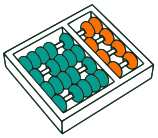

Monte Carlo inference

Samuel F. Chenatti

1 Introduction and motivation

This chapter focus on adopting the idea of Monte Carlo approximation to compute approximations for the prior and for any kind of calculation

The principle is simple: generate samples from the posterior and use them to calculate any quantity of interest. Mathematicaly, being f the quantity, we can describe this process as:

$$\boldsymbol{x}^{\boldsymbol{s}} \sim p(\boldsymbol{x}|\mathbb{D})$$

$$E[f|D]] \approx \frac{1}{S}\sum_{s=1}^{S} f(\boldsymbol{x}^{\boldsymbol{s}})$$

Where the accuracy of the approximation is determined by the number of samples S.

The next sections explores methods for sampling $\boldsymbol{x}^{\boldsymbol{s}}$ from the posterior.

1.1 Sampling from standart distributions

1.1.1 Cumulative Distribution Function method

We can sample from an univariate distribution using the idea of *inverse probability transform* from a *cumulative distribution function* F, presented in the introduction chapter.

Remember that F is a monotonically increasing function that maps the probability of the continuous random variable X assuming a value less or equal to x

$$F(x) = P(X \leq x) = u, 0 \leq u \leq 1$$

So the intuition is that $F^{-1}(u)$ tells us which value of x would make $F(x) = u$.

The method then consists in simply sampling u from a uniform distribution and plugging it into F^{-1} to obtain a sample from F.

Proof:

1.1.2 Box-Muller Method

In this method we first sample points z_1, z_2 uniformly distributed inside a circle of radius $r = 1$, that is, $z_1^s + z_2^2 < 1$. We define x_i in terms of z_i as follows:

$$x_i = z_i\left(\frac{-2\ln(r^2)}{r^2}\right)^{\frac{1}{2}}$$

$$r^2 = z_1^2 + z_2^2$$

From the multivariate change of variables formula (see chapter 2.6.2.1 from MLAPP [1]), we have:

$$p(x_1, x_2) = p(z_1, z_2)\left|\frac{\partial(z_1, z_2)}{\partial(x_1, x_2)}\right| = \left[\frac{1}{2\pi}exp(-\frac{1}{2}x_1^2)\right]\left[\frac{1}{2\pi}exp(-\frac{1}{2}x_2^2)\right]$$

The rightmost term of the above equation is the exactly formula of sampling x_1 and x_2 independently from a Gaussian distribution.

1.1.3 Rejection Sampling

The implementation

Be $p(x)$ the distribution we want to sample from and $\bar{p}(x)$ its unnormalized form. Let $q(x)$ be our *proposal distribution* that satisfies the relation $Mq(x) \geq \bar{p}$ for some constant M that provides an upper envelope to the function $\bar{p}(x)$. We then can use the following algorithm to draw samples from p using the following algorithm:

Algorithm 1 Rejection Sampling

```
S ← {}
while |S| < N do
    x ~ q(x)
    u ~ U(0,1)
    if u ≤ p(x)/(Mq(x)) then
        S ← x {Accept x as a valid sample}
    end if
end while
```

Proof:

Let S be the set of accepted points (x,u). Let S_0 be the set of accepted points (x,u) for which $x \leq x_0$

$$P(x \leq x_0 | x \text{ accepted}) = \frac{P(x < x_0)}{P(x \text{ accepted})} = \frac{\int_x \int_u \mathbb{I}((x,u) \in S_0) q(x) du dx}{\int_x \int_u \mathbb{I}((x,u) \in S) q(x) du dx} = \frac{int_{-\infty}^{x_0} \bar{p} dx}{int_{-\infty}^{\infty} \bar{p} dx} = p(x)$$

In terms of pratical implementation, given the distribution $p(x), a \leq x \leq b$, the rejection sampling consists in taking $x_i \sim U(a,b) = q(x)$ and $y_i \sim U(0,1)$ and accepting it if $y_i < p(x_i)$. When sampling, the accepted points will then cover the area under the curve delimited by $p(x)$.

1.1.4 Importance Sampling

Given a integral of the form:

$$\mathbb{E} = \int f(x) p(x) d(x)$$

We can use *importance sampling*, a Monte Carlo based method, to approximate it. Lets define $q(x)$ as our proposal pdf. Then we can rewrite our integral as

$$\mathbb{E}[f] = \int f(x) \frac{p(x)}{q(x)} dx \approx \frac{1}{S} \sum_{s=1}^{S} w_s f(x^s)$$

Notice that we are now sampling from $q(x)$ instead of $p(x)$. We call $w_s \triangleq \frac{p(x)}{q(x)}$ the *importance weights*. We are now left with the problem of finding the proposal pdf, which we resolve by minimizing the variance of our estimation:

$$\text{var}_q \left[\frac{p(x) f(x)}{q(x)} \right] = \mathbb{E} \left[\frac{p^2(x) f^2(x)}{q^2(x)} \right] - \left(\mathbb{E} \left[\frac{p(x) f(x)}{q(x)} \right] \right)^2 = \int \frac{p^2(x) f^2(x) dx}{q(x)} - \int p(x) f(x) dx$$

Since the rightmost integral is constant with respect to q we can ignore it.

By Jensen's inequality (see theorem 2.8.1 from MLAPP [1]):

$$\mathbb{E}_{q(x)} \left[f^2(x) w^2(x) \right] \geq \mathbb{E}_{q(x)} [|f(x) w(x)|]^2 = \left(\int |f(x)| p(x) dx \right)^2$$

$$q^*(x) = \frac{\left| f(x) \right| p(x)}{\int \left| f(x') \right| p(x') dx'}$$

So the lower bound is obtained by the optimal importance distribution:

$$q^*(x) = \frac{|f(x)| p(x)}{\int |f(x')| p(x') dx'}$$

1.1.5 Particle Filtering

Before diving into the *particle filtering* it is recommended to read chapters 17 and 18 from MLAPP [1] book (or 13 from PRML). Here we will take some intuition and notation from *state space models*.

