10-708: Probabilistic Graphical Models, Spring 2020

1: Monte Carlo Methods

Lecturer: Eric P. Xing *Scribe: Aditi Chaudhary, Neha Cheemalavagu, Ke Ni*

1 Approach to Inference

In the previous class, we discussed methods for graph inference. For graphs of manageable size, exact inference algorithms such as elimination algorithm, message passing can be used. However, for complex graphs we turn to approximate inference method such as Mean Field Approximation (MFE) and Loopy Belief Propagation (BP). Both methods pose the inference problem as optimization. The key idea behind MFE is that it is a product of singleton marginals. In Loopy BP, message passing is applied over the any graph, even though it might not be converge to the exact solution. Today we are going to talk about another inference method for arbitrary distribution.

Closed-form representation For a given closed-form distribution say Gaussian, inference is straightforward as we can simply integrate the probability distribution.

Sample-based representation For an arbitrary distribution, we can draw samples $X_n \sim P((X)$ where $n = 1, N$ and use the following approximation $E_p f(x) = \sum_i \frac{1}{N} f(x_i)$ to get the sample-based representation. The key idea is that instead of manipulating the $P(X)$, we instead base it solely on the samples from the distribution.

Naive Sampling We first consider the naive sampling, where we sample according to the probabilities specified by the Bayesian Network. We traverse the graph in the topological order and draw samples for each random variable based on its local marginal distribution. Inference for a given query is then done on the counting the samples drawn which satisfy the query. The major drawback is that for large models, it might be difficult to get good estimates for rare events as we might not have enough samples. So, we might not have a good estimate of the approximate if the sample size is too small.

Monte Carlo Methods For distributions which have irregular or arbitrary distribution, it might be difficult to draw samples using the naive method. This is where Monte Carlo methods come in the picture. The key idea is that how to make use of the examples, because not all examples are useful for a given inference query. The main challenges that these methods attempt to answer are:

1. How to draw samples for irregular or arbitrary distributions?
2. How to know which samples are useful?
3. How do we know how much to sample?

Rejection Sampling Suppose the true distribution $\Pi(X)$ is difficult to sample form. Instead, we consider the un-normalized distribution $\Pi'(X)$ to evaluate since that doesn't involve computing the partition function. We instead sample from a simpler distribution $Q(x)$. The procedure goes as follows:

1. Draw x_0 from $Q(x)$
2. Accept x_0 with probability $p = \frac{\Pi'(x_0}{kQx}$ such that $0 <= p <= 1$

Figure 1 explains this as selecting a sample which falls under the white region of the un-normalized distribution and rejecting the sample if it falls under the grey region.

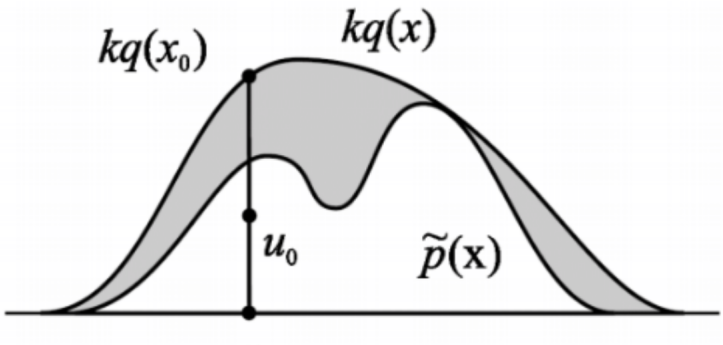

Figure 1: Figure of Rejection Sampling explained pictorically.

