B9140 Dynamic Programming & Reinforcement Learning Lecture number - Oct 3

Asynchronous DP, Real-Time DP and Intro to RL

Lecturer: Daniel Russo *Scribe: Kejia Shi, Yexin Wu*

Today's lecture looks at the following topic:

- Classical DP: asynchronous value iteration
- Real-time Dynamic Programming: RTDP (closest intersection between the classical DP and RL)
- RL: overview; look at policy evaluation; Monte Carlo (MC) vs Temporal Difference (TD)

1 Classical Dynamic Programming

1.1 Value Iteration

Algorithm 1: Value Iteration

```
Input: $J \in \mathbb{R}^n$
for $k = 0, 1, 2, ...$ do
    for state space $i = 1, 2, ..., n$ do
        $J'(i) = (TJ)(i)$
    end
    stop if ...
    $J = J'$
end
```

1.2 Gauss-Seidel VI

Gauss-Seidel Value Iterations is the most commonly used variant of asynchronous value iteration. This method updates one state at a time, while incorporating into the computation the interim results.

Algorithm 2: Gauss-Seidel Value Iteration

```
Input: $J \in \mathbb{R}^n$
for $k = 0, 1, 2, ...$ do
    for $i = 1, 2, ..., n$ do
        $J(i) = (TJ)(i)$    # not J' here
    end
    stop if (...some stopping criterion is met)
end
```

Proposition 1. *Asynchronous Value Iteration Consider an algorithm that starts with $J_0 \in \mathbb{R}^{|X|}$ and makes updates at a sequence of states $(x_0, x_1, x_2, ...)$, where for any k,*

$$J_{k+1}(x) = \begin{cases} (TJ_k)(x) & \text{if } x_k = x; \\ J_k(x) & \text{otherwise} \end{cases}$$

If each state is updated infinitely often, then $J_k \to J^*$ as $k \to \infty$.

Proof. **Case 1:** If $J_0 \leqslant TJ_0$, then we have

$$J_0 \leqslant TJ_0 \leqslant T^2 J_0 \leqslant ... \leqslant T^k J_0 \leqslant ... \leqslant J^*.$$

We also have $J_0 \leqslant TJ_0 = J_1$, since our assumption. We change only one state of them and get it bigger with other states untouched.

By monotonicity, we have $J_1 \leqslant TJ_0 \leqslant TJ_1$. Repeating this we have $J_2 \geqslant J_1$ and $J_2 \leqslant TJ_2$.

Inductively, $J_0 \leqslant J_1 \leqslant J_2 \leqslant ... \leqslant J_k$ and $J_k \leqslant TJ_k$, which implies $J_k \leqslant J^*$.

Let $J_\infty = \lim_{k\to\infty} J_k$. Certainly we have $J_\infty = TJ_\infty$. (run forever, get arbitrarily close)

Note: $J_{k+1}(x_k) - J_k(x_k) = TJ_k(x_k) - J_k(x_k)$: RHS: Bellman gap; LFS: goes to 0 when $k \to \infty$.

Case 2: If $J_0 \geqslant TJ_0$, then similar to case 1, just flip everything around.

Case 3: For any J_0, take $\delta > 0$, $e = (1, 1, ..., 1)$, we have

$$J^- \equiv J^* - \delta e \leqslant J_0 \leqslant J^* + \delta e \equiv J^+.$$

Let J_k^-, J_k^+ be the output of asynchronous value iteration applied to J^- and J^+.

Monotonicity gives us $J_k^- \leqslant J_k \leqslant J_k^+$. One can show that $J^- \leqslant TJ_-$, $J^+ \leqslant TJ^+$.
So $J^+ \to J^*$ and $J^- \to J^*$, which means $J_k \to J^*$. □

Note: Another way to prove is through contractions. Here we use monotonicity. Intuitively, everywhere we update, it's a right direction with its momentum maintained.

Why we do this?

