IEOR 8100: Reinforcement learning

Lecture 4: Approximate dynamic programming

By Shipra Agrawal

Deep Q Networks discussed in the last lecture are an instance of approximate dynamic programming. These are iterative algorithms that try to find fixed point of Bellman equations, while approximating the value-function/Q-function a parametric function for scalability when the state space is large. Similar to Q-learning, function approximation versions of TD-learning and value-iteration can be obtained. Here, we will examine some versions of these algorithms in terms of their convergence properties.

1 TD(0) and TD(1)

In TD-learning we are only interested in evaluating a policy given π – you can think of this a special case of Q-learning on an MDP with only one action available for every state. Therefore, we are interested in computing value function $V^\pi(s)$ for all s. In the function approximation version, we learn a parametric approximation $\tilde{V}_\theta(s)$. For example, the function $\tilde{V}_\theta(s,a)$ could simply be a linear function in θ and features $\tilde{V}_\theta(s) = \theta_0 f_0(s) + \theta_1 f_1(s) + \ldots + \theta_n f_n(s)$, or output of a deep neural net. The parameter θ can map the feature vector $f(s)$ for any s to its value-function, thus providing a compact representation. Instead of learning the $|S|$ dimensional value vecor TD-learning algorithm will learn the parameter θ.

Here is a least squares approximation based modification (similar to Q-learning with function approximation) of TD(0) and TD(1) respectively.

Algorithm 1 Approximate TD(0) method for policy evaluation

1: Initialization: Given a starting state distribution D_0, policy π, the method evaluates $V^\pi(s)$ for all states s.
Initialize θ.
Initialize episode $e = 0$.
2: **repeat**
3: $e = e + 1$.
4: Set $t = 1, s_1 \sim D_0$. Choose step sizes $\alpha_1, \alpha_2, \ldots$.
5: Perform TD(0) updates over an episode:
6: **repeat**
7: Take action $a_t \sim \pi(s_t)$. Observe reward r_t, and new state s_{t+1}.
8: (*) **Let** $\delta_t := r_t + \gamma \hat{\tilde{V}}_\theta(s_{t+1}) - \tilde{V}_\theta(s_t)$.
9: (*) **Update** $\theta \leftarrow \theta + \alpha_t \delta_t \nabla_\theta \tilde{V}_\theta(s_t)$.
10: $t = t + 1$
11: **until** until episode terminates
12: **until** termination condition

Recall in $TD(1)$, the target is computed using infinite lookahead as $target(s_{t+1}) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1}, \ldots =: \mathcal{R}_t$. Effectively, the only difference is to replace Step 8 in TD(0) by:

TD(1) algorithm

8: Let $\delta_t := \mathcal{R}_t - \tilde{V}_\theta(s_t)$.

In tabular case, both these algorithms converge to the true value (they are both instances of the stochastic approximation method); we saw that depending on the MDP, one may be more efficient than the other. However, these tradeoffs change when we use function approximation. Below is any example where these methods do converge *but* some times to a function approximation that is far from true value function! The examples show that this can happen even if there exists some value of θ for which the function approximation $\tilde{V}_\theta$ is close to V^π.

Counter-example TODO: Example covered in class: from "A Counterexample to Temporal Differences Learning", Bertsekas 1995. This example also applies to Q-learning.

2 Fitted Value iteration

Let's take a step back and consider an easier case: the problem of solving a known MDP instead of reinforcement learning. That is, we know the MDP model or can simulate the MDP model in any state and action. If state space was small, we could use value iteration for computing optimal policy. However, to handle large state space, we consider approximate value iteration, also known as 'fitted' value iteration.

Recall that the tabular value iteration algorithm starts with an initial value vector v^0, and in every iteration i, simply performs the update $v^i = Lv^{i-1}$ (see (1)).

Tabular value-iteration (Recall)

1. Start with an arbitrary initialization $\mathbf{v}^0$. Specify $\epsilon > 0$
2. **Repeat** for $k = 1, 2, \ldots$ **until** $\|\mathbf{v}^k(s) - \mathbf{v}^{k-1}(s)\|_\infty \leq \epsilon \frac{(1-\gamma)}{2\gamma}$:
   - for every $s \in S$, improve the value vector as:

$$\mathbf{v}^k(s) = [Lv^{k-1}](s) := \max_{a \in A} R(s,a) + \gamma \sum_{s'} P(s,a,s') v^{k-1}(s'), \tag{1}$$

In the fitted value iteration, we replace this to a scalable update, by fitting a compact representation $\tilde{V}_\theta$ to v_i in every iteration.