We check the correctness of the approach. Based on the algorithm we have: $P(x_0) = \frac{\frac{Q(x_0)*\Pi'(x_0)}{kQ(x_0)}}{\int Q(x)*\Pi'(x)kQ(x)dx}$. The $Q(x)$ terms being common across numerator and denominator get cancelled and give us: $P(x_0) = \frac{\Pi'(x_0)}{\int \Pi'(x)dx} = \Pi(x)$

One major pitfall with this method is that if the $Q(x)$ distribution is not chosen well, we end up with lot of rejected samples, as many as the shaded area in the Figure 1. In the class, we discussed a thought example, where we have $Q \sim N(\mu, \sigma_p^{\frac{2}{d}})$ and $P \sim N(\mu, \sigma_p^{\frac{2}{d}})$ where d is the dimention. However, even for a very similar distribution, in high-dimentional space the optimal acceptance rate is roughly only $\frac{1}{20000}$. We can do adaptive rejection samping which covers the envelope with piecewise functions.

Importance Sampling Suppose sampling from the target distribution $P(x)$ is hard, we therefore sample from a simpler distribution $Q(x)$. We first look at the un-normalized case:

Unnormalized importance sampling

1. Suppose we draw M samples $x_m \sim Q(x)$.
2. We then calculate the respective weights $w_m = \frac{P(x_m)}{Q(x_m)}$ to re-weight the $f(X)$.
3. We then compute $< f(X) >= \frac{1}{M} \sum_m f(x_m) w_m$

Since we re-weight every example, there are no samples wasted. This is called un-normalized because w_m are just weights and need not necessarily sum to 1. We check the correctness of the expectation of the target distribution. $< f(X) >= \int f(x)P(x)dx$
$< f(X) >= \int f(x)\frac{P(x)}{Q(x)}Q(x)dx$
We approximate the integral using the samples we draw from $Q(x)$ which gives us: $< f(X) >\sim \frac{1}{M}\sum_m f(x_m)\frac{P(x_m)}{Q(x_m)}$ where $x_m \sim Q(x)$
$< f(X) >\sim \frac{1}{M}\sum_m f(x_m)w_m$

Normalized importance sampling To get a better estimate of the target distribution, we now consider $P(x) = \frac{P'(x)}{\alpha}$. It turns out we can also obtain α from the samples. Let $R(x) = \frac{P'(X)}{Q(X)}$. We note thet this $R(x)$ is different from the w_m used earlier. w_m was the true ratio of distributions whereas R is the ratio of un-normazlied distribution and the proposed distribution. We show the correctness of that: $< R(x) >_Q= \int \frac{P'(x)}{Q(x)}Q(x)dx$
$< R(x) >_Q= \int P'(x)dx = \alpha$ which is the normazliation constant. Then to calculate the expectation for true distribution: $< f(x) >_P= \int f(x)P(x)dx = \frac{1}{\alpha}\int f(x)\frac{P'(x)}{Q(x)}Q(x)dx$
$< f(x)_P >= \frac{\int f(x)r(x)Q(x)dx}{\int r(x)Q(x)dx}$
$< f(x)_P >\sim \frac{\sum_m f(x_m)r_m}{\sum_m r_m}$ where $x_m \sim Q(x)$
$< f(x)_P >\sim \sum_m f(x_m)w_m$ where $w_m = \frac{r_m}{\sum_m r_m}$

The procedure for sampling is same as before, with the weights now computed as shown above. In the un-normalized sampling, since we don't have to compute the normalization constant, the w_m are available immediately. However, in this case the weights can be computed only after enough samples have been drawn using which the normalization constant can be computed.

The major pitfall for importance sampling is again if the $P$and Q distributions do not match. Consider the example shown in Figure 2: If more samples from the high-density region of $P(x)$ but which falls under the

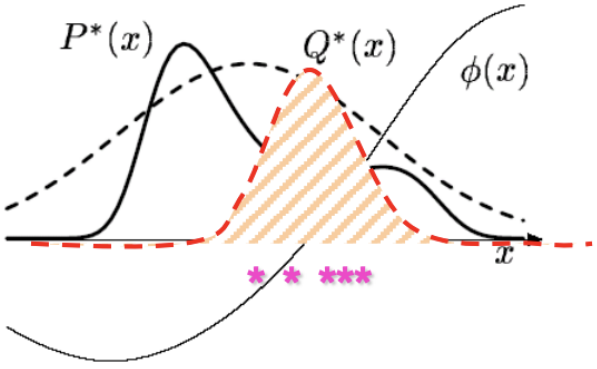

Figure 2: Figure of importance Sampling explained pictorically.

tail of $Q(x)$, we will end up getting less of these high-weight samples as they are under the tail of $Q(x)$. Similarly, samples from the high-density $Q(x)$ if they fall under tail of $P(x)$, we will end with many low-weight samples which will be less important. And since the convergence criterion is based on the change in the $< f(x) >$ value, we might get stuck in the local estimate.

