Lecture 8: PGM — Inference

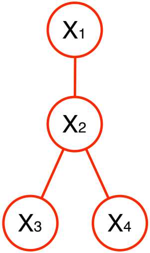
Qinfeng (Javen) Shi

15 September 2014

Intro. to Stats. Machine Learning
COMP SCI 4401/7401

Table of Contents I

Marginal and MAP



Marginal inference: $P(x_i) = \sum_{x_j : j \neq i} P(x_1, x_2, x_3, x_4)$

MAP inference: $(x_1^*, x_2^*, x_3^*, x_4^*) = \operatorname{argmax}_{x_1, x_2, x_3, x_4} P(x_1, x_2, x_3, x_4)$

In general, $x_i^* \neq \operatorname{argmax}_{x_i} P(x_i)$

Marginals

When do we need marginals? Marginals are used to compute

- normalisation constant
  $Z = \sum_{x_i} q(x_i) = \sum_{x_j} q(x_j) \quad \forall i, j = 1, \ldots.$
  log loss in CRFs is $-\log P(x_1, \ldots, x_n) = \log(Z) + \ldots$
- expectations like $\mathbb{E}_{P(x_i)}[\phi(x_i)]$ and $\mathbb{E}_{P(x_i,x_j)}[\phi(x_i, x_j)]$, where
  $\psi(x_i) = \langle \phi(x_i), w \rangle$ and $\psi(x_i, x_j) = \langle \phi(x_i, x_j), w \rangle$
  Gradient of CRFs risk contains above expectations.

MAP

When do we need MAP?

- find the most likely configuration for $(x_i)_{i \in \mathcal{V}}$ in testing.
- find the most violated constraint generated by $(x_i^\dagger)_{i \in \mathcal{V}}$ in training (*i.e.* learning), *e.g.* by cutting plane method (used in SVM-Struct) or by Bundle method for Risk Minimisation (Teo JMLR2010).

Variable elimination

$$
\begin{aligned}
&\max_{x_1,x_2,x_3,x_4} P(x_1,x_2,x_3,x_4)\\
&= \max_{x_1,x_2,x_3,x_4} \psi(x_1,x_2)\psi(x_2,x_3)\psi(x_2,x_4)\psi(x_1)\psi(x_2)\psi(x_3)\psi(x_4)\\
&= \max_{x_1,x_2}\left[\ldots \max_{x_3}\Big(\psi(x_2,x_3)\psi(x_3)\Big)\max_{x_4}\Big(\psi(x_2,x_4)\psi(x_4)\Big)\right]\\
&= \max_{x_1}\left[\psi(x_1)\max_{x_2}\Big(\psi(x_2)\psi(x_1,x_2)m_{3\to 2}(x_2)m_{4\to 2}(x_2)\Big)\right]\\
&= \max_{x_1}\Big(\psi(x_1)m_{2\to 1}(x_1)\Big) \Rightarrow x_1^* = \operatorname*{argmax}_{x_1}\Big(\psi(x_1)m_{2\to 1}(x_1)\Big)
\end{aligned}
$$

Max-product

Variable elimination for MAP $\Rightarrow$ Max-product:

$$x_i^* = \operatorname*{argmax}_{x_i} \left(\psi(x_i) \prod_{j \in Ne(i)} m_{j \to i}(x_i) \right)$$

$$m_{j \to i}(x_i) = \max_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in Ne(j) \setminus \{i\}} m_{k \to j}(x_j) \right)$$

$Ne(i)$: neighbouring nodes of i (*i.e.* nodes that connect with i).

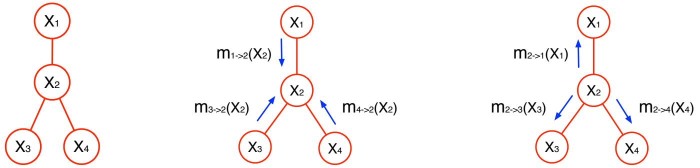
$Ne(j) \setminus \{i\} = \emptyset$ if j has only one edge connecting it. *e.g.* x_1, x_3, x_4.
For such node j,

$$m_{j \to i}(x_i) = \max_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \right)$$

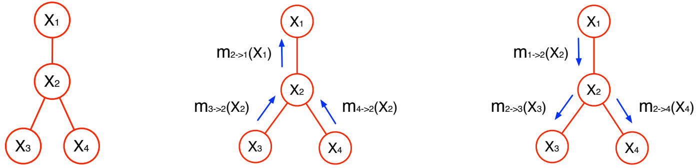
Easier computation!

