Approximate Inference

Xin Zhang
Peking University

Adapted from the slides of "Pattern Recognition and Machine Learning" Chapter 10 & 11

Recap: Converting Directed to Undirected

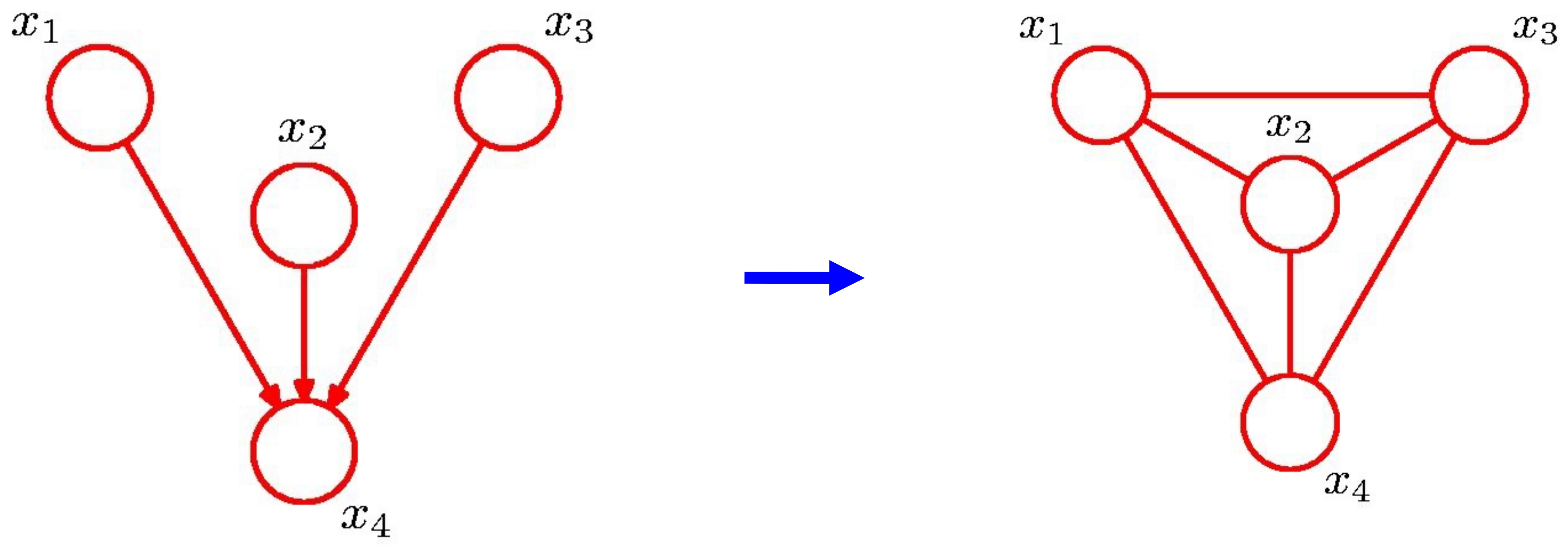
1. Add links between all pairs of parents for each node (moralization)

2. Drop arrows, which results in a moral graph

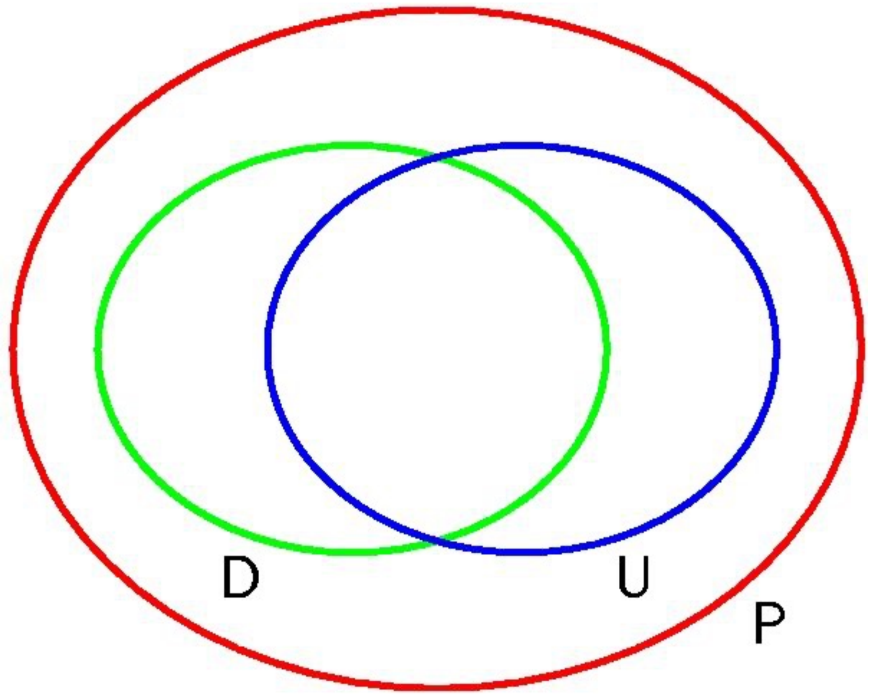
3. Initialize all of the clique potentials to 1. Take each conditional distribution factor and multiply it into one of the clique potentials

4. $Z = 1$

Converting Directed to Undirected Graphs

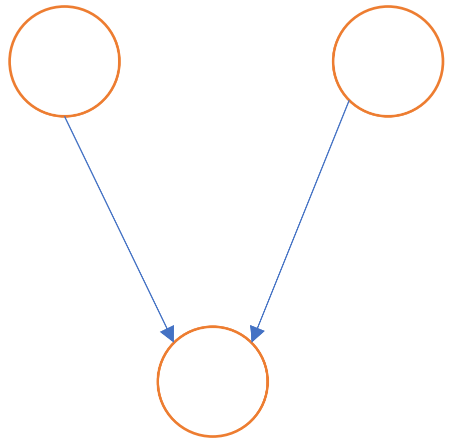


Directed vs. Undirected Graphs

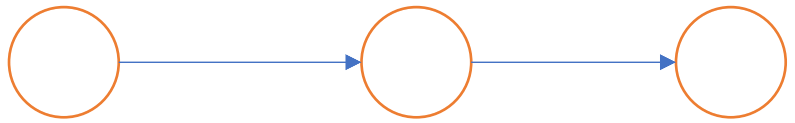


Distributions that can be perfectly represented by two types of graphs in terms of conditional independence

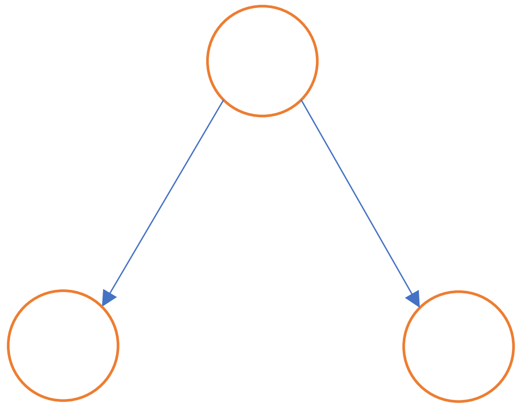
Can you convert the following directed graphs into undirected while keeping conditional independence?



