Lecture : Inference in Graphical Models

Riashat Islam

Reasoning and Learning Lab
McGill University

11th October 2017

Exact Inference

Variable Elimination and Belief Propagation

Inference

Inference corresponds to using the distribution to answers questions about the environment.

---

examples

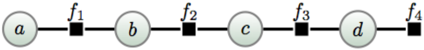
- What is the probability $p(x = 4|y = 1, z = 2)$?
- What is the most likely joint state of the distribution $p(x, y)$?
- What is the entropy of the distribution $p(x, y, z)$?
- What is the probability that this example is in class 1?
- What is the probability the stock market will do down tomorrow?

Computational Efficiency

- Inference can be computationally very expensive and we wish to characterise situations in which inferences can be computed efficiently.
- For singly-connected graphical models, and certain inference questions, there exist efficient algorithms based on the concept of message passing.
- In general, the case of multiply-connected models is computationally inefficient.

Sum-Product Algorithm

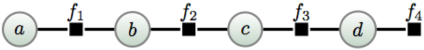
$p(a,b,c,d) \propto f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$ a, b, c, d binary variables



$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&\propto \sum_{b,c,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \Rightarrow 2^3 \text{ sums}
\end{aligned}
$$

Sum-Product Algorithm

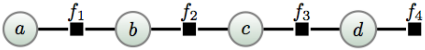
$p(a,b,c,d) \propto f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$ $\quad a,b,c,d$ binary variables



$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&\propto \sum_{b,c,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \Rightarrow 2^3 \text{ sums} \\
&= \sum_{b} f_1(a,b) \sum_{c} f_2(b,c) \sum_{d} f_3(c,d)\, f_4(d) \Rightarrow 2 \times 3 \text{ sums}
\end{aligned}
$$

Sum-Product Algorithm

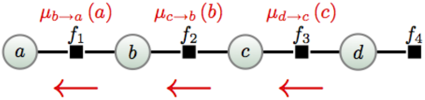
$p(a,b,c,d) \propto f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$ $\quad a,b,c,d$ binary variables



$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&\propto \sum_{b,c,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \\
&= \underbrace{\sum_b f_1(a,b) \underbrace{\sum_c f_2(b,c) \underbrace{\sum_d f_3(c,d)\, f_4(d)}_{\mu_{d\to c}(c)}}_{\mu_{c\to b}(b)}}_{\mu_{b\to a}(a)}
\end{aligned}
$$

Sum-Product Algorithm

$p(a,b,c,d) \propto f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$ $\quad a,b,c,d$ binary variables

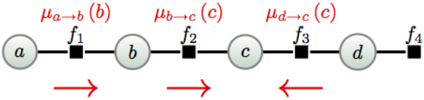


Passing variable-to-variable messages from d up to a

$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a,b,c,d) \\
&\propto \sum_{b,c,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d) \\
&= \underbrace{\sum_b f_1(a,b) \underbrace{\sum_c f_2(b,c) \underbrace{\sum_d f_3(c,d)\, f_4(d)}_{\mu_{d\to c}(c)}}_{\mu_{c\to b}(b)}}_{\mu_{b\to a}(a)}
\end{aligned}
$$

Sum-Product Algorithm

For $p(c)$ need to send messages in both directions

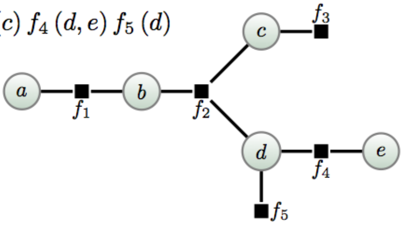


$$p(c) \propto \sum_{a,b,d} f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(d)$$

$$= \underbrace{\sum_b \underbrace{\sum_a f_1(a,b)}_{\mu_{a\to b}(b)} f_2(b,c)}_{\mu_{b\to c}(c)} \underbrace{\sum_d f_3(c,d)\, f_4(d)}_{\mu_{d\to c}(c)}$$

