Monte Carlo Inference

Lesson No.11

Ivan Lima

1 Introduction

Exact inference is intractable for most probabilistic models of practical interest. One way to approximate inference is based on numerical sampling, known as Monte Carlo techniques. For most situations the posterior distribution is required primarily for the purpose of evaluating expectations.

The fundamental problem that we therefore wish to address involves finding the expectation of some function $f(x)$ with respect to a probability distribution $p(x)$. Here, the components of x might comprise discrete or continuous variables or some combination of the two. Thus in the case of continuous variables, we wish to evaluate the expectation [1]

$$\mathbb{E}(f|D) = \int f(x)p(x)dx \tag{1}$$

where the integral is replaced by summation in the case of discrete variables. The general idea behind sampling methods is to obtain a set of samples x_s (where $s = 1, ..., S$) drawn independently from the distribution $p(x)$. This allows the expectation 1 to be approximated by a finite sum

$$\mathbb{E}(f|D) = \frac{1}{S}\sum_{s=1}^{S} f(x_s) \tag{2}$$

The idea is very simple: generate some (unweighted) samples from the posterior, $x_s \sim p(x|D)$, and then use these to compute any quantity of interest, such as a posterior marginal, $p(x_1|D)$, or the posterior of the difference of two quantities, $p(x_1 - x_2|D)$, or the posterior predictive,$p(y|D)$, etc. All of these quantities can be approximated by equation 2 for some suitable function f. By generating enough samples, we can achieve any desired level of accuracy we like. The main issue is how do we efficiently generate samples from a probability distribution. Monte Carlo is a non-iterative methods for generating independent samples.

2 Sampling from standard distributions

In this section, we consider some simple strategies for generating random samples from a given distribution. Here we shall assume that an algorithm has been provided that generates pseudo-random numbers distributed uniformly over $(0, 1)$, and indeed most software environments have such a facility built in. [1]

2.1 Using the cdf

The simplest method for sampling from a univariate distribution is based on the **inverse probability transform**. Let F be a **cumulative density function** of some distribution we want to sample from, and let F^{-1} be its inverse. Then we have the following result theorem: If $U \sim U(0, 1)$ is a uniform rv, then $F^{-1}(U) \sim F$. We can prove this theorem as follows:

$$Pr(F^{-1}(U) \le x) \quad = \quad Pr(U) \le F(x)) \quad (applying \quad F \quad to \quad both \quad sides) \tag{3}$$

$$Pr(F^{-1}(U) \le x) \quad = \quad F(x) \quad because \quad Pr(U \le y) = y \tag{4}$$

where relation 3 follows since F is a monotonic function, and 4 follows since U is uniform on the unit interval.

Therefore, we can sample from any univariate distribution, for which we can evaluate its inverse cdf by generating a random number $U \sim U(0, 1)$ using a pseudo random number generator . Let u represent the height up the y axis. Then "slide along" the x axis until you intersect the F curve, and then "drop down" and return the corresponding x value. This corresponds to computing $x = F^{-1}(u)$ as Figure 1 shows.

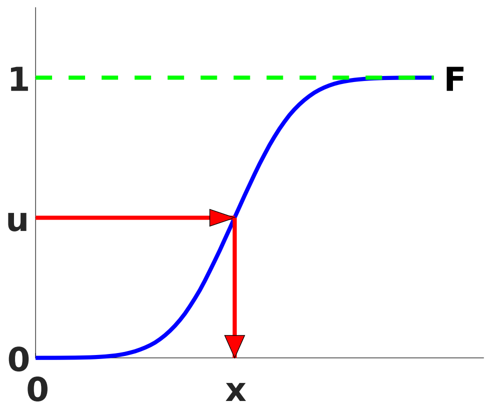


Figure 1. Sampling using an inverse CDF. Figure generated by sampleCdf

2.2 Sampling from a Gaussian (Box-Muller method)

The idea is sampling uniformly from a unit radius circle, and then use the change of variables formula to derive samples from a spherical $2d$ Gaussian. This can be thought of as two samples from a $1d$ Gaussian. In more detail, sample $z_1, z_2 \epsilon(-1,1)$ uniformly, and then discard pairs that do not satisfy $z_1^2 + z_2^2 \leq 1$. The result will be points uniformly distributed inside the unit circle, so $p(z) = \frac{1}{\pi}\prod$, (z inside the circle). Now define

$$x_i = z_i\left(\frac{-2ln(r^2)}{r^2}\right)^{\frac{1}{2}} \tag{5}$$

for $i = 1:2$, wherer $r^2 = z_1^2 + z_2^2$. Using the multivariate change of variables formula, we have

$$p(x_1, x_2) = p(z_1, z_2)\left|\frac{\partial(z_1, z_2)}{\partial(x_1, x_2)}\right| = \left[\frac{1}{\sqrt{2\pi}}exp\left(-\frac{1}{2}x_1^2\right)\right]\left[\frac{1}{\sqrt{2\pi}}exp\left(-\frac{1}{2}x_2^2\right)\right] \tag{6}$$

