STA 414/2104: Probabilistic Machine Learning

Week 3: Exact/Approximate Inference

Murat A. Erdogdu

University of Toronto

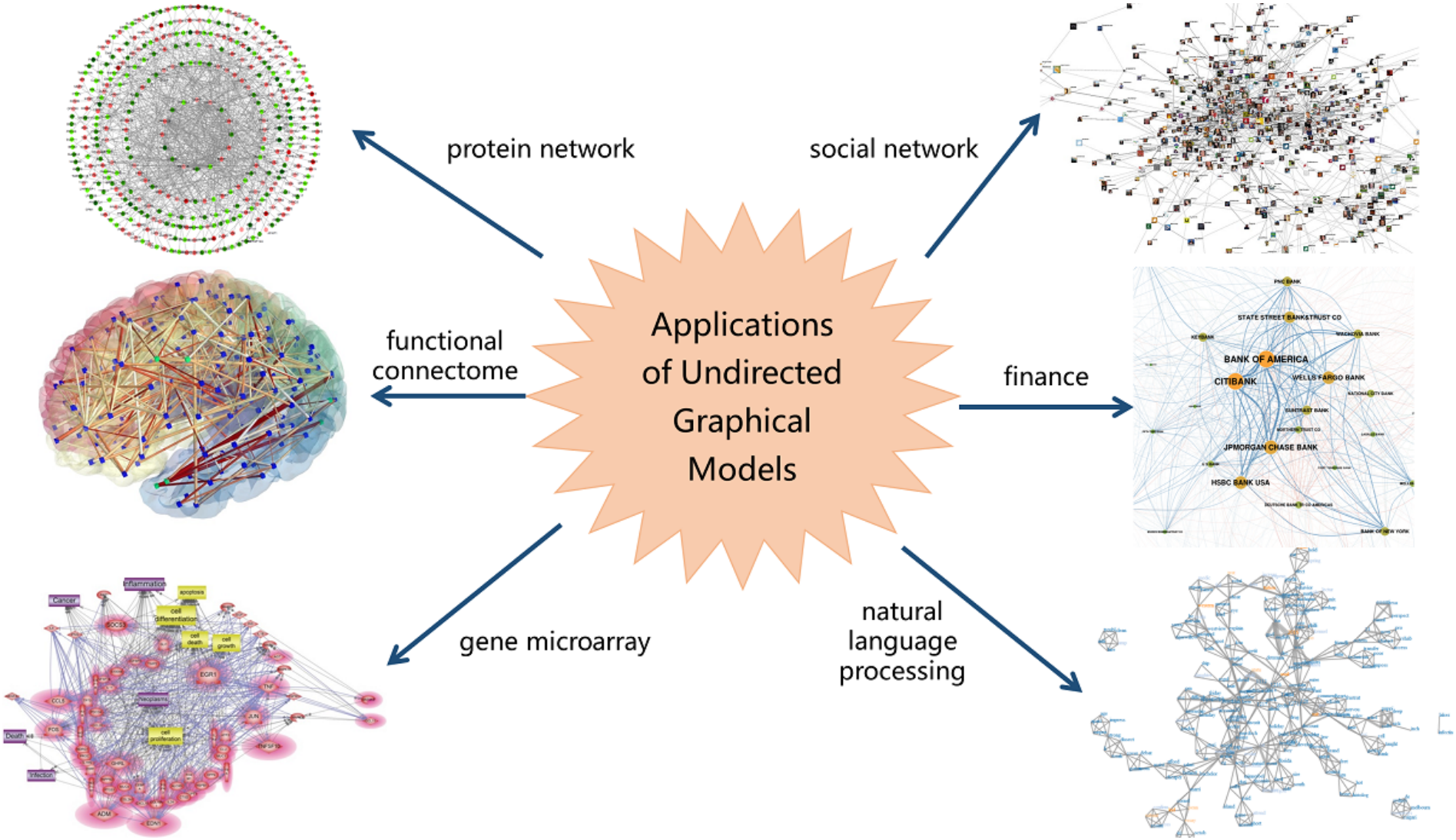
Today's lecture

Summary of the content:

- Exact inference on graphical models
- Variable elimination
- Intro to approximate inference

Some announcements:

- Assignment 1 is released this week.
- TA office hours next week.


protein network
social network
functional connectome
Applications of Undirected Graphical Models
finance
gene microarray
natural language processing
BANK OF AMERICA
CITIBANK
JPMORGAN CHASE BANK
HSBC BANK USA

Example

- This describes a probability distribution over 8 variables.
- In general, we are only interested in the conditional distribution of few.
- For example, ..

