Probabilistic Graphical Models

Monte Carlo Sampling

Sampling-Based (Monte Carlo) Estimation

Example: Estimate p for a Bernoulli distribution P

Toss a coin m times, compute $\hat{p} = E[X] = \frac{1}{M} \sum X_i$

Theoretical expectation: $\mathrm{p} = E[X] = \sum_x xP(x)$

Empirical Expectation: $\hat{p} = \hat{E}[X] \approx \frac{1}{M} \sum X_i$

Estimator is unbiased

$$E[\hat{p}] = p$$

Asymptotics: $\hat{p} \to p, \hat{p} \sim Norm(p, \frac{\sigma^2}{n})$

Sampling-Based (Monte Carlo) Estimation

Idea: Distribution P, you want to compute $\mathrm{E}[f(\boldsymbol{X})]$

Exact/Approximate Inference is very hard

Sample $D = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$ (i.i.d)

For a distribution P, function f:

Estimate $\widehat{E_P}[f(X)] = \frac{1}{M} \sum f(x[m])$

Properties

Estimator is unbiased: $\mathrm{E}[\hat{f}] = E[\widehat{E_D}[f(X)]] = E_P[f(X)]$

Variance of the estimator: $\sigma_{\hat{f}}^2 = \sigma_f^2 / n$

Weak Law of Large Numbers:

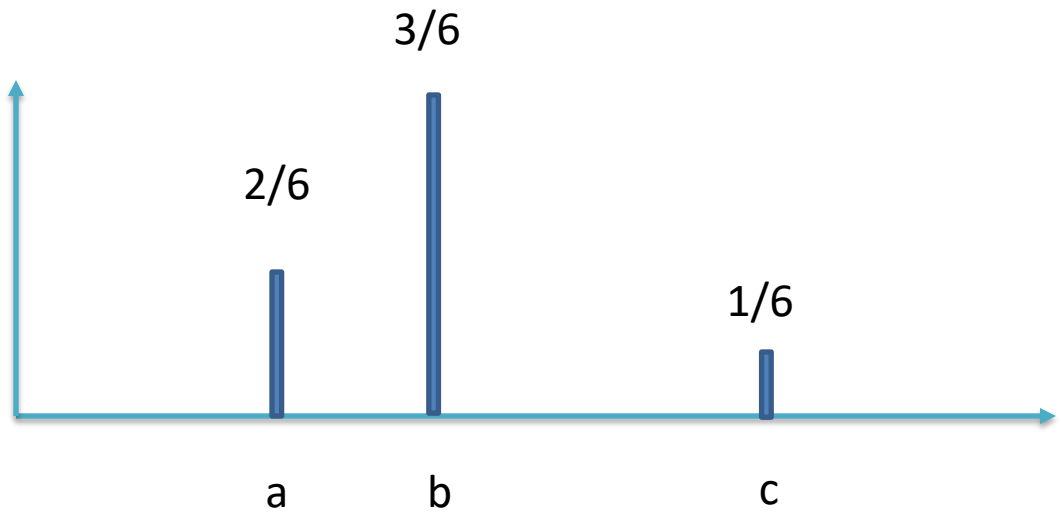
$$\lim_{n\to\infty} P\left(\left|\widehat{f} - E_p[f(X)]\right| < \epsilon\right) = 1$$

CLT: $\hat{f} \sim N(E_P[f(X)], \sigma_{\hat{f}}^2)$

How do we sample?

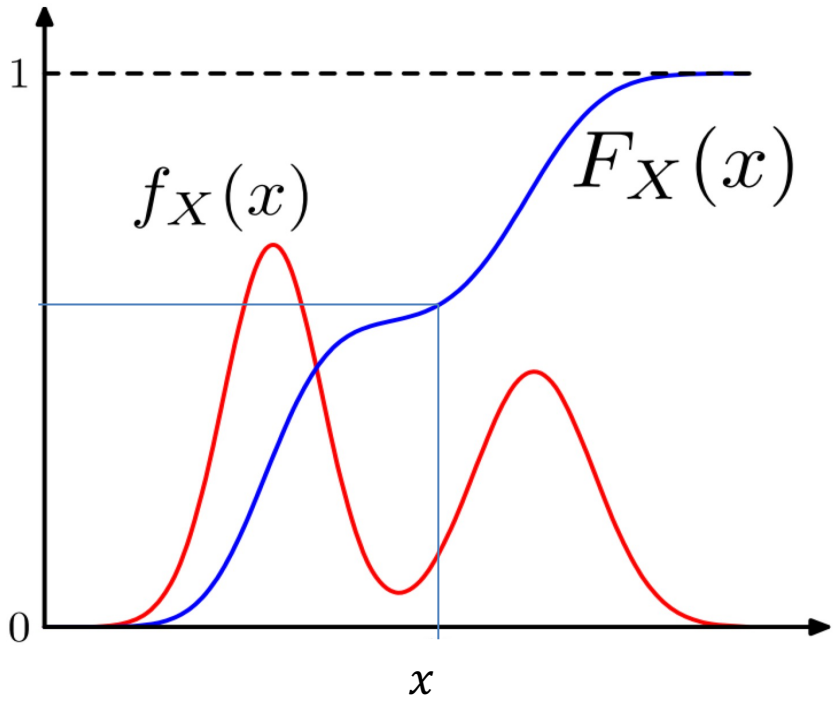
-Computers can produce random numbers from a uniform distribution.

-Inverse CDF transform-based sampling



Let $U_i = \mathrm{F}(\mathrm{x_i})$. Then $U_i \sim Uniform(0, 1)$

How do we sample?



Let F be a strictly monotonic cumulative distribution function, and let F^{-1} be its inverse function

Claim: If U is a uniform random variable on $[0,1]$ then $F^{-1}(U)$ has F as its CDF.

Proof:

$$\Pr(F^{-1}(U) \leq x)$$
$$= \Pr(U \leq F(x))$$
$$= F(x) \text{ (because } \Pr(U \leq u) = u, \text{ when U is uniform on } [0,1])$$

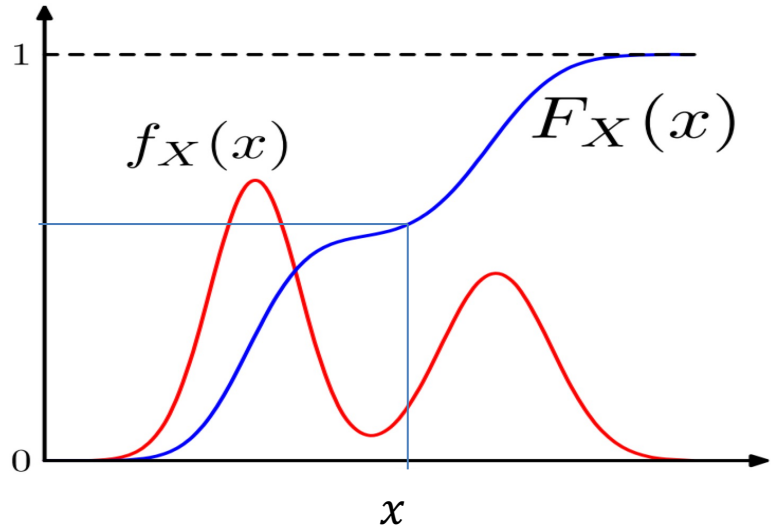
Let $U_i = \mathrm{F}(\mathrm{x_i})$. Then $U_i \sim Uniform(0,1)$

$$P(X_i \leq x)$$
$$= P(F_x^{-1}(u_i) \leq x) = F(x)$$

How do we sample?

1. Generate a random number u from the standard uniform distribution in the interval [0,1], i.e. from $U \sim \text{Unif}[0,1]$.
2. Find the generalized inverse of the desired CDF, i.e. $F_X^{-1}(u)$.
3. Compute $X'(u) = F_X^{-1}(u)$. The computed random variable $X'(U)$ has distribution F_X and thereby the same law as X.