Hence x_1 and x_2 are two independent samples from a univariate Gaussian. This is known as the Box-Muller method. To sample from a multivariate Gaussian, we first compute the Cholesky decomposition of its covariance matrix, $\Sigma = LL^T$, where L is lower triangular. Next we sample $x \sim \mathcal{N}(0, I)$ using the Box-Muller method. Finally we set $y = Lx + \mu$. This is valid since:

$$cov(\mathbf{y}) = \mathbf{L}cov[\mathbf{x}]\mathbf{L}^T = \mathbf{LIL}^T = \Sigma \tag{7}$$

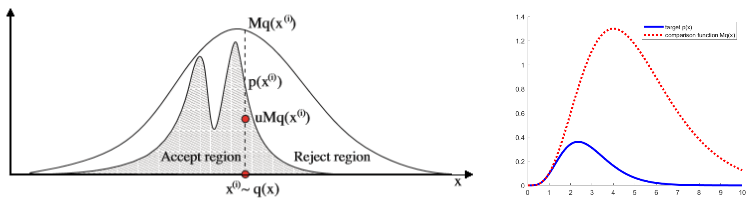
3 Rejection sampling

When the inverse cdf method cannot be used, one simple alternative is to use rejection sampling. In rejection sampling, we create a proposal distribution $q(x)$ which satisfies $Mq(x) \geq \tilde{p}(x)$, for some constant M, where $\tilde{p}(x) =$ is an unnormalized version of $p(x)$, i.e., $p(x) = \tilde{p}(x)/Z_p$ for some possibly unknown constant Z_p). The function $Mq(x)$ provides an upper envelope for $\tilde{p}$. We then sample $x \sim q(x)$, which corresponds to picking a random x location, and then we sample $u \sim U(0,1)$, which corresponds to picking a random height (y location) under the envelope. If $u > \frac{\tilde{p(x)}}{Mq(x)}$, we reject the sample, otherwise we accept it. See Figure 2 where the acceptance region is shown shaded, and the rejection region is the white region between the shaded zone and the upper envelope.

Since we generate with probability $q(x)$ and accept with probability $u > \frac{\tilde{p(x)}}{Mq(x)}$, the probability of acceptance is:

$$p(accept) = \int \frac{\tilde{p}(x)}{Mq(x)}q(x)dx = \frac{1}{M}\int \tilde{p}(x)dx \tag{8}$$

Hence we want to choose M as small as possible while still satisfying $Mq(x) \geq \tilde{p}(x)$.



(a) **(b)**

Figure 2. (a) Schematic illustration of rejection sampling. (b) Rejection sampling from a $Ga(\alpha = 5.7, \lambda = 2)$ distribution (solid blue) using a proposal of the form $MGa(k, \lambda - 1)$ (dotted red), where $k = \lfloor 5.7 \rfloor = 5$. The curves touch at $\alpha - k = 0.7$. Figure generated by rejectionSamplingDemo.

3.1 Application to Bayesian statistics

Suppose we want to draw (unweighted) samples from the posterior:

$$p(\theta|D) = p(D|\theta)p(\theta)/p(D) \tag{9}$$

We can use rejection sampling with $\tilde{p}(\theta) = p(D|\theta)p(\theta)$ as the target distribution, $q(\theta) = p(\theta)$ as our proposal, and $M = p(D|\tilde{\theta})$, where $\tilde{\theta} = argmax \quad p(D|\theta)$ is the MLE; this was first suggested in (Smith and Gelfand 1992). We accept points with probability

$$\frac{\tilde{p}(\theta)}{Mq(\theta)} = \frac{p(D|\theta)}{p(D|\hat{\theta})} \tag{10}$$

Therefore, samples from the prior that have high likelihood are more likely to be retained in the posterior. Of course, if there is a big mismatch between prior and posterior (which will be the case if the prior is vague and the likelihood is informative), this procedure is very inefficient.

3.2 Adaptive rejection sampling

We can automatically come up with a tight upper envelope $q(x)$ to any log concave density $p(x)$. The idea is to upper bound the log density with a piecewise linear function. We choose the initial locations for the pieces based on a fixed grid over the support of the distribution. We then evaluate the gradient of the log density at these locations, and make the lines be tangent at these points. Since the log of the envelope is piecewise linear, the envelope itself is piecewise exponential:

$$q(x) = M_i \lambda_i exp(-\lambda_i (x - x_{i-1})), \quad x_{i-1} < x \leq x_i \tag{11}$$

where x_i are the grid points. It is relatively straightforward to sample from this distribution. If the sample x is rejected, we create a new grid point at x, and thereby refine the envelope. As the number of grid points is increased, the tightness of the envelope improves, and the rejection rate goes down. This is known as adaptive rejection sampling (ARS) (Gilks and Wild 1992). As with standard rejection sampling, it can be applied to unnormalized distributions.

