

CSC 412/2506:  
Probabilistic Learning and Reasoning  
Week 4 : Message Passing and Monte Carlo

Murat A. Erdogdu

University of Toronto

# Overview

- Trueskill latent variable model
- Message passing

# Latent variables

- Latent variables are unobserved variables that govern certain properties in our probabilistic models.
- What to do when a variable  $z$  is unobserved but our model depends on it?
- If we never condition on  $z$  when in the inference problem, then we can just integrate it out.
- However, in certain cases, we are interested in the latent variables themselves, e.g. the clustering problems.
- More on latent variables when we cover Gaussian mixtures.

# The TrueSkill latent variable model

- TrueSkill model is a player ranking system for competitive games.
- The goal is to infer the skill of a set of players in a competitive game, based on observing who beats who.
- In the TrueSkill model, each player has a fixed level of skill, denoted  $z_i$ .
- We initially don't know anything about anyone's skill, but we assume everyone's skill is independent (e.g. an independent Gaussian prior).
- We never get to observe the players' skills directly, which makes this a latent variable model.

# TrueSkill model

- Instead, we observe the outcome of a series of matches between different players.
- For each game, the probability that player  $i$  beats player  $j$  is given by

$$p(i \text{ beats } j | z_i, z_j) = \sigma(z_i - z_j)$$

where sigma is the logistic function:  $\sigma(y) = \frac{1}{1 + \exp(-y)}$ .

- We can write the entire joint likelihood of a set of players and games as:

$$\begin{aligned} & p(z_1, z_2, \dots, z_N, \text{game 1, game 2, .. game } T) \\ &= \left[ \prod_{i=1}^N p(z_i) \right] \left[ \prod_{\text{games}} p(i \text{ beats } j | z_i, z_j) \right] \end{aligned}$$

# Posterior

- Given the outcome of some matches, the players' skills are no longer independent, even if they've never played each other.
- Computing the **exact** posterior over even two players' skills requires integrating over all the other players' skills:

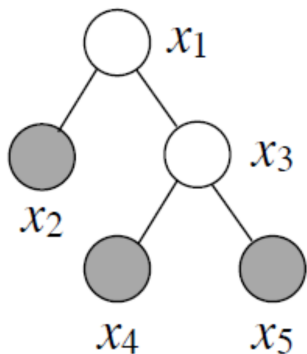
$$\begin{aligned} & p(z_1, z_2 | \text{game 1, game 2, ... game T}) \\ &= \int \cdots \int p(z_1, z_2, z_3 \dots z_N | \text{games}) dz_3 \dots dz_N \end{aligned}$$

- **Message passing** can be used to compute **approximate** posteriors!
- More on this model in Assignment 2.

# Variable Elimination Order and Trees

- **Last week:** we can do exact inference by variable elimination: I.e. to compute  $p(A|C)$ , we can marginalize  $p(A, B|C)$  over every variable in  $B$ , one at a time.
- Computational cost is determined by the graph structure, and the elimination ordering.
- Determining the optimal elimination ordering is hard.
- Even if we do, the resulting marginalization might also be unreasonably costly.
- Fortunately, for trees, any elimination ordering that goes from the leaves inwards towards any root will be optimal.
- You can think of trees as just chains which sometimes branch.

## Inference in Trees (MRF with no cycles)



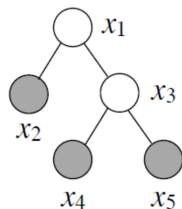
- A graph is  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  where  $\mathcal{V}$  is the set of vertices (nodes) and  $\mathcal{E}$  the set of edges
- For  $i, j \in \mathcal{V}$ , we have  $(i, j) \in \mathcal{E}$  if there is an edge between the nodes  $i$  and  $j$ .
- For a node in graph  $i \in \mathcal{V}$ ,  $N(i)$  denotes the neighbors of  $i$ , i.e.  $N(i) = \{j : (i, j) \in \mathcal{E}\}$ .
- Shaded nodes are observed, and denoted by  $\bar{x}_2, \bar{x}_4, \bar{x}_5$ .

