w^{(i)} f(x^{(i)})}{\sum_{i=1}^n w^{(i)}}$$

- (weighted) sample variance of $f(x)$: (fixing weights)

$$\mathrm{Var}[\bar{f}] = \frac{\sum_{i=1}^n \left(w^{(i)}\right)^2}{\left(\sum_{i=1}^n w^{(i)}\right)^2} \mathrm{Var}[f(x)]$$

- We define the **effective sample size** by

$$\mathrm{ESS} := \frac{\left(\sum_{i=1}^n w^{(i)}\right)^2}{\sum_{i=1}^n \left(w^{(i)}\right)^2}$$

Weighted Sample

The weighted sample $\{(x^{(i)}, w^{(i)})\}_{i=1}^{n}$ is called **properly weighted** w.r.t. $p(x)$ if for any "regular" function f, we have

$$\frac{\sum_{i=1}^{n} w^{(i)} f(x^{(i)})}{\sum_{i=1}^{n} w^{(i)}} \xrightarrow{P} \mathbb{E}_P[f(x)]$$

Remarks

- The weights do not have to be normalized. In most cases, we have a proportional form for them.
- In many cases, the weights are also random (depending on x). The previous variance form is an approximation.
- But the effecitve sample size tells how unevenly the weights are distributed.

Importance Sampling

The importance sampling adjusts the weight of the samples if the sampling distribution and the target distribution differ.

Importance Sampling for target distribution P

- Draw (unweighted) samples $\{x^{(i)}\}_{i=1}^n$ from the **sampling distribution** Q.
- Set the weights by

$$w^{(i)} = \frac{p(x^{(i)})}{q(x^{(i)})}$$

- $\{(x^{(i)}, w^{(i)})\}_{i=1}^n$ is a weighted sample that is properly weighted w.r.t. P.

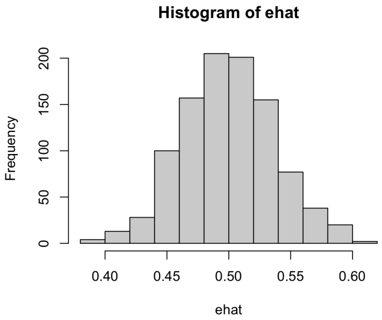
Justification:

$$\frac{\sum_{i=1}^n w^{(i)} f(x^{(i)})}{\sum_{i=1}^n w^{(i)}} \xrightarrow{P} \frac{\mathbb{E}_Q[wf(x)]}{\mathbb{E}_Q[w]} = \frac{\int \frac{p(x)}{q(x)} f(x) q(x) d\mu(x)}{\int \frac{p(x)}{q(x)} q(x) d\mu(x)} = \mathbb{E}_P[f(x)]$$

Example

Estiamte the expectation of $\mathrm{Beta}(2, 2)$ distribution.
Method 1: accept-reject sampling from $\mathrm{unif}[0, 1]$.

```
n = 50
r = 1000
c = dbeta(0.5, 2, 2)
x = matrix(runif(n*r), ncol=r)
p_accept = dbeta(x, 2, 2)/c
accept = runif(n*r) <= p_accept
ehat = colSums(x * accept) / colSums(accept)
hist(ehat)
```

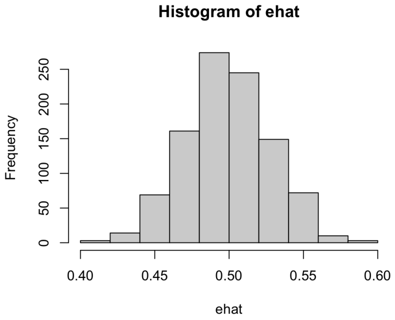


Execected sample size: $n/c \approx 33$.

Example

Estiamte the expectation of $\mathrm{Beta}(2, 2)$ distribution.
Method 2: importance sampling from $\mathrm{unif}[0, 1]$.

```
x = matrix(runif(n*r), ncol=r)
w = dbeta(x, 2, 2)
ehat = colSums(x * w) / colSums(w)
hist(ehat)
```



Expected effective sample size: $n/\mathbb{E}[w^2] \approx 42$.

Importance Sampling

- We only need to know the densities up to a constant (in proportional form).
- P should be absolutely continous w.r.t. Q.
- Q should be easy to sample from.
- The effective sample size depends on the distance between P and Q.
- **Change-of-measure property** for the importance sampling:
  If $\{x^{(i)}, w^{(i)}\}_{i=1}^n$ is properly weighted w.r.t. to a proability measure P, then $\{x^{(i)}, \tilde{w}^{(i)}\}_{i=1}^n$ is properly weighted w.r.t. another probability measure Q if and only if
  1. Q is absoluately continous w.r.t. P.
  2. and
  $$\tilde{w}^{(i)} \propto w^{(i)} \frac{q(x^{(i)})}{p(x^{(i)})}$$

Exercise: How to generate samples from an improper distribution (e.g. $p(x) \propto 1$)?

Importance Resampling

- The major drawback of the importance sampling is the possible **weight collapse**.
- Weight collapse means most of the weights are assigned to few samples.
- Small ESS is an indicator of weight collapse.
- It usually happens when the sampling distribution is significantly different from the target one.

- If weight collapse happens in the last step of sampling, we can mere do anything to reduce variance.
- If it happens in the intermediate step, we can reduce the weight collapse by **importance resampling**.

Importance Resampling

- Let $\{x^{(i)}, w^{(i)}\}_{i=1}^n$ be a weighted sample.
- Assign each data with a nonnegative **priority score** $\beta^{(i)}$.
- Draw $r_1, \ldots, r_m$ i.i.d. from the **Multinomial distribution** with probabilities $\propto \beta^{(i)}$:

$$p(r_j = i) = \frac{\beta^{(i)}}{\sum_{i=1}^n \beta^{(i)}}$$

- The new sample after resampling is $\{\tilde{x}^{(j)}, \tilde{w}^{(j)}\}_{j=1}^m$ with

$$\tilde{x}^{(j)} = x^{(r_j)}, \quad \tilde{w}^{(j)} \propto \frac{w^{(r_j)}}{\beta^{(r_j)}}$$

Importance Resampling

How to sample from multinomial distributions?

- Use the default PRNG for multinomial: inverse c.d.f. + bisectional search.
- Residual sampling:
  - get $\lfloor m\beta^{(i)} / \sum_i \beta^{(i)} \rfloor$ copies of index i.
  - for the rest, use the default multinomial sampling.
- Stratified: divide the indices into clusters and do multinomial sampling within each cluster.

How to choose priority scores?

- $\beta^{(i)} \propto 1$ — wasting time.
- $\beta^{(i)} \propto w^{(i)}$ — default way. resulting in an unweighted sample.
- $\beta^{(i)} \propto \sqrt{w^{(i)}}$ — least aggresive resampling.
- Other customizable priority scores depending on sampling needs.

Practice

- How to generate samples from $p(x, y)$ with known $p(x)$?
  - Draw $\{x^{(i)}\}_{i=1}^{n}$ from $p(x)$
  - Draw $y^{(i)}$ from $p(y \mid x^{(i)})$ for each i.
  - Return $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$

- How to generate samples from $p(x, y)$ with unknown $p(x)$?
  importance sampling / rejection sampling.

- How to generate samples from $p(x)$ with known $p(x, y)$?
  - Draw $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ from $p(x, y)$.
  - Return $\{x^{(i)}\}_{i=1}^{n}$