Fitted value-iteration

1. Start with an arbitrary initialization θ^0.
2. **Repeat** for $k = 1, 2, 3, \ldots$:
   - Evaluate $[L\tilde{V}_{\theta^{k-1}}](s)$ on a subset $s \in S_0$.
   - Use some regression technique to find a θ to fit data $(\tilde{V}_\theta(s), L\tilde{V}_{\theta^{k-1}}(s))_{s \in S_0}$. Set this θ as θ_k.

Example 1: Sampling + Least squares fitting [Munos and Szepesvári, 2008]. (This assumes number of actions is small, number of states is large. And, the MDP model is large but can be simulated for any arbitrary state s and action a).
In state k:

1. Sample states $X_1, X_2, \ldots, X_N$ from the state space S, using some distribution μ.
2. For each action a, and state X_i, take multiple samples of next state and rewards $\{Y_{i,a,j}, R_{i,a,j}\}_{j=1,\ldots,M}$.
3. Approximate $[L\tilde{V}_{\theta^{k-1}}](s)$ for $s \in \{X_1, X_2, \ldots, X_N\}$ as

$$\text{target}_i = \max_{a \in A} \frac{1}{M} \sum_{j=1}^{M} \left(R_{i,a,j} + \gamma \tilde{V}_{\theta^{k-1}}(Y_{i,a,j}) \right), \forall i = 1, \ldots, N$$

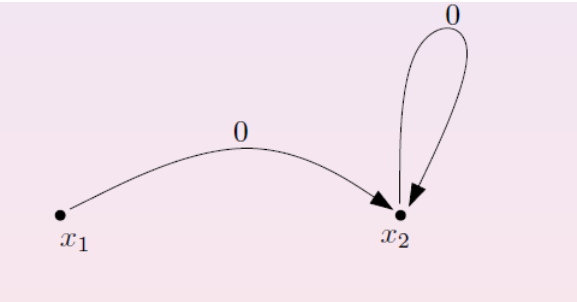
4. Use least squares new θ as:

$$\theta_k := \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} (\tilde{V}_\theta(i) - \text{target}_i)^2$$

Interestingly, it matters what regression technique is used. The example below shows that least square minimization doesn't always work!

2.1 Counter-example for least-square regression [Tsitsiklis and van Roy, 1996]

An MDP with two states x_1, x_2, 1-d features for the two states: $f_{x_1} = 1, f_{x_2} = 2$. Linear Function approximation with $\tilde{V}_\theta(x) = \theta f_x$.



$$\begin{aligned}
\theta_k &:= \arg\min_\theta \frac{1}{2}(\theta - \text{target}_1)^2 + (2\theta - \text{target}_2)^2 \\
&= \arg\min_\theta \frac{1}{2}(\theta - \gamma\theta^{k-1} f_{x_2})^2 + (2\theta - \gamma\theta^{k-1} f_{x_2})^2 \\
&= \arg\min_\theta \frac{1}{2}(\theta - \gamma 2\theta^{k-1})^2 + (2\theta - \gamma 2\theta^{k-1})^2
\end{aligned}$$

$$(\theta - \gamma 2\theta^{k-1}) + 2(2\theta - \gamma 2\theta^{k-1}) = 0 \Rightarrow 5\theta = 6\gamma\theta^{k-1}$$

$$\theta_k = \frac{6}{5}\gamma\theta_{k-1}$$

This diverges if $\gamma \geq 5/6$.

2.2 Convergence of non-expansive approximations

Operator view of Fitted value-iteration. A more general way to interpret fitted value iteration is that you have an operator M_A that takes a value vector v^i and projects it into the function space formed by functions of form $\tilde{V}_\theta$.

1. Start with an arbitrary initialization $V^0, \tilde{V}_{\theta^0} := M_A(V^0)$.
2. **Repeat** for $k = 1, 2, 3, \ldots$:
   - $\tilde{V}_{\theta^i} = M_A \circ L\tilde{V}_{\theta^{i-1}}$.

Now, to match the description earlier, consider operator M_A defined as follows: Fit a $\tilde{V}_\theta$ to $L\tilde{V}_{\theta^{i-1}}$ by comparing its values on a subset S_0 of states, using a regression technique. And, then return this $\tilde{V}_\theta$ as new function $\tilde{V}_{\theta^i}$ in the output space of M_A. Thus, M_A is effectively an approximation operator.

Equivalently,

1. Start with an arbitrary initialization v^0.
2. **Repeat** for $k = 1, 2, 3, \ldots$:
   - $v^i = (L \circ M_A)v^{i-1}$.

