CME307/MS&E311 Suggested Course Project III: Value-Iteration Method for MDP

February 11, 2022

As described in Lectures, Rainforcement Learning (RL) and Markov Decision Processes (MDP) provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker. MDPs are useful for studying a wide range of optimization problems solved via Dynamic Programming (DP), where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957). Modern applications include dynamic planning, reinforcement learning, social networking, and almost all other dynamic/sequential decision game strategy making problems in Mathematical, Physical, Management and Social Sciences.

As talked in class, the MDP problem with m states and total n actions can be formulated as a standard form linear program with m equality constraints and n variables:

$$\begin{array}{lccccc}
\min_{\mathbf{x}} & \sum_{j\in\mathcal{A}_1} c_j x_j + & \ldots & +\sum_{j\in\mathcal{A}_m} c_j x_j & & \\
\text{s.t.} & \sum_{j\in\mathcal{A}_1} (\mathbf{e}_1 - \gamma\mathbf{p}_j)x_j + & \ldots & +\sum_{j\in\mathcal{A}_m} (\mathbf{e}_m - \gamma\mathbf{p}_j)x_j & = & \mathbf{e}, \\
 & \ldots & x_j & \ldots & \geq & 0,\ \forall j,
\end{array} \tag{1}$$

where $\mathcal{A}_i$ represents the set of all actions available in state i, $\mathbf{p}_j$ is the state transition probabilities from state i to all states and c_j is the immediate cost when action j is taken, and $0 < \gamma < 1$ is the discount factor. Also, $\mathbf{e} \in R^m$ is the vector of ones, and $\mathbf{e}_i$ is the unit vector with 1 at the i-th position and zeros everywhere else. Variable $x_j,\ j \in \mathcal{A}_i$, is the state-action frequency or flux, or the expected present value of the number of times in which the process visits state i and takes state-action $j \in \mathcal{A}_i$. Thus, solving the problem entails choosing a state-action frequencies/fluxes that minimize the expected present value sum of total costs. The dual of the LP is

$$\begin{array}{lrcl} \text{maximize}_{\mathbf{y}} & \mathbf{e}^T\mathbf{y} = \sum_{i=1}^m y_i & & \\ \text{subject to} & y_1 - \gamma \mathbf{p}_j^T \mathbf{y} & \leq & c_j,\ j \in \mathcal{A}_1 \\ & & \vdots & \\ & y_i - \gamma \mathbf{p}_j^T \mathbf{y} & \leq & c_j,\ j \in \mathcal{A}_i \\ & & \vdots & \\ & y_m - \gamma \mathbf{p}_j^T \mathbf{y} & \leq & c_j,\ j \in \mathcal{A}_m. \end{array} \tag{2}$$

where y_i represents the cost-to-go value in state i.

1. **Question 1:** Prove that in (1) every basic feasible solution represent a policy, i.e., the basic variables have exactly one variable from each state i. Furthermore, prove each basic variable value is no less than 1, and the sum of all basic variable values is $\frac{m}{1-\gamma}$.

2. **Question 2:** Value Iteration Method: This is a first-order optimization method – starting with any vector $\mathbf{y}^0$, then iteratively update it

$$y_i^{k+1} = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k\},\ \forall i. \tag{3}$$

Prove the contraction result:

$$\|\mathbf{y}^{k+1} - \mathbf{y}^*\|_\infty \leq \gamma \|\mathbf{y}^k - \mathbf{y}^*\|_\infty,\ \forall k.$$

where $\mathbf{y}^*$ is the fixed-point or optimal value vector, that is,

$$y_i^* = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*\},\ \forall i.$$

3. **Question 3:** In the VI method, if starting with any vector $\mathbf{y}^0 \geq \mathbf{y}^*$ and assuming $\mathbf{y}^1 \leq \mathbf{y}^0$, then prove the following entry-wise monotone property:

$$\mathbf{y}^* \leq \mathbf{y}^{k+1} \leq \mathbf{y}^k,\ \forall k.$$

On the other hand, if we start from a vector such that

$$y_i^0 < \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^0\},\ \forall i$$

($\mathbf{y}^0$ in the interior of the feasible region), then prove the entry-wise monotone property:

$$\mathbf{y}^* \geq \mathbf{y}^{k+1} \geq \mathbf{y}^k,\ \forall k.$$

This monotone property has been used in a recent paper (see [SWWY17]) on the VI method using samples.