1

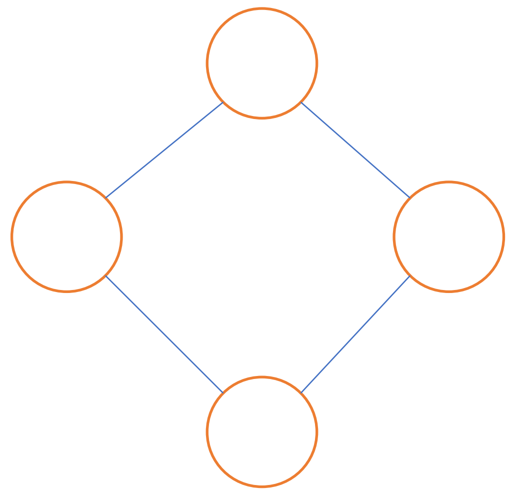


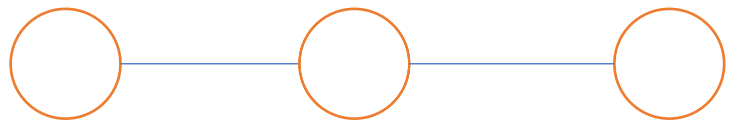
2

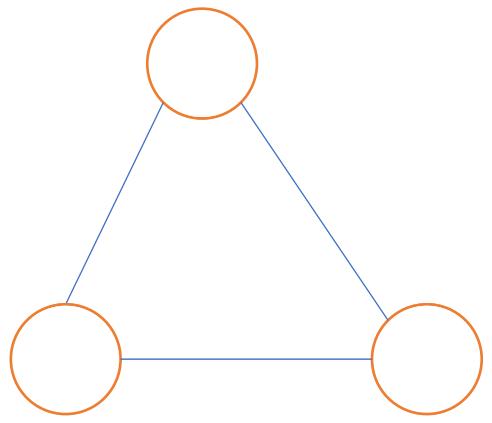


3

Can you convert the following undirected graphs into directed while keeping conditional independence?

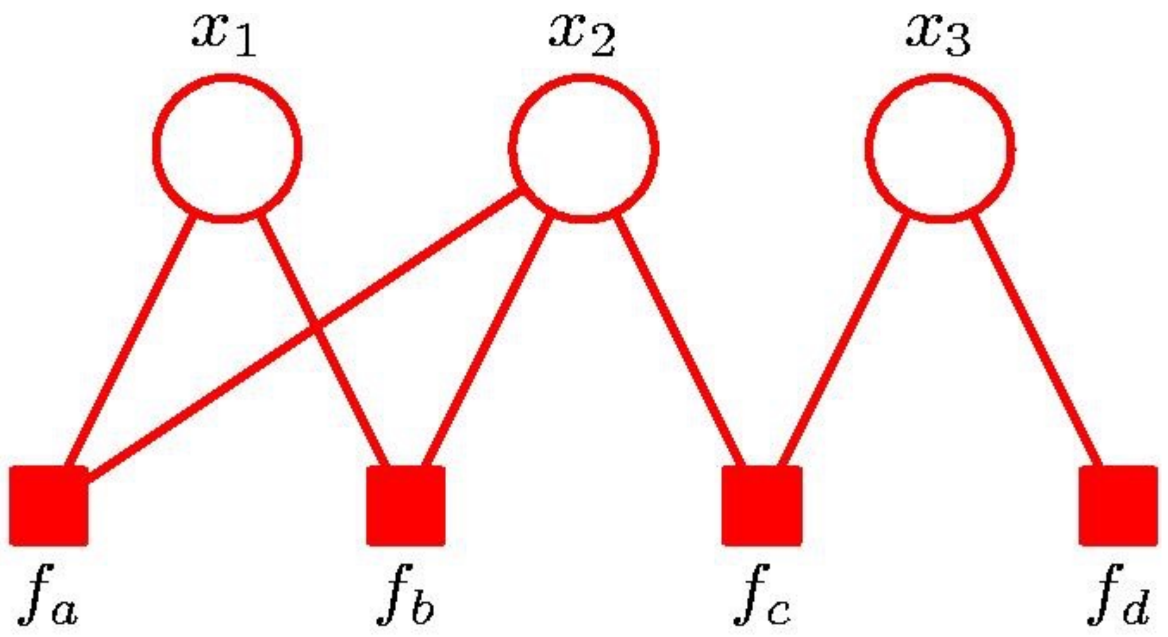

1


2


3

Factor Graphs

- Bipartite graph
- Two kinds of nodes:
  - Regular random variables
  - Factor nodes
- Factor node represents a function that maps assignments to its neighbors to a real number
- $p(\boldsymbol{x}) = \prod_s f_s(\boldsymbol{x_s})$



$$p(x_1, x_2, x_3) = \frac{1}{Z} f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

Sum-Product Algorithm

- Computes marginal probabilities with/out conditions
- Exact on tree-structure factor graphs
- Key idea: exchange sums and products using the distributive law

$$ab + ac = a(b + c)$$

The Sum-Product Algorithm

- To compute local marginals:
  - Pick an arbitrary node as root
  - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
  - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
  - Compute the product of received messages at each node for which the marginal is required, and normalize if necessary.

The Max-Sum Algorithm

- Efficient algorithm that finds an assignment to all variables that maximizes the probability

- Similar to Sum-Product, but it uses the distributive law on max and sum:

$$\max(a+b, a+c) = a + \max(b, c).$$

Sum-Product vs. Max-Sum

Sum-Product

$$\mu_{f\to x}(x) = \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{x_m \in ne(f)\setminus \mathrm{x}} \mu_{x_m \to f}(x_m)$$

$$\mu_{x\to f}(x) = \prod_{l \in ne(x)\setminus \mathrm{f}} \mu_{f_l \to x}(x)$$

a(b+c) = ab+bc

Max-Sum

$$\mu_{f\to x}(x) = \max_{x_1,\dots,x_M} [ln f(x, x_1, \dots, x_M) + \sum_{x_m \in ne(f)\setminus \mathrm{x}} \mu_{x_m \to f}(x_m)]$$

$$\mu_{x\to f}(x) = \sum_{l \in ne(x)\setminus \mathrm{f}} \mu_{f_l \to x}(x)$$

a+max(b,c) =max(a+b, a+c)

What about inference on general graphs?