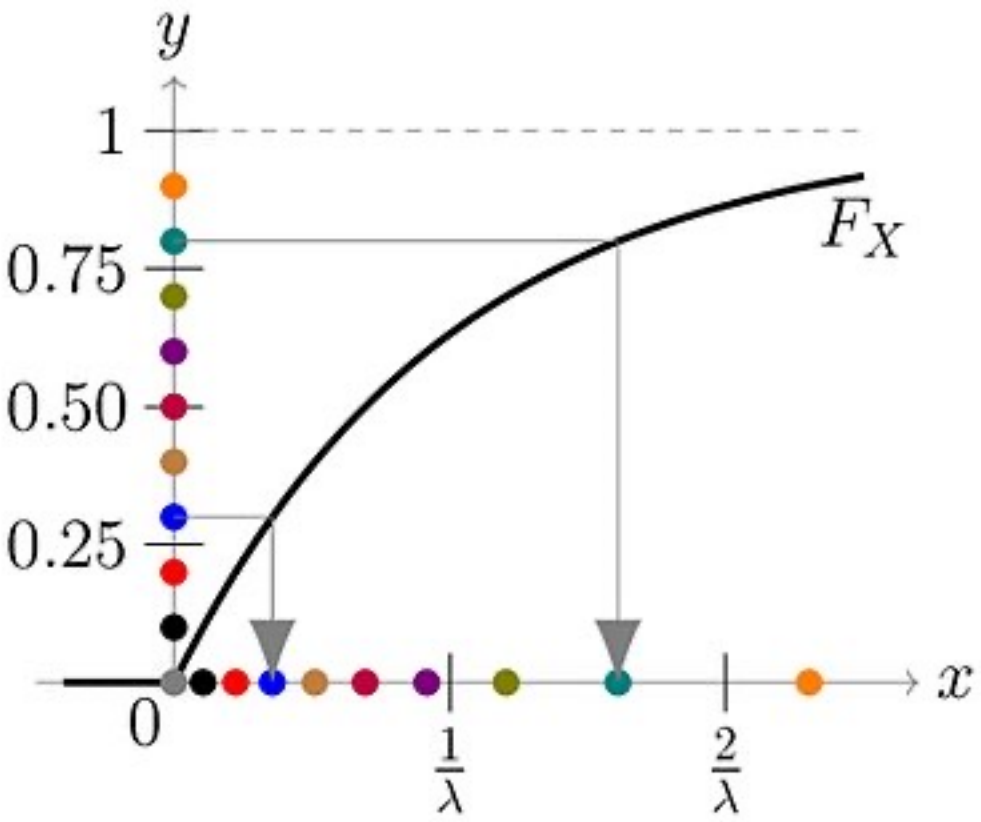
Let $U_i = \mathrm{F}(\mathrm{x_i})$. Then $U_i \sim Uniform(0,1)$



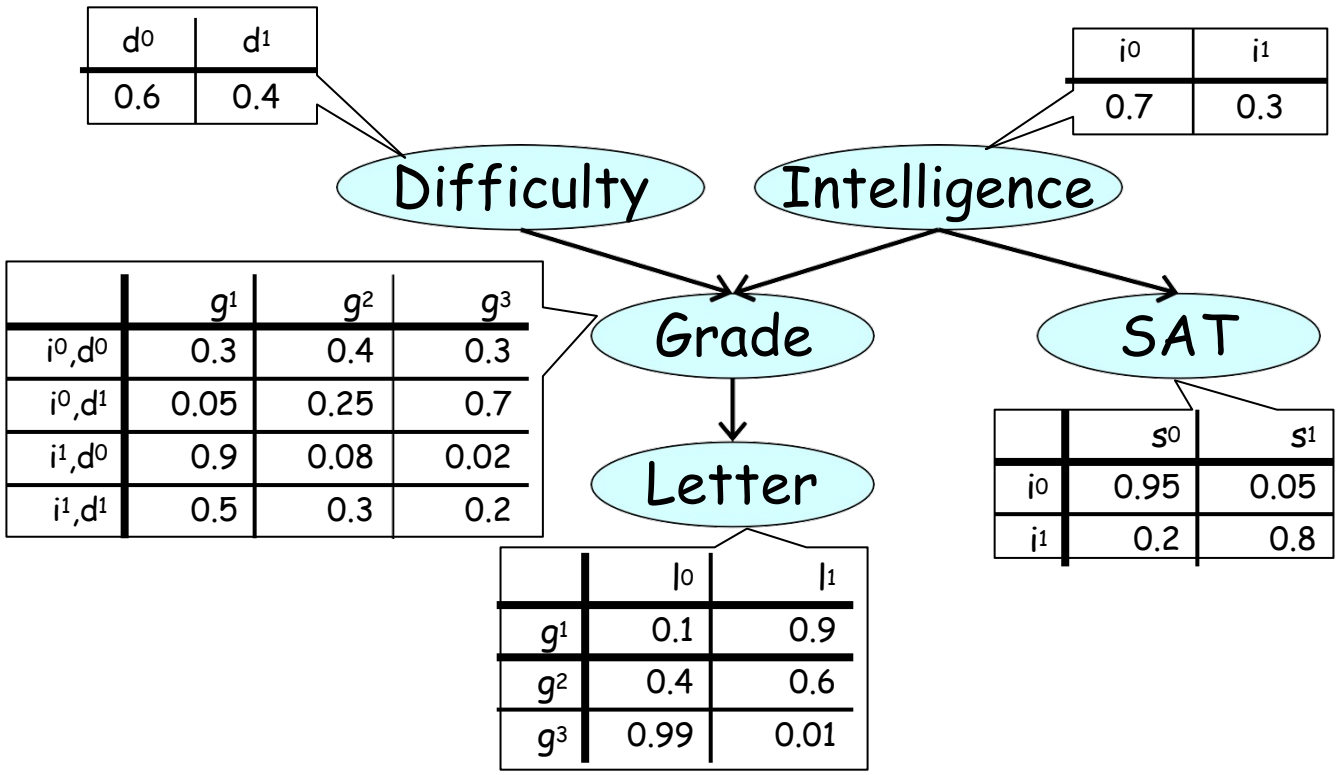
Example: Exponential Distribution

- Draw some u_i from a $U \sim \text{Unif}(0,1)$
- Compute $x_i = F_X^{-1}(u_0) = -\frac{1}{\lambda}\ln(1-u_i)$
- This x_i has exponential distribution.

How about $F(x) = 1 - exp(-\sqrt{x})$



Forward sampling from a BN



Daphne Koller

Sampling-Based (Monte Carlo) Estimation

Idea: Distribution P, you want to compute $P(\boldsymbol{X} = \boldsymbol{x}) = \boldsymbol{E}[I(\boldsymbol{X} = \boldsymbol{x})]$

Exact/Approximate Inference is very hard

Sample $D = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$ (i.i.d)

Estimate $\hat{P}(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{M} \sum I(\boldsymbol{X} = \boldsymbol{x})$

Sampling-Based (Monte Carlo) Estimation

$$T_{\mathcal{D}} = \frac{1}{M}\sum_{m=1}^{M} X[m]$$

Hoeffding Bound:

$$P_{\mathcal{D}}(T_{\mathcal{D}} \notin [p-\epsilon, p+\epsilon]) \leq 2e^{-2M\epsilon^2}$$

For additive bound ϵ on error with probability > 1-δ:

$$M \geq \frac{\ln(2/\delta)}{2\epsilon^2}$$

Chernoff Bound:

$$P_{\mathcal{D}}(T_{\mathcal{D}} \notin [p(1-\epsilon), p(1+\epsilon)]) \leq 2e^{-Mp\epsilon^2/3}$$

For multiplicative bound ϵ on error with probability > 1-δ:

$$M \geq 3\frac{\ln(2/\delta)}{p\epsilon^2}$$

Daphne Koller

Forward Sampling for Querying

- Goal: Estimate P(Y=y)
  - Generate samples from BN
  - Compute fraction where Y=y