Weighted Re-Sampling There are two possible solutions. The first is to use heavy tail $Q(x)$, but it might lead to lot of samples wasted. The other idea is to use weighted re-sampling. The idea is that we stop using previous examples, but we do another round of re-sampling. The procedure is as follows:

1. Draw samples from the Q
2. Construct weights $w_m = \frac{\frac{P(x_m)}{Q(x_m)}}{\sum_l \frac{P(x_l)}{Q(x_l)}}$
3. Sub-sample x from with probability as w_m .

This re-weights the samples and amplifies the high-importance samples. The low-probability will get suppressed.

Limitations of Monte Carlo Methods There are some fundamental issues with these methods. The weighted re-sampling method lets us re-shape the proposed distribution $Q(x)$. But since it is not an online algorithm, it requires us to store large number of samples and again re-draw samples from those. If the samples are not good, then the procedure needs to repeated again, so it is not very handy in practice. Online algorithms' advantage is that there is no storage cost but the main problem is that the $Q(x)$ function might not match the $P(x)$.

2 Markov Chain Monte Carlo (MCMC)

- MCMC algorithms have adaptive proposals
- Instead of $Q(x')$ they use $Q(x'|x)$ in which x' is the new state being sampled and x is the previous sample
- As, x changes, $Q(x'|x)$ can also change as a function of x'

2.1 Metropolis-Hastings Algorithm

1. Initialize starting state $x^{(0)}$ and set $t = 0$
2. Burn-in: while samples have not converged, draw samples...
   - $x = x^{(t)}$
   - $t = t + 1$
   - sample $x^* \sim Q(x^*|x)$ # draw from proposal
   - sample $u \sim Uniform(0, 1)$ # draw acceptance threshold
     - if $u < A(x^*|x) = min\left(1, \frac{P(x^*)Q(x|x^*)}{P(x)Q(x^*|x)}\right)$, then $x^t = x^*$ # transition
     - else $x^t = x$ # stay in current state
   - Take samples from $P(x)$: reset $t = 0$, for $\mathbf{t} = 1 : N$
     - $x(t + 1) \leftarrow$ Draw sample $x(t)$

2.2 Markov Chains Overview

- **Markov Chain:** a sequence of random variables $x^{(1)}, x^{(2)}, ..., x^{(n)}$ with the Markov Property
- **Markov Property:** $P(x^{(n)} = x|x^{(1)}, ..., x^{(n-1)}) = P(x^{(n)} = x|x^{(n-1)})$
- **Transition kernel:** $P(x^{(n)} = x|x^{(n-1)})$
  - The next state depends only on the previous state
- Random variables $x^{(i)}$ can be vectors
  - $x^{(t)}$ is defined as the t-th sample of **all** variables in a graphical model
  - $X^{(t)}$ represents the entire state of the graphical model at time, t
- We study homogeneous Markov Chains, in which the transition kernel is fixed with time
  - We will call the kernel $T(x'|x)$, where x is the previous state and x' is the next state $P(x^{(t)} = x|x^{(t-1)})$