What is the probability of a getting a job given you have 3.1 GPA?

Inference as Conditional Distribution

- We explore inference in probabilistic graphical models (PGMs).
  - x_E = The observed evidence
  - x_F = The unobserved variable we want to infer
  - $x_R = \boldsymbol{x} - \{x_F, x_E\}$ = Remaining variables, extraneous to query.

- Focus on computing the conditional probability distribution

$$p(x_F|x_E) = \frac{p(x_F, x_E)}{p(x_E)} = \frac{p(x_F, x_E)}{\sum_{x_F} p(x_F, x_E)}$$

- for which, we marginalize out these extraneous variables, focussing on the joint distribution over evidence and subject of inference:

$$p(x_F, x_E) = \sum_{x_R} p(x_F, x_E, x_R)$$

Variable elimination

Order in which we marginalize affects the computational cost!

Our main tool is **variable elimination**:

- A simple and general **exact inference** algorithm in any probabilistic graphical model (DAGMs or MRFs).
- Computational complexity depends on the graph structure.
- Dynamic programming avoids enumerating all variable assignments.

Example: Simple chain

- Lets start with the example of a simple chain

$$A \to B \to C \to D$$

  where we want to compute $p(D)$, with no evidence variables.
- We have

$$x_F = \{D\},\ x_E = \{\},\ x_R = \{A, B, C\}$$

- We saw last lecture that this graphical model describes the factorization of the joint distribution as:

$$p(A, B, C, D) = p(A)p(B|A)p(C|B)p(D|C)$$

- Assume each variable can take on k different values.

Example: Simple chain

- The goal is to compute the marginal $p(D)$:

$$p(D) = \sum_{A,B,C} p(A, B, C, D)$$

- However, if we do this sum naively, cost is exponential $O(k^{n=4})$:

$$\begin{aligned} p(D) &= \sum_{A,B,C} p(A, B, C, D) \\ &= \sum_{C} \sum_{B} \sum_{A} p(A)p(B|A)p(C|B)p(D|C) \end{aligned}$$

- Instead, choose an **elimination ordering**:

$$\begin{aligned} p(D) &= \sum_{C,B,A} p(A, B, C, D) \\ &= \sum_{C} p(D|C) \left(\sum_{B} p(C|B) \Big(\sum_{A} p(A)p(B|A) \Big) \right). \end{aligned}$$

Example: Simple chain

- This reduces the complexity by first computing terms that appear across the other sums.

$$p(D) = \sum_C p(D|C) \sum_B p(C|B) \sum_A p(A)p(B|A)$$

-

$$\begin{aligned} p(D) &= \sum_C p(D|C) \sum_B p(C|B) \sum_A p(A)p(B|A) \\ &= \sum_C p(D|C) \sum_B p(C|B)p(B) \\ &= \sum_C p(D|C)p(C) \end{aligned}$$

- The cost of performing inference on the chain in this manner is $\mathcal{O}(nk^2)$. In comparison, generating the full joint distribution and marginalizing over it has complexity $\mathcal{O}(k^n)$!

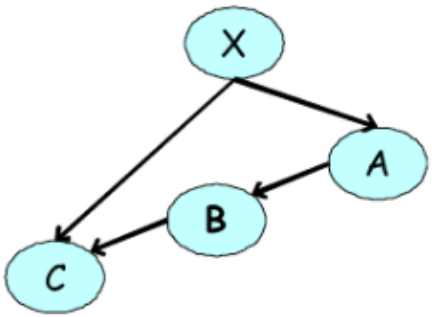
Best Elimination Ordering

- The complexity of variable elimination depends on the elimination ordering!
- Unfortunately, finding the best elimination ordering is NP-hard.

