November 19, 2017 Lecture 20, Nov. 13, 2017

Bayesian Inference

Lecturer: Justin Domke *Scribe: Lakshmi Vikraman*

1 Summary

Last class we discussed how Bayesian inference works with the help of a few examples. This class, we will review the main concepts and then describe the Metropolis algorithm for sampling from the posterior distribution.

2 Bayesian inference Review

As seen in the last class, Bayesian inference consists of two steps:

- Calculate the posterior using the data

$$Pr(M|Data) \propto Pr(M) \prod_{i=1}^{N} Pr(Y_i|X_i, M)$$

  Here $Pr(M|Data)$ is the posterior, M the monkeys (i.e. the models), X the weather and Y the colored blocks.

- Make predictions using the posterior. Given a new X', we want to predict Y'

$$Pr(Y'|X', Data) = \sum_{i=1} Pr(M|Data) Pr(Y'|X', M)$$

What is the issue? We are summing over all models M. There are often an infinite number of models which leads to computational difficulties while performing the summation.

How to approach this issue? We use a simple strategy where we sample models from the posterior and average them.

- Draw a "sample" of models,

$$M^1, M^2, ..., M^T \sim Pr(M|Data)$$

- Approximate the distribution over Y'

$$Pr(Y'|X', Data) \approx \frac{1}{T} \sum_{t=1}^{T} Pr(Y'|X', M^t)$$

Notice that we don't have $Pr(M|Data)$. This information is captured in the samples.
The next issue is to figure out how to sample from the posterior which is computationally difficult.

3 Sampling from shapes

3.1 Sampling from a square

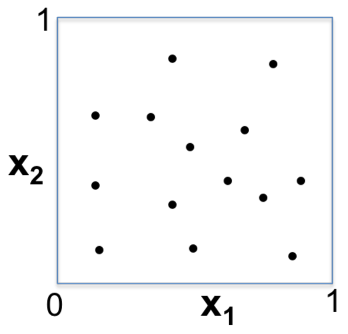

Figure 3.1: Sampling from a square

Consider a square with sides of length 1. To sample points from this square, we do the following.

- Draw $X_1 \sim U(0, 1)$
- Draw $X_2 \sim U(0, 1)$
- Return (X_1, X_2)

3.2 Sampling from a circle

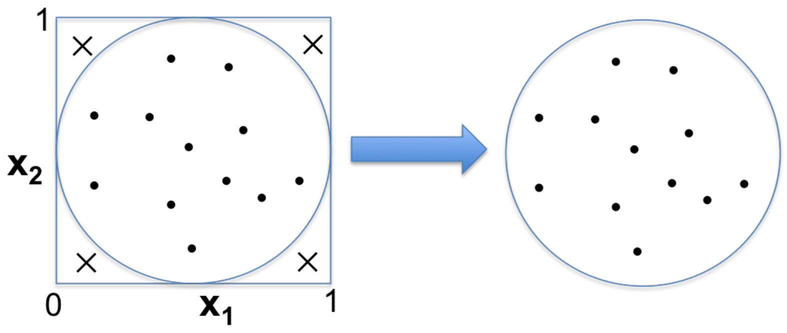

Figure 3.2: Rejection Sampling

Consider a circle with radius 0.5. To generate samples from the circle we can inscribe the circle in a square and then sample points from the square (as mentioned in the earlier section). Then we throw away the points which do not fall within the circle. This method is called "rejection sampling".

- Draw $X_1 \sim U(0, 1)$

- Draw $X_2 \sim U(0,1)$
- If $(X_1 - 0.5)^2 + (X_2 - 0.5)^2 \leq (0.5)^2$, return (X_1, X_2)

This doesn't work in high dimensions because the most of the area of the square is concentrated around the boundary. This algorithm will reject 99.99% of the samples. Therefore, in general for Bayesian Inference we cannot use rejection sampling.

4 Metropolis Algorithm

Assume that we want to sample from a distribution $p(M)$. But we only have $\hat{p}(M)$.

$$\begin{aligned}\hat{p}(M) &= p(M) \times Z \\ &= Pr(M) \prod_{i=1}^{N} Pr(Y_i|X_i, M)\end{aligned}$$

- $p(M) = Pr(M|Data)$is the posterior
- $Z = Pr(Data)$is the normalization constant
- $Pr(M)$ is the prior
- $\prod_{i=1}^{N} Pr(Y_i|X_i, M)$ is the likelihood

We will sample directly from $\hat{p}(M)$.