For additive bound ϵ on error with probability > 1-δ

$$M \geq \frac{\ln(2/\delta)}{2\epsilon^2}$$

For multiplicative bound ϵ on error with probability > 1-δ

$$M \geq 3\frac{\ln(2/\delta)}{P(y)\epsilon^2}$$

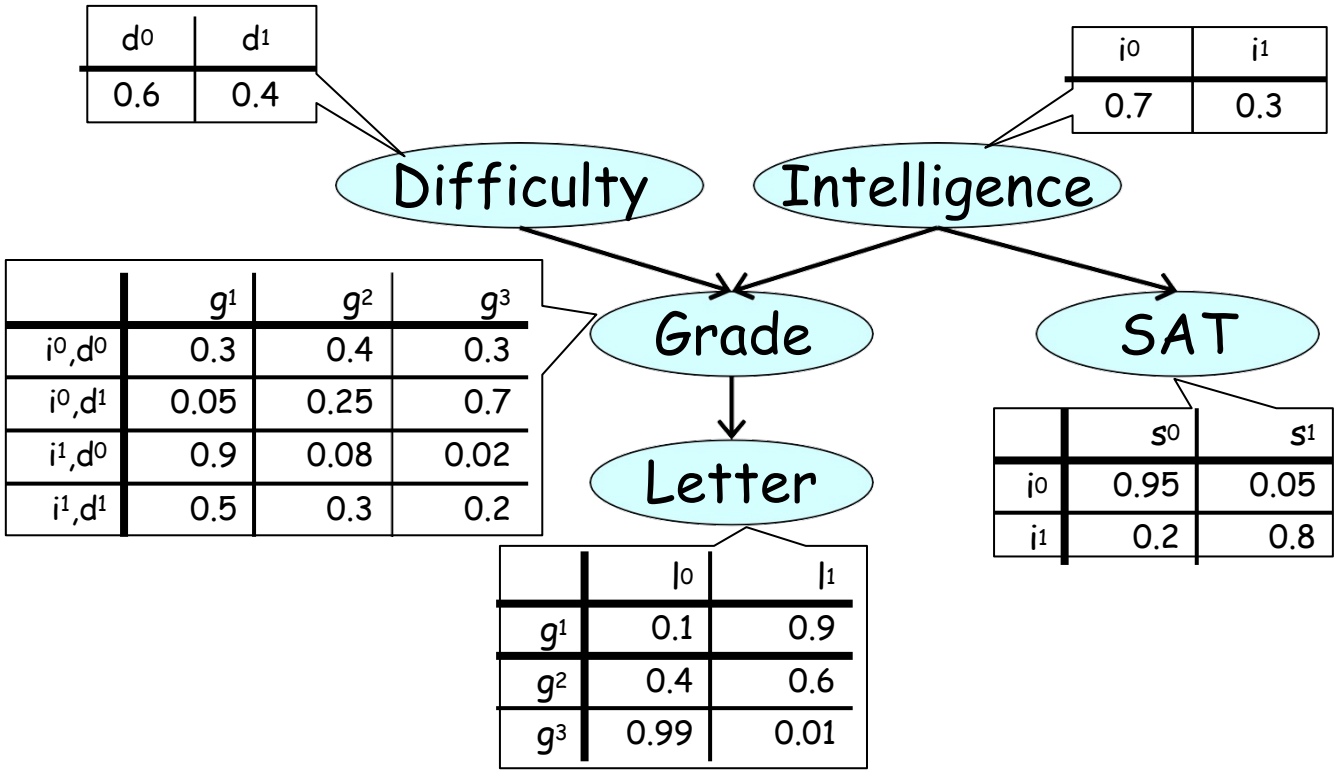
Queries with Evidence

- Goal: Estimate P(Y=y | **E**=**e**)
- Rejection sampling algorithm
  - Generate samples from BN
  - Throw away all those where **E**≠**e**
  - Compute fraction where Y=y

Expected fraction of samples kept ~ P(**e**)

samples needed rows exponentially with # of observed variables

Likelihood Weighting Sampling



Example: Force s^1

$$P(s^1|i^0) = 0.05$$
$$P(s^1|i^1) = 0.8$$

Example: Force s^1, l^0, sample $\mathrm{d}^1, \mathrm{i}^0, g^2$

$$P(l^0|g^2) = 0.4$$
$$P(s^1|i^0) = 0.05$$

Likelihood Weighting Sampling

Algorithm 12.2 Likelihood-weighted particle generation

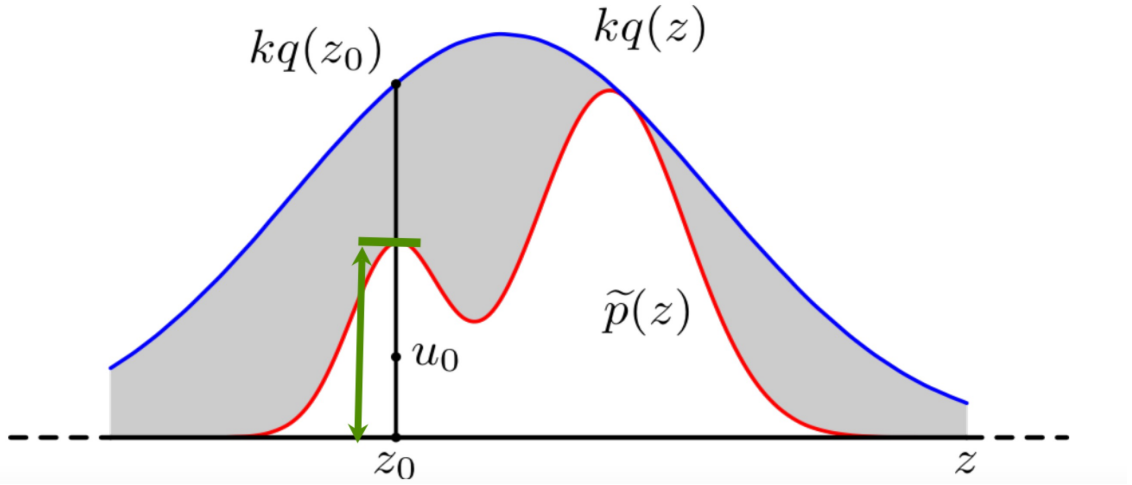
```
    Procedure LW-Sample (
      B,    // Bayesian network over X
      Z = z    // Event in the network
    )
1     Let X_1, ..., X_n be a topological ordering of X
2     w ← 1
3     for i = 1, ..., n
4       u_i ← x⟨Pa_{X_i}⟩    // Assignment to Pa_{X_i} in x_1, ..., x_{i-1}
5       if X_i ∉ Z then
6         Sample x_i from P(X_i | u_i)
7       else
8         x_i ← z⟨X_i⟩    // Assignment to X_i in z
9         w ← w · P(x_i | u_i)    // Multiply weight by probability of desired value
10    return (x_1, ..., x_n), w
```

$$\widehat{P_D}(\boldsymbol{y} \mid \boldsymbol{e}) = \frac{\sum_{m=1}^{M} w[m] 1\{\boldsymbol{y}[m] = \boldsymbol{y}\}}{\sum_{m=1}^{M} w[m]}$$

General Framework: Rejection Sampling

- Access to easy-to-sample distribution $\mathrm{Q}(x)$
- Constant k such that $P(\mathrm{x}) \leq kQ(x)$

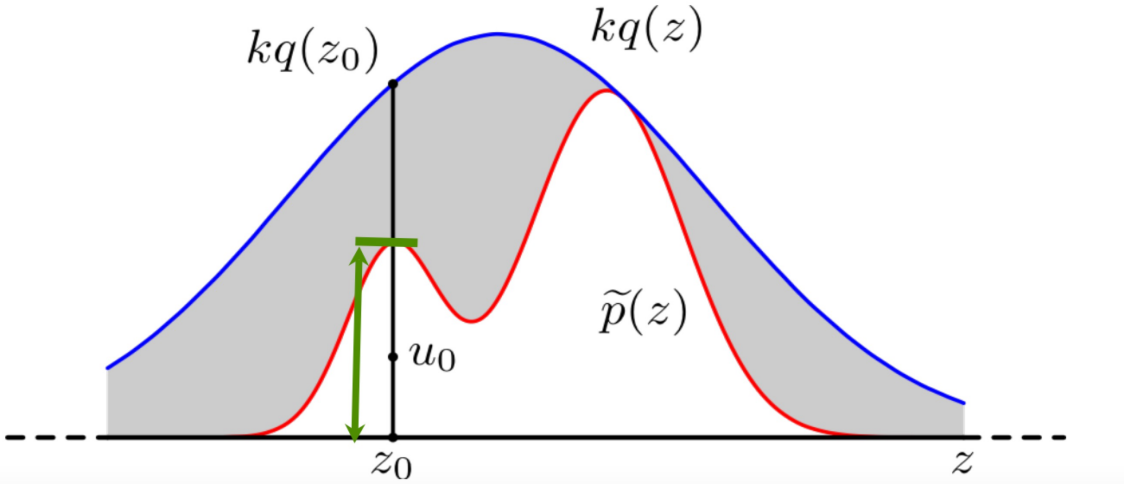
- Algorithm
  1. Sample from $\mathrm{Q}(x)$
  2. Keep samples in proportion to $\frac{P(x)}{k \cdot \mathrm{Q}(z)}$ and reject the rest



General Framework: Rejection Sampling

- Access to easy-to-sample distribution $\mathrm{Q}(x)$
- Constant k such that $\mathrm{P}(\mathrm{x}) \leq kQ(x)$

- Algorithm
  1. Sample $\mathrm{x_i} \sim \mathrm{Q(x)}$
  2. Sample $y_i \sim Uni[0, kQ(x_i)]$
  3. If $\mathrm{P}(x_i) \leq y_i$ keep, otherwise reject.