The joint distribution in the general case is

$$p(x_{1:n}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \psi(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j).$$



# Inference in Trees



- Joint distribution is

$$p(x_{1:n}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \psi(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j).$$

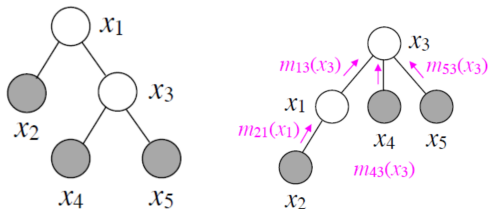
- Want to compute  $p(x_3 | \bar{x}_2, \bar{x}_4, \bar{x}_5)$ .
- We have

$$p(x_3 | \bar{x}_2, \bar{x}_4, \bar{x}_5) \propto p(x_3, \bar{x}_2, \bar{x}_4, \bar{x}_5).$$

$$p(x_3 | \bar{x}_2, \bar{x}_4, \bar{x}_5) = \frac{1}{Z_E} \sum_{x_1} \psi_1(x_1) \psi_3(x_3) \psi_2(\bar{x}_2) \psi_4(\bar{x}_4) \psi_5(\bar{x}_5) \psi_{12}(\bar{x}_2, x_1) \psi_{34}(\bar{x}_4, x_3) \psi_{35}(\bar{x}_5, x_3) \psi_{13}(x_1, x_3)$$

- Let's write the variable elimination algorithm.

# Inference in Trees



$$\begin{aligned}
 p(x_3 \mid \bar{x}_2, \bar{x}_4, \bar{x}_5) &= \frac{1}{Z^E} \sum_{x_1} \psi_1(x_1) \psi_3(x_3) \psi_2(\bar{x}_2) \psi_4(\bar{x}_4) \psi_5(\bar{x}_5) \psi_{12}(\bar{x}_2, x_1) \psi_{34}(\bar{x}_4, x_3) \psi_{35}(\bar{x}_5, x_3) \psi_{13}(x_1, x_3) \\
 &= \frac{1}{Z^E} \underbrace{\psi_4(\bar{x}_4) \psi_{34}(\bar{x}_4, x_3)}_{m_{43}(x_3)} \underbrace{\psi_5(\bar{x}_5) \psi_{35}(\bar{x}_5, x_3)}_{m_{53}(x_3)} \psi_3(x_3) \sum_{x_1} \psi_1(x_1) \psi_{13}(x_1, x_3) \underbrace{\psi_2(\bar{x}_2) \psi_{12}(\bar{x}_2, x_1)}_{m_{21}(x_1)} \\
 &= \frac{1}{Z^E} \psi_3(x_3) m_{43}(x_3) m_{53}(x_3) \underbrace{\sum_{x_1} \psi_1(x_1) \psi_{13}(x_1, x_3) m_{21}(x_1)}_{m_{13}(x_3)} \\
 &= \frac{1}{Z^E} \psi_3(x_3) m_{43}(x_3) m_{53}(x_3) m_{13}(x_3) = \frac{\psi_3(x_3) m_{43}(x_3) m_{53}(x_3) m_{13}(x_3)}{\sum_{x_3} \psi_3(x_3) m_{43}(x_3) m_{53}(x_3) m_{13}(x_3)}
 \end{aligned}$$

Slide credit: S. Ermon

## Message Passing on Trees

We perform variable elimination from leaves to root, which is the sum product algorithm to compute all marginals. Belief propagation is a message-passing between neighboring vertices of the graph.