Algorithm 1 Metropolis Algorithm

```
1: Initialize:
     $M^0$ = initial state
2: for $t$ = 0,1,....,$T-1$ do
3:     $M' \sim q(M'|M^t)$
4:     if rand() $\leq \frac{\hat{p}(M')}{\hat{p}(M^t)}$ then
5:         $M^{t+1} \leftarrow M'$
6:     else
7:         $M^{t+1} \leftarrow M^t$
   return $M^1$,............,$M^T$
```

The algorithm works by first picking an initial point from the distribution. It then picks the next point based on a proposal distribution and then calculates the ratio of their probabilities. If the ratio is high, then the new point is accepted, else it is rejected. Hence the algorithm performs a random walk which is biased based on whether the point is accepted or rejected.

We will go run through a demo in class and implement Metropolis algorithm for $\hat{p}(M)$ shown below. The code is up on moodle.

$$\hat{p}(M) = \frac{7}{10} \exp(-\frac{1}{2}(m - 1.5)^2) + \frac{3}{10} \exp(-\frac{1}{2}(m + 1.5)^2)$$

The proposal function used in the code is proposal$(M) = M + x * \text{np.random.rand()}$. Figure 4.1 shows how the algorithm generates samples which converge to the true distribution after 5000 iterations for $x = 1$ in the proposal function. The green curve is $\hat{p}(M)$ and blue curve plots the samples from the algorithm.

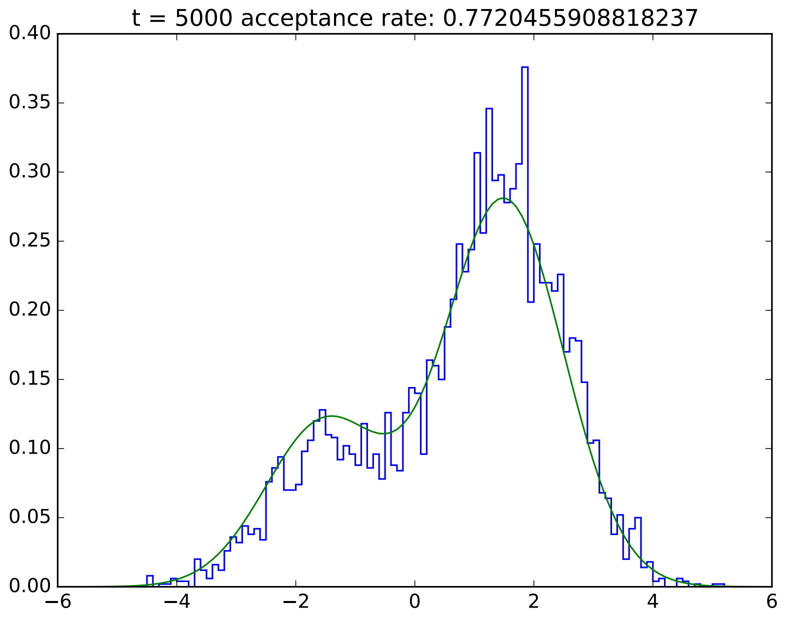

Figure 4.1: Metropolis Algorithm execution output for x=1

- The value x is important. This is used to tune the proposal distribution. This is the noise which is added to the current M to get the new M'. If this value is high, then the new points are much further from M, having low probabilities which results in the points getting rejected and the algorithm gets stuck. For eg: when the value is 3, the acceptance rate is around 7%. When the value is increased to 300, the acceptance rate is very low and the algorithm converges very slowly.
- When this value is very low, the acceptance rate is very high, but the samples stay within a small region and takes a long time to converge. This is called "Drunkard's walk".
- The bottom line is that the proposal function should be tuned to take longer steps even if the rejection rate is high.
- If we have a $\hat{p}(M)$ with very well separated gaussians, then the algorithm spends a lot of time in one gaussian and moves over to the next and spends a lot of time there and then moves back again. If we take large steps then it moves back and forth faster. It is also possible to take small steps, but this will take much longer to converge.

Mixing time : This is the number of iterations required for the algorithms to return samples from $\hat{p}(M)$. In real world problems, it is not possible to know if the algorithm has converged since we don't know the true distribution.

5 Detailed Balance

The metropolis algorithm works because of a property called detailed balance. A random walk algorithm has detailed balance with respect to $p(M)$ if

$$p(M)Pr(M \to N) = p(N)Pr(N \to M)$$

$p(M)$ and $p(N)$ are the distributions.
$Pr(M \to N)$ and $Pr(N \to M)$ comes from the random walk algorithm.
$Pr(M \to N)$ is the probability in state M at time t and state N at time $t + 1$.
These probabilities are invariant with respect to time.

Intuitively this means

$$Pr(M \to N) = q(N|M)\min(1, \frac{\hat{p}(N)}{\hat{p}(M)}), \text{where } M \neq N$$