Bayesian Inference and MCMC

Aryan Arbabi

Partly based on MCMC slides from CSC412

Fall 2018

Bayesian Inference - Motivation

- Consider we have a data set $D = \{x_1, ..., x_n\}$. E.g each x_i can be the outcome of a coin flip trial.
- We are interested in learning the dynamics of the world to explain how this data was generated ($p(D|\theta)$)
- In our example θ is the probability of observing head in a coin trial
- Learning θ will enable us to also predict future outcomes ($P(x'|\theta)$)

Bayesian Inference - Motivation

- The primary question is how to infer θ
- Observing the sample set D gives us some information about θ, however there is still some uncertainty about it (specially when we have very few samples)
- Furthermore we might have some prior knowledge about θ, which we are interested to take into account
- In Bayesian approach we embrace this uncertainty by calculating the posterior $p(\theta|D)$

Bayes rule

- Using Bayes rule we know:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \propto P(D|\theta)P(\theta)$$

- Where $P(D|\theta)$ is the data likelihood, $P(\theta)$ is the prior, and $P(D)$ is called the evidence
- In Maximum Likelihood estimation (MLE) we find a θ that maximizes the likelihood:

$$\arg\max_{\theta}\{P(D|\theta)\}$$

- In Maximum a posteriori (MAP) estimation, the prior is also incorporated:

$$\arg\max_{\theta}\{P(D|\theta)P(\theta)\}$$

Bayesian Inference

- Alternatively, instead of learning a fixed point-value for θ, we can incorporate the uncertainty around θ
- We can predict the probability of observing a new sample x' by marginalizing over θ:

$$P(x'|D) = \int_{\theta} P(\theta|D)P(x'|\theta)d\theta$$

- In cases such as when the model is simple and conjugate priors are being used the posterior and the above integral can be solved analytically
- However in many practical cases it is difficult to solve the integral in closed form

Monte Carlo methods

- Although it might be difficult to solve the previous integral, however if we can take samples from the posterior distribution, it can be approximated as

$$\int_{\theta} P(\theta|D)P(x'|\theta)d\theta \simeq \frac{1}{n} \sum_{1\leq i\leq n} P(x'|\theta^{(i)})$$

- Where $\theta^{(i)}$s are samples from the posterior:

$$\theta^{(i)} \sim P(\theta|D)$$

- This estimation is called Monte Carlo method

Monte Carlo methods

- In its general form, Monte Carlo estimates the following expectation

$$\int_x P(x) f(x) dx \approx \frac{1}{S} \sum_{1 \leq s \leq S} f(x^{(s)})$$

$$x^{(s)} \sim P(x)$$

- It is useful wherever we need to compute difficult integrals:
  - Posterior marginals
  - Finding moments (expectations)
  - Predictive distributions
  - Model comparison

Bias and variance of Monte Carlo

- Monte Carlo is an unbiased estimation:

$$\mathbb{E}[\frac{1}{S}\sum_{1\leq s\leq S} f(x^{(s)})] = \frac{1}{S}\sum_{1\leq s\leq S} \mathbb{E}[f(x^{(s)})] = \mathbb{E}[f(x)]$$

- The variance reduces proportional to S:

$$Var(\frac{1}{S}\sum_{1\leq s\leq S} f(x^{(s)})) = \frac{1}{S^2}\sum_{1\leq s\leq S} Var(f(x^{(s)}))) = \frac{1}{S} Var(f(x))$$

How to sample from $P(x)$?

- One way is to first sample from a Uniform[0,1] generator:

$$u \sim Uniform[0,1]$$

- Transform the sample as:

$$h(x) = \int_{\text{inf}}^{x} p(x')dx'$$

$$x(u) = h^{-1}(u)$$

- This assumes we can easily compute $h^{-1}(u)$, which is not always true

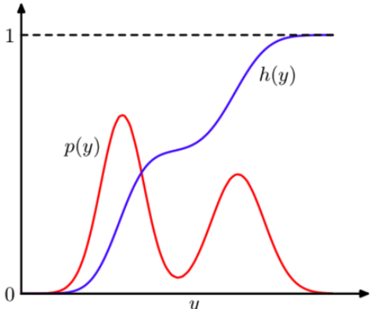


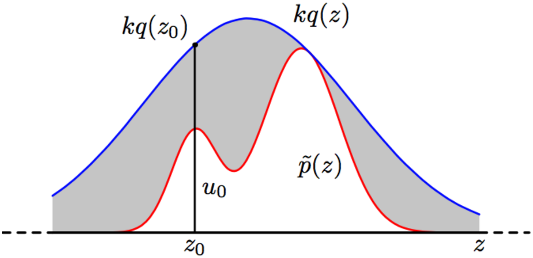
Figure from PRML, Bishop (2006)

