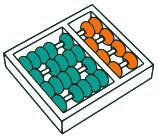

Markov Chain Monte Carlo

Lesson No. 12

Rosa Yuliana Gabriela Paccotacya Yanque - RA:263068

1 Introduction

Markov Chain Monte Carlo (MCMC) Inference allows sampling from a large class distribution in a high-dimensional space, this can not be done using just Monte Carlo methods since it presents limitations in spaces of high dimensionality.

MCMC aims to calculate $p^*(x)$,that can be a posterior or prior that cannot be calculated due to its intractability, using Markov Chains.

A Markov Chain [1] is defined to be a series of random variables $z^{(1)},...,z^{(M)}$ that holds the following property:

$$p(z^{(m+1)}|z^{(1)},..,z^{(m)}) = p(z^{(m+1)}|z^{(m)}) \tag{1}$$

meaning that the probability of a state in time $m+1$ will be only dependant of the state at m moment. A Markov Chain has a matrix with the transition of probabilities through states. We can calculate the Probability distribution for the next state by multiplying the current state vector with the transition matrix.Eventually, this multiplication will converge in a stationary state, which will be the stationary distribution specifically for the transition matrix given. This is important because allows to define the probability for every state of a system at a random time. This process of going to a next state given the probabilities of a current state is callesd as a random walk.

In a few words, MCMC constructs a Markov Chain whose stationary distribution is our traget $p^*(x)$. So we will perform a random walk on the state space, in such a way that the time spent in each state is proportional to $p^*(x)$, and by drawing samples from the chain, we can perform Monte Carlo integration wrt p^* [2].

When contrasting Variational Inference to MCMC, each approach has different advantages, On the one hand, Variational Inference is deterministic, is faster for small to medium problems, it is easy to decide when to stop and provides a lower bound on the log-likelihood. On the other hand, MCMC is easier to implement, its applicability extends to a variety of challenging models and it is faster in huge models or datasets due to it works with specific values instead of distributions.

There are different algorithms based on MCMC, the most known are GIBBS Sampling, and Metropolis-Hasting, these will be reviewed in this summary along other important features about MCMC.

2 Gibbs sampling

It samples each variable in turn, conditioned on the values of all the other variables in the distribution. That is, given a joint sample x^s of all the variables, we generate a new sample x^{s+1} by sampling each component in turn, based(conditioned) on the most recent values of the other variables. For example, if we have $D = 3$ random variables: X_1, X_2, X_3, in the first iteration we sample

- $x_1^{s+1} \sim p(x_1|x_2^s, x_3^s)$
- $x_2^{s+1} \sim p(x_2|x_1^{s+1}, x_3^s)$
- $x_3^{s+1} \sim p(x_2|x_1^{s+1}, x_2^{s+1})$

The algorithm for Gibbs Sampling is as listed below:

- Initialize $x^{(0)}$
- For iteration $i = 1, 2, ...$ till converging:
  - sample $x_1^{(i)} \sim p(X_1 = x_1 | X_2 = x_2^{(i-1)}, X_3 = x_3^{(i-1)}, ..., X_D = x_D^{(i-1)})$
  - sample $x_2^{(i)} \sim p(X_2 = x_2 | X_1 = x_1^{(i)}, X_3 = x_3^{(i-1)}, ..., X_D = x_D^{(i-1)})$
  - sample $x_D^{(i)} \sim p(X_D = x_D | X_1 = x_1^{(i)}, X_2 = x_2^{(i)}, ..., X_D = x_{D-1}^{(i)})$

To do Gibbs Sampling we must know only the immediate neighbors, so, it is a distributed algorithm but not a parallel one, since samples must be generated sequentially. Furthermore, it is necessary to discard some of the initial samples(where convergence has not been reached yet) until the Markov chain has burned in, or entered its stationary distribution.

2.1 Collapsed Gibbs Sampling

There are some cases where it is possible to analytically integrate out some of the unknown quantities, and then sample the rest. This algorithm is more efficient since the sampling is done in a lower dimensional space, however it is only worth applying it, if integration can be done quickly.

In a few words, we sample z and integrate out θ. Thus, the θ parameters do not participate in the Markov chain; consequently we can draw conditionally independent samples $\theta^s \sim p(\theta | z^s, D)$, which will have much lower variance than samples drawn from the joint state space.

The whole process is called Rao-Blackwellisation since it satisfies the same-titled theorem, that guarantees that the variance obtained analytically integrating out θ will never be higher than the variance of a direct MC estimate.

3 Metropolis-Hastings algorithm

Metropolis-Hastings(MH) is a generalized algorithm that uses a proposal distribution or kernel that allows us to calculate the probability q to move from the current state x to a new state x'. After proposing a move to x', it has to be decided whether to accept this proposal or not according to some formula. If the proposal is accepted, the new state is x', otherwise the new state is the same as the current state, x (i.e., we repeat the sample). More details about proposal distributions can be found at [2].