Intermediate Factors

The same algorithm both for DAGMs and MRFs:

- Introduce nonnegative **factors** ϕ (like for MRFs).
- e.g. in a simple DAG model:



$$
\begin{aligned}
p(A,B,C) &= \sum_X p(X)p(A|X)p(B|A)p(C|B,X) \\
&= \sum_X \phi_1(X)\phi_2(A,X)\phi_3(A,B)\phi_4(X,B,C) \\
&= \phi_3(A,B)\sum_X \phi_1(X)\phi_2(A,X)\phi_4(X,B,C) \\
&= \phi_3(A,B)\tau(A,B,C)
\end{aligned}
$$

- Marginalizing over X we introduce a new factor, denoted by τ.

Sum-Product Inference

- Abstractly, computing $p(x_F|x_E)$ is given by the **sum-product** algorithm:

$$p(x_F|x_E) \propto \tau(x_F, x_E) = \sum_{x_R} \prod_{C \in \mathcal{F}} \psi_C(x_C)$$

  where $\mathcal{F}$ is a set of potentials or factors.
- For DAGMs, $\mathcal{F}$ is given by the the sets of the form

$$\{i\} \cup \text{parents}(i) \qquad \text{for all nodes } i.$$

- For MRFs, $\mathcal{F}$ is given by the set of maximal cliques.

Example

- This describes a factorization:

$$p(C, D, I, G, S, L, H, J) = p(C)p(D|C)p(I) \times p(G|D, I)p(L|G)p(S|I)p(J|S, L)p(H|J, G)$$

We have

$$\mathcal{F} = \{\{C\}, \{C, D\}, \{I\}, \{G, D, I\}, \{L, G\}, \{S, I\}, \{J, S, L\}, \{H, J, G)\}\}$$

We are interested in the probability of getting a job, $p(J)$.

We perform exact inference as follows.

Example $\left(\mathcal{F} = \{\{C\}, \{C, D\}, \{I\}, \{G, D, I\}, \{L, G\}, \{S, I\}, \{J, S, L\}, \{H, J, G)\}\}\right)$

Elimination Ordering $\prec \{C, D, I, H, G, S, L\}$

$$p(J) = \sum_L \sum_S \psi(J, L, S) \sum_G \psi(L, G) \sum_H \psi(H, G, J) \sum_I \psi(S, I)\psi(I) \sum_D \psi(G, D, I) \underbrace{\sum_C \psi(C)\psi(C, D)}_{\tau(D)}$$

$$= \sum_L \sum_S \psi(J, L, S) \sum_G \psi(L, G) \sum_H \psi(H, G, J) \sum_I \psi(S, I)\psi(I) \underbrace{\sum_D \psi(G, D, I)\tau(D)}_{\tau(G,I)}$$

$$= \sum_L \sum_S \psi(J, L, S) \sum_G \psi(L, G) \sum_H \psi(H, G, J) \underbrace{\sum_I \psi(S, I)\psi(I)\tau(G, I)}_{\tau(S,G)}$$

$$= \sum_L \sum_S \psi(J, L, S) \sum_G \psi(L, G)\tau(S, G) \underbrace{\sum_H \psi(H, G, J)}_{\tau(G,J)}$$

$$= \sum_L \sum_S \psi(J, L, S) \underbrace{\sum_G \psi(L, G)\tau(S, G)\tau(G, J)}_{\tau(J,L,S)}$$

$$= \sum_L \underbrace{\sum_S \psi(J, L, S)\tau(J, L, S)}_{\tau(J,L)}$$

$$= \underbrace{\sum_L \tau(J, L)}_{\tau(J)}$$

$$= \tau(J)$$

Do we need to normalize the final τ?

Complexity of Variable Elimination Ordering

- We discussed previously that variable elimination ordering determines the computational complexity. This is due to how many variables appear inside each sum.
- Different elimination orderings will involve different number of variables appearing inside each sum.
- The complexity of the VE algorithm is (roughly)

$$O(mk^{N_{\max}})$$

where

  - m is the number of initial factors.
  - k is the number of states each random variable takes (assumed to be equal here).
  - N_i is the number of random variables inside each sum $\sum_i$.
  - $N_{\max} = \max_i N_i$ is the number of variables inside the largest sum.