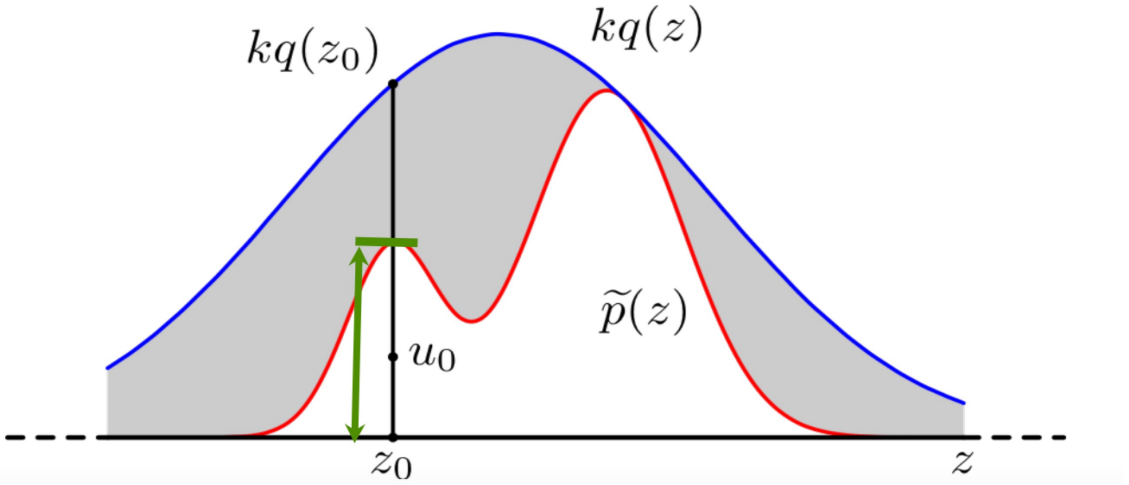


General Framework: Rejection Sampling

- Access to easy-to-sample distribution $\mathrm{Q}(x)$
- Constant k such that $\tilde{P}(\mathrm{x}) \leq kQ(x)$

- Algorithm
  1. Sample $\mathrm{x_i} \sim \mathrm{Q(x)}$
  2. Sample $y_i \sim Uni[0, kQ(x_i)]$
  3. If $\tilde{P}(x_i) \leq y_i$ keep, otherwise reject.

Example Uses Gaussian proposal q to draw samples from multimodal distribution p



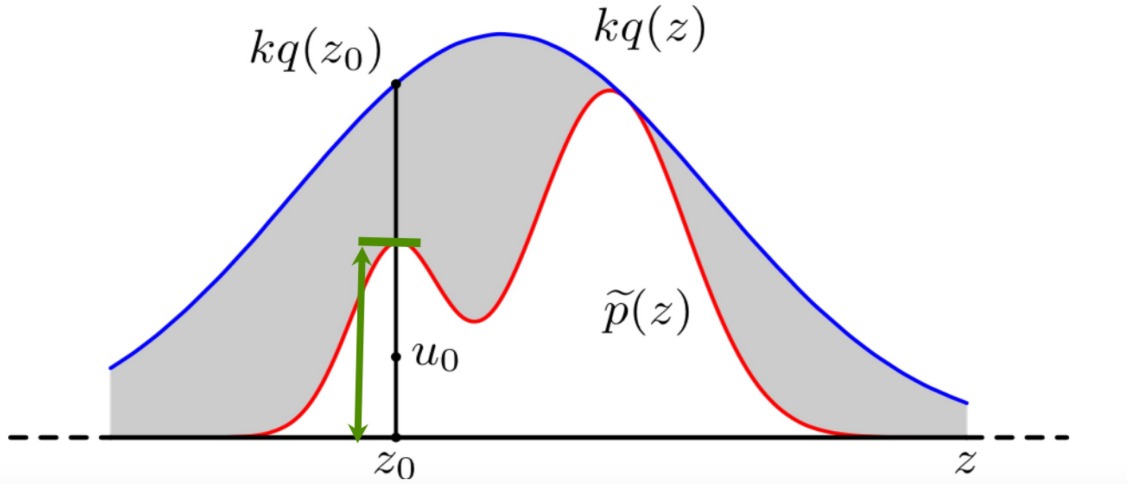
General Framework: Rejection Sampling

- Access to easy-to-sample distribution $\mathrm{Q}(x)$
- Constant k such that $\tilde{P}(\mathrm{x}) \leq kQ(x)$

- Algorithm
  1. Sample $\mathrm{x_i} \sim \mathrm{Q(x)}$
  2. Sample $y_i \sim Uni[0, kQ(x_i)]$
  3. If $\tilde{P}(x_i) \leq y_i$ keep, otherwise reject.

Can be very wasteful
Needs to be bounded

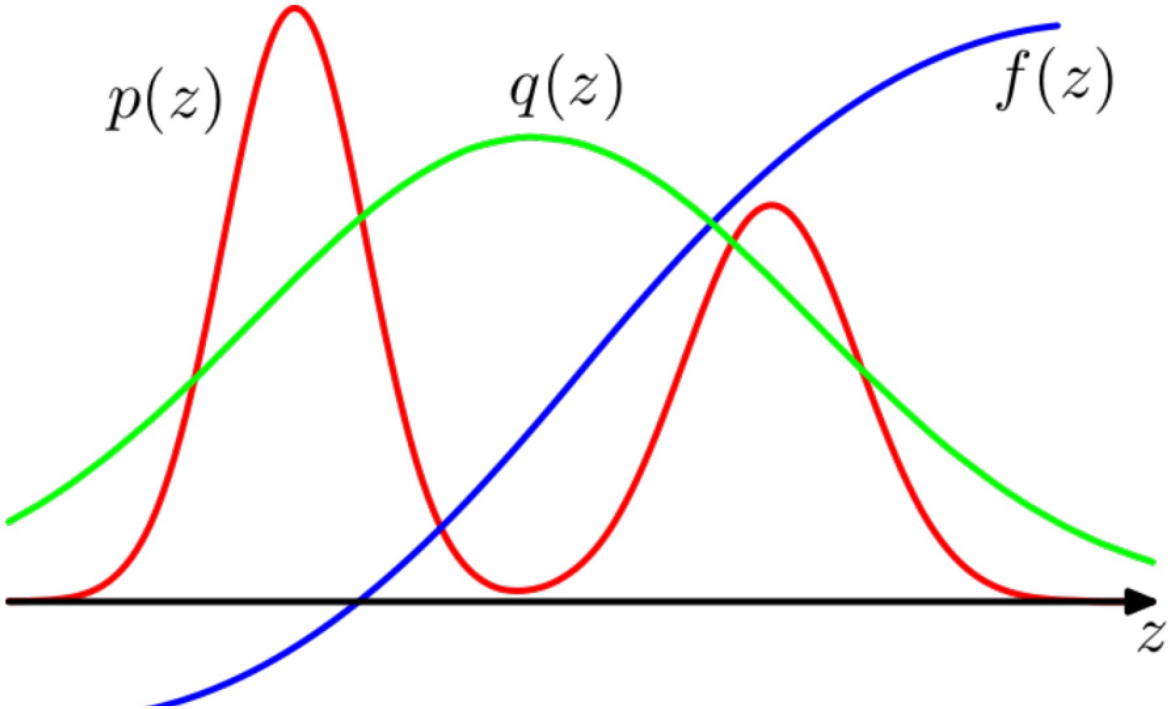
Example Uses Gaussian proposal q to draw samples from multimodal distribution p



Importance Sampling (unnormalized)

- Access to easy-to-sample distribution $Q(x)$
- $\mathrm{Q}(x) > 0$ whenever $P(X) > 0$
- You want to compute $E[f(X)]$

- Idea: Sample from $\mathrm{Q}(x)$
- Weigh the sample proportionally to $\mathrm{w}(\mathrm{x}) = \frac{P(x)}{Q(x)}$

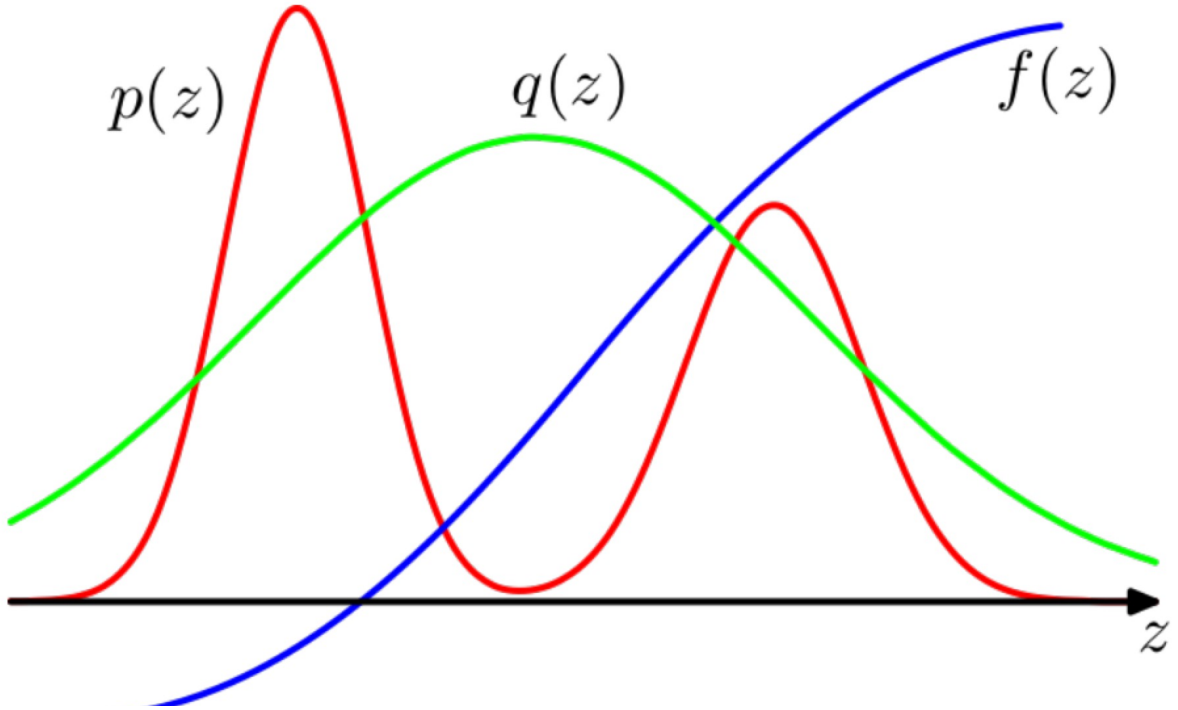


Importance Sampling (unnormalized)

