COS 513: FOUNDATIONS OF PROBABILISTIC MODELING
LECTURE 16

JIA LI,
JUAN CARLOS NIEBLES

1. MOTIVATION.

We have discussed latent variable models such as mixture models, factor analysis and Bayesian exponential families. In these cases, the exact posterior distribution of interest is easy to compute. Also, recall that we already have discussed a general purpose algorithm for posterior inference, the *Junction Tree* algorithm, which can be effectively applied to those models.

For more complex models, exact inference is not possible, because the posterior cannot be computed. For instance, a small change in the functional form of a conjugate prior to a non-conjugate prior, renders the posterior computation intractable, even if the structure of the graphical model stays the same (Figure 1).

2. APPROXIMATE POSTERIOR INFERENCE.

Consider the mixture model represented by the graphical model in Figure 2. This model is a generalized version of the Gaussian mixture model. Here we have introduced the mixture component centers as random variables, and

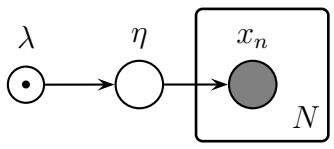

FIGURE 1. If the random variables in this model are normally distributed, we can compute posteriors analytically. However, if the distribution of η is not Gaussian, the posterior is uncomputable.

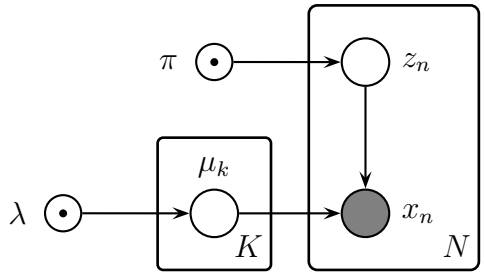

FIGURE 2. Mixture model with random cluster centers.

placed a prior on them. The generative process is characterized by:

$$\mu_k \sim N(0, \lambda) \qquad k = 1 \dots K, \tag{1}$$

$$z_n \sim Mult(\pi) \qquad n = 1 \dots N, \tag{2}$$

$$x_n | z_n, \mu_{1:K} \sim N(\mu_{z_n}, \sigma^2). \tag{3}$$

In the Bayesian setting, we are interested in estimating a posterior distribution of the mixture component centers $\mu_{1:K}$, instead of a single point estimate. That is, our interest is the posterior distribution

$$p(\mu_{1:K} | x_{1:N}), \tag{4}$$

which is not easy to compute. Let's see why. Suppose that the parameter π is fixed and $k = 3$. In that case, the posterior of interest is

$$p(\mu_1, \mu_2, \mu_3 | x_{1:N}) = \frac{p(\mu_1)p(\mu_2)p(\mu_3)\prod_{n=1}^{N} p(x_n|\mu_{1:3})}{\int_{\mu_1,\mu_2,\mu_3} p(\mu_1)p(\mu_2)p(\mu_3)\prod_{n=1}^{N} p(x_n|\mu_{1:3})}. \tag{5}$$

The expression in the numerator can be calculated by

$$p(x_n|\mu_{1:3}) = \sum_{i=1}^{3} \pi_i p(x_n|\mu_i). \tag{6}$$

On the other hand, the expression in the denominator can be rewritten as

$$p(x_{1:N}) = \int_{\mu_1}\int_{\mu_2}\int_{\mu_3} p(\mu_1)p(\mu_2)p(\mu_3)\prod_{n=1}^{N}\sum_{i=1}^{3}\pi_i p(x_n|\mu_i) \tag{7}$$

which is difficult to compute due to the product of sums in the integral argument. Alternatively, we could expand the denominator by marginalizing the mixture assignment z_n first:

$$p(x_{1:N}) = \sum_{z_{1:N}} p(z_{1:N}) \int_{\mu_1}\int_{\mu_2}\int_{\mu_3} p(\mu_1)p(\mu_2)p(\mu_3)\prod_{n=1}^{N} p(x_n|\mu_{z_n}) \tag{8}$$

(9)

$$p(x_{1:N}) = \sum_{z_{1:N}} p(z_{1:N}) \left(\int_{\mu_1} p(\mu_1) \prod_{\{n, z_n = 1\}} p(x_n|\mu_1) \right) \times \left(\int_{\mu_2} p(\mu_2) \prod_{\{n, z_n = 2\}} p(x_n|\mu_2) \right) \left(\int_{\mu_3} p(\mu_3) \prod_{\{n, z_n = 3\}} p(x_n|\mu_3) \right).$$

Even though we can manage to compute each integral in the parenthesis, the outter sum has 3^N terms. In general, we would have K^N terms, an exponential number of terms that makes the computation infeasible. This result suggests that performing exact inference in this model is not possible. But we still have hope: we must find an approximate algorithm to compute the posterior distribution of interest.

This is an example of a general rule of thumb: **practical Bayesian models require approximate posterior inference**.

3. SAMPLING.

In a general sampling setting, we have a target distribution $p(x)$ which cannot be computed. We will approximate it with a collection of samples:

$$p(x) \approx \frac{1}{M} \sum_{i=1}^{M} \delta_{x^{(i)}}(x) \tag{10}$$

where $x^{(i)}$ are samples from the target distribution.

In the following, we will discuss two methods to obtain such samples: **rejection sampling** and **importance sampling**.