Sum-Product Algorithm

$$p(a,b,c,d,e) \propto f_1(a,b)\, f_2(b,c,d)\, f_3(c)\, f_4(d,e)\, f_5(d)$$



Need to define factor-to-variable messages and variable-to-factor messages

$$p(a) \propto \underbrace{f_1(a,b) \underbrace{\sum_{c,d} f_2(b,c,d) \underbrace{f_3(c)}_{\mu_{c\to f_2}(c)=\mu_{f_3\to c}(c)} \underbrace{\underbrace{f_5(d)}_{\mu_{f_5\to d}(d)} \underbrace{\sum_e f_4(d,e)}_{\mu_{f_4\to d}(d)}}_{\mu_{d\to f_2}(d)}}_{\mu_{b\to f_1}(b)=\mu_{f_2\to b}(b)}}_{\mu_{f_1\to a}(a)}$$

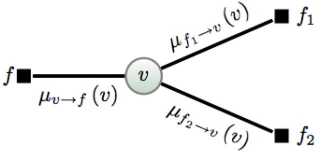
$\Rightarrow$ **Marginal inference for a singly-connected structure is easy.**

Sum-Product Algorithm for Factor Graphs

Variable to factor message

$\mu_{v \to f}(v) = \prod_{f_i \sim v \setminus f} \mu_{f_i \to v}(v)$

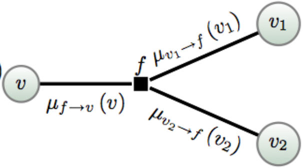
Messages from extremal variables are set to 1



Factor to variable message

$\mu_{f \to v}(v) = \sum_{\{v_i\}} f(v, \{v_i\}) \prod_{v_i \sim f \setminus v} \mu_{v_i \to f}(v_i)$

Messages from extremal factors are set to the factor

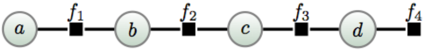


Marginal

$p(v) \propto \prod_{f_i \sim v} \mu_{f_i \to v}(v)$

Max Product Algorithm

$p(a, b, c, d) \propto f_1(a, b)\, f_2(b, c)\, f_3(c, d)\, f_4(d)$ $\quad a, b, c, d$ binary variables

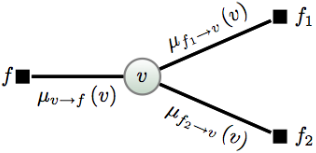


$$
\begin{aligned}
\max_{a,b,c,d} p(a, b, c, d) &= \max_{a,b,c,d} f_1(a, b)\, f_2(b, c)\, f_3(c, d)\, f_4(d) \\
&= \max_a \underbrace{\max_b f_1(a, b) \underbrace{\max_c f_2(b, c) \underbrace{\max_d f_3(c, d)\, f_4(d)}_{\mu_{d\to c}(c)}}_{\mu_{c\to b}(b)}}_{\mu_{b\to a}(a)}
\end{aligned}
$$

Max Product Algorithm

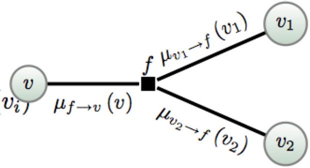
Variable to factor message

$\mu_{v\to f}(v) = \prod_{f_i \sim v \setminus f} \mu_{f_i \to v}(v)$



Factor to variable message

$\mu_{f\to v}(v) = \max_{\{v_i\}} f(v, \{v_i\}) \prod_{v_i \sim f \setminus v} \mu_{v_i \to f}(v_i)$



Marginal

$v^* = \underset{v}{\operatorname{argmax}} \prod_{f_i \sim v} \mu_{f_i \to v}(v)$

Message Passing

- Also known as **belief propagation** or **dynamic programming**
- Note that for non-branching graphs (they look like lines), only variable to variable messages are required.
- For message passing to work we need to be able to distribute the operator over the factors (which means that the operator algebra is a semiring) and that the graph is singly-connected.
- Provided the above conditions hold, marginal inference scales linearly with the number of nodes in the graph.