4. **Question 4:** Rather than go through all state values in each iteration, we modify the VI method, call it RamdomVI: In the kth iteration, randomly select a subset of states B^k and do

$$y_i^{k+1} = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k\}, \ \forall i \in B^k. \tag{4}$$

   In RandomVI, we only update a subset of state values at random in each iteration.

   What can you tell the convergence of the RandomVI method? Does it make a difference with the classical VI method? How is the sample size affect the performance? Use simulated computational experiments to verify your claims.

   Rather than randomly select a subset of all states in each iteration, suppose we build an "influence tree" from a given subset of states, say B, for all sates, denoted by $I(B)$, that are connected by any state in B. Then when states in B are updated in the current iteration, then selected a subset of states in $I(B)$ for updating in the next iteration. Redo the computational experiments using this strategy for a sparsely connected ($\mathbf{p}_j$ is a very sparse distribution vector for each action j) MDP network. In doing so, many unimportant or irrelevant states may be avoided which results a state-reduction.

5. **Question 5:** Here is another modification, called CyclicVI: In the kth iteration do

   - Initialize $\tilde{\mathbf{y}}^k = \mathbf{y}^k$.
   - For $i = 1$ to m

$$\tilde{y}_i^k = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \tilde{\mathbf{y}}^k\} \tag{5}$$

   - $\mathbf{y}^{k+1} = \tilde{\mathbf{y}}^k$.

   In the CyclicVI method, as soon as a state value is updated, we use it to update the rest of state values.

   What can you tell the convergence of the CyclicVI method? Does it make a difference with other VI methods? Use simulated computational experiments to verify your claims. How is this cyclic method related to the method at the bottom of Question 4?

6. **Question 6:** In the CyclicVI method, rather than with the fixed cycle order from 1 to m, we follow a random permutation order, or sample without replacement to update the state values. More precisely, in the kth iteration do

   0. Initialize $\tilde{\mathbf{y}}^k = \mathbf{y}^k$ and $B^k = \{1, 2, ..., m\}$
   1. − Randomly select $i \in B^k$

      −

$$\tilde{y}_i^k = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \tilde{\mathbf{y}}^k\} \tag{6}$$

- remove i from B^k and return to Step 1.

3. $\mathbf{y}^{k+1} = \tilde{\mathbf{y}}^k$.

We call it the randomly permuted CyclicVI or RPCyclicVI in short

What can you tell the convergence of the RPCyclicVI method? Does it compare with other VI methods? Use simulated computational experiments to verify your claims.

In this project, you may generate a Maze Game in 2D by assign actions with different costs and probability distributions to test your algorithms.

Question 7: Tic-Tac-Toe Game: In this problem, we want to develop the optimal strategy for the cross-player. We assume that the cross-player plays first, and the opponent is a random player. That is, the opponent puts a circle in an empty square with equal probability in each round. Please formulate the 3×3 tic-tac-toe game as an MDP problem and find the optimal policy.

In addition, what can you tell about the optimal first step for the cross player in the 4×4 tic-tac-toe game?

References

[Ber13] Dimitri P Bertsekas. *Abstract dynamic programming.* Athena Scientific, Belmont, MA, 2013.

[HMZ13] Thomas Dueholm Hansen, Peter Bro Miltersen, and Uri Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *J. ACM*, 60(1):1:1–1:16, February 2013.

[How60] Ronald A. Howard. *Dynamic programming and Markov processes.* The MIT press, Cambridge, MA, 1960.

[LDK95] Michael L Littman, Thomas L Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 394–402. Morgan Kaufmann Publishers Inc., 1995.

[Put14] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

[Sch13] Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. In *Advances in Neural Information Processing Systems*, pages 386–394, 2013.

[SWWY17] Aaron Sidford, Mengdi Wang, Xian Wu, Yinyu Ye. Variance Reduced Value Iteration and Faster Algorithms for Solving Markov Decision Processes. SODA2018 and https://arxiv.org/abs/1710.09988

[Ye11] Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.