4. REJECTION SAMPLING.

Assume we can compute the target distribution $p(x)$ but cannot sample from it. However, suppose we can sample from a proposal function $q(x)$ defined over the same sample space. Additionally, let q be such that there exists some C for which the following holds:

$$p(x) \leq C \cdot q(x) \tag{11}$$

The rejection sampling algorithm is described in Algorithm 1.

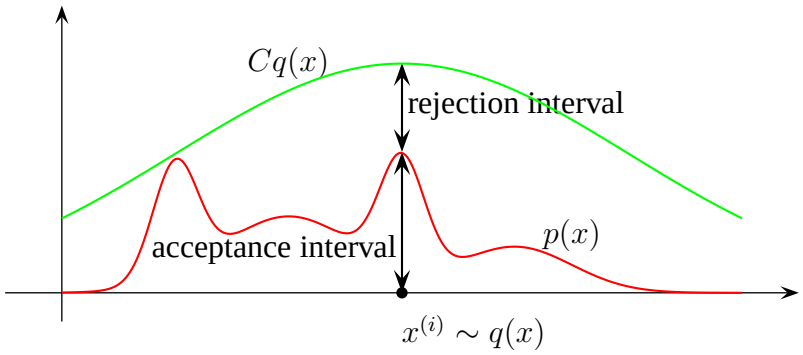

FIGURE 3. Sampling from $p(x)$, with a proposal function $q(x)$. First, we sample $x^{(i)}$ from the proposal distribution $q(x)$. Second, we obtain u from a uniform in the interval $(0, 1)$. Lastly, we accept $x^{(i)}$ if u falls in the normalized "acceptance interval".

Input: target distribution $p(x)$, proposal distribution $q(x)$
Output: M samples from $p(x)$
$i = 1$;
repeat
  $x^{(i)} \sim q(x)$;
  $u \sim$ Unif$(0, 1)$;
  if $u < \dfrac{p(x^{(i)})}{Cq(x^{(i)})}$ **then**
    accept $x^{(i)}$;
    increment i
  end
until $i = M$;

Algorithm 1: Rejection Sampling

Note that a good sampler tends to have a low scaling factor C. The bigger C is, the larger the rejection area is.

5. IMPORTANCE SAMPLING.

The Importance sampling algorithm provides a method to compute samples from an expectation

$$\mathbb{E}[f(x)] = \int_x f(x)p(x)dx. \tag{12}$$

Suppose we can compute $p(x)$ and we can sample from a proposal distribution $q(x)$, which has the same support as $p(x)$. The expectation in

Equation 12 can be formulated as:

$$\mathbb{E}[f(x)] = \int_x f(x)\frac{p(x)q(x)}{q(x)}dx \tag{13}$$

$$\approx \frac{1}{M}\sum_i f(x^{(i)})w(x^{(i)}) \tag{14}$$

where $x^{(i)} \sim q(x)$ and $w(x^{(i)}) = \frac{p(x^{(i)})}{q(x^{(i)})}$ is the sample-specific weight.

The criteria to assess if a proposal distribution q is appropiately chosen can be:

- It is easy to sample from q.
- q is close to p.

A nice feature of the importance sampling algorithm is that samples can be reused with different target distributions p. However, the downside is that it performs poorly when the sample space is high dimensional.

Finally, note that the two sampling methods described above assume that $p(x)$ can be computed. In practice, it is not always possible to calculate $p(x)$ exactly. Some alternative sampling methods that circumvent that restriction are provided by the MCMC techniques described in the following.

6. Markov Chain Monte Carlo (MCMC).

The MCMC sampling methods scale well with the dimensionality. In such sampling techniques, we only need to know the target distribution $p(x)$ up to a constant:

$$p(x) = \tilde{p}(x)/Z \longleftarrow \text{normalization factor} \tag{15}$$

During the MCMC sampling procedure, we are going to draw a sequence of states $x^{(t)}$, where x is a configuration of our set of random variables. The proposal distribution q depends on the previous state $x^{(t)}$, which is denoted by $q(x|x^{(t)})$.

The overall sampling procedure is composed by two steps:

- Draw a sample x^* from the proposal
- "Accept" according to a possibly random criterion

6.1. Metropolis algorithm.

The Metropolis algorithm was introduced in 1953. The procedure assumes a symmetric proposal $q(x_1|x_2) = q(x_2|x_1)$. A candidate sample x^* is drawn from q, and accepted with probability

$$A\left(x^*, x^{(t)}\right) = min\left(1, \frac{p(x^*)}{p(x^{(t)})}\right) = min\left(1, \frac{\tilde{p}(x^*)}{\tilde{p}(x^{(t)})}\right). \tag{16}$$

Notice that since we only need to compute the ratio $\frac{p(x^*)}{p(x^{(t)})}$, we can work directly with $\tilde{p}$ instead. Thus, we do not need to obtain the normalization constant Z since it will be canceled out.

If the candidate is accepted, we set $x^{(t+1)} = x^*$, otherwise we set the new state to $x^{(t+1)} = x^{(t)}$.

An important property of the Metropolis algorithm is that as long as $q(x_1|x_2) > 0$ for all (x_1, x_2), the empirical distribution $p_M(x^{(t)})$ converges to $p(x)$ as $t \to \infty$ and $M \to \infty$.

MCMC algorithms define a Markov chain on X whose stationary distribution is $p(x)$.

The general procedure to obtain independent samples from p is

- Run the Markov chain for a long time.
- Collect samples at some time lag.