- NP-complete

- Counting problem

Is the following description right?

- Factor graph can be only used in probabilistic inference.

Is the following description right?

- In general, the sum-product algorithm is imprecise for Bayesian networks that are not trees.

Is the following description right?

- The sum-product algorithm is imprecise for any Markov Random Field that is not a tree.

Is the following statement right?

- To compute the most likely value for a joint distribution, one can calculate the marginal probabilities of each variable, and take the values with the highest probabilities.

Is the following statement correct?

- Loopy belief propagation is approximate, but it monotonically converges to the precise value if infinite time is given.

Motivation

- A central task in applying probabilistic models is to evaluate P(Z | X)

Latent variables

Observed values

- And calculate expectations
- Example
    - X: training data
    - Z: parameters
    - Expectations: predictions or parameters themselves

Motivation

- However, as we see in graphical model inference, exact solution is not always possible:
  - Curse of dimensionality
  - The posterior distribution has a highly complex form making expectations not analytically tractable
    - Continuous case: no closed-form analytical form
    - Discrete: cannot perform summarization because there are too many hidden variables

This Class

- General approximate inference techniques for various probabilistic models.
  - Deterministic
    - Can never generate exact results
    - The approximation has an analytical form
  - Stochastic (Sampling-Based)
    - Gives exact results when infinite resources are given
    - Can be computationally demanding

Deterministic Approximate Methods

- Variational inference ⟵ **Brief introduction**

- Expectation propagation

Variational Inference: Background

- Originates from the calculus of variations
  - A functional maps a function to a value

$$\mathrm{H}[p] = \int p(x) \ln p(x)\, \mathrm{d}x$$

  - Functional derivative: expresses how the value of a functional changes in response to infinitesimal changes to the input function
  - Variational method: find an input function that maximizes or minimizes the functional
  - Exact if the input function can be of any form; in general, we restrict it to some range

Variational Inference: Main Idea

- Goal: approximate p(Z | X) and p(X)

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \mathrm{KL}(q\|p)$$

- Where

$$\begin{aligned} \mathcal{L}(q) &= \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z} \\ \mathrm{KL}(q\|p) &= -\int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z} \end{aligned}$$

Variational Inference : Main Idea

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \mathrm{KL}(q\|p)$$

- Here L(q) is a functional. What happens if we maximizes it?
- We have KL(q||p) =0.
- This implies q(Z) = p(Z|X)!
- In other words, we can approximate p(Z|X) using q(Z) by minimizing KL(q||p) or maximizing L(q)

L(q) is called evidence lower bound(ELBO)

Variational Inference: Main Idea

- In order to make the problem tractable, we need to restrict q to a family of distributions

- Example: $q(Z \mid w)$, find w using nonlinear optimization

Variational Inference: Factorized Distributions

- We can divide the latent variables **Z** into disjoint groups $\mathbf{Z_1}$, …, $\mathbf{Z_M}$:

$$q(\mathbf{Z}) = \prod_{i=1}^{M} q_i(\mathbf{Z}_i).$$

- No restriction on the forms of q_i

- Corresponds to an approximation framework in physics called mean field theory

- Optimize L(q) with respect to each $q_i(Z_i)$ in turn

Variational Inference: Factorized Distributions

$q_i = q_i(\mathbf{Z_i})$. Optimize with respect to q_j while keeping other q_i's constant

$$\begin{aligned}
\mathcal{L}(q) &= \int \prod_i q_i \left\{ \ln p(\mathbf{X}, \mathbf{Z}) - \sum_i \ln q_i \right\} \mathrm{d}\mathbf{Z} \\
&= \int q_j \left\{ \int \ln p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i \, \mathrm{d}\mathbf{Z}_i \right\} \mathrm{d}\mathbf{Z}_j - \int q_j \ln q_j \, \mathrm{d}\mathbf{Z}_j + \text{const} \\
&= \int q_j \ln \widetilde{p}(\mathbf{X}, \mathbf{Z}_j) \, \mathrm{d}\mathbf{Z}_j - \int q_j \ln q_j \, \mathrm{d}\mathbf{Z}_j + \text{const}
\end{aligned}$$

Where $\ln \widetilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const.}$

Variational Inference: Factorized Distributions

$$\mathcal{L}(q) = \underbrace{\int q_j \ln \widetilde{p}(\mathbf{X}, \mathbf{Z}_j)\,\mathrm{d}\mathbf{Z}_j - \int q_j \ln q_j \,\mathrm{d}\mathbf{Z}_j}_{-KL(\tilde{p}(\boldsymbol{X}, Z_j)||q_j)} + \text{const}$$

Solution: $$\ln q_j^\star(\mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})] + \text{const.}$$

Essentially, the optimal solution for q_j is obtained by taking the expectation of p(**X,Z**) with respect to all the other factors

Variational Inference: Factorized Distributions

- Algorithm workflow:
  - Initialize all factors appropriately
  - Cycle through all factors to run the optimization and update the factors
  - Convergence is guaranteed because bound is convex with respect to each factor (Boyd and Vandenberghe, 2004)

Variational Inference in Probabilistic Programming

- http://docs.webppl.org/en/master/optimization/index.html

- https://probmods.org/chapters/inference-algorithms.html

Sampling Methods: Introduction

- Also called Monte Carlo methods

- Suppose want to evaluate E(f) when its inputs are from distribution **z**, we can replace