Approximate Inference

Sampling Methods

Inference for Graphical Models

Before we looked into **Exact Inference :**

- Can be slow in many cases!

Approximate Inference : Sampling Methods represent desired distribution with a set of samples $\rightarrow$ as more samples are used, obtain more accurate representation

Sampling

Fundamental problem we address:

- How to obtain samples from a probability distribution $p(\mathbf{z})$
- This could be a conditional distribtion $p(\mathbf{z}|\mathbf{e})$

We wish to evaluate expectations such as

$$\mathbb{E}[f] = \int f(z)p(z)dz \tag{1}$$

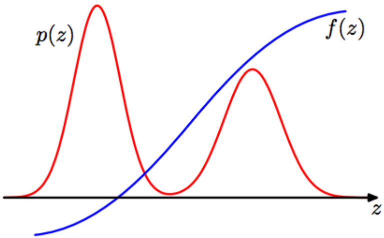
- e.g mean when $f(z) = z$

For complicated p(z), this is difficult to do exactly, so approximate as

$$\hat{f} = \frac{1}{Z}\sum_{l=1}^{L} f(z^{(l)}) \tag{2}$$

where $\{z^{(l)}|l = 1, ...L\}$ are independent samples from p($\mathbf{z}$)

Sampling



- Approximate

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

where $\{\boldsymbol{z}^{(l)} | l = 1, \ldots, L\}$ are independent samples from $p(\boldsymbol{z})$

Simple Monte Carlo

Statistical sampling can be applied to any expectation:

In general:

$$\int f(x)P(x)\,\mathrm{d}x \;\approx\; \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}), \quad x^{(s)} \sim P(x)$$

Example: making predictions

$$\begin{aligned} p(x|\mathcal{D}) &= \int P(x|\theta,\mathcal{D})P(\theta|\mathcal{D})\,\mathrm{d}\theta \\ &\approx \frac{1}{S}\sum_{s=1}^{S} P(x|\theta^{(s)},\mathcal{D}), \quad \theta^{(s)} \sim P(\theta|\mathcal{D}) \end{aligned}$$

Properties of Monte Carlo

Estimator: $\int f(x)P(x)\,\mathrm{d}x \approx \hat{f} \equiv \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}), \quad x^{(s)} \sim P(x)$

Estimator is unbiased:

$$\mathbb{E}_{P(\{x^{(s)}\})}\left[\hat{f}\right] = \frac{1}{S}\sum_{s=1}^{S}\mathbb{E}_{P(x)}[f(x)] = \mathbb{E}_{P(x)}[f(x)]$$

Variance shrinks $\propto 1/S$:

$$\mathrm{var}_{P(\{x^{(s)}\})}\left[\hat{f}\right] = \frac{1}{S^2}\sum_{s=1}^{S}\mathrm{var}_{P(x)}[f(x)] = \mathrm{var}_{P(x)}[f(x)]\,/S$$

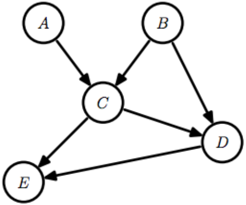
"Error bars" shrink like $\sqrt{S}$

Sampling from a Bayesian Network

Ancestral pass for directed graphical models:

— sample each top level variable from its marginal

— sample each other node from its conditional once its parents have been sampled



Sample:

$$A \sim P(A)$$
$$B \sim P(B)$$
$$C \sim P(C|A,B)$$
$$D \sim P(D|B,C)$$
$$E \sim P(D|C,D)$$

$$P(A,B,C,D,E) = P(A)\,P(B)\,P(C|A,B)\,P(D|B,C)\,P(E|C,D)$$

Sampling from Bayesian Networks

- Sampling from discrete Bayesian networks with no observations is straight-forward, using ancestral sampling
- Bayesian network specifies factorization of joint distribution

$$P(z_1, \ldots, z_n) = \prod_{i=1}^{n} P(z_i | pa(z_i))$$

- Sample in-order, sample parents before children
  - Possible because graph is a DAG
- Choose value for z_i from $p(z_i | pa(z_i))$