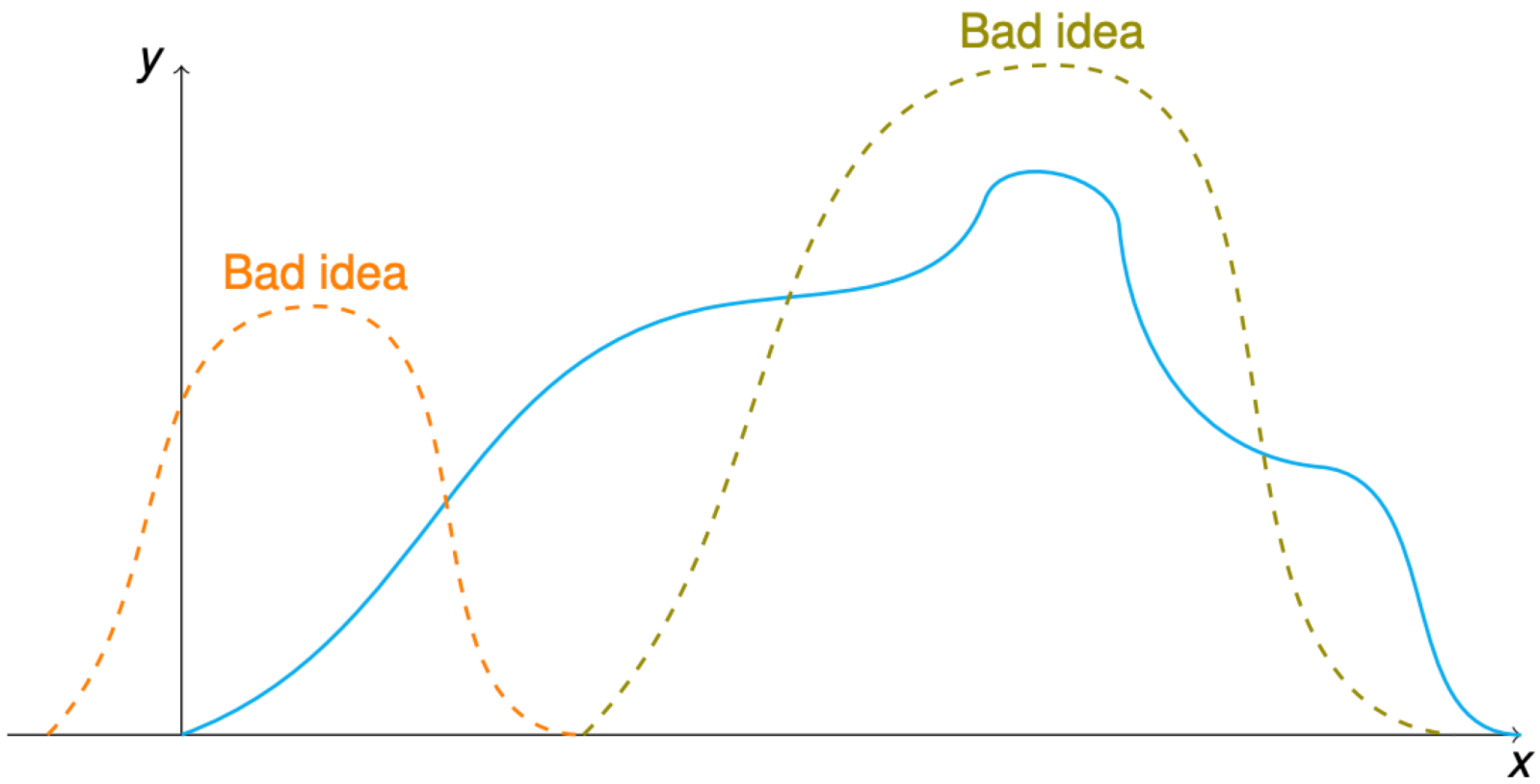
$$E_{P(X)}[f(X)] = E_{Q(X)}\left[f(X)\frac{P(X)}{Q(X)}\right]$$

$$\cong \frac{1}{M}\sum_{m=1}^{M}\frac{p\left(x^{(m)}\right)}{q\left(x^{(m)}\right)}f\left(x^{(m)}\right)$$

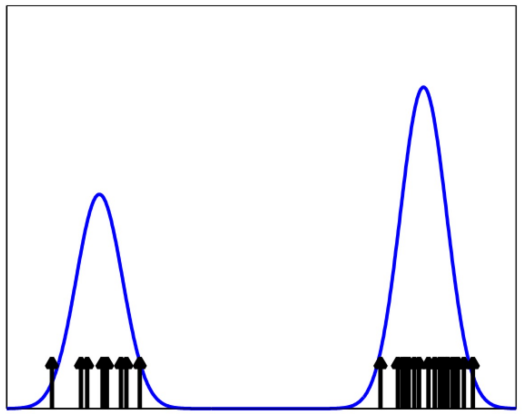
Where x_m are sampled from $q(x)$



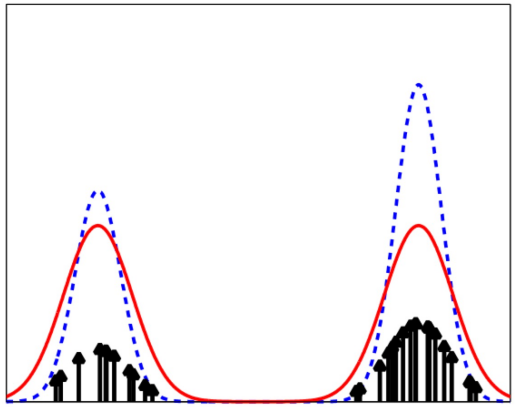
Selecting Good Proposals



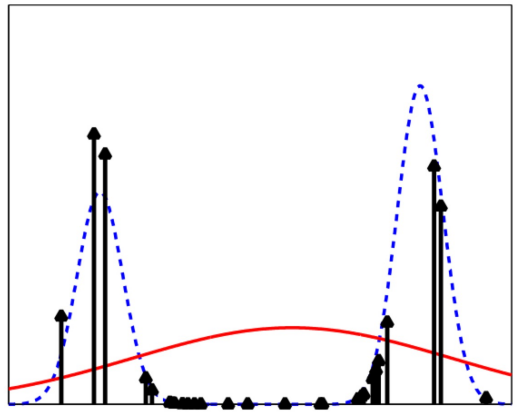
Selecting Good Proposals



Target Distribution



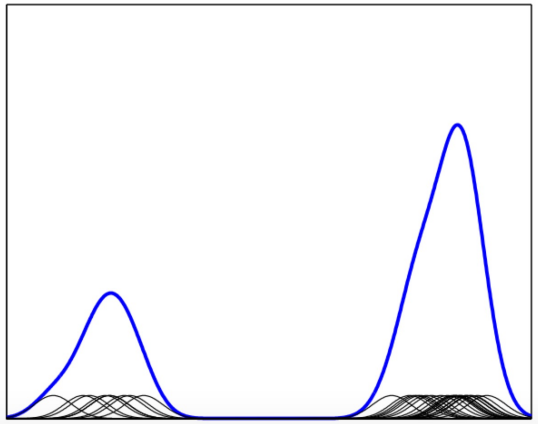
Good Proposal

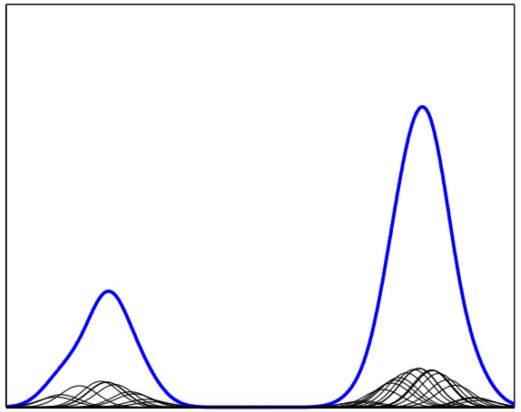


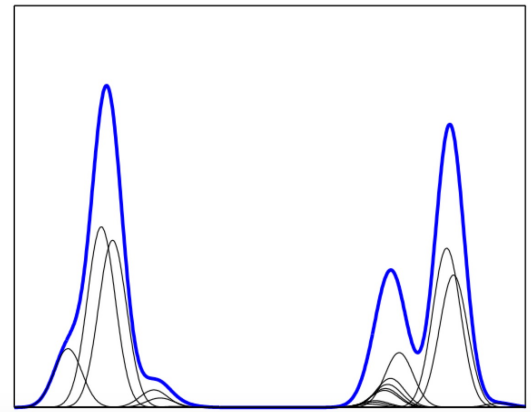
Poor Proposal

Kernel or Parzen window estimators interpolate to predict density:

$$\hat{p}(x) = \sum_{\ell=1}^{L} w^{(\ell)} \mathcal{N}(x; x^{(\ell)}, \Lambda) \qquad w^{(\ell)} \propto \frac{p(x^{(\ell)})}{q(x^{(\ell)})}$$

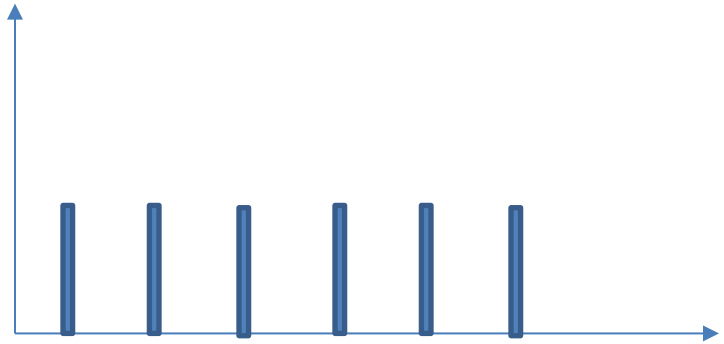






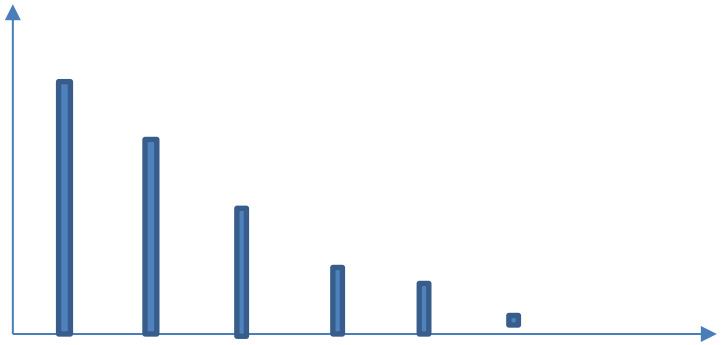
Example:

Sample from a fair die to simulate data from an unfair die



$E[X] = 3.5$

$P(X = 1) = 1/6$



$E[X] = 1.9$

$P(X = 1) = 4/6$

Bias/Variance of the importance sampling estimator

For data sets $\mathcal{D}$ sampled from Q, we have that:

$$\mathbb{E}_{\mathcal{D}}\left[\hat{\mathbb{E}}_{\mathcal{D}}(f)\right] = \boldsymbol{E}_{Q(\boldsymbol{X})}[f(\boldsymbol{X})w(\boldsymbol{X})] = \mathbb{E}_{P(\boldsymbol{X})}[f(\boldsymbol{X})]$$

We can also estimate the distribution of this estimator around its mean. Letting $\epsilon_{\mathcal{D}} = \hat{\mathbb{E}}_{\mathcal{D}}(f) - \mathbb{E}_{P}[f(x)]$, we have that, since $M \to \infty$:

$$\mathbb{E}_{\mathcal{D}}[\epsilon_{\mathcal{D}}] \sim \mathcal{N}\left(0; \sigma_Q^2/M\right)$$

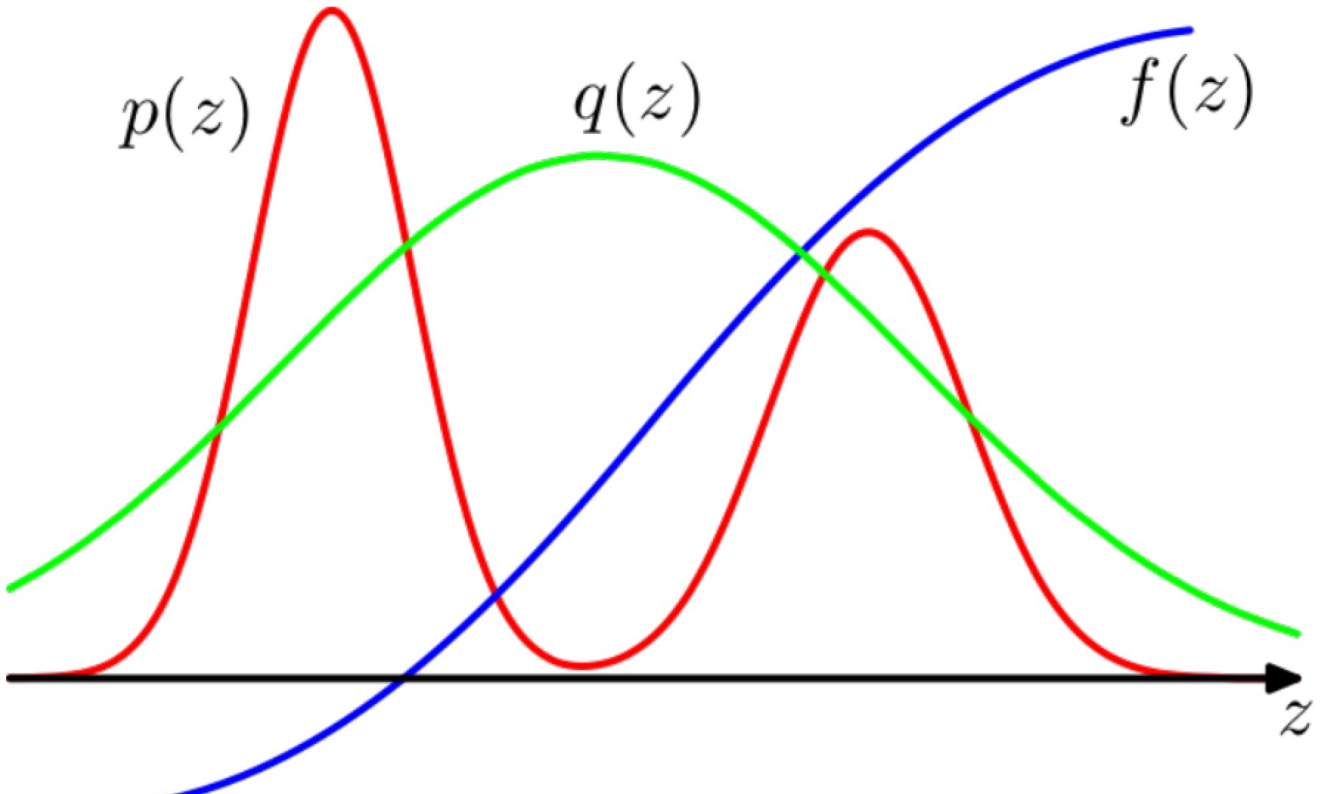
where

$$\begin{aligned}\sigma_Q^2 &= \boldsymbol{E}_{Q(\boldsymbol{X})}[(f(\boldsymbol{X})w(\boldsymbol{X}))^2] - \mathbb{E}_{Q(\boldsymbol{X})}[(f(\boldsymbol{X})w(\boldsymbol{X}))]^2 \\ &= \boldsymbol{E}_{Q(\boldsymbol{X})}[(f(\boldsymbol{X})w(\boldsymbol{X}))^2] - \left(\mathbb{E}_{P(\boldsymbol{X})}[f(\boldsymbol{X})]\right)^2\end{aligned}$$

Importance Sampling (normalized)

What if you have an unnormalized distribution p?