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}$$

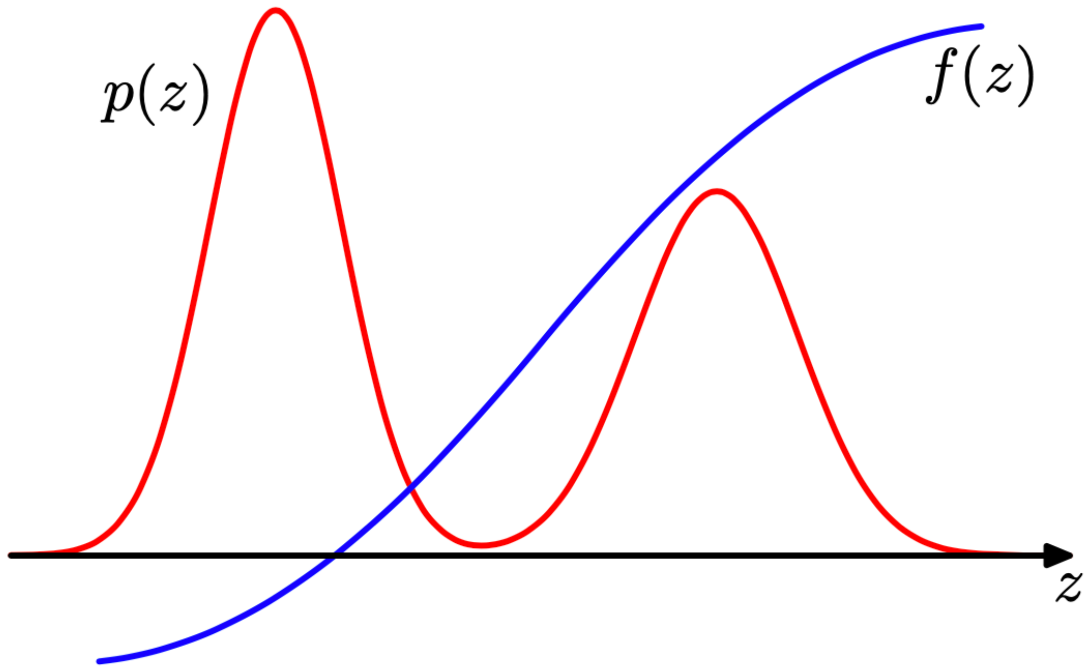
- With

$$\hat{f} = \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})$$

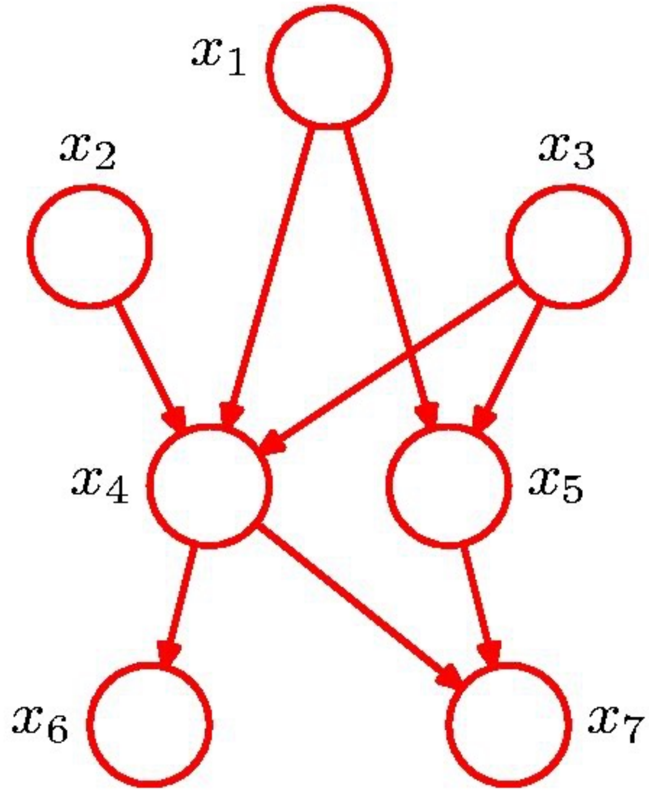
$\mathbf{z}_1, \ldots, \mathbf{z}_l$ are samples from p

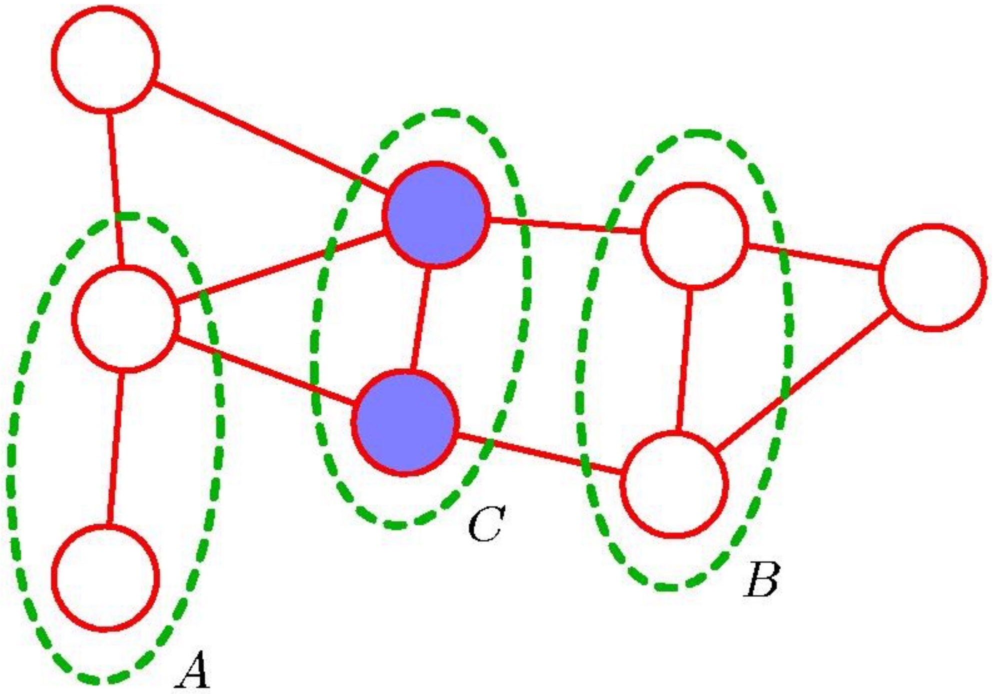
Sampling Methods: Introduction

- New problem: how to sample independent samples effectively?



What about graphical models?





Standard Distributions: Transformation

- Seed distributions z which we know how to draw samples from:

  uniform(0,1)

- To sample from a given distribution y, we can define function f, so that

$$y = f(x)$$

- Key challenge: how to define f so we can use the samples from z to calculate y?