2.3 Markov Chains Key Concepts

- **Probability distributions over states:** $\pi^{(t)}(x)$ is a distribution over the state of the system x, at time t
  - When considering MCs, we do not think of the system as being in one state, but rather as having a distribution over states
  - For graphical models, x represents **all** variables
- **Transitions:** states transition from $x^{(t)}$ to $x^{(t+1)}$ according to transition kernel $T(x'|x)$
  - We can also transition entire distributions:

$$\pi^{(t+1)}(x') = \sum_x \pi^{(t)}(x)T(x'|x)$$

  - At time t, state x has a probability mass of $\pi^{(t)}(x)$ and the transition probability redistributes this mass to other states x'
- **Stationary distributions:** $\pi(x)$ is a stationary distribution if it does not change under the transition kernel:

$$\pi(x') = \sum_x \pi(x)T(x'|x), \text{ for all x'}$$

- To understand why stationary distributions are important in MCMC, we need to define the following notions:
  - **Irreducible:** an MC is irreducible if you can get form any state x to any other state x' with a probability > 0 in a finite number of steps
    - No states are unreachable in the defined state space
  - **Aperiodic:** an MC is aperiodic if you can return to any state x at any time
    - Periodic MCs have states that need ≥ 2 time steps to return to (cycles)
  - **Ergodic (or regular):** an MC is ergodic if it is irreducible and aperiodic
    - Ergodicity is important and implies that you can reach the stationary distribution $\pi_{st}(x)$ no matter what the initial distribution $\pi^{(0)}(x)$

  - All good MCMC algorithms must satisfy ergodicity so that you cannot have an initialization that will never converge
- **Reversible (detailed balance):** an MC is reversible if there exists a distribution $\pi(x)$ such that the detailed balance condition is satisfied:

$$\pi(x')T(x|x') = \pi(x)T(x'|x)$$

  - The probability of $x' \to x$ is the same as $x \to x'$
  - Reversible MCs **always** have a stationary distribution
  - Proof:

$$\begin{aligned}
\pi(x')T(x|x') &= \pi(x)T(x'|x) \\
\sum_x \pi(x')T(x|x') &= \sum_x \pi(x)T(x'|x) \\
\pi(x')\sum_x T(x|x') &= \sum_x \pi(x)T(x'|x) \\
\pi(x') &= \sum_x \pi(x)T(x'|x)
\end{aligned}$$

  - The last line is the definition of the stationary proof

2.4 Why MH Works

- In MH, we draw samples x' according to $Q(x'|x)$ and then accept or reject the sample according to $A(x'|x)$
- So, the transition kernel is:

$$T(x'|x) = Q(x'|x)A(x'|x)$$

- Proof that MH satisfies detailed balance. Recall that...

$$A(x'|x) = min\left(1, \frac{P(x')Q(x|x')}{P(x)Q(x'|x)}\right)$$

This implies that if $A(x'|x) < 1$, then $\frac{P(x)Q(x'|x)}{P(x')Q(x|x')} > 1$ and thus $A(x|x') = 1$
if $A(x'|x) < 1$, then $\frac{\pi(x)Q(x'|x)}{\pi(x')Q(x|x')} > 1$ and thus $A(x|x') = 1$

  - Suppose $A(x'|x) < 1$ and $A(x|x') = 1$ Then,

$$\begin{aligned}
A(x'|x) &= \frac{P(x')Q(x|x')}{P(x)Q(x'|x)} \\
P(x)Q(x'|x)A(x'|x) &= P(x')Q(x|x') \\
P(x)Q(x'|x)A(x'|x) &= P(x')Q(x|x')A(x|x') \\
P(x)T(x'|x) &= P(x')T(x|x')
\end{aligned}$$

  - The last line is the definition of detailed balance

- This means that the MH algorithm leads to the stationary distribution $P(x)$
- Since we defined $P(x)$ to be the true distribution of x, this suggests that the algorithm eventually converges to the true distribution

2.5 Caveats

- We know MH eventually converges to the true distribution $P(x)$, but we have no guarantees as to when this will occur
  - **Burn-in period:** represents the un-converged part of the Markov Chain, which we throw away
  - Knowing when to halt burn-in is an art and there are techniques to help determine this

