CSC2535
Appendix to Lecture 4: Introduction to Markov Chain Monte Carlo

Andriy Mnih

The problem

- Training and performing inference in probabilistic models often requires computing expectations with respect to complex distributions.
  - Computing these expectations directly is usually infeasible.
  - They can, however, be efficiently approximated by generating samples from the distribution using Markov Chain Monte Carlo (MCMC) and averaging over the samples.
- We might also want to generate samples from a probabilistic model to see what input vectors the model finds probable (i.e. “believes in”).

Monte Carlo

- Basic idea: To estimate an intractable expectation

$$E[f(X)] = \sum_x P(x) f(x)$$

  do the following:

  - Generate *K* samples from *P(x)* (call them $x^1, \ldots, x^K$)
  - Set $E = \frac{1}{K} \sum_{k=1}^{K} f(x^k)$
- *E* is an estimate of the expectation.
  - Using more samples produces more accurate estimates.
    - The samples don't have to be independent, but fewer samples are required to achieve the desired accuracy if they are independent.
  - In the limit of the infinite number of samples, *E* is equal to the expectation being esimated.

Why sampling is difficult

- We usually work with distributions over high-dimensional vectors.
  - In the discrete case, the number of joint configurations is exponential in the number of random variables.
    - Therefore, even enumerating all the possible configurations is infeasible.
  - In the continuous case, rejection sampling or importance sampling can be used (in theory).
    - However, these methods are exponentially inefficeint in high-dimensional spaces.
- This means that sampling from the distribution of interest directly is infeasible in both cases.

Markov Chain Monte Carlo

- Markov Chain Monte Carlo methods do not sample from the distribution of interest *P(x)* directly. Instead, they sample from a sequence of distributions that converges to *P(x)*.
- The state vector *x* stores the current assignment of values to the vector of random variables, which can be viewed as a "sample in the making".
- An MCMC method makes random changes to the state vector using the transition probabilities *T(x, y)*.
  - *T(x, y)* is the probability of going to state *y* given that we are currently in state *x*.
  - These probabilities define a Markov chain that converges to *P(x)*. This means that after sufficienly many transitions the state vector is a sample from *P(x)*.

Transition probabilities

- Transition probabilities are almost never specified explicitly. Instead they are defined algorithmically.
  - Different MCMC methods are simply different ways of making the transitions (and thus defining *T(x,y)*).
    - For example, a transition can be made by generating a proposed new state *y* from some simple distribution "centered" at the current state *x* and accepting or rejecting this proposal based on *P(x)* and *P(y)*.
  - To ensure convergence of the Markov chain to *P(x)*, *T* has to satisfy $P(x) = \sum_y P(y) T(x, y)$ for all *x*.
- Examples of MCMC algorithms used in machine learning are Hybrid Monte Carlo, Gibbs sampling, and various Metropolis algorithms.

Gibbs sampling

- Suppose we have a distribution such that sampling from its conditional distributions $P\left(x_i \middle| \{x_j\}_{j \neq i}\right)$ is easy.
  - This is the case, for example, if the conditionals are multinomial or Gaussian.
- Then we can generate samples from this distribution using Gibbs sampling.
- Gibbs sampling cycles through the state vector, updating one vector component at a time by sampling it from the corresponding conditional distribution.
  - Components can be visited in a determinstic or random order, as long as every component is visited infinitely often (i.e. “once in a while”).
- If some of the components are strongly correlated, the Markov chain can take a long time to converge.

Gibbs sampling algorithm

x = initial value

repeat

for i = 1 to n

x_i = sample from $P\left(x_i \middle| \{x_j\}_{j \neq i}\right)$

until convergence

Finding conditional distributions

- To find the conditional distribution for x_i:
  - Write down the joint distribution *P(x)* for the model
  - Factor out the terms containing x_i
  - Normalize the product of these terms with respect to x_i to get $P(x_i | \{x_j\}_{j \neq i})$