- Distributed computation: most DPs have too many states, the real case is that you can't update states in a line. Some are slower than the others. (Communication delays, see textbook Chap 2.6)
- Form the basis of learning from interaction: most RL algos look at TJ_k at x_k.

2 Real-Time Dynamic Programming

This is a variant of asynchronous value iteration where states are sampled by an agent that, at all times, makes decisions greedily under her current guess of the value function. The following proposition gives a first

Algorithm 3: Real-Time Dynamic Programming

```
Input: $J_0$
1 for $k = 0, 1, 2, ...$ do
2     observe $x_k$ the current state
3     play $u_k = \arg\min_u g(x, u) + \gamma \Sigma_{x'} P(x, u, x') J_k(x')$
4     update $J_{k+1}(x) = TJ_k(x)$ if $x = x_k$; otherwise $J_{k+1}(x) = J_k(x)$
5 end
```

convergence result for RTDP. For details see [BBS95]. Note that unlike the previous case, this proposition does not require that *every* state is visited infinitely often.

Proposition 2. *Under RTDP, J_k converges to some vector J_∞, $J_\infty(x) = TJ_\infty(x)$ at all x visited i.o.*

Note: This proposition is unsatisfying since it does not imply convergence to J^*. To guarantee convergence to J^* using the previous result, we would generally need to ensure each state is updated infinitely often. However, it's not clear the goal should be to find J^*, at all. Instead, it may be sufficient that eventually the action's chosen by the agent are optimal.

Fix 1: We may add randomness to action selection in an effort to ensure that every state is visited infinitely often. If each state is reachable from any other state, this will work, and is enough to ensure asymptotic convergence of J_k to J^*. However, it may take an exceptionally long time to reach visit certain states. One reason is that random exploration can be very inefficient, and may take such a strategy time exponential in the size of the state space to reach a state that could be reached efficiently under a particular policy. But it may also be the case that some states are almost impossible to reach under any policy. It seems that these states are essentially irrelevant to minimizing discounted costs.

Fix 2: Start optimistic. Assume $J_0 \leqslant TJ_0$. This can be ensure by picking a J_0 that has very small values (e.g. if expected costs are non-negative, it suffices to take $J_0 = 0$)
From the above we have $J_0 \leqslant J_1 \leqslant J_2 \leqslant ...$ and so on. (This means we always believe expected costs are lower than is possible, and updates always consist of raising our expectations about expected costs.)

Proposition 3. *If $J_0 \leqslant TJ_0$, there exists a (random) time K after which all actions are optimal. That is $u_k = \mu^*(x_k)$ for any $K > k$.*

Note $J_k \leqslant J_{k+1} \leqslant ... \leqslant J^*$, so $J_k \to J_\infty$. (bounded monotone sequences converge.) This implies the policy also converges, $\mu_k \to \mu_\infty$:
The following argument holds for any sample path, (omitting a set of measure zero).

- Let V be the set of states visited i.o. (this exists and is nonempty for any sample path).
- The agent's eventual policy μ_∞ must have zero probability of leaving V. (Precisely, $P(x, \mu_\infty(x), x') = 0$ for all $x' \in V^c$). Otherwise, with probability 1 the agent would *eventually* leave V.
  - It is as if the agent plays on a sub-MDP where all sates in V^c have been deleted, and all actions that might reach V^c are deleted.
  - As a result, the estimates $J_\infty(x)$ are accurate for all $x \in V$. That is $J_\infty(x) = J_{\mu_\infty}(x)$, the agent's true expected cost-to-go under the current policy.
- How do we conclude the actions chosen by μ_∞ on V are optimal?
  - Consider some action $u \neq \mu_\infty(x)$. Since u is not chosen, it must be that

    $$g(x, \mu_\infty(x)) + \gamma \sum_{x' \in X} P(x, \mu_\infty \infty, x') J_\infty(x') \leq g(x, u) + \gamma \sum_{x' \in X} P(x, u, x') J_\infty(x')$$