Until now we have only dealt with data assuming it as independently and identically distributed. If we face a dataset composed of sequential data (such as text, where each word is related to the previous word in the context) we can assume it as iid in order to achieve a reasonable models such as we do in Naive Bayes classifiers for text documents. But sometimes this assumption can not be enough (for example, in the NB classifier two sentences with the same words in different orders would lead to the same class) and we need to capture the "time-dependecy" of our samples.

This is where Markov Models comes into play. Suppose we want to model our text sentence breaking it into tokens (here we will assume words). Then each word corresponds to a *state* x_t which comes after a previous state x_{t-1}. This scheme can be seem in figure 1.

Figure 1. Each node x_t at the graph represents a observation in the Markov Model whereas each edge represents the relation between them.

We can model the joint probability of the sequence of observations as follows:

$$p(x_1, x_2, .., x_N) = \prod_{n=1}^{N} p(x_n | x_1, ..., x_{n-1}) \tag{1}$$

Assuming that each conditional probability on the right side is independent of all previous observations with the exception of the immediate one as showed in figure 1 (what we call Markov Assumption), we can write:

$$p(x_1, x_2, .., x_N) = p(x_1) \prod_{n=2}^{N} p(x_n | x_{n-1})$$

Which implies:

$$p(x_n | x_1, ..., x_{n-1}) = p(x_n | x_{n-1})$$

The conditional distribution $p(x_n|x_{n-1})$ tells us that to predict the next value in a sequence we only need the value of the immediately last state. This what we call a *first order Markov chain*. Since this restriction could not capture the full relationship of states, we can relax it by modeling our data with higher order Markov chains by stablishing that a state x_t should depend on the last two preceding states x_{t-1} and x_{t-2} on equation 1.

Unfortunately increasing the order of Markov chains make computations impractical since the underline model parameters should increase as well. We can then "hack" the assumption by creating a discrete latent variable z_t to represent our true state (or hidden state) that can be measured with noise by the observations x_t for which the Markov assumption still holds.

We now have a *transition model* $p(z_t|z_{t-1})$ and an *observation model* $p(x_t|z_t)$. We call this a *Hidden Markov Model*.

We now define a *State Space model* which is just like a *HMM*, but our hidden states are continuous.

$$\boldsymbol{z}_t = g(\boldsymbol{u}_t, \boldsymbol{z}_{t-1}, \boldsymbol{\epsilon}_t)$$
$$\boldsymbol{y}_t = h(\boldsymbol{z}_t, \boldsymbol{u}_t, \boldsymbol{\delta}_t)$$

Here g is the *transition model* and h is the *observation model*. The goal is then to estimate our belief state $p(\boldsymbol{z}_t|\boldsymbol{y}_{1:t}, \theta)$.

In plain english, we are trying to answer the following question: what is the probability of the current state being $\boldsymbol{z}_t$ given the last observations were $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, ..., \boldsymbol{y}_{t-1}\}$?

The classic algorithm for updating the belief state described in section 18.3.1 of MLAPP [1] can be computationally expensive. It turns out that we can approximate the belief state using a weighted set of particles:

$$p(\boldsymbol{z}_{1:t}|\boldsymbol{y}_{1:t}) \approx \sum_{s=1}^{S} \hat{w}_t^s \delta_{\boldsymbol{z}_{1:t}^s(\boldsymbol{z_{1:t}})}$$

Where $\hat{w}_t^s$ is the normalized weight of sample s at time t.

As we have seem before we can update the belief state using importance sampling. Taking the proposal as $q(\boldsymbol{z}_{1:t}^s|\boldsymbol{y}_{1:t})$, we have:

$$w_t^s \propto \frac{p(\boldsymbol{z}_{1:t}^s|\boldsymbol{y}_{1:t})}{q(\boldsymbol{z}_{1:t}^s|\boldsymbol{y}_{1:t})} \rightarrow \hat{w}_t^s = \frac{w_t^s}{\sum_{s'} w_t^{s'}} \tag{2}$$

Where $\hat{w}$ is the normalized version of w.

Taking $p(\boldsymbol{z}_{1:t}^s|\boldsymbol{y}_{1:t})$ from equation 2 and following the Markov assumptions we can get

$$p(\boldsymbol{z}_{1:t}^s|\boldsymbol{y}_{1:t}) \propto p(y_t|z_t)p(z_t|z_{t-1})p(z_{1:t-1}|y_{1:t-1})$$

We also set our proposal density to have the form:

$$q(z_{1:t}|y_{1:t}) = q(z_t|z_{1:t-1}, y_{1:t})q(z_{1:t-1}|y_{1:t-1})$$

And by plugging it back into the equation 2, we have:

$$w_t^s \propto w_{t-1}^s \frac{p(y_t|z_t^s)p(z_t^s|z_{t-1}^s)}{q(z_t^s|z_{t-1}^s, y_t)}$$

And we can approximate the posterior density using:

$$p(z_t|y_{1:t}) \propto \sum_{s=1}^{S} \hat{w}_t^s \delta_{\boldsymbol{z}_{1:t}^s(\boldsymbol{z_{1:t}})}$$

We can iteractively take each old sample s, propose an extesion using $z_t^s \sim q(z_t|z_{t-1}^s, y_t)$ and use equation 2 to give it a weight.

References

[1] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.