The algorithm is summarized as follows:

- Initialize x^0
- for s=0,1,2,...
  - Define $x = x^s$
  - $x' \sim q(x'|x)$
  - Compute acceptance probability:
$$\alpha = \frac{\tilde{p}(x')q(x|x')}{\tilde{p}(x)q(x'|x)}$$
  - Compute $r = min(1, \alpha)$
  - Sample fron a uniform distribution $u \sim U(0,1)$
  - Set new sample $x^{(s+1)}$ to:

$$x^{s+1} = \begin{cases} x', & \text{if } u < r \\ x^s, & \text{if } u \geq r \end{cases}$$

Gibbs Sampling algorithm presented before is a special case derived from MH. Particularly, the sequence of proposals will be of the form:

$$q(x'|x) = p(x'_i | \mathbf{x}_{\_i}) \mathbb{I}(\mathbf{x}'_{\_i} = \mathbf{x}_{\_i}) \tag{2}$$

that is, we move to a new state sampled from its full conditional, but $\mathbf{x}\_i$ is left unchanged. So, the acceptance rate r of this proposal is 1 that is 100%.

The proof is as follows:

$$\alpha = \frac{p(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p(\mathbf{x})q(\mathbf{x}'|\mathbf{x}')} = \frac{p(x'\_i|\mathbf{x}'\_i)p(\mathbf{x}\_i)p(x\_i|\mathbf{x}'\_i)}{p(x\_i|\mathbf{x}\_i)p(\mathbf{x}\_i)p(x'\_i|\mathbf{x}\_i)} \tag{3}$$

$$= \frac{p(x'\_i|\mathbf{x}\_i)p(\mathbf{x}\_i)p(x\_i|\mathbf{x}\_i)}{p(x\_i|\mathbf{x}\_i)p(\mathbf{x}\_i)p(x'\_i|\mathbf{x}\_i)} = 1 \tag{4}$$

4 Speed and accuracy of MCMC

There are some points to take into account when working with MCMC that affect the speed and the accuracy:

4.1 Bur-in Phase

It is the initial period when doing when doing the random walk the convergence has not been reached and samples has to be ignored since its stationary distribution do not come from p^*. So, sampling before converging must not be done cause it can lead to erroneous conclusions.

4.2 Mixing Time

It is the amount of time it takes for a Markov chain to converge to the stationary distribution, and forget its initial state. The mixing time depends on the transition matrix, however, computing it can be hard in high-dimensional and/or continuous state spaces. So, an alternative approach is to examine the geometry of the state space using their conductance, that is chains with low conductance have high mixing time. MCMC does not work well in cases with high mixing time.

4.3 Practical convergence diagnostics

Another approach to avoid computing transition matrix, is to diagnose when there is non-convergence. So, we run multiple chains from very different overdispersed starting points, and to plot the samples of some variables of interest. This is called a trace plot. If the chain has mixed, it should have "forgotten" where it started from, so the trace plots should converge to the same distribution, and thus overlap with each other. In this way the computational efficiency is reduced, but not the statistical validity [2].

‘

4.4 Accuracy of MCMC

The samples produced by MCMC are auto-correlated, and this reduces their information content relative to independent or "perfect" samples. We can quantify this as follows. Suppose we want to estimate the mean of $f(X)$, for some function f, where $X \sim p()$. Denote the true mean by

$$f^* = \mathbb{E}[f(X)] \tag{5}$$

A Monte Carlos estimate is given by

$$\bar{f} = \frac{1}{S}\sum_{s=1}^{S} f_s \tag{6}$$

where $f_s \triangleq f(x_s)$ and $x_s \sim p(x)$. An MCMC estimate of the variance of this estimate is given by

$$Var_{MCMC}[\bar{f}] = Var_{MC}(\bar{f} + \frac{1}{S^2}\sum_{s\neq t}\mathbb{E}[(f_s - f^*)(f_t - f^*)] \tag{7}$$

where the first term is the Monte Carlo estimate of the variance if the samples weren't correlated, and the second term depends on the correlation of the samples. We can measure this using the autocorrelation functions(ACF) that is:
Define the sample-based auto-correlation at lag t of a set of samples f_1,...,f_S as follows:

$$p_t \triangleq \frac{\frac{1}{S-t}\sum_{s=1}^{S-t}(f_s-\bar{f})(f_{s+t}-\bar{f})}{\frac{1}{S-1}\sum_{s=1}^{S}(f_s-\bar{f})^2} \tag{8}$$

4.5 How many chains?

In practice, it is common to run a medium number of chains (say 3) of medium length that could be 100,000 steps, and to take samples from each after discarding the first half of the samples.

References

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

[2] K. P. Murphy, *Machine Learning : A Probabilistic Perspective*, 1st. Cambridge, Mass. [u.a.]: MIT Press, 2013.