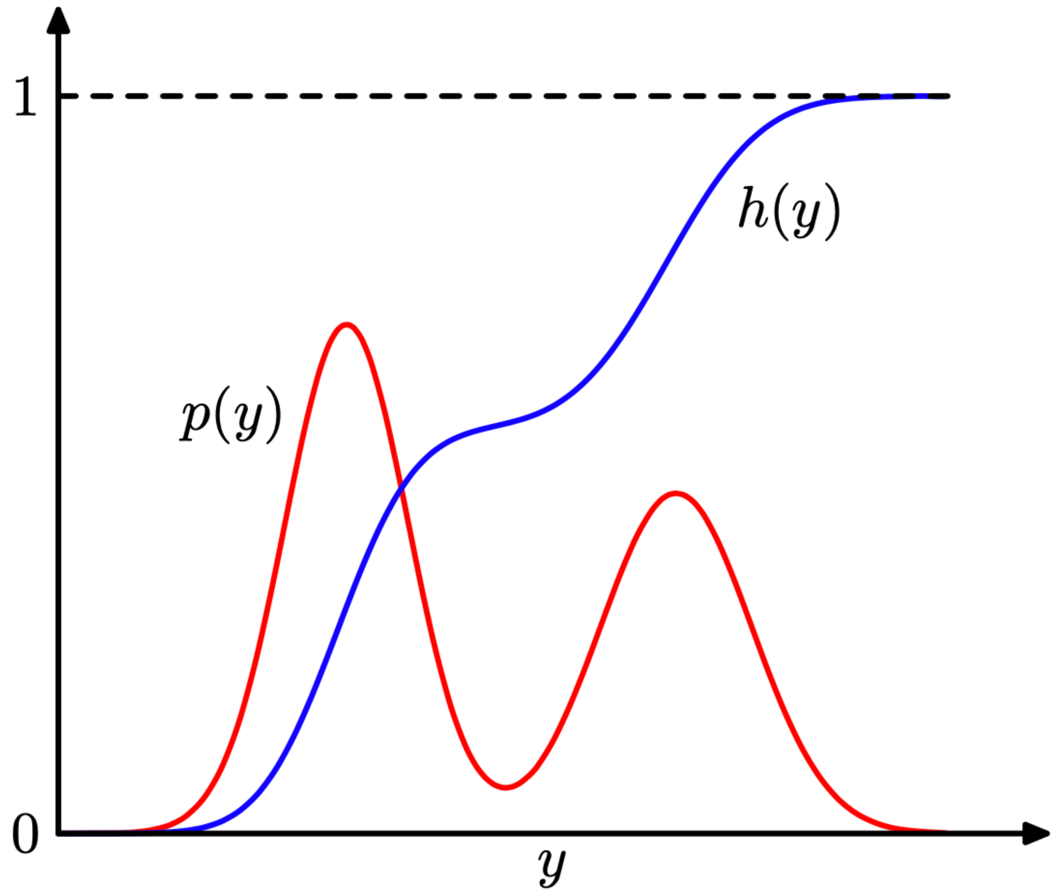
Standard Distributions: Transformation

- Key idea
  - Interpret numbers in uniform distribution z as probabilities
  - Find h(y), such that the probability regarding y is z:

$$z = h(y) = P(\hat{y} < y) = \int_{-\infty}^{y} p(\hat{y}) d\hat{y}$$

  - f = h^{-1}
  - Function h is called the cumulative distribution function (CDF) of distribution y
  - Function f is called the inverse CDF

Standard Distributions: Transformation



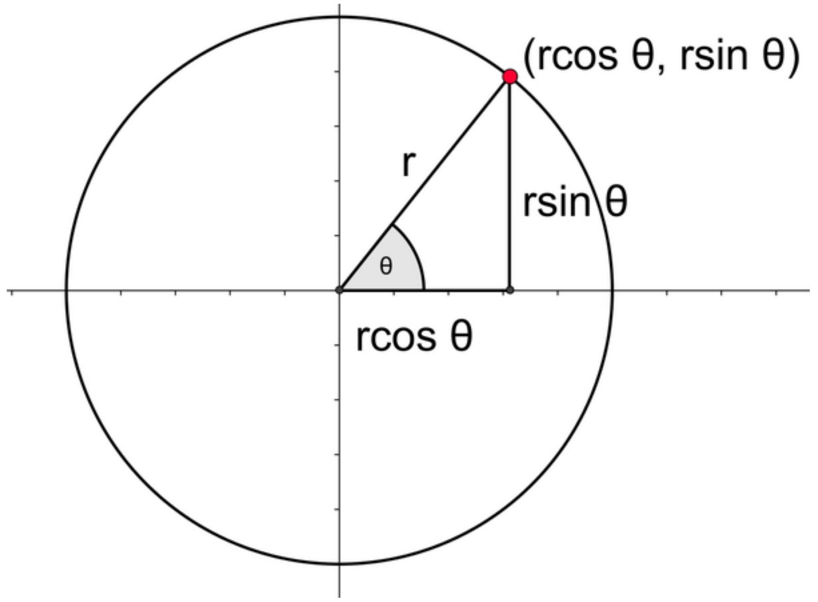
Transformation Method: Examples

- Exponential distribution: $p(y) = \lambda \exp(-\lambda y)\,, 0 \leq y < \infty$
- $h(y) = 1 - \exp(-\lambda y)\,, y = -\lambda^{-1}\ln(1 - z)$

- Gaussian: Box-Muller method (inverse CDF of Gaussian is not well-defined)
  - Assume $z_1, z_2 = uniform(-1,1)$
  - Draw samples from z_1, z_2, and only keep $z_1^2 + z_2^2 \leq 1$. Now we get $p(z_1, z_2) = \frac{1}{\pi}$
  - Let $y_1 = z_1\sqrt{\left(\frac{-2lnz_1}{r^2}\right)}\,, \mathrm{y}_2 = z_2\sqrt{\left(\frac{-2lnz_2}{r^2}\right)}$, where $r^2 = z_1^2 + z_2^2$
  - $p(y_1, y_2) = [\frac{1}{\sqrt{2\pi}}\exp(-y_1^2/2)][\frac{1}{\sqrt{2\pi}}\exp(-y_2^2/2)]$

More on Box-Muller Method

- https://www.quora.com/What-is-an-intuitive-explanation-of-the-Box-Muller-transform



$$\theta \sim 2\pi * uniform(0,1)$$
$$r \sim \sqrt{-2\ln(uniform(0,1))}$$

$$y_1 = rsin\theta$$
$$y_2 = rcos\theta$$

Rejection Sampling: Introduction

- Allows sampling from relatively complex distributions with constraints

- One of the basic inference methods in probabilistic programming

- Basic idea: sample from a proposal distribution and accept some samples

Rejection Sampling: Assumptions

- Sampling from **z** is hard, but we can evaluate p(**z**) for any value of **z** up to some normalizing constant Z:

$$p(z) = \frac{1}{Z_p}\widetilde{p}(z)$$

- There exists a proposal distribution q(**z**) which we know how to sample form and

- There exists a constant k such that $kq(z) \geqslant \widetilde{p}(z)$

Rejection Sampling: Algorithm

- Each step, generates two numbers:
  - $z_0 \sim z$, according to $\tilde{p}$
  - $u_0 \sim \text{uniform}(0, kq(z_0))$