    But by he discussion above, the left hand side equals $J_\mu(x)$, and so the true cost of following μ is less than estimated cost of playing u with cost-to-go under J_∞.
  - From here, the key is that $J_\infty(x) \leq J^*(x)$ for all $x \in V^c$. The agent may not have an accurate estimate of the value function, but is *optimistic*, in sense that she underestimates costs at V^c.
  - In particular, for $x \in V$, and $u \neq \mu_\infty(x)$

    $$g(x, u) + \gamma \sum_{x' \in X} P(x, u, x') J_\infty(x') \leq g(x, u) + \gamma \sum_{x' \in X} P(x, u, x') J^*(x')$$

    Therefore, the cost of playing u with cost-to-go estimate J_∞ *underestimates* the real cost from playing u, even if actions thereafter are chosen optimally. If $J_\mu(x)$ is below this under-estimate, u cannot be optimal.

3 Reinforcement Learning

Now we don't know the transition probabilities of the the MDP, and must use sample data to reason about cost-to-go functions and learn effective policies.

3.1 Challenges

Reinforcement learning requires overcoming several substantial challenges:

1. Curse of Dimensionality
2. Learning from Interaction
3. Learning with Delayed Consequences

Challenge 1. Curse of Dimensionality Suppose a state is a vector of 10 state variables, This appears to be a relatively compact representation, but if each variable takes on a finite number of possible values, say 100, then the there are 100^{10} states in the MDP. Another simple example is the game of tetris. (Such episodic game will end.) Suppose we have a 10*10 grid, then we instantly have a state space with 2^{100} states.

In the second example, how do we model such MDP? A person watching a someone play tetris is unlikely to have every seen a tetris game with that exact configuration of pieces. Nevertheless, they can probably tell quickly if the player is in a good shape by looking at the current board. This is because the observer is able to recognize abstract features of the screen configurations that are relevant to predicting future payouts, and this lets them generalize from previously seen states to reason about the current state.
We might think about approximating value to go as a linear combination of basis functions. For example, suppose $J^*(x) \approx \phi(x)^T\theta$, then the state $\Phi(x) \in \mathbb{R}^{20}$ is defined as
$\phi_1, ..., \phi_{10}$ column heights,
$\phi_{11}, ..., \phi_{19}$ absolute inter-column height differences ϕ_{20} max height.

It is true that $(\phi_{11}, \ldots, \phi_{20})$ are functions of the first ten features, but they are not linear combinations of them. Including them allows for a richer set of functions to be approximated as linear combination of the features

As another example, let us reconsider the inventory control example considered in the first lecture. Suppose now that order arrive after a lead time of L periods from when they are placed. During interim, customers arrive and purchase inventories, and the firm may wish to place another order before the already ordered inventory arrives. Holding and ordering inventory is costly, but there is also a shortage cost the inventory/seller is charged when a customer demand is unmet due to a stockout. How do we describe the system? The system is not Markovian if the state only encodes how much inventory is on hand and how much has yet to be delivered. Instead, we need to know how inventory is on hand, how much is arriving tomorrow, how much is arriving the next day, the day after that, and so on. Therefore we describe the state with $L+1$ components. Then the total states is exponential in L again. And we can never solve this!

The problem is: how do you derive an effective policy, or construct value function estimates, without having to perform computation at all all possible states?

Challenge 2. Learning from Interaction This part is about how to gather data effectively and will be taught during the final part of the class.

Return to the Tetris example. Now we have a simulator that allows us to evaluate the reward accumulated and state dynamics under any policy. We could presumably learn from the data generated by such a simulator, but can only learn from the scenarios our algorithm actually generates. How should we select actions?