- The message sent from variable  $j$  to  $i \in N(j)$  is

$$m_{j \rightarrow i}(x_i) = \sum_{x_j} \psi_j(x_j) \psi_{ij}(x_i, x_j) \prod_{k \in N(j)/i} m_{k \rightarrow j}(x_j)$$

- ▶ If  $x_j$  is observed, the message is

$$m_{j \rightarrow i}(x_i) = \psi_j(\bar{x}_j) \psi_{ij}(x_i, \bar{x}_j) \prod_{k \in N(j)/i} m_{k \rightarrow j}(\bar{x}_j)$$

- Once the message passing stage is complete, we can compute our beliefs as

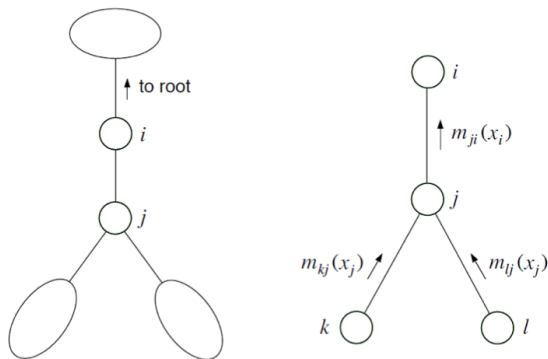
$$b(x_i) \propto \psi_i(x_i) \prod_{j \in N(i)} m_{j \rightarrow i}(x_i).$$

- Once normalized, beliefs are the marginals we want to compute!

# Message Passing on Trees

The message sent from variable  $j$  to  $i \in N(j)$  is

$$m_{j \rightarrow i}(x_i) = \sum_{x_j} \psi_j(x_j) \psi_{ij}(x_i, x_j) \prod_{k \in N(j)/i} m_{k \rightarrow j}(x_j)$$

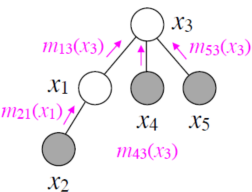


Each message  $m_{j \rightarrow i}(x_i)$  is a vector with one value for each state of  $x_i$ .

# Inference in Trees: Compute $p(x_3|\bar{x}_2, \bar{x}_4, \bar{x}_5)$

$$m_{j \rightarrow i}(x_i) = \sum_{x_j} \psi_j(x_j) \psi_{ij}(x_i, x_j) \prod_{k \in N(j)/i} m_{k \rightarrow j}(x_j)$$

$$b(x_i) \propto \psi_i(x_i) \prod_{j \in N(i)} m_{j \rightarrow i}(x_i).$$



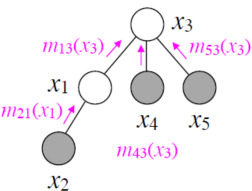
- $m_{5 \rightarrow 3}(x_3) = \psi_5(\bar{x}_5) \psi_{35}(x_3, \bar{x}_5)$
- $m_{2 \rightarrow 1}(x_1) = \psi_2(\bar{x}_2) \psi_{12}(x_1, \bar{x}_2)$
- $m_{4 \rightarrow 3}(x_3) = \psi_4(\bar{x}_4) \psi_{34}(x_3, \bar{x}_4)$
- $m_{1 \rightarrow 3}(x_3) = \sum_{x_1} \psi_1(x_1) \psi_{13}(x_1, x_3) m_{2 \rightarrow 1}(x_1)$
- $b(x_3) \propto \psi_3(x_3) m_{1 \rightarrow 3}(x_3) m_{4 \rightarrow 3}(x_3) m_{5 \rightarrow 3}(x_3)$

This is the same as variable elimination, so

$$p(x_3|\bar{x}_2, \bar{x}_4, \bar{x}_5) = b(x_3)$$

# Belief Propagation on Trees

## Belief Propagation Algorithm on Trees



- Choose root  $r$  arbitrarily
- Pass messages from leafs to  $r$
- Pass messages from  $r$  to leafs
- These two passes are sufficient on trees!
- Compute beliefs (marginals)

$$b(x_i) \propto \psi_i(x_i) \prod_{j \in \mathcal{N}(i)} m_{j \rightarrow i}(x_i), \quad \forall_i$$

One can compute them in two steps:

- Compute unnormalized beliefs  $\tilde{b}(x_i) \propto \psi_i(x_i) \prod_{j \in \mathcal{N}(i)} m_{j \rightarrow i}(x_i)$
- Normalize them  $b(x_i) = \tilde{b}(x_i) / \sum_{x_i} \tilde{b}(x_i)$ .