Max-product

Order matters: message $m_{2\to3}(x_3)$ requires $m_{1\to2}(x_2)$ and $m_{4\to2}(x_2)$.



Alternatively, leaves to root, and root to leaves.



Sum-product

Variable elimination for marginal $\Rightarrow$ Sum-product:

$$P(x_i) = \frac{1}{Z}\left(\psi(x_i) \prod_{j \in Ne(i)} m_{j\to i}(x_i)\right)$$

$$m_{j\to i}(x_i) = \sum_{x_j}\left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in Ne(j)\backslash\{i\}} m_{k\to j}(x_j)\right)$$

Extension

To avoid over/under flow, often operate in the log space.

Max/sum-product is also known as Message Passing and Belief Propagation (BP).

In graphs with loops, running BP for several iterations is known as Loopy BP (neither convergence nor optimal guarantee in general).

Extend to Junction Tree Algorithm (exact, but expensive) and Clusters-based BP.

LP Relaxations

Assume pairwise MRFs with graph $G(\mathcal{V}, \mathcal{E})$

$$P(\mathbf{X} \mid \mathbf{Y}) = \frac{1}{Z} \prod_{(i,j) \in \mathcal{E}} \psi_{i,j}(x_i, x_j) \prod_{i \in \mathcal{V}} \psi_i(x_i)$$

$$= \frac{1}{Z} \exp\left(\sum_{(i,j) \in \mathcal{E}} \theta_{i,j}(x_i, x_j) + \sum_{i \in \mathcal{V}} \theta_i(x_i) \right)$$

$$\text{MAP} \quad \mathbf{X}^* = \operatorname*{argmax}_{\mathbf{X}} \prod_{(i,j) \in \mathcal{E}} \psi_{i,j}(x_i, x_j) \prod_{i \in \mathcal{V}} \psi_i(x_i)$$

$$= \operatorname*{argmax}_{\mathbf{X}} \sum_{(i,j) \in \mathcal{E}} \theta_{i,j}(x_i, x_j) + \sum_{i \in \mathcal{V}} \theta_i(x_i)$$

LP Relaxations

$$\operatorname*{argmax}_{\mathbf{x}} \sum_{(i,j)\in\mathcal{E}} \theta_{i,j}(x_i, x_j) + \sum_{i\in\mathcal{V}} \theta_i(x_i)$$

$\Leftrightarrow$ the following Integer Program:

$$\operatorname*{argmax}_{\{q\}} \sum_{(i,j)\in\mathcal{E}} \sum_{x_i,x_j} q_{i,j}(x_i, x_j)\theta_{i,j}(x_i, x_j) + \sum_{i\in\mathcal{V}} \sum_{x_i} q_i(x_i)\theta_i(x_i)$$

$$\text{s.t.} \quad q_{i,j}(x_i, x_j) \in \{0, 1\}, \sum_{x_i,x_j} q_{i,j}(x_i, x_j) = 1, \sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j).$$

Relax to Linear Program:

$$\operatorname*{argmax}_{\{q\}} \sum_{(i,j)\in\mathcal{E}} \sum_{x_i,x_j} q_{i,j}(x_i, x_j)\theta_{i,j}(x_i, x_j) + \sum_{i\in\mathcal{V}} \sum_{x_i} q_i(x_i)\theta_i(x_i)$$

$$\text{s.t.} \quad q_{i,j}(x_i, x_j) \in [0, 1], \sum_{x_i,x_j} q_{i,j}(x_i, x_j) = 1, \sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j).$$