The classic RL example of river swim highlights some of subtelties in designing exploration schemes. Suppose the swimmer starts at state 1 and across the lake there's an island. There are $n-1$ states along the river $2, 3, ..., n$ and the island is labeled as $n+1$. The swimmer can either go right (advance the state) or left (decrease the state) by one step in each period, but of course reaching the island requires going right n consecutive times.
In reality, the swimmer can stick around at starting point or the first several states, unless he/she decides to go to island purposefully. But in most of the time, we don't know about if there's a purpose.

Challenge 3: Learning with Delayed Consequences This is what we are going to learn during the next few classes. It is difficult to use historical data to evaluate whether a particular action is effective in a particular state, since the action influence not just the reward in the current period, but indirectly, the rewards in future periods through impacting the next state.

3.2 Policy Evaluation

First challenge: Given a policy μ, we sample data from μ. How do we estimate J_μ?
we can still start simple by considering the following setup :

- A look-up table representation of J_μ,
- We observe repeated episodes of an episodic (or indefinite horizon) MDP
  - State space $x \cup \{t\}$,
  - The terminal state t is costless and absorbing: when it's over it's over.
  - Assume t is reached with probability 1 under μ.

Since $J_\mu(t) = 0$, it's easier to work with $(J_\mu(x))_{x\in X}$. The Bellman equation is as follows, (notice there's no discount factor here .)

$$(T_\mu J)(x) = g_\mu(x) + \Sigma_{x'\in X} P_\mu(x, x')J(x').$$

Suppose we have a data set generated by applying μ for N independent episodes. After one episode terminates, the MDP resets either to a deterministic state or to a state drawn independently from some probability distribution. At episode $n \in \{1, \ldots, N\}$, we observe $(x_0^{(n)}, c_0^{(n)}, x_1^{(n)}, c_1^{(n)}, ...x_{\tau_n}^{(n)}, c_{\tau_n}^{(n)})$. Here c is the instantaneous cost/reward - the one step accumulation. τ_n is the termination time of episode n.

How should we estimate $J_\mu(x)$? This remains a open question, but two the RL literature generally considers two types of methods: Monte Carlo and Temporal Difference. We introduce MC now.

Monte Carlo
The essential idea is to evaluate $J_\mu(x)$ by taking a sample average of the returns after visiting x over all such episodes that visit x. In particular, *first-visit* Monte carlo works as follows: if x is visited in episode n, and k is the first period in which x is visited, then calculate $\Sigma_{i=k}^{\tau_n} c_i^n$. By the strong Markov property, $\Sigma_{i=k}^{\tau_n} c_i^n$ has expected value $J_\mu(x)$. Averaging over all such episodes produces an unbiased estimate.

Consider another example from [SB2017]. There is a MDP with 2 states A and B. Each episode starts with random initialization, so some episodes begin at A and others begin at B. You observe 8 data points $\{(A, 0, B, 0), (B, 1), (B, 1), (B, 1), (B, 1),$

$(B, 1), (B, 1), (B, 0)\}$. Using MC to calculate $\hat{J}_\mu(A)$ and $\hat{J}_\mu(B)$: $\hat{J}_\mu(B) = \frac{3}{4}$, $\hat{J}_\mu(A) = 0$.

The state A is only visited in one epsiode, and the fact that the subsequent state B generated a reward of 0 in this state may be spurious.
Another way to approach the problem is to think that from A the agent often incurs zero immediate cost and then transitions to B. Borrowing from Bellman's equation, we calculate $\hat{J}_\mu(A) = 0 + \hat{J}_\mu(B) = \frac{3}{4}$.

Note: Some literature considers MC unbiased but with higher variance, and TD to be biased but have higher variance. In this case, TD is also unbiased. More analysis and comparisons are coming.

References

[BBS95] A. G. Barto S. J. Bradtke and S. P. Singh, "Learning to act using real-time dynamic programming" *Artificial intelligence, 1995, 72(1-2), 81-138.*

[SB2017] R. S. Sutton A. G. Barto and S.P. Singh, "Reinforcement Learning: An Introduction. Second Edition (Draft)" *(2017)*