# Loopy Belief Propagation

- What if the graph (MRF) we have is not a tree and have cycles?
- Keep passing messages until convergence.
- This is called **Loopy Belief Propagation**.
- This is like when someone starts a rumour and then hears the same rumour from someone else, making them more certain it's true.
- We won't get the exact marginals, but an approximation.
- But turns out it is still very useful!

# Loopy Belief Propagation

Loopy BP:

- Initialize all messages uniformly:

$$m_{i \rightarrow j}(x_j) = [1/k, \dots, 1/k]^\top$$

where  $k$  is the number of states  $x_j$  can take.

- Keep running BP updates until it “converges”:

$$m_{j \rightarrow i}(x_i) = \sum_{x_j} \psi_j(x_j) \psi_{ij}(x_i, x_j) \prod_{k \in \mathcal{N}(j)/i} m_{k \rightarrow j}(x_j)$$

and (sometimes) normalized for stability.

- It will generally not converge, but that’s generally ok.
- Compute beliefs

$$b(x_i) \propto \psi_i(x_i) \prod_{j \in \mathcal{N}(i)} m_{j \rightarrow i}(x_i).$$

This algorithm is still very useful in practice, without any theoretical guarantee (other than trees).



## Sum-product vs. Max-product

- The algorithm we learned is called **sum-product BP** and approximately computes the **marginals** at each node.
- For MAP inference, we maximize over  $x_j$  instead of summing over them. This is called **max-product BP**.
- BP updates take the form

$$m_{j \rightarrow i}(x_i) = \max_{x_j} \psi_j(x_j) \psi_{ij}(x_i, x_j) \prod_{k \in N(j) \neq i} m_{k \rightarrow j}(x_j)$$

- After BP algorithm converges, the beliefs are **max-marginals**

$$b(x_i) \propto \psi_i(x_i) \prod_{j \in N(i)} m_{j \rightarrow i}(x_i).$$

- MAP inference:

$$\hat{x}_i = \arg \max_{x_i} b(x_i).$$

# Summary

- This algorithm is still very useful in practice, without much theoretical guarantee (other than trees).
- Loopy BP multiplies the same potentials multiple times. It is often over-confident.
- Loopy BP can oscillate, but this is generally ok.
- Loopy BP often works better if we normalize messages, and use momentum in the updates.
- The algorithm we learned is called **sum-product BP**. If we are interested in MAP inference, we can maximize over  $x_j$  instead of summing over them. This is called **max-product BP**.

# Monte Carlo: Overview

- Ancestral Sampling
- Simple Monte Carlo
- Importance Sampling
- Rejection Sampling

# Sampling

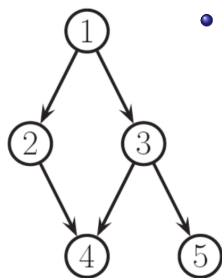
- A sample from a distribution  $p(x)$  is a single realization  $x$  whose probability distribution is  $p(x)$ . Here,  $x$  can be high-dimensional or simply real valued.
- We assume the density from which we wish to draw samples,  $p(x)$ , can be evaluated to within a multiplicative constant. That is, we can evaluate a function  $\tilde{p}(x)$  such that

$$p(x) = \frac{\tilde{p}(x)}{Z}.$$

## Warm up: Ancestral Sampling

- Given a DAGM, and the ability to sample from each of its factors given its parents, we can sample from the joint distribution over all the nodes by **ancestral sampling**, which simply means sampling in a topological order.
- At each step, sample from any conditional distribution that you haven't visited yet, whose parents have all been sampled.