2.6 Gibbs Sampling Overview

- **Gibbs Sampling:** an MCMC algorithm that samples each random variable of a graphical model, one at a time
  - GS is a special case of the MH algorithm
- GS algorithms are relatively simple to derive for many graphical models, such as mixture models and Latent Dirichlet allocation
- They have reasonable computation and memory requirements because they sample a single random variable at a time
- They also can be Rao-Blackwellized (integrate out some random variables) to decrease the sampling variance

2.7 Gibbs Sampling Algorithm

1. Let variables $x_1, ..., x_n$ represent random variables in our graphical model
2. Initialize the starting values of $x_1, ..., x_n$
3. Repeat the following until convergence:
   (a) Choose an ordering of the n variables (can be fixed or random)
   (b) For each variable x_i in the selected order:
      i. Sample x form $P(x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$ (the conditional distribution of x_i given the current values of all other variables)
      ii. Update $x_i \leftarrow x$

As soon as we update the current variable x_i, we **immediately** use the updated value for sampling the subsequent variables x_j

2.8 Recall Markov Blankets

- The conditional $P(x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$ in the GS algorithm can be simplified using Markov Blankets
  - Let $MB(x_i)$ be the Markov Blanket of x_i, then,
$$P(x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = P(x_i|MB(x_i))$$
- For a Bayesian network, the Markov Blanket of x is the set of parents, children and co-parents
- For a Markov Random Field, the Markov Blanket of x is its immediate neighbors

3 Topic Models: Collapsed Gibbs

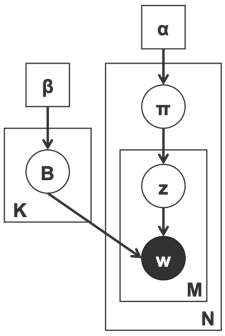

Figure 3: Graph representation of topic model (LDA) from Prof. Xing's slide

3.1 Graph Model Review

LDA graphical model mentioned in previous lectures can utilize collapsed Gibbs to complete inference task. Since previous scribe should have concluded LDA well, we review the model graph briefly here: we have V-dimensional topic vectors for K topics (denoted as B node). Each topic vector is sampled from a conjugate Dirichlet prior distribution, $Dirichlet(\beta)$. Similarly π_i is sampled from $Dirichlet(\alpha_i)$, $i \in \{1, ..., M\}$ where M is the number of documentation. z_{ij} is a topic index sampled from $Multinomial(\pi_i)$, with i, j denoting documentation ID and j^{th} token in a document. Finally, $w_{ij} \sim Multinomial(TopicVector_{z_{ij}})$.

3.2 Algorithm Overview

Similar to Gibbs sampling, we sample all z at every iteration until convergence, for all documents and all tokens. A simplified core algorithm pseudocode for inferring z is below.

1. Initialize the starting values of $z_{1,\cdot}, ..., z_{n,\cdot}$
2. Repeat the following until convergence:
   (a) For each variable z_{ij} in the token order and some documentation order
      i. Sample z from $P(z_{ij}|z_{-ij}, w)$
      ii. Update $z_{ij} \leftarrow z$

3.3 Compute $P(z_{ij}|z_{-ij}, w)$

We can derive this probability density function from two Dirichlet-Multinomial conditional distributions. The result is shown below.

$$P(z_{ij}) \propto \frac{n^{w_i}_{-i,j} + \beta}{n^{\cdot}_{-i,j} + W \cdot \beta} \cdot \frac{n^{d_i}_{-i,j} + \alpha}{n^{d_i}_{-i} + T \cdot \alpha} \tag{1}$$

$n_{-i,j}^{w_i}$ is the number of word positions a excluding i such that $w_a = w_i, z_{aj} = z_{ij}$; $n_{-i,j}^{\cdot}$ is the number of word positions a with $z_{aj} = z_{ij}$ in all documents excluding w_i; $n_{-i,j}^{d_i}$ is the number of word positions a in the current document excluding i such that $z_{aj} = z_{ij}$; $n_{-i}^{d_i}$ is the number of word positions a in the current document, which maybe ignored with α terms.