Ancestral Sampling

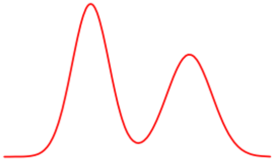
Given a DAG model

- Start by sampling from $P(x_1)$
- Then sample from $P(x_2|x_1)$
- Then sample from $P(x_3|x_2)$
- Finally sample from $P(x_4|x_3)$

$\{x_1, x_2, x_3, x_4\}$ is a sample from the joint distribution

Basic Sampling Algorithm

How to generate samples from simple non-uniform distributions assuming we can generate samples from uniform distribution.



Define: $h(y) = \int_{-\infty}^{y} p(\hat{y}) d\hat{y}$

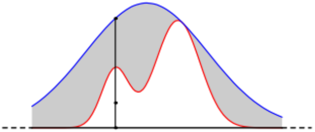
Sample: $z \sim U[0, 1]$.

Then: $y = h^{-1}(z)$ is a sample from $p(y)$.

Problem: Computing cumulative $h(y)$ is just as hard!

Rejection Sampling

Sampling from *target distribution* $p(z) = \tilde{p}(z)/\mathcal{Z}_p$ is difficult. Suppose we have an easy-to-sample *proposal distribution* $q(z)$, such that $kq(z) \geq \tilde{p}(z), \forall z$.

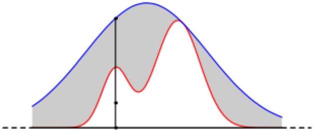


Sample z_0 from $q(z)$.
Sample u_0 from Uniform$[0, kq(z_0)]$

The pair (z_0, u_0) has uniform distribution under the curve of $kq(z)$.

If $u_0 > \tilde{p}(z_0)$, the sample is rejected.

Rejection Sampling



Probability that a sample is accepted is:

$$
\begin{aligned}
p(\text{accept}) &= \int \frac{\tilde{p}(z)}{kq(z)} q(z) dz \\
&= \frac{1}{k} \int \tilde{p}(z) dz
\end{aligned}
$$

The fraction of accepted samples depends on the ratio of the area under $\tilde{p}(z)$ and $kq(z)$.

Hard to find appropriate $q(z)$ with optimal k.

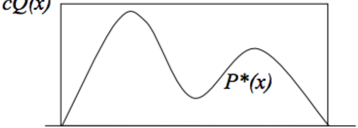
Useful technique in one or two dimensions. Typically applied as a subroutine in more advanced algorithms.

Rejection Sampling

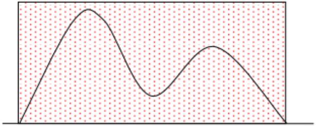
Need a proposal density Q(x) [e.g. uniform or Gaussian], and a constant c such that c(Qx) is an upper bound for P*(x)

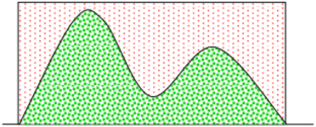
Example with Q(x) uniform

upper bound



generate uniform random samples in upper bound volume



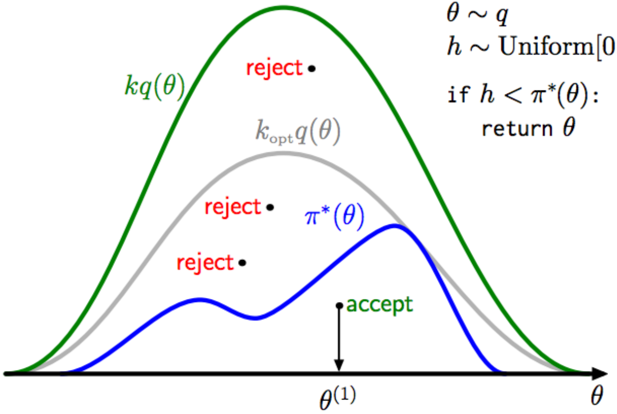


accept samples that fall below the P*(x) curve

the marginal density of the x coordinates of the points is then proportional to P*(x)

Note the relationship to Monte Carlo integration.