# Ancestral Sampling Example



- The graph factorizes according to the local conditional probabilities

$$\begin{aligned} p(x_1, \dots, x_N) &= \prod_i^N p(x_i | \text{parents}(x_i)) \\ &= p(x_1) p(x_2 | x_1) p(x_3 | x_1) p(x_4 | x_2, x_3) p(x_5 | x_3) \end{aligned}$$

- Start by sampling from  $p(x_1)$ .
- Then sample from  $p(x_2 | x_1)$  and  $p(x_3 | x_1)$ .
- Then sample from  $p(x_4 | x_2, x_3)$ .
- Finally, sample from  $p(x_5 | x_3)$ .

# Main objectives of sampling

We will be using Monte Carlo methods to solve one or both of the following problems.

- **Problem 1:** To generate samples  $\{x^{(r)}\}_{r=1}^R$  from a given probability distribution  $p(x)$ .
- **Problem 2:** To estimate expectations of functions,  $\phi(x)$ , under this distribution  $p(x)$

$$\Phi = \mathbb{E}_{x \sim p(x)} [\phi(x)] = \int \phi(x)p(x)dx$$

$\phi$  is called a test function.

## Example

Examples of test functions  $\phi(x)$ :

- the **mean** of a function  $f$  under  $p(x)$  by finding the expectation of the function  $\phi_1(x) = f(x)$ .
- the **variance** of  $f$  under  $p(x)$  by finding the expectations of the functions  $\phi_1(x) = f(x)$  and  $\phi_2(x) = f(x)^2$

$$\phi_1(x) = f(x) \Rightarrow \Phi_1 = \mathbb{E}_{x \sim p(x)} [\phi_1(x)]$$

$$\phi_2(x) = f(x)^2 \Rightarrow \Phi_2 = \mathbb{E}_{x \sim p(x)} [\phi_2(x)]$$

$$\Rightarrow \text{var}(f(x)) = \Phi_2 - (\Phi_1)^2$$



# Estimation problem

We start with the estimation problem using simple Monte Carlo:

- **Simple Monte Carlo:** Given  $\{x^{(r)}\}_{r=1}^R \sim p(x)$  we can estimate the expectation  $\mathbb{E}_{x \sim p(x)} [\phi(x)]$  using the estimator  $\hat{\Phi}$ :

$$\Phi := \mathbb{E}_{x \sim p(x)} [\phi(x)] \approx \frac{1}{R} \sum_{r=1}^R \phi(x^{(r)}) := \hat{\Phi}$$

- The fact that  $\hat{\Phi}$  is a consistent estimator of  $\Phi$  follows from the Law of Large Numbers (LLN).

# Basic properties of Monte Carlo estimation

- **Unbiasedness:** If the vectors  $\{x^{(r)}\}_{r=1}^R$  are generated independently from  $p(x)$ , then the expectation of  $\hat{\Phi}$  is  $\Phi$ .

$$\begin{aligned}\mathbb{E}[\hat{\Phi}] &= \mathbb{E}\left[\frac{1}{R} \sum_{r=1}^R \phi(x^{(r)})\right] = \frac{1}{R} \sum_{r=1}^R \mathbb{E}[\phi(x^{(r)})] \\ &= \frac{1}{R} \sum_{r=1}^R \mathbb{E}_{x \sim p(x)}[\phi(x)] = \frac{R}{R} \mathbb{E}_{x \sim p(x)}[\phi(x)] \\ &= \Phi\end{aligned}$$

## Simple properties of Monte Carlo estimation

- **Variance:** As the number of samples of  $R$  increases, the variance of  $\hat{\Phi}$  will decrease with rate  $\frac{1}{R}$

$$\begin{aligned}\text{var}[\hat{\Phi}] &= \text{var}\left[\frac{1}{R} \sum_{r=1}^R \phi(x^{(r)})\right] \\ &= \frac{1}{R^2} \text{var}\left[\sum_{r=1}^R \phi(x^{(r)})\right] \\ &= \frac{1}{R^2} \sum_{r=1}^R \text{var}\left[\phi(x^{(r)})\right] \\ &= \frac{R}{R^2} \text{var}[\phi(x)] \\ &= \frac{1}{R} \text{var}[\phi(x)]\end{aligned}$$

Accuracy of the Monte Carlo estimate depends on the variance of  $\phi$ .