(z_0, u_0) are uniform under the curve of $kq(z_0)$

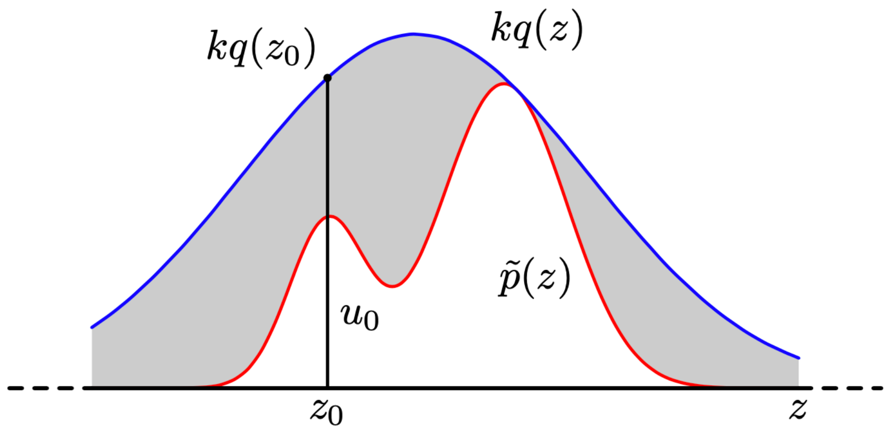
- If $u_0 > \tilde{p}(z_0)$, then the sample is rejected, otherwise z_0 is accepted

(z_0, u_0) are uniform under the curve of $\tilde{p}(z_0)$

- The set of kept samples are distributed according to p

Rejection Sampling: Analysis

$$\begin{aligned} p(\text{accept}) &= \int \{\widetilde{p}(z)/kq(z)\}\, q(z)\, \mathrm{d}z \\ &= \frac{1}{k}\int \widetilde{p}(z)\, \mathrm{d}z. \end{aligned}$$

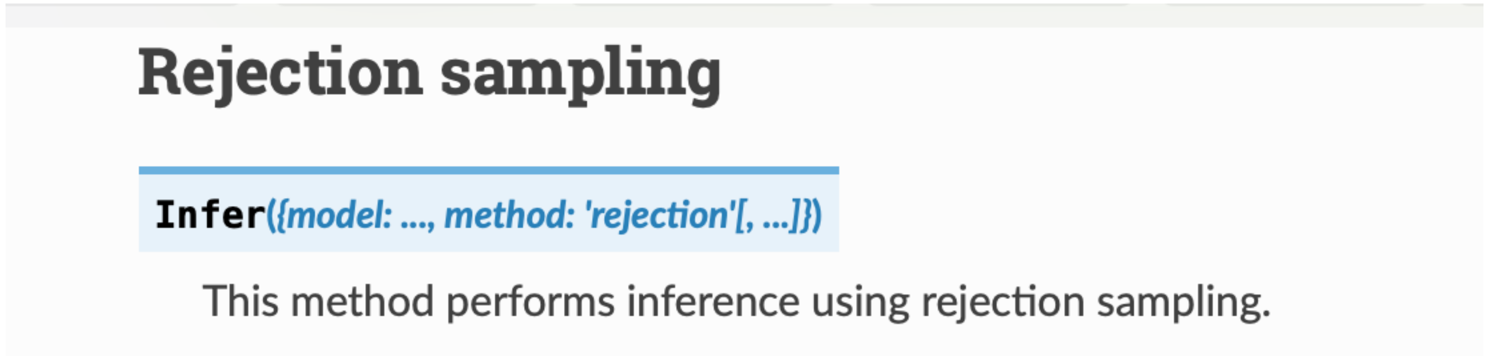


Efficiency is decided by the ratio of the grey area and the white area:

1. Choose k as small as possible
2. The proposal distribution should be as close to the real distribution as possible

Rejection Sampling in Prob. Prog.

- Much simpler. Usually it just throws away the samples that do not meet the conditions



Rejection sampling

```
Infer({model: ..., method: 'rejection'[, ...]})
```

This method performs inference using rejection sampling.

Importance Sampling: Background

- Goal: evaluate the expectation of a function f(z) where z is from distribution p(z)

- Naïve idea: sampling using a grid

$$\mathbb{E}[f] \simeq \sum_{l=1}^{L} p(\mathbf{z}^{(l)}) f(\mathbf{z}^{(l)}).$$

- Problem: does not scale with number of dimensions

Importance Sampling: Idea

- Motivation: only points where p(z) or f(z)*p(z) are large matter

- Idea: again using a proposal distribution but we don't discard samples

$$E(f) = \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz \approx \frac{1}{L}\sum_{l=1}^{L}\frac{p(z^l)}{q(z^l)}f(z^l)$$

- $\frac{p(z^l)}{q(z^l)}$ are called importance weights. They are used to correct bias introduced by sampling the wrong distribution

We don't get right samples but samples with correcting weights

What if I want to get the right samples with importance sampling?

Sampling-Importance-Resampling

- Alternative to rejection sampling when k is hard to find

- Steps:
  - Draw samples $\mathrm{z}_1, z_2, \ldots, z_L$ using importance sampling
  - Logging importance weights $w_1, w_2, \ldots, w_L$
  - Construct a discrete distribution $(\mathrm{z}_1, z_2, \ldots, z_L)$ whose probabilities are given by $w_1, w_2, \ldots, w_l$. Sample from this distribution

- Precise when $L \to \infty$

Markov Chain Monte Carlo

- Goal: find a sampling method that works well with a large family of distribution with high dimensions
  - Problem with rejection and importance sampling: high dimensionality

- Main Idea:
  - Still based on using a proposal distribution
  - But the proposal distribution is based on current state: $q(z \mid z^{\tau})$
  - Decide whether to accept z^*as the next state based on $q(z \mid z^{\tau})$. If accepted, $z^{\tau+1} = z^*$. Otherwise, $z^{\tau+1} = z^{\tau}$
  - Samples form a Markov chain

- Assumption: we can efficient evaluate $\tilde{p}(z) = Z * p(z)$

Markov Chain Monte Carlo

- Originated from physics
- Often used in optimization
  - Similar to simulated annealing

Metropolis Algorithm (Metropolis *et al.*, 1953)