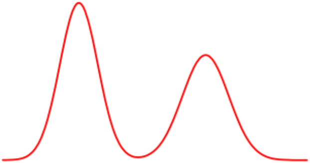
Rejection Sampling



while true:
    $\theta \sim q$
    $h \sim \text{Uniform}[0, kq(\theta)]$
    if $h < \pi^*(\theta)$:
        return θ

Importance Sampling

Suppose we have an easy-to-sample *proposal distribution* $q(z)$, such that $q(z) > 0$ if $p(z) > 0$.



$$
\begin{aligned}
\mathbb{E}[f] &= \int f(z)p(z)dz \\
&= \int f(z)\frac{p(z)}{q(z)}q(z)dz \\
&\approx \frac{1}{N}\sum_n \frac{p(z^n)}{q(z^n)}f(z^n), \quad z^n \sim q(z)
\end{aligned}
$$

The quantities $w^n = {}^{p(z^n)}/_{q(z^n)}$ are known as **importance weights**.
Unlike rejection sampling, all samples are retained.
But wait: we cannot compute $p(z)$, only $\tilde{p}(z)$.

Importance Sampling

Let our proposal be of the form $q(z) = \tilde{q}(z)/\mathcal{Z}_q$:

$$\begin{aligned}
\mathbb{E}[f] &= \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz = \frac{\mathcal{Z}_q}{\mathcal{Z}_p}\int f(z)\frac{\tilde{p}(z)}{\tilde{q}(z)}q(z)dz \\
&\approx \frac{\mathcal{Z}_q}{\mathcal{Z}_p}\frac{1}{N}\sum_n \frac{\tilde{p}(z^n)}{\tilde{q}(z^n)}f(z^n) = \frac{\mathcal{Z}_q}{\mathcal{Z}_p}\frac{1}{N}\sum_n w^n f(z^n), \qquad z^n \sim q(z)
\end{aligned}$$

But we can use the same importance weights to approximate $\frac{\mathcal{Z}_p}{\mathcal{Z}_q}$:

$$\frac{\mathcal{Z}_p}{\mathcal{Z}_q} = \frac{1}{Z_q}\int \tilde{p}(z)dz = \int \frac{\tilde{p}(z)}{\tilde{q}(z)}q(z)dz \approx \frac{1}{N}\sum_n \frac{\tilde{p}(z^n)}{\tilde{q}(z^n)} = \frac{1}{N}\sum_n w^n$$

Hence:

$$\mathbb{E}[f] \approx \frac{1}{N}\sum_n \frac{w^n}{\sum_n w^n} f(z^n) \qquad \text{Consistent but biased.}$$

Probelms

If our proposal distribution $q(z)$ poorly matches our target distribution $p(z)$ then:

- Rejection Sampling: almost always rejects
- Importance Sampling: has large, possibly infinite, variance (unreliable estimator).

For high-dimensional problems, finding good proposal distributions is very hard. What can we do?

Markov Chain Monte Carlo.

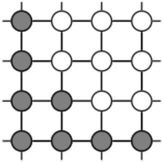
Summary so far

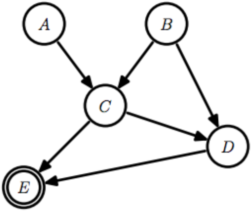
- Sums and integrals, often expectations, occur frequently in statistics
- **Monte Carlo** approximates expectations with a sample average
- **Rejection sampling** draws samples from complex distributions
- **Importance sampling** applies Monte Carlo to 'any' sum/integral