QP Relaxations

Imposing $q_{i,j}(x_i, x_j) = q_i(x_i)q_j(x_j)$ yields QP:

$$\operatorname*{argmax}_{\{q\}} \sum_{(i,j)\in\mathcal{E}} \sum_{x_i,x_j} q_i(x_i)q_j(x_j)\theta_{i,j}(x_i, x_j) + \sum_{i\in\mathcal{V}} \sum_{x_i} q_i(x_i)\theta_i(x_i)$$

$$\text{s.t.} \quad q_i(x_i) \in [0, 1], \sum_{x_i} q_i(x_i) = 1.$$

Rewrite objective function as

$$\mathbf{y}^T \Theta \mathbf{y} + \mathbf{y}^T \boldsymbol{\theta},$$

where $\mathbf{y} = [q_i(x_i)]_{i\in V, x_i\in Val(X)} \in \mathbb{R}^N$, $\boldsymbol{\theta} = [\theta_i(x_i)]_{i,x_i} \in \mathbb{R}^N$, and $\Theta = [\theta_{i,j}(x_i, x_j)]_{i,x_i,j,x_j} \in \mathbb{R}^{N\times N}$.

QP Relaxations

Problem: $\mathbf{y}^T \Theta \mathbf{y} + \mathbf{y}^T \boldsymbol{\theta}$ may not be concave *i.e.* Θ may not be negative semidefinite (NSD).

[1] In general for $y_i \in [0, 1]$, $y_i - y_i^2 \geq 0$, thus $g(\mathbf{d}, \mathbf{y}) \geq 0$

QP Relaxations

Problem: $\mathbf{y}^T \Theta \mathbf{y} + \mathbf{y}^T \boldsymbol{\theta}$ may not be concave *i.e.* Θ may not be negative semidefinite (NSD).

Solution: To find a $\mathbf{d} \in \mathbb{R}^N$, such that $(\Theta - \mathbf{diag}(\mathbf{d}))$ is NSD. Given $\mathbf{d}$, we have
$\mathbf{y}^T \Theta \mathbf{y} + \mathbf{y}^T \boldsymbol{\theta} = \mathbf{y}^T(\Theta - \mathbf{diag}(\mathbf{d}))\mathbf{y} + (\mathbf{y} + \mathbf{d})^T \boldsymbol{\theta} + g(\mathbf{d}, \mathbf{y})$,
where the gap $g(\mathbf{d}, \mathbf{y}) = \mathbf{y}^T \mathbf{diag}(\mathbf{d})\mathbf{y} - \mathbf{d}^T \boldsymbol{\theta} = \sum_{i=1}^N d_i(y_i - y_i^2)$.
Note for $y_i \in \{0, 1\}$, $y_i - y_i^2 = 0$, thus $g(\mathbf{d}, \mathbf{y}) = 0$ [1]. So we know

$$\mathbf{y}^T \Theta \mathbf{y} + \mathbf{y}^T \boldsymbol{\theta} \approx \underbrace{\mathbf{y}^T(\Theta - \mathbf{diag}(\mathbf{d}))\mathbf{y} + (\mathbf{y} + \mathbf{d})^T \boldsymbol{\theta}}_{\textit{Concave}}$$

$$\operatorname*{argmax}_{\mathbf{y}} \mathbf{y}^T(\Theta - \mathbf{diag}(\mathbf{d}))\mathbf{y} + (\mathbf{y} + \mathbf{d})^T \boldsymbol{\theta}$$

[1] In general for $y_i \in [0, 1]$, $y_i - y_i^2 \geq 0$, thus $g(\mathbf{d}, \mathbf{y}) \geq 0$

Break

Take a break ...

Understanding samples

In fact, there is no way to check 'a sample' is from a distribution or not — two totally different distributions can generate the same sample. For example, $uniform[0, 1]$ and gaussian $N(0, 1)$ can both generate a sample with value 0. Looking at a sample with value = 0 alone, how do you know its distribution for sure? What we really check (and know for sure) is the way that the samples were generated. When we say a procedure generates a sample from a distribution P, what we really mean is that keeping sampling this way (by the procedure), the normalised histogram H^n with n samples is going to converge to the distribution P. That is $H^n \to P$ as $n \to \infty$. If we don't know the way that the samples were generated, we never know what's the distribution for sure — we can only guess (e.g. using statistical tests) based on a number of available samples.