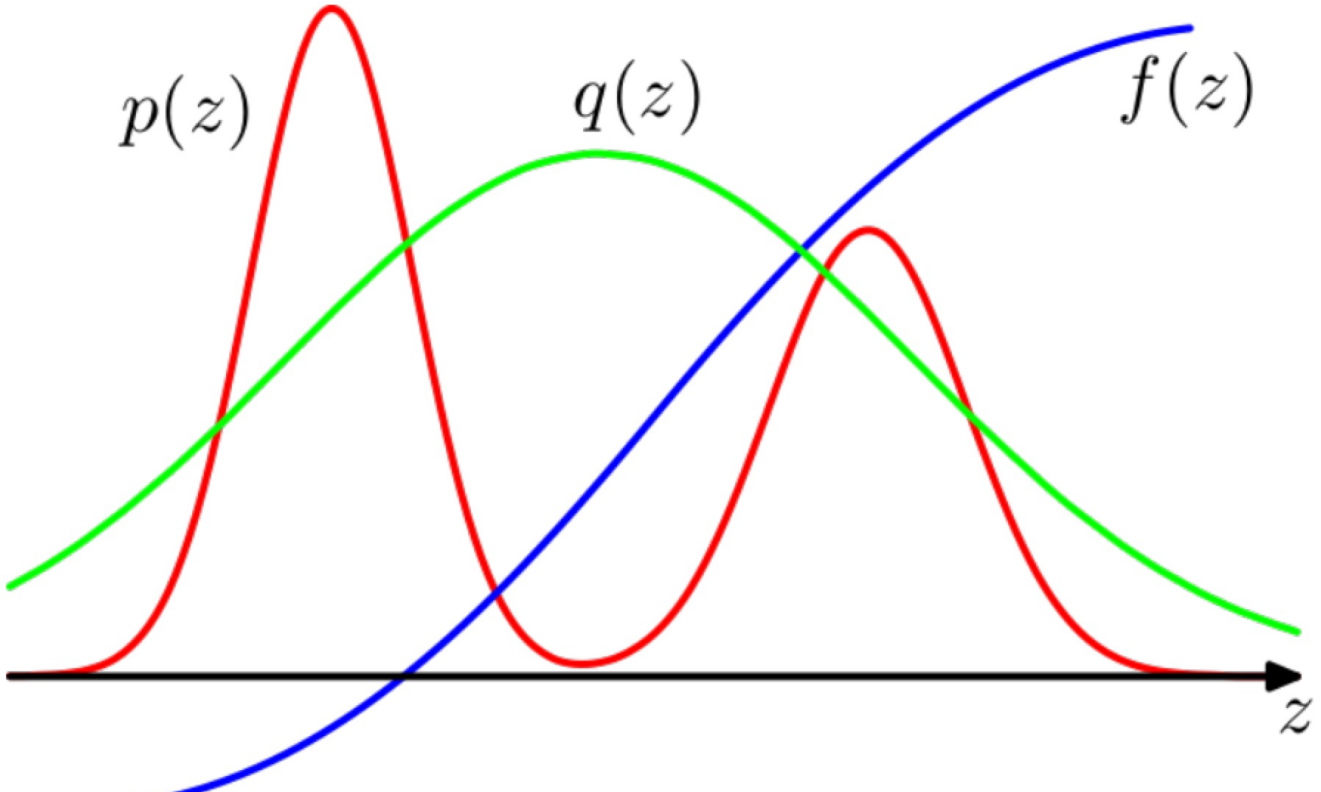
- Access to easy-to-sample distribution $Q(x)$
- $\mathrm{Q}(x) > 0$ whenever $\tilde{P}(x) = ZP(X) > 0$
- You want to compute $E[f(X)]$



Importance Sampling (normalized)

What if you have an unnormalized distribution p?

- Access to easy-to-sample distribution $Q(x)$
- $Q(x) > 0$ whenever $\tilde{P}(x) = ZP(X) > 0$
- You want to compute $E[f(X)]$



- $E_P[f(X)] = \frac{1}{Z} E_Q\left[\frac{\tilde{P}(X)}{Q(X)} f(X)\right]$
- $w(x) = \frac{\tilde{P}(X)}{Q(X)}$
- $E_Q[w(X)] = Z$

Bias/Variance of the normalized importance sampling estimator

$$\hat{\mathbb{E}}_{\mathcal{D}}(f) = \frac{\sum_{m=1}^{M} f(\boldsymbol{x}[m]) w(\boldsymbol{x}[m])}{\sum_{m=1}^{M} w(\boldsymbol{x}[m])}$$

Not unbiased:

$$\mathbb{E}_{\mathcal{D}}\left[\hat{\mathbb{E}}_{\mathcal{D}}(f)\right] \neq \mathbb{E}_{P(\boldsymbol{X})}[f(\boldsymbol{X})]$$

Variance: $$\mathrm{Var}_P\left[\hat{\mathbb{E}}_{\mathcal{D}}(f(\boldsymbol{X}))\right] \approx \frac{1}{M}\mathrm{Var}_P[f(\boldsymbol{X})]\left(1 + \mathrm{Var}_Q[w(\boldsymbol{X})]\right)$$

Summary: Importance Sampling

1. Simulate from tractable distribution

$$\{x^{(\mathrm{m})}\}_{\mathrm{m}=1}^{M} \sim Q(\mathrm{x})$$

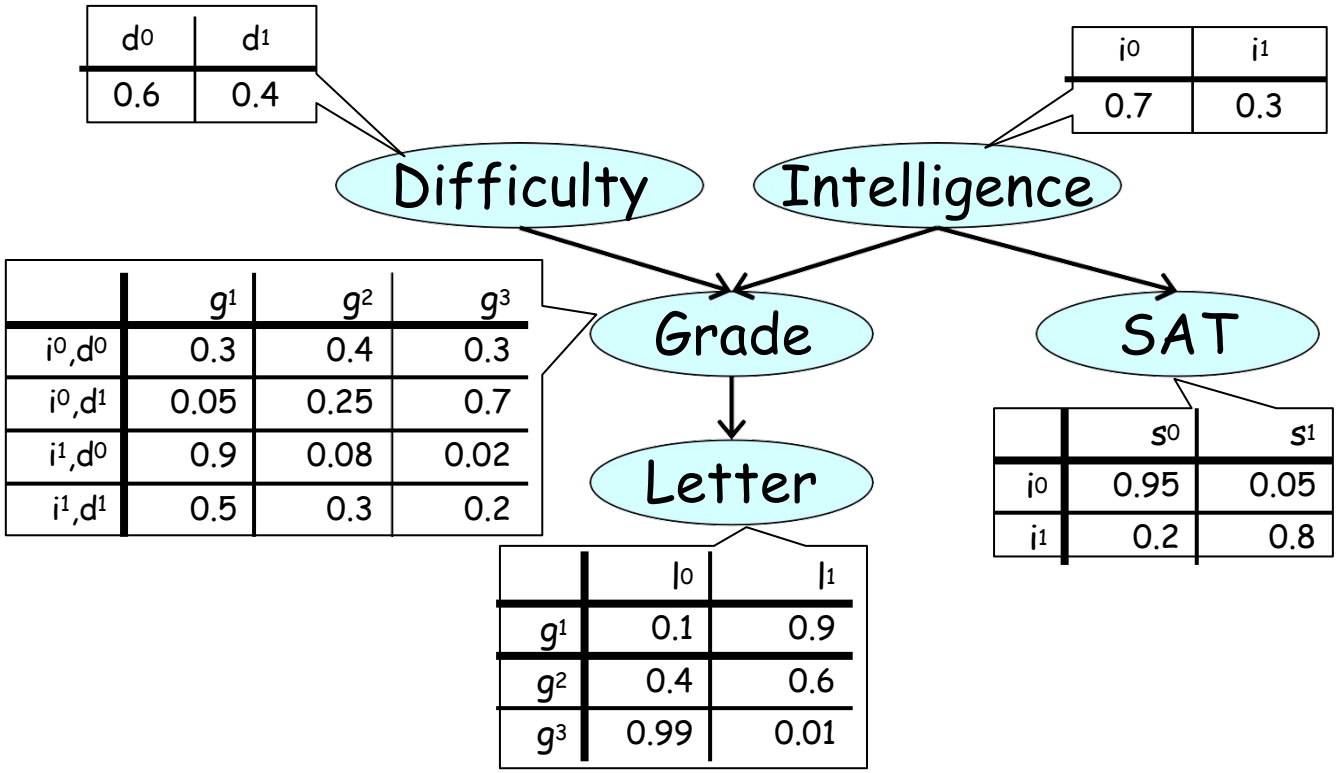
2. Compute importance weights & normalize

$$\widetilde{w}^{(\mathrm{m})} = \frac{\tilde{P}\left(x^{(\mathrm{m})}\right)}{Q\left(x^{(\mathrm{m})}\right)}, \mathrm{w}^{(m)} = \frac{\widetilde{w}^{(\mathrm{m})}}{\sum_{i=1}^{M} \tilde{r}^{(i)}}$$

3. Compute importance-weighted expectation

$$\mathbf{E}_P[f(\mathrm{x})] \approx \sum_{\mathrm{m}=1}^{M} w^{(\mathrm{m})} f\left(x^{(\mathrm{m})}\right) \equiv \hat{f}$$

Importance Sampling in BNs



We want to sample from $P(D, I, S, L|G = g^2)$

We want to bias our sampling towards parts of the space where this event holds

For L, this is easy; Sample from $P(L|G = g^2)$

For the rest?