Application to Large Problems

We often can't decompose $P(X)$ into low-dimensional conditionals

Undirected graphical models: $P(x) = \frac{1}{\mathcal{Z}} \prod_i f_i(x)$





Posterior of a directed graphical model

$$P(A, B, C, D \,|\, E) = \frac{P(A, B, C, D, E)}{P(E)}$$

We often don't know $\mathcal{Z}$ or $P(E)$

Gibbs Sampling

For large graphical models, given a multivariate distribution, it is simpler to sample from a conditional distribution than to marginalize by integrating over a joint distribution

- We want samples approximate the joint distribution of all variables
- The marginal distribution of any subset of variables can be approximated by simply considering the samples for that subset of variables (Markov Blanket in Bayes Nets!)
- The expected value of any variable can be approximated by averaging over all the samples

Gibbs Sampling

A method with no rejections:

- Initialize $\mathbf{x}$ to some value
- Pick each variable in turn or randomly and resample $P(x_i|\mathbf{x}_{j\neq i})$

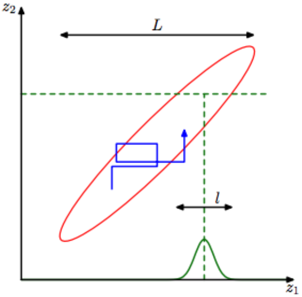
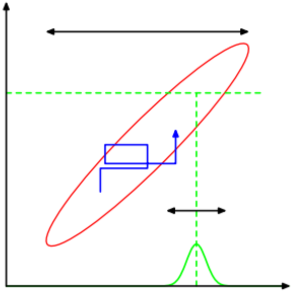


Figure from PRML, Bishop (2006)

Gibbs Sampler

Consider sampling from $p(z_1, ..., z_N)$.



Initialize z_i, $i = 1, ..., N$

For t=1,...,T

Sample $z_1^{t+1} \sim p(z_1 | z_2^t, ..., z_N^t)$

Sample $z_2^{t+1} \sim p(z_2 | z_1^{t+1}, x_3^t, ..., z_N^t)$

...

Sample $z_N^{t+1} \sim p(z_N | z_1^{t+1}, ..., z_{N-1}^{t+1})$

Gibbs sampler is a particular instance of M-H algorithm with proposals $p(z_n | \mathbf{z}_{i \neq n})$ $\rightarrow$ accept with probability 1. Apply a series (component-wise) of these operators.

Gibbs Sampling for Bayes Nets

1. Initialization
   - Set evidence variables E, to the observed values e
   - Set all other variables to random values (e.g. by forward sampling)

   This gives us a sample $x_1, \ldots, x_n$.
2. Repeat (as much as wanted)
   - Pick a non-evidence variable X_i uniformly randomly)
   - Sample x_i' from $P(X_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$.
   - Keep all other values: $x_j' = x_j, \forall j \neq i$
   - The new sample is $x_1', \ldots, x_n'$
3. Alternatively, you can march through the variables in some predefined order

Why Gibbs works in Bayes Nets

- The key step is sampling according to $P(X_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. How do we compute this?
- In Bayes nets, we know that a variable is conditionally independent of all others given its Markov blanket (parents, children, spouses)

$$P(X_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(X_i|\text{MarkovBlanket}(X_i))$$

- So we need to sample from $P(X_i|\text{MarkovBlanket}(X_i))$
- Let $Y_j, j = 1, \ldots, k$ be the children of X_i. It is easy to show that:

$$\begin{aligned} P(X_i = x_i|\text{MarkovBlanket}(X_i)) \quad &\propto \quad P(X_i = x_i|\text{Parents}(X_i)) \cdot \\ &\cdot \quad \prod_{j=1}^{k} P(Y_j = y_j|\text{Parents}(Y_j)) \end{aligned}$$

Summary

Ways of doing inference in graphical models...

Exact Inference

- Message Passing algorithms

Approximate Inference (Sampling Methods)

- Monte-Carlo Sampling
- Importance Sampling
- Gibbs Sampling in Bayesian Networks

Note : We will cover Sampling Methods in more details when we talk about Approximate Inference and Variational Methods (later...)