Overview

- Monte Carlo
- Importance sampling
- Acceptance-rejection sampling
- $\cdots$

Monte Carlo

Monte Carlo methods are a class of computational algorithms that rely on repeated random sampling to compute their results.

repeat
  draw sample(s)
  compute result according to the samples
until sampled enough (or the result is stable)

Monte Carlo

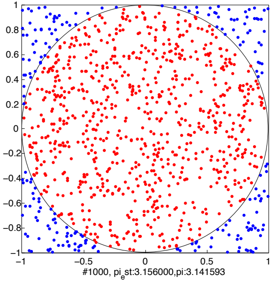
To estimate π (area of a circle with radius r is $S_c = \pi r^2$).
Idea:

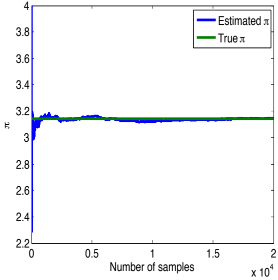
- draw a circle ($r = 1$) and a rectangle $(2r \times 2r)$ enclosing the circle. We know the area of the rectangle is $S_{rec} = (2r)^2$. If we can estimate the area of the circle, then we can estimate π by $\pi = S_c / r^2$.
- Draw a sample point from the rectangle area uniformly. The chance of it being within the circle is S_c / S_{rec}. So if we throw enough points, we have $N_{within} / N_{total} \approx S_c / S_{rec}$. Thus $S_c \approx S_{rec} N_{within} / N_{total}$. Theorefore,

$$\pi \approx \frac{S_{rec} N_{within} / N_{total}}{r^2} = 4 N_{within} / N_{total}$$

See a matlab demo.

Monte Carlo


#1000, pi_est:3.156000,pi:3.141593


Estimated π
True π
π
Number of samples
x 10^4

Monte Carlo

To estimate an expectation:
Generate samples $x_i \sim q(X), i = 1, \ldots, N$.

$$\begin{aligned}\mathbb{E}_{X \sim q(X)}[f(X)] &\approx \hat{\mathbb{E}}_{X \sim q(X)}[f(X)] \\ &= \frac{1}{N}\sum_{i=1}^{N} f(x_i),\end{aligned}$$

Importance sampling

To compute $\mathbb{E}_{X\sim p(X)}[f(X)]$.
Assume $p(x)$ (target distribution) is hard to sample from directly, and $q(x)$ (proposal distribution) is easy to sample from and $q(x) > 0$ when $p(x) > 0$.

$$\begin{aligned}\mathbb{E}_{X\sim p(X)}[f(X)] &= \int_x p(x)f(x)dx \\ &= \int_x q(x)\frac{p(x)}{q(x)}f(x)dx \\ &= \mathbb{E}_{X\sim q(X)}[\frac{p(X)}{q(X)}f(X)].\end{aligned}$$

$$\hat{\mathbb{E}}_{X\sim p(X)}[f(X)] = \hat{\mathbb{E}}_{X\sim q(X)}[\frac{p(X)}{q(X)}f(X)],$$

$$\text{where } \hat{\mathbb{E}}_{X\sim q(X)}[f(X)] = \frac{1}{N}\sum_{i=1}^{N} f(x_i), x_i \sim q(X), i = 1, \ldots, N.$$

Acceptance-rejection sampling

Target: to sample X from $p(x)$.
Given: $q(x)$ easy to sample from.
Find a constant M such that $M \cdot q(x) \geq p(x), \ \forall \ x$.
repeat
  step 1: sample $Y \sim q(y)$
  step 2: sample $U \sim Uniform[0, 1]$
  if $U \leq \frac{p(y)}{M \cdot q(y)}$ **then**
    then $X = Y$;
  else
    reject and go to step 1.
  end if
until sampled enough

Acceptance-rejection sampling

Proof:

$$\because\ Pr(accept|X=x) = \frac{p(x)}{M\cdot q(x)} \quad \text{and} \quad Pr(X=x) = q(x)$$

$$\therefore\ Pr(accept) = \int_x Pr(accept|X=x)\cdot Pr(X=x)dx$$

$$= \int_x \frac{p(x)}{M\cdot q(x)}\cdot q(x)dx = \frac{1}{M} \quad (\text{ thus don't want } M \text{ big})$$

$$\therefore\ Pr(X|accept) = \frac{Pr(accept|X)\cdot P(X)}{Pr(accept)}$$

$$= \frac{\frac{p(x)}{M\cdot q(x)}\cdot q(x)}{\frac{1}{M}} = p(x).$$

Understanding AR sampling (1)

I guess the most confusing part, is why M comes in. So let's look at the case without M first.