(In an efficient implementation, $u^{i-1} = M_A v^{i-1}$ probably has a more compact representation, so the first view may be better for implementation)

The above view allows us to view fitted value iteration as just value iteration with a different operator: $v^i = Lv^{i-1}$ is replaced by $\tilde{V}^i = (M_A \circ L)\tilde{V}^{i-1}$. Therefore, as long as the new operator $(M_A \circ L)$ is also γ-contraction,

the results proven earlier for convergence of value iteration will hold. A sufficient condition is that the operator M_A is non-expansive:

$$\|M_A v - M_A v'\|_\infty \leq \|v - v'\|_\infty$$

Then,

$$\begin{aligned} \|(M_A \circ L)v - (M_A \circ L)v'\|_\infty &\leq \|Lv - Lv'\|_\infty \\ &\leq \gamma \|v - v'\|_\infty \end{aligned}$$

Similarly, in the other view, $v^i = Lv^{i-1}$ is replaced by $v^i = (T \circ M_A)v^{i-1}$ which is also a γ contraction. so, v^i converges.

Sufficient condition for convergence: Averager [Gordon, 1995].

Lemma 1. *The opertor M_A is non-expansive if it is an averager, that is,*

$$[M_A v^k](s) = \sum_{i \in S} w_{i,s} v^k(i) + w_{0,s}$$

where

$$\sum_i w_{i,s} + w_{0,s} = 1, w_{i,s} \geq 0$$

Proof. This is non-expansive because

$$M_A v = Wv + w_0$$

$$\|M_A v - M_A v'\|_\infty = \max_s |\sum_{i \in S} w_{i,s}(v(i) - v'(i))| \leq \max_s \max_i |v(i) - v'(i)| = \|v - v'\|_\infty$$

□

2.3 Converges to what

The fitted value iteration converges with exponential rate under above condition, but to what? Let v^i converges to fixed point U^*. That is, $U^* = L(M_A(U^*))$. We show that under the contraction properties, U^* is as close as it can be to V^* (the optimal value function), in the sense that:

$$\|U^* - V^*\|_\infty \leq \frac{\gamma}{1-\gamma} \|M_A V^* - V^*\|_\infty$$

The proof is as follows.

$$\begin{aligned} \|U^* - V^*\| &= \|L(M_A(U^*) - LV^*\| \\ &\leq \gamma \|M_A U^* - V^*\| \\ \|M_A U^* - V^*\| &\leq \|M_A U^* - M_A V^*\| + \|M_A V^* - V^*\| \\ &\leq \|U^* - V^*\| + \|M_A V^* - V^*\| \\ \|U^* - V^*\| &\leq \gamma \|U^* - V^*\| + \gamma \|M_A V^* - V^*\| \\ \|U^* - V^*\| &\leq \frac{\gamma}{(1-\gamma)} \|M_A V^* - V^*\| \end{aligned}$$

2.4 What about least squares?

TODO: PAC bounds in [Munos and Szepesvári, 2008].

(Book Chapter reference: Chapter 3 of Szepesvári [1999])

3 Discussion

What next? So far the methods we have seen are based on the value function approach, where all the function approximation effort went into scalable estimation of value function. The action selection policy was represented implicitly as greedy policy with respect to the estimated value functions. This approach has several limitations. First, it is oriented toward finding deterministic policies, whereas the optimal policy is often stochastic, selecting different actions with specific probabilities. Second, an arbitrarily small change in the estimated value of an action can cause it to be, or not be, selected. Such discontinuous changes have been identified as a key obstacle to establishing convergence assurances for algorithms following the value-function approach (Bertsekas and Tsitsiklis, 1996). For example, Q-learning, Sarsa, and dynamic programming methods have all been shown unable to converge to any policy for simple MDPs and simple function approximators (Gordon, 1995, 1996; Baird, 1995; Tsitsiklis and van Roy, 1996; Bertsekas and Tsitsiklis, 1996). This can occur even if the best approximation is found at each step before changing the policy, and whether the notion of "best" is in the mean-squared-error sense or the slightly different senses of residual-gradient, temporal-difference, and dynamic-programming methods.

Next, we will look at policy gradient methods that use function approximation for directly approximating the optimal policy and eliminate some of the above limitations.

References

Geoffrey J. Gordon. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*, pages 261 – 268. Morgan Kaufmann, San Francisco (CA), 1995.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *J. Mach. Learn. Res.*, 9:815–857, June 2008. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1390681.1390708`.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, 1999. URL `http://old.sztaki.hu/ szcsaba/papers/RLAlgsInMDPs-lecture.pdf`.

John N. Tsitsiklis and Benjamin van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1):59–94, Mar 1996.