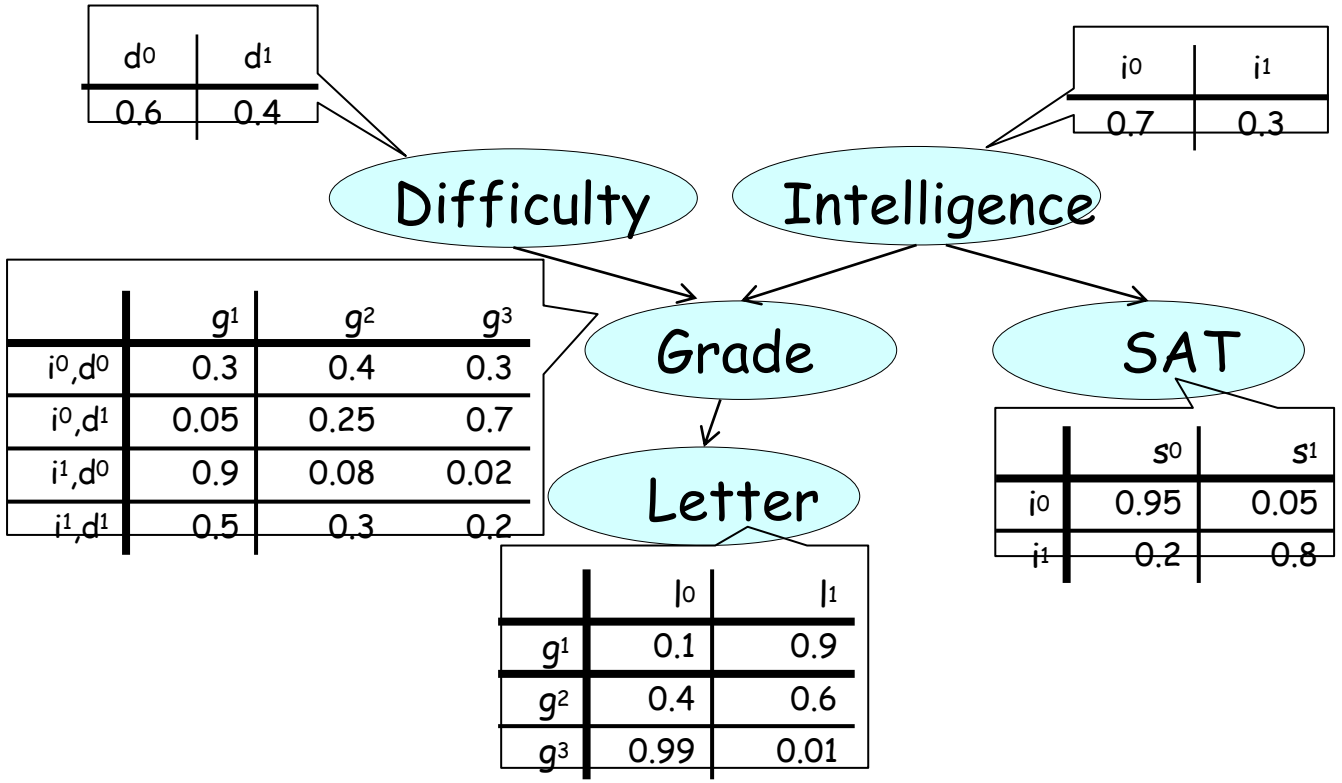
Mutilated Network Proposal Distribution

Let $\mathcal{B}$ be a network, and $Z_1 = z_1, \ldots, Z_k = z_k$, abbreviated $\boldsymbol{Z} = \boldsymbol{z}$, an instantiation of variables.

We define the mutilated network $\mathcal{B}_{\boldsymbol{Z}=\boldsymbol{z}}$ as follows:

- Each node $Z_i \in \boldsymbol{Z}$ has no parents in $\mathcal{B}_{\boldsymbol{Z}=\boldsymbol{z}}$; the CPD of Z_i in $\mathcal{B}_{Z=z}$ gives probability 1 to $Z_i = z_i$ and probability 0 to all other values $z_i' \in \text{Val}(Z_i)$.
- The parents and CPDs of all other nodes $X \notin \boldsymbol{Z}$ are unchanged.

Mutilated Network Proposal Distribution



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

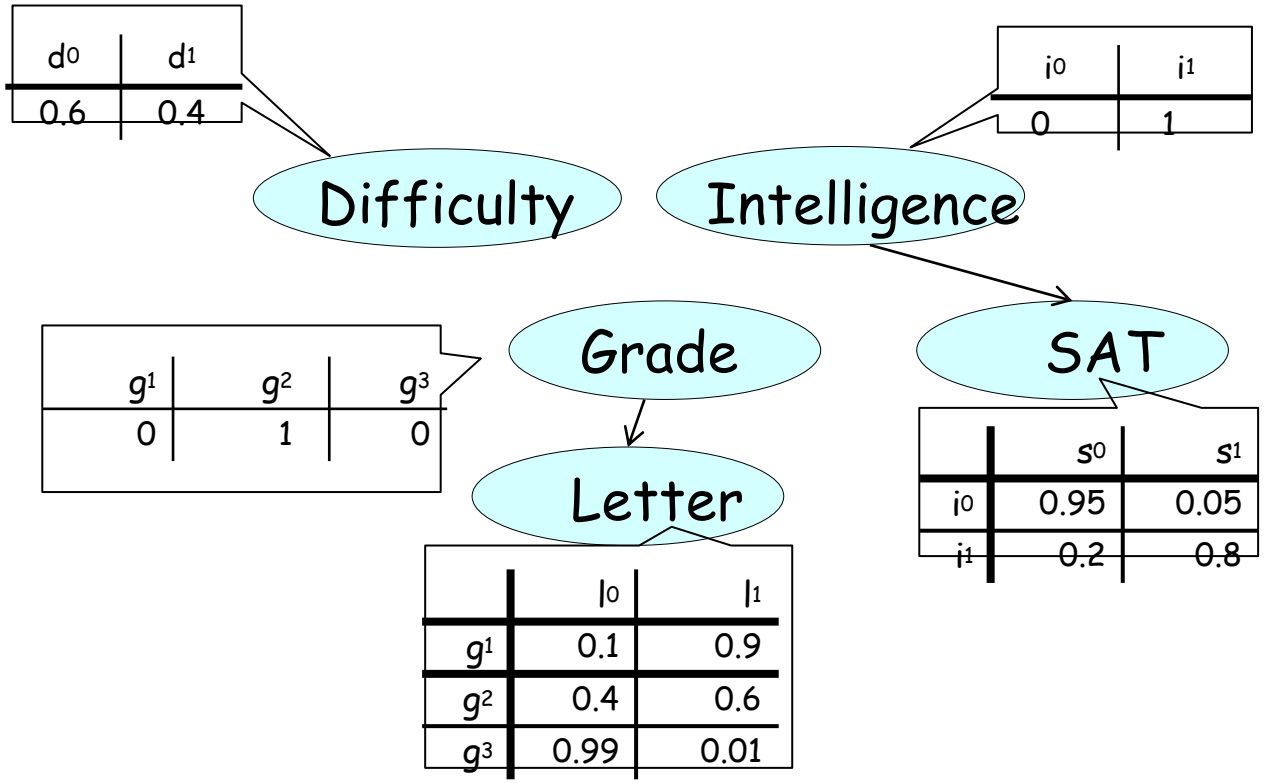
	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Assume you want to sample from

$$I = i^1, G = g^2$$

Mutilated Network Proposal Distribution



Assume you want to sample from

$$I = i^1, G = g^2$$

Mutilated Graph $\mathcal{B}_{Z=z}$, proposal distribution $P_{\mathcal{B}_{Z=z}}$

Weight of a sample $w(\xi) = \frac{P_{\mathcal{B}}(\xi)}{P_{\mathcal{B}_{Z=z}}(\xi)}$.

Summary

- Generating samples from a BN is easy
- (ϵ, δ)-bounds exist, but usefulness is limited:
  - Additive bounds: useless for low probability events
  - Multiplicative bounds: # samples grows as 1/P(y)

- Forward sampling generally infeasible for MNs

Two ideas for sampling with evidence:
Throw away samples (rejection sampling)
Weigh samples (Importance sampling)

Summary

Monte Carlo Estimators can compute unbiased expectations with good properties

Rejection sampling is straightforward but can be very wasteful

Importance sampling can reduce the variance of the estimators if you find a good proposal