After we obtain the formula proportional to $P(z_{ij}|z_{-ij}, w)$, we can normalize probabilities for all z_{ij} and obtain the distribution. Note that now we would like to estimate new π and topic word vectors according to our new z sample, if the process does not converge.

4 Proof: Gibbs Sampling is MH

- Gibbs Sampling proposal distribution: $Q(x_i', x_{-i}|x_i, x_{-i}) = P(x_i'|x_{-i})$
- apply MH to this proposal

$$\begin{aligned} A(x_i', x_{-i}|x_i, x_{-i}) &= min(1, \frac{P(x_i', x_{-i}) \cdot Q(x_i, x_{-i}|x_i', x_{-i})}{P(x_i, x_{-i}) \cdot Q(x_i', x_{-i}|x_i, x_{-i})}) \\ &= min(1, \frac{P(x_i', x_{-i}) \cdot P(x_i|x_{-i})}{P(x_i, x_{-i}) \cdot P(x_i'|x_{-i})}) \\ &= min(1, \frac{P(x_i'|x_{-i}) \cdot P(x\_i) \cdot P(x_i|x_{-i})}{P(x_i|x_{-i}) \cdot P(x\_i) \cdot P(x_i'|x_{-i})}) \\ &= min(1, 1) \\ &= 1 \end{aligned} \tag{2}$$

As we can observe, GS is a special case of MH with a proposal accepting everything.

5 Practical Aspects of MCMC

- Evaluate $Q(x'|x)$
  - acceptance rate
  - autocorrelation function
- When to stop burn-in
  - plot the sample values vs. time
  - plot log-likelihood vs. time

5.1 Acceptance Rate

- Definition of acceptance Rate: the fraction of samples that MH accepts.
  - General guideline: 0.5 is a good acceptance rate.
  - If $P(x)$ and $Q(x'|x)$ are both Gaussian, the optimal acceptance rate is around 0.45 for $D = 1$ and around 0.23 as D goes to infinity.

- Tradeoff regarding choosing $Q(x'|x)$
  - Low-variance (narrow) proposals have high acceptance rate and proposed samples are close leading to more iterations to explore P(x) fully.
  - High-variance proposals explore P(x) faster, but rejects many samples which slows the algorithm.

5.2 Autocorrelation(AC) Function

AC indicates how correlated adjacent samples are, which means we want lower AC. AC function of a random variable x is:

$$R_x(k) = \frac{\sum_{t=1}^{n-k}(x_t - \hat{x}) \cdot (x_{t+k} - \hat{x})}{\sum_{t=1}^{n-k}(x_t - \hat{x})^2} \tag{3}$$

Sample Size Inflation Factor (SSIF) can be estimated by $R_x(1)$

$$S_x = \frac{1 + R_x(1)}{1 - R_x(1)} \tag{4}$$

The effective sample size is n/S_x, where n is the number of samples.

5.3 Monitor burn-in: Sample vs. Time

We can monitor convergence by plotting all samples from multiple MH runs. Well-mixed chains in the sample-time plots indicate good convergence. However, using this method needs us to visualize random variables, which may be high dimensional data.

5.4 Monitor burn-in: Likelihood vs. Time

To solve high-dimensional data issue mentioned above, we can plot $\log P(x_i|x_{-i}$, meaning we plot complete log-likelihood of a random variable x depending all other random variables in the model.

6 Summary

- **Direct Sampling:** is difficult to populate in high dimensional space
- **Rejection Sampling:** Creates sampling like in direct sampling, but only counts samples which are consistent with evidences
- **Likelihood Weighting:** Sample variables and calculate evidence weight, but only create samples which support evidences
- **MCMC:** use adaptive proposals $Q(x'|x)$ to sample from the true distribution $P(x)$
  - **Metropolis-Hastings:** allows you to specify $Q(x'|x)$, but choosing a good one is important
  - **Gibbs:** set $Q(x'|x)$ to the conditional distribution $P(x'|x)$; the acceptance rate is always 1, but that results in slow exploration of the space
  - Knowing when to halt burn in is an art