- A basic algorithm
- Assumption: the proposal distribution is symmetric

$$q(\mathbf{z}_A|\mathbf{z}_B) = q(\mathbf{z}_B|\mathbf{z}_A)$$

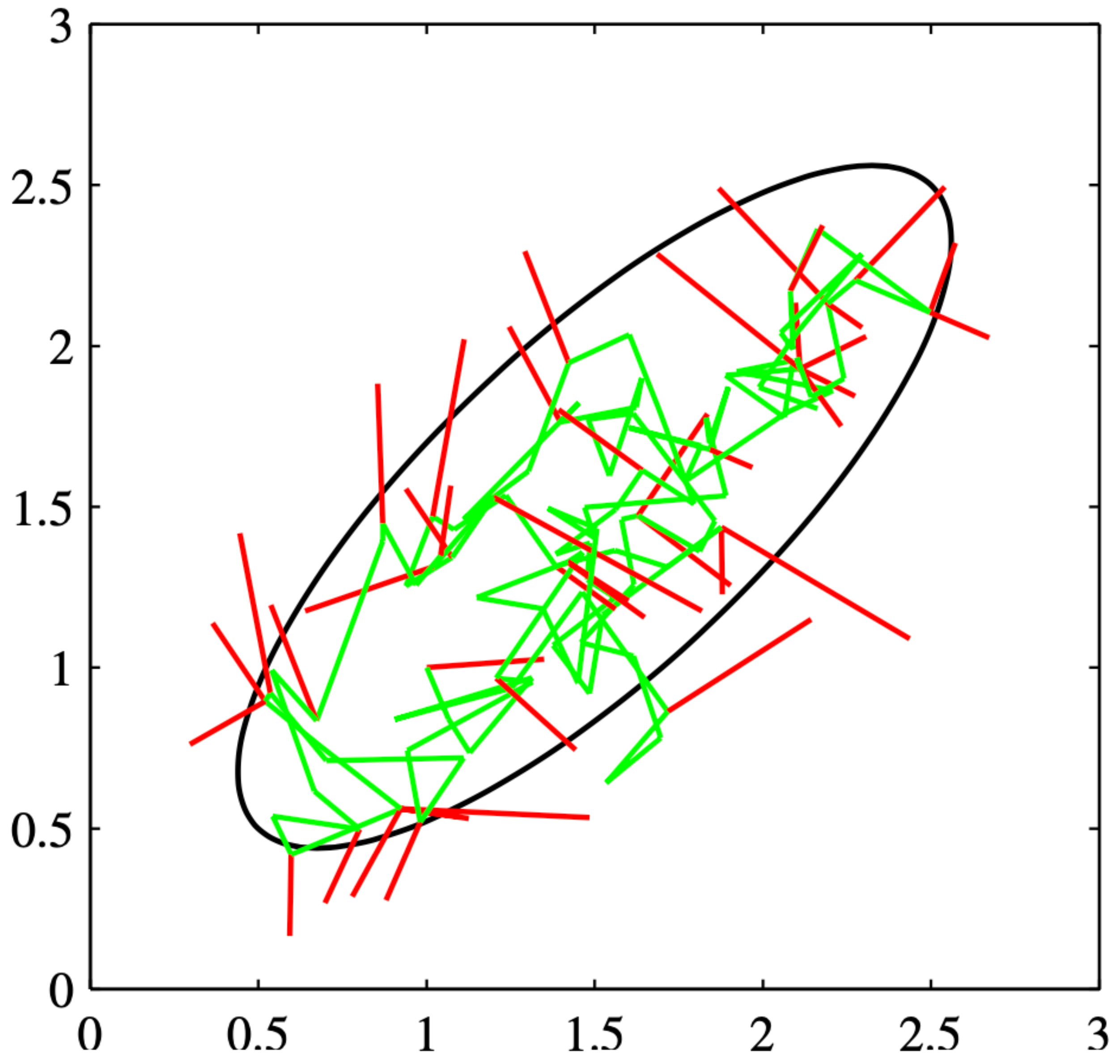
- Steps:
  - Choose some point as the initial state z_0
  - If the current state is z^τ, draw z^* from $q(z \mid z^\tau)$. Accept z^* with probability

$$A(\mathbf{z}^\star, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{\widetilde{p}(\mathbf{z}^\star)}{\widetilde{p}(\mathbf{z}^{(\tau)})}\right).$$

  - If z^* is accepted, $z^{\tau+1} = z^*$. Otherwise, $z^{\tau+1} = z^\tau$. Loop to the above step.

Multiple samples of z^τ

A simple illustration using Metropolis algorithm to sample from a Gaussian distribution whose one standard-deviation contour is shown by the ellipse. The proposal distribution is an isotropic Gaussian distribution whose standard deviation is 0.2. Steps that are accepted are shown as green lines, and rejected steps are shown in red. A total of 150 candidate samples are generated, of which 43 are rejected.



Metropolis Algorithm: Why it works?

- **Theorem**: as long as $q(z_A \mid z_B)$ is always positive, when $\tau \rightarrow \infty$, z $\rightarrow p$

- We will prove it using properties of Markov chains

Markov Chains

- A first-order Markov chain:

$$p(\mathbf{z}^{(m+1)}|\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}) = p(\mathbf{z}^{(m+1)}|\mathbf{z}^{(m)}).$$

- Transition probability: $T\left(\boldsymbol{z}^{(m)}, \boldsymbol{z}^{(m+1)}\right) = p(\boldsymbol{z}^{(m+1)}|\boldsymbol{z}^{(m)})$
- A Markov chain is called homogeneous if all transition probabilities are the same
- Stationary distribution of a Markov chain: each step in the chain does not change the distribution.
  - A step in Markov chain: a variable go to the next one by multiplying the transition probability

More on Stationary Distribution

- For a homogeneous Markov chain, a stationary distribution is

$$p^\star(\mathbf{z}) = \sum_{\mathbf{z}'} T(\mathbf{z}', \mathbf{z}) p^\star(\mathbf{z}').$$

- A Markov chain can have more than one stationary distribution
  - Example: identity transition function

- A sufficient condition to make a distribution stationary

$$p^\star(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = p^\star(\mathbf{z}') T(\mathbf{z}', \mathbf{z})$$

More More on Stationary Distribution

Detailed balance: $p^\star(\mathbf{z})T(\mathbf{z},\mathbf{z}') = p^\star(\mathbf{z}')T(\mathbf{z}',\mathbf{z})$

$$\sum_{\mathbf{z}'} p^\star(\mathbf{z}')T(\mathbf{z}',\mathbf{z}) = \sum_{\mathbf{z}'} p^\star(\mathbf{z})T(\mathbf{z},\mathbf{z}') = p^\star(\mathbf{z})\sum_{\mathbf{z}'} p(\mathbf{z}'|\mathbf{z}) = p^\star(\mathbf{z}).$$

- A Markov chain is said to be *reversible* if it satisfies detailed balance
- A *ergodic* Markov chain converges to the same distribution regardless the initial distribution
  - The system does not return to the same state at fixed intervals
  - The expected number of steps for returning to the same state is finite