Denote the histogram formed by n samples from $q(x)$ as H_q^n, the histogram formed by n samples from $p(x)$ as H_p^n, the histogram formed by n accepted samples from AR sampling procedure as H^n. For a sample $x \sim q(x)$, if $p(x) < q(x)$, it suggests if you accept all the x and keep sampling this way, the histogram you will get is H_q^n. But what you really want to get, is a way that the resulting histogram H becomes H_p^n. Rejecting some portion of x can make the histogram H has the same shape as H_p at point x. In other words, the histogram H has more counts at point x than H_p, so we remove some counts to make $H(x) = H_p(x)$. (Take a moment to think this through).

Understanding AR sampling (2)

What if for a sample $x \sim q(x), p(x) > q(x)$? The histogram H_q^n already has less counts than H_p^n at x. What do we do? Well, we can sample $M \times n$ points from $q(x)$ to build H_q^{Mn} first. Now H_q^{Mn} should have more counts than H_p^n at x (because we choose a M such that $p(x) < Mq(x)$ for all x. If not, choose a larger M). Visually, H_q^{Mn} encloses H_p^n. At point x, we only want to keep $H_p^n(x)$ many samples from totally $H_q^{Mn}(x)$ many. This is how uniform sampling and M came in. We sample $u \sim Uniform[0, Mq(x)]$, accept x when $u < p(x)$ (equivalent to sample $u \sim Uniform[0, 1]$, accept x when $u < p(x)/Mq(x)$). As a result, after Mn samples, we will get a H close to H_p^n. Moreover,

$$\lim_{n\to\infty} H^n = \lim_{n\to\infty} H_p^n = p.$$

Understanding AR sampling (3)

Here we can choose any M such that $p(x) < Mq(x)$ for all x. The bigger M is, the more samples (Mn samples) you need to approximate H_p^n. That's why in practice, people want to use the smallest M (such that $p(x) < Mq(x)$ for all x) to reduce the number of rejected samples.

Sampling in PGM inference

Overview:

- Forward sampling
- Likelihood weighting sampling
- Importance sampling inference
- $\cdots$

Forward sampling

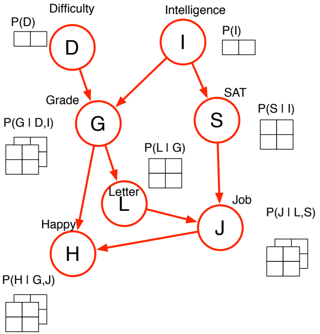
Given an ordering of subsets of random variables $\{X^i\}_{i=1}^n$ (knowing parents to generate children).

for $i = 1$ **to** n **do**

  $\mathbf{u}^i \leftarrow Pa(\mathbf{x}^{i-1})$

  sample $\mathbf{x}^i$ from $P(X^i | \mathbf{u}^i)$

end for



Forward sampling

Assume $\{\mathbf{x}_i\}_{i=1}^M$ are M samples from $P(X)$, we can approximately compute

- expectation:

$$\mathbb{E}_{X \sim P(X)}[f(X)] \approx \frac{1}{M} \sum_{i=1}^{M} f(\mathbf{x}_i)$$

- MAP solution: $\operatorname{argmax}_{\mathbf{x}} P(\mathbf{x}) \approx \operatorname{argmax}_{\mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^M} P(\mathbf{x})$
- marginal: $P(\mathbf{x}) \approx N_{X=\mathbf{x}} / N_{total}$
- sample from $P(X|\,\mathbf{e})$ when evidences $\mathbf{e}$:
  sample from $P(X)$ first, and reject $\mathbf{x}$ when it does not agree on $\mathbf{e}$.

Forward sampling

Problems?

Forward sampling

Problem: Rejection step in estimating $P(X|\mathbf{e})$ wastes too many samples when $P(\mathbf{e})$ is small. In real applications, $P(\mathbf{e})$ is almost always very small.