## Normalizing constant

- Assume we know the density  $p(x)$  up to a multiplicative constant

$$p(x) = \frac{\tilde{p}(x)}{Z}$$

- There are two difficulties:

- ▶ We do not generally know the normalizing constant,  $Z$ . The main difficulty is computing it

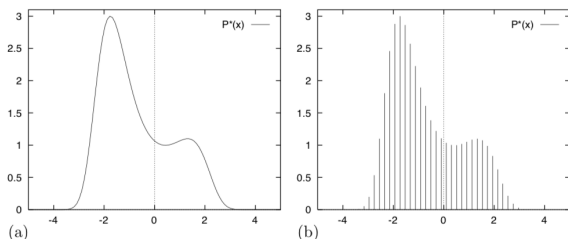
$$Z = \int \tilde{p}(x) dx$$

which requires computing a high-dimensional integral.

- ▶ Even if we did know  $Z$ , the problem of drawing samples from  $p(x)$  is still a challenging one, especially in high-dimensional spaces.

## Bad Idea: Lattice Discretization

Imagine that we wish to draw samples from the density  $p(x) = \frac{\tilde{p}(x)}{Z}$  given in figure (a).

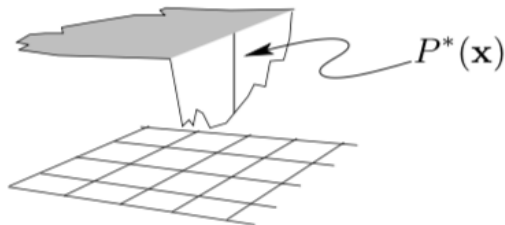


- How to compute  $Z$ ?
- We could discretize the variable  $x$  and sample from the discrete distribution (figure (b)).
- In figure (b) there are 50 uniformly spaced points in one dimension. If our system had,  $D = 1000$  dimensions say, then the corresponding number of points would be  $50^D = 50^{1000}$ . Thus, the cost is exponential in dimension!

## An analogy

Imagine the tasks of drawing random water samples from a lake and finding the average plankton concentration. Let

- $\tilde{p}(\mathbf{x})$  = the depth of the lake at  $\mathbf{x} = (x, y)$
- $\phi(\mathbf{x})$  = the plankton concentration as a function of  $\mathbf{x}$
- $Z$  = the volume of the lake =  $\int \tilde{p}(\mathbf{x}) d\mathbf{x}$



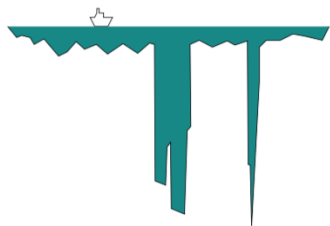
The average concentration of plankton is therefore

$$\Phi = \frac{1}{Z} \int \phi(\mathbf{x}) \tilde{p}(\mathbf{x}) d\mathbf{x}.$$

## An analogy

You can take the boat to any desired location  $\mathbf{x}$  on the lake, and can measure the depth,  $\tilde{p}(\mathbf{x})$ , and plankton concentration,  $\phi(\mathbf{x})$ , at that point. Therefore,

- **Problem 1** is to draw water samples at random such that each sample is equally likely to come from any point within the lake.
- **Problem 2** is to find the average plankton concentration.

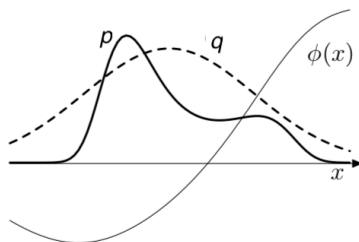


A slice through a lake that includes some canyons.