Example

Elimination Ordering $\prec \{C, D, I, H, G, S, L\}$

- Here are all the initial factors:

$$\mathcal{F} = \{\{C\}, \{C, D\}, \{I\}, \{G, D, I\}, \{L, G\}, \{S, I\}, \{J, S, L\}, \{H, J, G)\}\}$$

$$\implies m = |\Phi| = 8$$

- Here are the sums, and the number of variables that appear in them

$$\underbrace{\sum_{C} \psi(C)\psi(C, D)}_{N_C=2} \quad \underbrace{\sum_{D} \psi(G, D, I)\tau(D)}_{N_D=3} \quad \underbrace{\sum_{I} \psi(S, I)\psi(I)\tau(G, I)}_{N_I=3}$$

$$\underbrace{\sum_{H} \psi(H, G, J)}_{N_H=3} \quad \underbrace{\sum_{G} \psi(L, G)\tau(S, G)\tau(G, J)}_{N_G=4} \quad \underbrace{\sum_{S} \psi(J, L, S)\tau(J, L, S)}_{N_S=3}$$

$$\underbrace{\sum_{L} \tau(J, L)}_{N_L=2} \implies \text{the largest sum is } N_G = 4.$$

- For simplicity, assume all variables take on k states. So the complexity of the variable elimination under this ordering is $O(8 \cdot k^4)$.

Summary

Variable elimination:

- Variable elimination can be used for exact inference in PGMs.
- The ordering in variable elimination can significantly reduce the computational complexity.
- The overall complexity of the variable elimination algorithm can be computed.

Approximate Inference: Overview

- In general, we do not have access to any (or some) of the (conditional) distributions, say $p(x)$.
- For this reason, we need tools to approximate them with $\hat{p}(x)$ when performing inference!
- Two common approaches:
  - Generate samples $x_i \sim p(x)$ and use them for estimation.
  - Directly approximate $\hat{p}(x)$, e.g. with neural networks.

Sampling approach

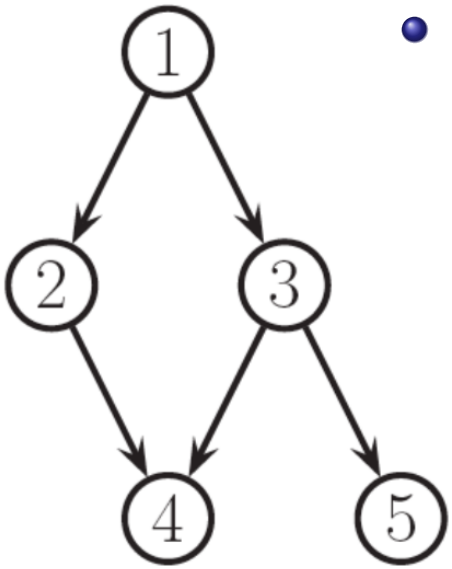
- A sample from a distribution $p(x)$ is a single realization x whose probability distribution is $p(x)$. Here, x can be high-dimensional or simply real valued.
- We assume the density from which we wish to draw samples, $p(x)$, can be evaluated to within a multiplicative constant. That is, we can evaluate 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 topoplogical order.
- At each step, sample from any conditional distribution that you haven't visited yet, whose parents have all been sampled.
- For this algorithm, we assume that we can sample from conditional distributions.

Ancestral Sampling Example



- The graph factorizes according to the local conditional probabilities

$$p(x_{1,\ldots,N}) = \prod_{i}^{N} p(x_i|\text{parents}(x_i))$$
$$= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2,x_3)p(x_5|x_3)$$

- 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 $p(x)$

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

ϕ 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 $\phi_2(x) = f(x)^2$

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

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

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

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 $\underset{x\sim p(x)}{\mathbb{E}}[\phi(x)]$ using the estimator $\hat{\Phi}$:

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

- The fact that $\hat{\Phi}$ is a consistent estimator of Φ 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 Φ.

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

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 ϕ.