It is clear that we want to make our proposal $q(x)$ as close as possible to the target distribution $p(x)$, while still being an upper bound. But this is quite hard to achieve, especially in high dimensions. To see this, consider sampling from $p(x) = \mathcal{N}(0, \sigma_p^2 I)$ using as a proposal $q(x) = \mathcal{N}(0, \sigma_p^2 I)$. Obviously we must have $\sigma_q^2 \geq \sigma_p^2$ in order to be an upper bound. In D dimensions, the optimum value is given by $M = (\sigma_q/\sigma_p)^D$. The acceptance rate is $1/M$ (since both p and q are normalized), which decreases exponentially fast with dimension. For example, if σ_q exceeds σ_p by just 1%, then in 1000 dimensions the acceptance ratio will be about 1/20,000. This is a fundamental weakness of rejection sampling.

4 Importance sampling

We now describe a Monte Carlo method known as importance sampling for approximating integrals of the form

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

The idea is to draw samples x in regions which have high probability, $p(x)$, but also where $|f(x)|$ is large. The result can be super efficient, meaning it needs less samples than if we were to sample from the exact distribution $p(x)$. The reason is that the samples are focused on the important parts of space. For example, suppose we want to estimate the probability of a rare event. Define $f(x) = \Pi(x \epsilon E)$, for some set E. Then it is better to sample from a proposal of the form $q(x) \propto f(x)p(x)$ than to sample from $p(x)$ itself.

Importance sampling samples from any proposal, $q(x)$. It then uses these samples to estimate the integral as follows:

$$E[f] = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{S}\sum_{S=1}^{S} w_s f(x_s) = \widehat{I} \tag{13}$$

where $w_s \cong \frac{p(x_s)}{q(x_s)}$ are the importance weights. Note that, unlike rejection sampling, we use all the samples. How should we choose the proposal? A natural criterion is to minimize the variance of the estimate $\widehat{I} = \sum_{S=1}^{S} w_s f(x_s)$:

$$var_{q(x)}\left[f(x)w(x)\right] = \mathbb{E}_{q(x)}\left[f^2(x)w^2(x)\right] - I^2 \tag{14}$$

Since the last term is independent of q, we can ignore it. By Jensen's inequality, we have the following lower bound:

$$E_{q(x)}\left[f^2(x)w^2(x)\right] \geq \left(E_{q(x)}\left[\left|f(x)w(x)\right|\right]\right)^2 = \left(\int \left|f(x)\right| p(x)dx\right)^2 \tag{15}$$

The lower bound is obtained when we use the optimal importance distribution:

$$q^*(x) = \frac{\left|f(x)\right| p(x)}{\int \left|f(\acute{x})\right| p(\acute{x})d\acute{x}} \tag{16}$$

When we don't have a particular target function $f(x)$ in mind, we often just try to make $q(x)$ as close as possible to $p(x)$. In general, this is difficult, especially in high dimensions, but it is possible to adapt the proposal distribution to improve the approximation. This is known as adaptive importance sampling (Oh and Berger 1992).

An obvious problem with this approach is that the number of terms in the summation grows exponentially with the dimensionality of x. Furthermore, as we have already noted, the kinds of probability distributions of interest will often have much of their mass confined to relatively small regions of x space and so uniform sampling will be very inefficient because in high-dimensional problems, only a very small proportion of the samples will make a significant contribution to the sum. We would really like to choose the sample points to fall in regions where $p(x)$ is large, or ideally where the product $p(x)f(x)$ is large. **bis2006**

5 Particle filtering

Particle filtering (PF) is a Monte Carlo, or simulation based, algorithm for recursive Bayesian inference. It approximates the predict-update cycle described [2], section 18.3.1. It is very widely used in many areas, including tracking, time-series forecasting, online parameter learning, etc. For a book-length treatment, see (Doucet et al. 2001); for a good tutorial, see (Arulampalam et al. 2002), or [2], chapter 23, section 5.

5.1 The proposal distribution

The simplest and most widely used proposal distribution is to sample from the prior:

$$q(z_t|z_{t-1}^S, y_t) = p(z_t|z_{t-1}^S) \tag{17}$$

In this case, the weight update simplifies to

$$w_t^S \propto w_{t-1}^S p(y_t|z_t^S) \tag{18}$$

This can be thought of a "generate and test" approach: we sample values from the dynamic model, and then evaluate how good they are after we see the data. This is the approach used in the condensation algorithm (which stands for "conditional density propagation") used for visual tracking (Isard and Blake 1998). However, if the likelihood is narrower than the dynamical prior (meaning the sensor is more informative than the motion model, which is often the case), this is a very inefficient approach, since most particles will be assigned very low weight. It is much better to actually look at the data y_t when generating a proposal.

For an a overview on applications, such as robot localization, visual object tracking and time series forecasting, [2] chapter 23, section 5.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press Cambridge, Massachusetts London,England, 2012.