Monte Carlo Methods

A very general approach to Bayesian computation — applicable even to very high-dimensional problems — is to obtain a *sample* of points from the posterior distribution, and use it to make *Monte Carlo* estimates.

A single sample point will contain values for all the unknown parameters, hyperparameters, etc. — everything not known, except what we don't care about or have integrated away analytically.

We use this sample to approximate expected values by averages over the sample points. For example, from K values, $\theta^{(1)}, \ldots \theta^{(K)}$, for a parameter, sampled from $P(\theta \,|\, \text{data})$, we can approximate the predictive probability that $Y = 1$ by

$$P(Y=1 \mid \text{data}) \;\; \approx \;\; \frac{1}{K}\sum_{j=1}^{K} P(Y=1 \mid \theta^{(j)})$$

Obtaining a Sample by Simulating a Markov Chain

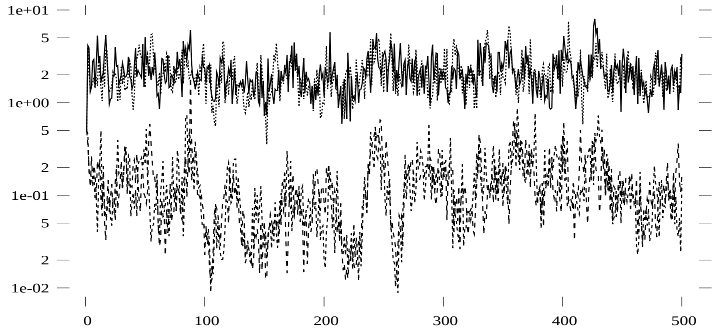
When the posterior distribution is too complex to sample from directly, we can instead simulate a *Markov chain* that will converge (asymptotically) to the posterior distribution.

States from the latter portion of this Markov chain will come (approximately) from the posterior distribution, though they will not be independent. We can use these states to make Monte Carlo estimates.

Finding such a Markov chain sounds hard, but fortunately there are general schemes that make this possible even for difficult problems. This *Markov chain Monte Carlo (MCMC)* approach is therefore very general. MCMC can also be very slow in some circumstances, but despite this, it is often the only viable approach to Bayesian inference using complex models.

An Example of Markov Chain Sampling

Here is a Markov chain run for a MLP network classification model, showing the four hyperparameters controlling the relevance of the four input variables:



The chain starts out in a state that isn't typical, but rapidly moves to fairly good point. It takes a while more for the distribution to apparently stabilize, however — iterations from about 50 or 100 on might be used to make predictions.

There high autocorrelation for the two hyperparameters with low values (only the equivalent of about 20 independent points in the last 400 iterations). Fortunately, the exact degree of irrelevance of largely irrelevant inputs isn't crucial.

The Metropolis Algorithm

The first MCMC algorithm dates back to 1953, and is still popular today. In this *Metropolis algorithm*, designed to sample from $\pi(x)$, a Markov chain transition from state x to state x' operates as follows:

1) A "candidate", x^*, is proposed according to some probabilities $S(x, x^*)$, satisfying the symmetry condition, $S(x, x^*) = S(x^*, x)$.
2) This candidate, x^*, is accepted as the next state with probability

$$\min\left[1,\; \pi(x^*)/\pi(x)\right]$$

If x^* is accepted, then $x' = x^*$. If x^* is instead rejected, then $x' = x$.

Provided the chain can't get trapped in one region, one can show that with transitions defined in this way, the distribution of the state gets arbitrarily close to π after sufficiently many transitions.

A good choice of proposal distribution, S, is crucial to getting the chain to converge to π rapidly.

Crucially, it's enough to be able to evaluate some function proportional to $\pi(x)$. We don't need the normalizing constant; it cancels when computing $\pi(x^*)/\pi(x)$.