Normalizing constant

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

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

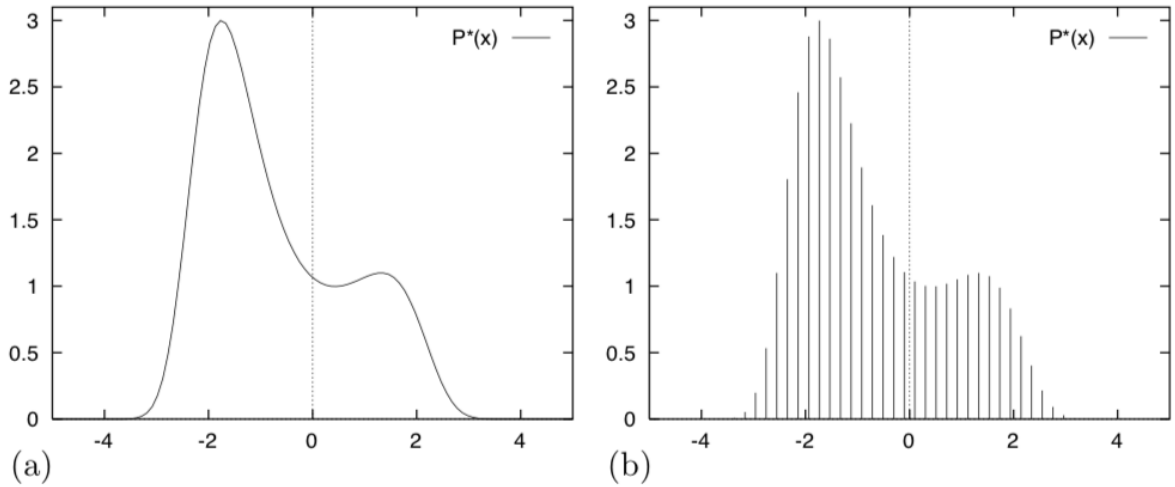
- There are two difficulties:
  - We do not generally know the normalizing constant, Z. The main diffuculty is computing it

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

    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).

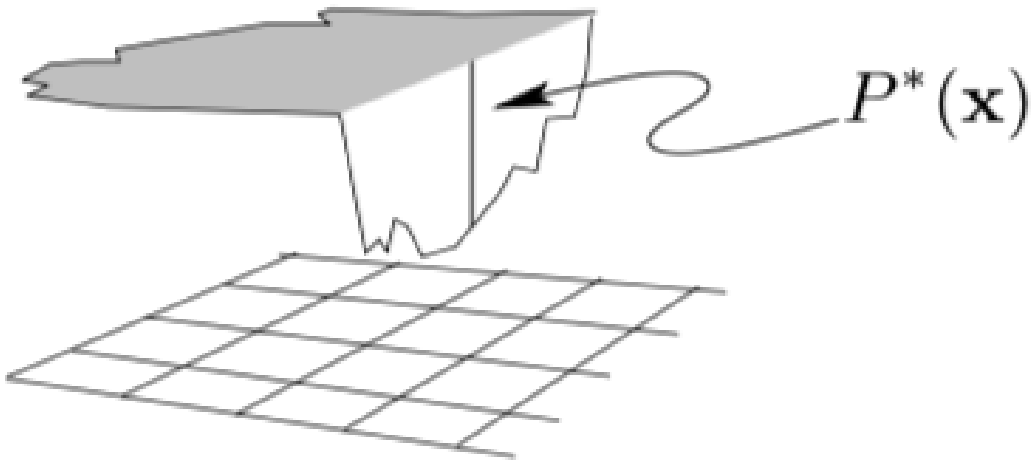


- How to compute Z?
- We could discretize the variable x and sample from the discrete distribution (figure (b)).
- In figure (b) there are 50 uniformly spaced points in one dimension. If our system had, $D = 1000$ dimensions say, then the corresponding number of points would be $50^D = 50^{1000}$. Thus,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