Proof about Metropolis Algorithm

- **Theorem**: as long as $q(z_A \mid z_B)$ is always positive, when $\tau \to \infty$, z $\to p$
- **Proof**
  - The Markov chain is ergodic (omitted)
  - The Markov chain satisfies detailed balance
    - The transition probability is

$$T(z_n, z_{n+1}) = q(z_{n+1}|z_n)\times\min(1, \frac{p(z_{n+1})}{p(z_n)})$$

$$\begin{aligned} p(z_n)T(z_n, z_{n+1}) &= q(z_{n+1}|z_n)\times\min(p(z_n), p(z_{n+1})) \\ &= q(z_n|z_{n+1})\times\min(p(z_n), p(z_{n+1})) \\ &= p(z_{n+1})\times q(z_n|z_{n+1})\times\min(\frac{p(z_n)}{p(z_{n+1})}, 1) \\ &= p(z_{n+1})T(z_{n+1}, z_n) \end{aligned}$$

The Metropolis-Hastings algorithm

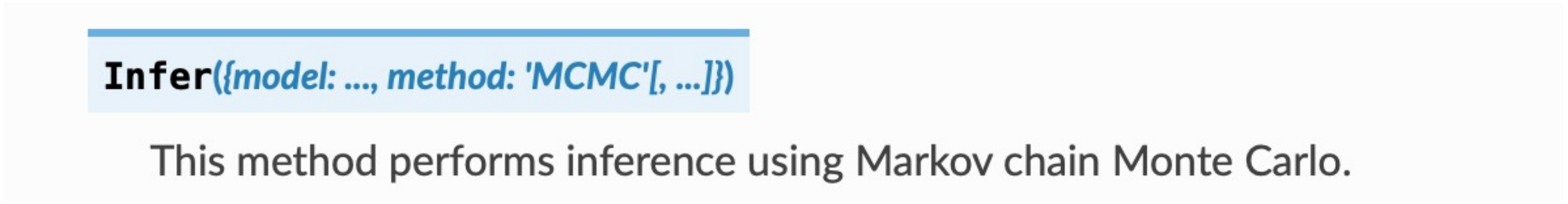
- Generalization of the Metropolis algorithm
  - No restriction on the proposal distribution
  - Now the accepting probability is defined as

$$A_k(\mathbf{z}^\star, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{\widetilde{p}(\mathbf{z}^\star) q_k(\mathbf{z}^{(\tau)}|\mathbf{z}^\star)}{\widetilde{p}(\mathbf{z}^{(\tau)}) q_k(\mathbf{z}^\star|\mathbf{z}^{(\tau)})}\right).$$

$$\begin{aligned} p(\mathbf{z}) q_k(\mathbf{z}|\mathbf{z}') A_k(\mathbf{z}', \mathbf{z}) &= \min\left(p(\mathbf{z}) q_k(\mathbf{z}|\mathbf{z}'), p(\mathbf{z}') q_k(\mathbf{z}'|\mathbf{z})\right) \\ &= \min\left(p(\mathbf{z}') q_k(\mathbf{z}'|\mathbf{z}), p(\mathbf{z}) q_k(\mathbf{z}|\mathbf{z}')\right) \\ &= p(\mathbf{z}') q_k(\mathbf{z}'|\mathbf{z}) A_k(\mathbf{z}, \mathbf{z}') \end{aligned}$$

MCMC in Probabilistic Programming

- Used widely



Infer({model: ..., method: 'MCMC'[, ...]})

This method performs inference using Markov chain Monte Carlo.

Final Notes on MCMC

- Based on the theory of Markov chain
- Can handle a wide range of distributions with high dimensions
- You don't even need to know p, but just the ratio
- A common choice for proposal distribution: Gaussian centered around the current state
- Samples are not independent. What should we do?

More MCMC Methods

- Gibbs sampling
- Slice sampling
- The Hybrid Monte Carlo Algorithm

Summary

- Approximate methods
  - Deterministic (variational inference)
    - Fast but can never get the precise results
  - Stochastic (sampling-based)
    - Slower but can converge the precise result if infinite samples are taken

Summary

- Variational Inference
  - Goal: find a distribution q(Z) that approximates p(Z|X)
  - Idea: by maximizing the evidence lower bound(ELBO)

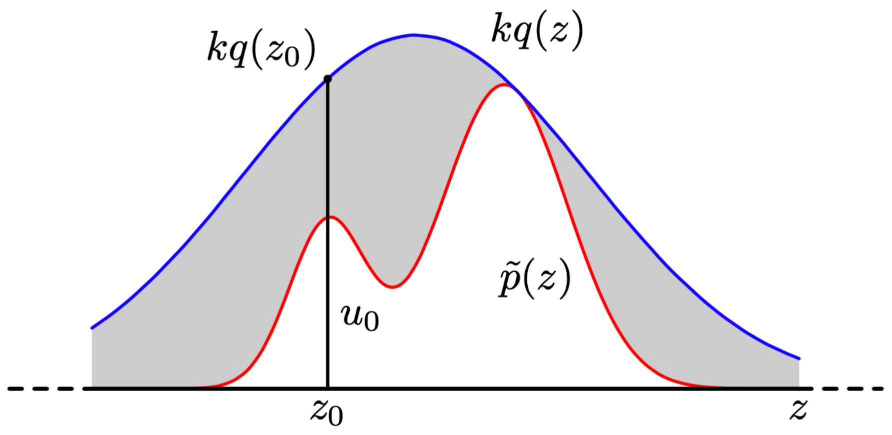
$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z}$$

$$\mathrm{KL}(q \| p) = -\int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z}$$

Summary

- Sample from standard distributions:
  - CDF^{-1}(uniform(0,1))

- Rejection sampling



z ~ q(z), h~ uniform(0, kq(z))
Discard z if h > $\tilde{p}(z)$

Summary

- Importance sampling
  - Also based on proposal distribution
  - Reweight the samples using ratio of probabilities in the two distributions

- Markov Chain Monte Carlo
  - The proposal distribution is the probability of next sample given the current sample
  - Accept a sample if it satisfies some property
  - Forms a Markov Chain
  - Converges to the right distribution because the chain is ergodic and satisfies detailed balance with the desired distribution

Summary

	Metropolis	Metropolis-Hasting
Constraints on the proposal distribution	Symmetric	None
Accepting probability	$\min(1, \frac{p(z')}{p(z)})$	$\min(1, \frac{p(z')q(z'\|z)}{p(z)q(z\|z')})$

Next Class

- Theoretical foundations of probabilistic programming
  - Before moving to inference in probabilistic programming, we first need to understand the problem