- To correctly estimate  $\Phi$ , our method must implicitly discover the canyons and find their volume relative to the rest of the lake.

# Estimation tool: Importance Sampling

**Importance sampling** is a method for estimating the expectation of a function  $\phi(x)$ .



- The density from which we wish to draw samples,  $p(x)$ , can be evaluated up to normalizing constant,  $\tilde{p}(x)$

$$p(x) = \frac{\tilde{p}(x)}{Z_p}$$

- There is a simpler density,  $q(x)$  from which it is easy to sample from and easy to evaluate up to normalizing constant (i.e.  $\tilde{q}(x)$ )

$$q(x) = \frac{\tilde{q}(x)}{Z_q}$$



## Estimation tool: Importance Sampling

- In importance sampling, we generate  $R$  samples from  $q(x)$

$$\{x^{(r)}\}_{r=1}^R \sim q(x)$$

- If these points were samples from  $p(x)$  then we could estimate  $\Phi$  by

$$\Phi = \mathbb{E}_{x \sim p(x)} [\phi(x)] \approx \frac{1}{R} \sum_{r=1}^R \phi(x^{(r)}) = \hat{\Phi}$$

That is, we could use a simple Monte Carlo estimator.

- But we sampled from  $q$ . We need to correct this!
- Values of  $x$  where  $q(x)$  is greater than  $p(x)$  will be over-represented in this estimator, and points where  $q(x)$  is less than  $p(x)$  will be under-represented. Thus, we introduce weights.

- Introduce weights:  $\tilde{w}_r = \frac{\tilde{p}(x^{(r)})}{\tilde{q}(x^{(r)})}$  and notice that

$$\frac{1}{R} \sum_{r=1}^R \tilde{w}_r \approx \mathbb{E}_{x \sim q(x)} \left[ \frac{\tilde{p}(x)}{\tilde{q}(x)} \right] = \int \frac{\tilde{p}(x)}{\tilde{q}(x)} q(x) dx = \frac{Z_p}{Z_q}$$

- Finally, we rewrite our estimator under  $q$

$$\Phi = \int \phi(x) p(x) dx = \int \phi(x) \cdot \frac{p(x)}{q(x)} \cdot q(x) dx \approx \frac{1}{R} \sum_{r=1}^R \phi(x^{(r)}) \frac{p(x^{(r)})}{q(x^{(r)})} = (*)$$

- However, the estimator relies on  $p$ . It can only rely on  $\tilde{p}$  and  $\tilde{q}$ .

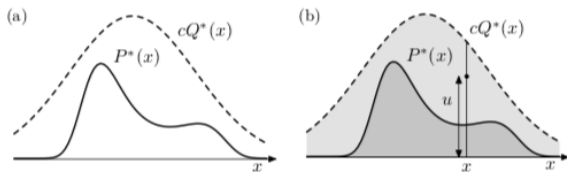
$$\begin{aligned} (*) &= \frac{Z_q}{Z_p} \frac{1}{R} \sum_{r=1}^R \phi(x^{(r)}) \cdot \frac{\tilde{p}(x^{(r)})}{\tilde{q}(x^{(r)})} = \frac{Z_q}{Z_p} \frac{1}{R} \sum_{r=1}^R \phi(x^{(r)}) \cdot \tilde{w}_r \\ &\approx \frac{\frac{1}{R} \sum_{r=1}^R \phi(x^{(r)}) \cdot \tilde{w}_r}{\frac{1}{R} \sum_{r=1}^R \tilde{w}_r} = \sum_{r=1}^R \phi(x^{(r)}) \cdot w_r = \hat{\Phi}_{iw} \end{aligned}$$

where  $w_r = \frac{\tilde{w}_r}{\sum_{r=1}^R \tilde{w}_r}$  and  $\hat{\Phi}_{iw}$  is our importance weighted estimator.