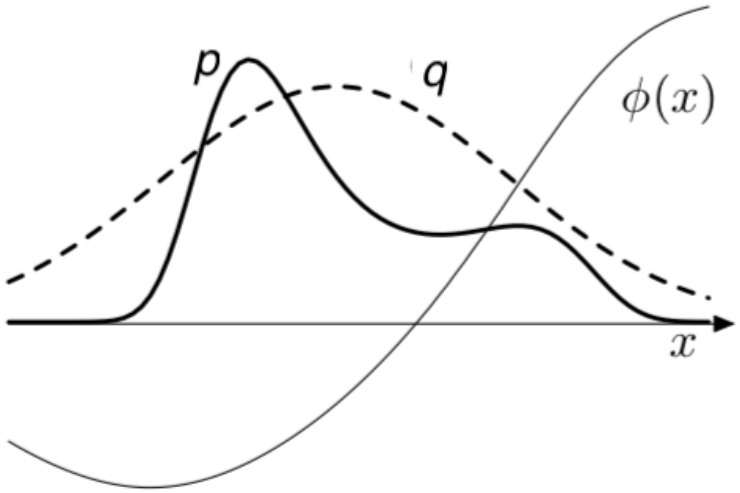
- **Problem 1** is to draw water samples at random such that each sample is equally likely to come from any point within the lake.
- **Problem 2** is to find the average plankton concentration.

A slice through a lake that includes some canyons.

- To correctly estimate Φ, 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. It is also 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 Φ by

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

  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

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

- 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 $\tilde{q}$.

$$(*) = \frac{Z_q}{Z_p}\frac{1}{R}\sum_{r=1}^{R}\phi(x^{(r)})\cdot\frac{\tilde{p}(x^{(r)})}{\tilde{q}(x^{(r)})} = \frac{Z_q}{Z_p}\frac{1}{R}\sum_{r=1}^{R}\phi(x^{(r)})\cdot\tilde{w}_r$$

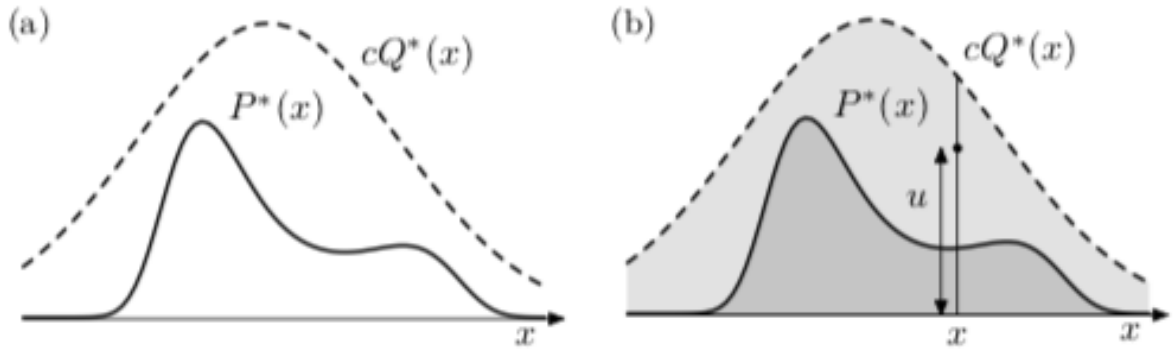
$$\approx \frac{\frac{1}{R}\sum_{r=1}^{R}\phi(x^{(r)})\cdot\tilde{w}_r}{\frac{1}{R}\sum_{r=1}^{R}\tilde{w}_r} = \sum_{r=1}^{R}\phi(x^{(r)})\cdot w_r = \hat{\Phi}_{iw}$$

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

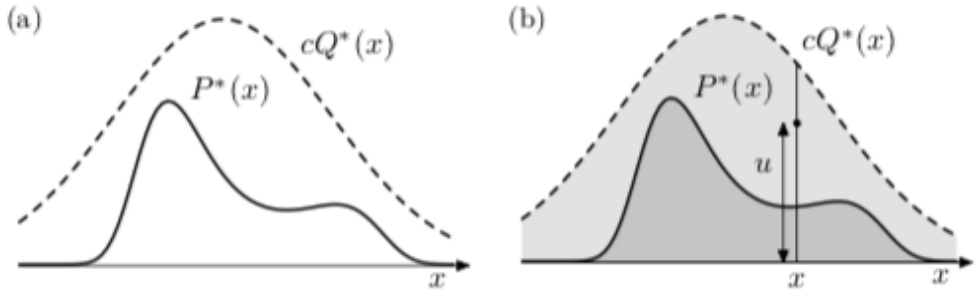
Sampling tool: Rejection sampling

- 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

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

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

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, message passing!