Monte Carlo Matrix Inversion and Reinforcement Learning

Tianyu Li

April 2 2017

1 Preliminary

Finite state Markov decision process is defined as a tuple $\langle \alpha_0, \alpha_\infty, P, S, A \rangle$, where α_0 is the initial weights, α_∞ is the final weight, P describes the transition probability from state $x_t = i$ to $x_{t+1} = j$ with probability p_{ij}, and S, A are set of states and actions respectively. Let r_t be the reward it receives at time step t and γ be the discounting factor, the *evaluation function*, mapping states to their exected, infinite-horizon discounted returns:

$$V_i = E\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} | x_0 = i\} \tag{1}$$

It also has the form as a linear system of equations describing local consistency:

$$V = R + \gamma P V \tag{2}$$

and further more:

$$(I - \gamma P)V = R \tag{3}$$

2 Monte Carlo Methods for Solving Systems of Linear Equations

Intuition

This Monte Carlo approach is motivated by considering the statistical evaluation (expectation) of a simple sum. For example, given a series $\{a_k\}$, let $\{p_k\}$ denote the probability corresponding to a_k, then we can view the sum of all a_k as an expectation of a_k/p_k with the probability being p_k, i.e.: $\sum_k a_k = \sum_k p_k(a_k/p_k)$, where $p_k = Pr(Z = a_k/p_k)$.

Formulation

From equation (3) we can expand the series into the following:

$$V = (I - \gamma P)^{-1} R = R + \gamma P R + \gamma^2 P^2 R + \cdots \tag{4}$$

Note for this equation it is essentially a sum for a infinite series and we now want to evaluate the expectation of the series.

Now analogue to the previous series sum case, we want to find a random variable Z, such that the sequence in equation (4) can be written in an expectation form. Define random variable Z as the discounted reward at time step k with a particular trajectory $i_0 \to i_1 \to \cdots \to i_k$ divided by the probability of reaching to step k with the same trajectory, governed by a Markov process defined over the transition $\tilde{P}$, where k is drawn from a geometric distribution with parameter p_{step}, i.e., $Pr(k\ state\ transitions) = p_{step}^k(1 - p_{step})$. Therefore the probability mass function of Z can be written as:

$$Pr\{Z = \frac{\gamma^k \prod_{j=1}^k p_{i_{j-1},i_j} r_{i_k}}{p_{step}^k (1 - p_{step}) \prod_{j=1}^k \tilde{p}_{i_{j-1},i_j}}\} = \gamma^k (1-\gamma) \prod_{j=1}^k p_{i_{j-1},i_j} \tag{5}$$

By taking the expectation of the random variable Z we thus obtained the expected discounted reward for a particular trajectory $i_0 \to i_1 \to \cdots \to i_k$. Therefore, by taking the sample average of the sampled values of Z, we will then be able to obtain V_i, which is the i-th component of the value function, also the value function starting at state i.

Note here the average should be over all possible trajectories. Sample average of the sampled values of Z is only the expected discounted reward for a particular trajectory. In order to obtain the estimation for the reward of length k, we need to take account of all possible trajectories.

Another similar idea comes from Wasow[1]. Let us define the *truncated* Neumann series as the value function truncated at time step N, which is:

$$\tilde{V} = R + \gamma P R + \gamma^2 P^2 R + \cdots + \gamma^N P^N R \tag{6}$$

This can be viewed as R plus the expected value of the sum of N random variables, where we have:

$$E(Z_k) = \gamma^k P^k R \tag{7}$$

By using the same trick as before, we have:

$$Pr(Z_k = \frac{\gamma^k P^k R}{\tilde{P}^k}) = \tilde{P}^k \tag{8}$$

Therefore, by taking the average of the sampled Z_k we will then be able to obtain the k-th component of the value function.

2.1 Complexity

In [2], curtiss establishes a theoretical comparison of the complexity by this Monte-Carlo method and the traditional linear iterative method, and developed a 95%-confidence interval for the complexity. This complexity involves a factor ζ defined as $||V^{(\infty)} - V_M|| < \zeta||V^{(\infty)} - V^{(0)}||$ for the Monte Carlo method, while $||V^{(\infty)} - V^{(n)}|| < \zeta||V^{(\infty)} - V^{(0)}||$ for the linear method.

For Monte Carlo method, the complexity is $\frac{1}{1-\gamma}(1 + \frac{2}{\zeta^2})$, while for iterative method, by dynamic programming recursion, the complexity is $(1 + \frac{log\zeta}{log\gamma})n^2 + n$.

We can see that Monte Carlo's complexity does not depend on n, while for iterative method it does and it is to the order of 2. This gives us an intuition that for a fixed amount of computation, for large enough n, the Monte Carlo method is likely to produce better estimates. However, one thing worth pointing out is that this analysis is only for computing one component of the solution vector, while for the worst case scenario, we have to compute n componets, adding a n facor to the complexity of Monte Carlo method.

3 Reference

[1]Wasow, Wolfgang R. "A note on the inversion of matrices by random walks." Mathematical Tables and Other Aids to Computation 6.38 (1952): 78-81.
[2]Curtiss, J. H. A theoretical comparison of the efficiencies of two classical methods and a monte carlo method for computing one component of the solution of a set of linear algebraic equations. Courant Institute of Mathematical Sciences, New York University, 1954.
[3]Barto, Andrew G., and Michael Duff. "Monte Carlo matrix inversion and reinforcement learning." Advances in Neural Information Processing Systems (1994): 687-687.