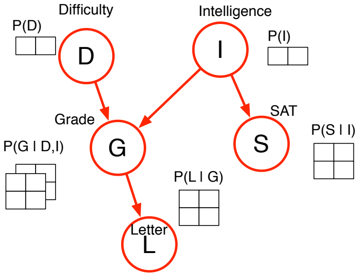
Question: how do we avoid rejecting samples?

Forward sampling

How about setting the observed random variables to the observed values, and then doing forward sampling on the rest?

Forward sampling

Let's see if it works.
To sample from $P(D, I, G, L|S = 0)$ from a simplified PGM.



Fixing $S = 0$, and then sample D, I, G, L.
Does this give the same result comparing to forward sampling with rejection?

Forward sampling

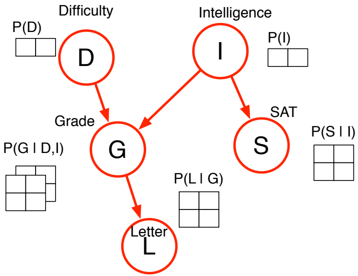
No! It doesn't.
The samples are not from $P(D, I, G, L|S = 0)$ at all!
Fixing this lead to Likelihood weighting sampling.

Likelihood weighting sampling

Input: $\{Z^i = \mathbf{z}^i\}_i$ are observed.
Step 1: set $\{Z^i\}_i$ to the observed values.
Step 2: forward sampling the unobserved variables.
Step 3: weight the sample by $\prod_i P(\mathbf{z}^i | Pa(\mathbf{z}^i))$

Likelihood weighting sampling inference

To sample from $P(D, I, G, L|S = 0)$ from the following PGM.

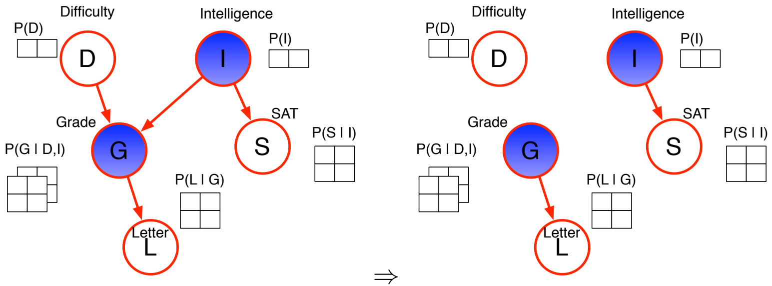


Fix $S = 0$, and forward sample D, I, G, L. Then weight the sample by $P(D, I, G, L|S = 0)$.
Does this give the same result comparing to forward sampling with rejection?

Likelihood weighting sampling

$$
\begin{aligned}
&\mathbb{E}_{X \sim P(D,I,G,L|S=0)}[f(D, I, G, L, 0)] \\
&\approx \frac{1}{N} \sum_{j=1}^{N} [f(d_j, i_j, g_j, l_j, 0) \cdot P(d_j, i_j, g_j, l_j | S = 0)]
\end{aligned}
$$

Importance sampling inference



(a) $\mathcal{G}_1$ with $p(X)$ (b) $\mathcal{G}_2$ with $q(X)$

Mutilate

Sample $\{\mathbf{x}_i\}_{i=1}^N$ from $q(X)$.

$$\hat{\mathbb{E}}_{X\sim p(X)}[f(X)] = \frac{1}{N}\sum_{i=1}^{N}\frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)}f(\mathbf{x}_i).$$

MAP Inference Revisit

- Primal
  - Variable Elimination
  - Message Passing
    - Max-product, (Loopy) BP)
    - Junction Tree Algorithm and Clusters-based BP
  - Optimisation Approaches
    - Linear Programming (LP) Relaxations
    - Quadratic programming (QP).
    - Semidefinite programming (SDP), Second-Order Cone Programming (SOCP)
    - ...
  - Sampling Approaches
  - Special potentials (Graph Cut)
  - ...
- Dual
  - GMPLP, Dual decomposition
  - ...

Marginal Inference Revisit

Many MAP inference methods can be converted to marginal ones (vice versa).

max-product $\rightarrow$ sum-product.
LP based dual methods $\rightarrow$ marginal (via $+$ entropy term (kikuchi approximation)).

That's all

Thanks!