Rejection Sampling

- Another approach is to define a simple distribution $q(z)$ and find a k where for all z:

$$kq(z) \geq p(z)$$

- Draw $z_0 \sim q(z)$
- Draw $u \sim Uniform[0, kq(z_0)]$
- Discard if $u > p(z_0)$



Rejection sampling in high dimensions

- Curse of dimensionality makes rejection sampling inefficient
- It is difficult to find a good $q(x)$ in high dimensions and the discard rate can get very high
- For example consider $P(x) = N(0, I)$, where x is D dimensional
- Then for $q(x) = N(0, \sigma I)$ (with $\sigma \geq 1$), the acceptance rate will be σ^{-D}

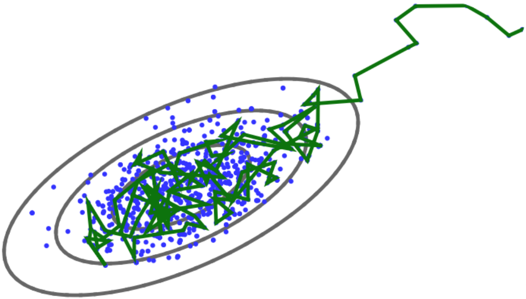
Markov Chains

- A Markov chain is a stochastic model for a sequence of random variables that satisfies the Markov property
- A chain has Markov property if each state is only dependent on the previous state
- It is also called memoryless property
- E.g for the sequence $x^{(1)}, ..., x^{(n)}$ we would have:

$$P(x^{(i)}|x^{(1)}, ..., x^{(i-1)}) = P(x^{(i)}|x^{(i-1)})$$

Markov Chain Monte Carlo (MCMC)

- An alternative to rejection sampling is to generate dependent samples
- Similarly, we define and sample from a proposal distribution
- But now we maintain the record of current state, and proposal distribution depends on it
- In this setting the samples form a Markov Chain



Markov Chain Monte Carlo (MCMC)

- Several variations of MCMC have been introduced
- Some popular variations are: Metropolis-Hasting, Slice sampling and Gibbs sampling
- They differ on aspects like how the proposal distribution is defined
- Some motivations are reducing correlation between successive samples in the Markov chain, or increasing the acceptance rate

Gibbs Sampling

- A simple, general MCMC algorithm
- Initialize $\mathbf{x}$ to some value
- Select an ordering for the variables $x_1, ..., x_d$ (can be random or fixed)
- Pick each variable x_i according to the order and resample $P(x_i|\mathbf{x}_{-i})$
- There is no rejection when taking a new sample

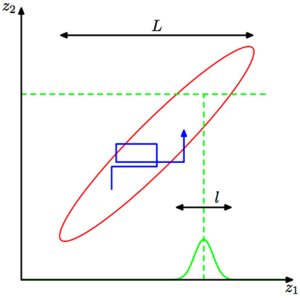
Gibbs Sampling

- For example consider we have three variables $P(x_1, x_2, x_3)$
- At each round t, we take samples from the following distributions:

$$x_1^{(t)} \sim P(x_1 | x_2^{(t-1)}, x_3^{(t-1)})$$

$$x_2^{(t)} \sim P(x_2 | x_1^{(t)}, x_3^{(t-1)})$$

$$x_3^{(t)} \sim P(x_3 | x_1^{(t)}, x_2^{(t)})$$



Monte Carlo methods summary

- Useful when we need approximate methods to solve sums/integrals
- Monte Carlo does not explicitly depend on dimension, although simple methods work only in low dimensions
- Markov chain Monte Carlo (MCMC) can make local moves. By assuming less it is more applicable to higher dimensions
- It produces approximate, correlated samples
- Simple computations and easy to implement

Probabilistic programming languages

- In probablistic programming languages, such as Stan we can describe Bayesian models and perform inference
- Models can be described by defining the random variables, model parameters and their distributions
- Given a model description and a data set, Stan can then perform Bayesian inference using methods such as MCMC