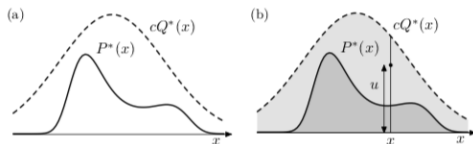
## Sampling tool: Rejection sampling

- We want expectations under  $p(x) = \tilde{p}(x)/Z_p$  which is a very complicated one-dimensional density.
- Assume that we have a simpler proposal density  $q(x)$  which we can evaluate (within a multiplicative factor  $Z_q$ , as before), and from which we can generate samples, i.e.  $\tilde{q}(x) = Z_q \cdot q(x)$ .
- Further assume that we know the value of a constant  $c$  such that

$$c\tilde{q}(x) > \tilde{p}(x) \quad \forall x$$



# Sampling tool: Rejection sampling



The procedure is as follows:

1. Generate two random numbers.
  - 1.1 The first,  $x$ , is generated from the proposal density  $q(x)$ .
  - 1.2 The second,  $u$  is generated uniformly from the interval  $[0, c\tilde{q}(x)]$  (see figure (b) above: book's notation  $P^* = \tilde{p}$ ,  $Q^* = \tilde{q}$ ).
2. Accept or reject the sample  $x$  by comparing the value of  $u$  with the value of  $\tilde{p}(x)$ 
  - 2.1 If  $u > \tilde{p}(x)$ , then  $x$  is rejected
  - 2.2 Otherwise  $x$  is accepted;  $x$  is added to our set of samples  $\{x^{(r)}\}$  and the value of  $u$  discarded.

# Why does rejection sampling work?

1.  $x \sim q(x)$
2.  $u|x \sim \text{Unif}[0, c\tilde{q}(x)]$
3.  $x$  is accepted if  $u \leq \tilde{p}(x)$ .

For any set  $A$

$$\mathbb{P}_{x \sim p}(x \in A) = \int_A p(x) dx = \int \mathbf{1}_{\{x \in A\}} p(x) dx = \mathbb{E}_{x \sim p}[\mathbf{1}_{\{x \in A\}}].$$

$$\begin{aligned} \mathbb{P}_{x \sim q}(x \in A | u \leq \tilde{p}(x)) &= \mathbb{P}_{x \sim q}(x \in A, u \leq \tilde{p}(x)) / \mathbb{E}_{x \sim q}[\mathbb{P}(u \leq \tilde{p}(x) | x)] \\ &= \mathbb{E}_{x \sim q}[\mathbf{1}_{\{x \in A\}} \mathbb{P}(u \leq \tilde{p}(x) | x)] / \mathbb{E}_{x \sim q}\left[\frac{\tilde{p}(x)}{c\tilde{q}(x)}\right] \\ &= \mathbb{E}_{x \sim q}\left[\mathbf{1}_{\{x \in A\}} \frac{\tilde{p}(x)}{c\tilde{q}(x)}\right] / \frac{Z_p}{cZ_q} \\ &= \mathbb{P}_{x \sim p}(x \in A) \frac{Z_p}{cZ_q} / \frac{Z_p}{cZ_q} \\ &= \mathbb{P}_{x \sim p}(x \in A) \end{aligned}$$

# Rejection sampling in many dimensions

- In high-dimensional problems, the requirement that  $c\tilde{q}(x) \geq \tilde{p}(x)$  will force  $c$  to be huge, so acceptances will be very rare.
- Finding such a value of  $c$  may be difficult too, since we don't know where the modes of  $\tilde{p}$  are located nor how high they are.
- In general  $c$  grows exponentially with the dimensionality, so the acceptance rate is expected to be exponentially small in dimension

$$\text{acceptance rate} = \frac{\text{area under } \tilde{p}}{\text{area under } c\tilde{q}} = \frac{Z_p}{cZ_q}$$

# Summary

- Estimating expectations is an important problem, which is in general hard. We learned 3 sampling-based tools for this task:
  - ▶ Simple Monte Carlo
  - ▶ Importance Sampling
  - ▶ Rejection Sampling
- Next lecture, we will learn to generate samples from a particular distribution.