Advanced Topics in Reinforcement Learning

Lecture 12: Function Approximation for On-policy Prediction

Josiah Hanna
University of Wisconsin — Madison

Announcements

- Homework 3 due Thursday of next week.
- Everyone should have received project proposal feedback.
- Begin reading chapter 11 for next week.
- Midterm survey and evaluation.

Project RL Algorithms

- We probably have not covered algorithms you will want to use in your final project.
- We have covered the fundamental ideas of more advanced algorithms.
  - Q-learning —> Deep Q-Networks (DQN)
  - Expected SARSA —> Actor-Critic Methods —> DDPG, TD3, SAC
  - Monte Carlo Policy Iteration —> Policy Gradient Methods —> Proximal Policy Optimization

This Week and Next

- Today: overview of function approximation for on-policy prediction.
- Thursday: feature construction methods and on-policy control.
- Next week: off-policy prediction with function approximation.

Function Approximation in RL

- How different from the tabular case?
  - Generalize value estimates across similar states.
- What is the benefit?
  - May only visit any given state once.
  - Too many states to store an individual value estimate for each.
- What do we lose?
  - Accurate approximation everywhere.
  - The policy improvement theorem.

Function Approximation in RL

- Form of the value estimate:

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

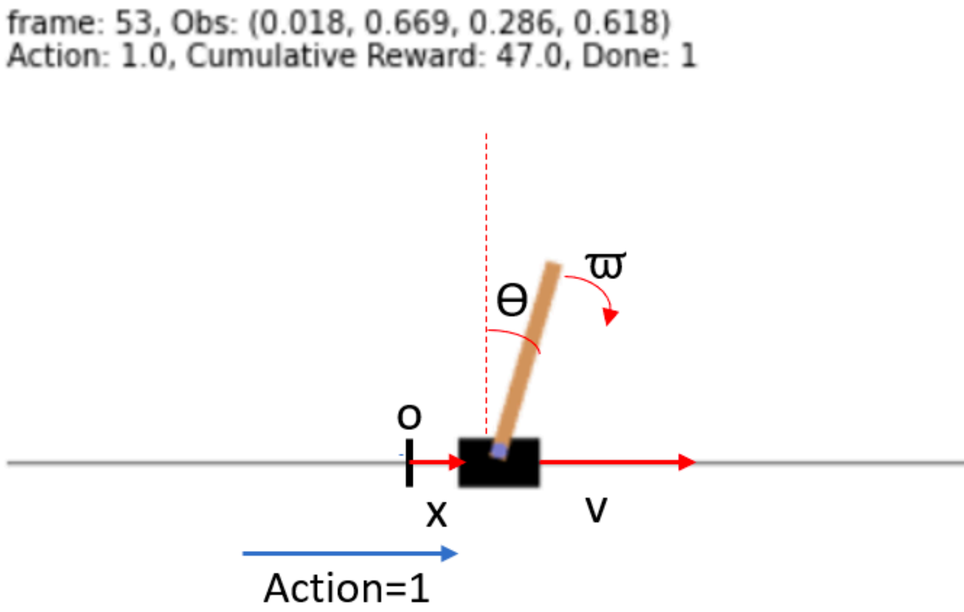
- $\mathbf{w} \in \mathbb{R}^d$ with $d \ll |\mathcal{S}|$.
- Changing $\mathbf{w}$ changes the value estimate at multiple states.
- (Tabular methods are a special case with $d = |\mathcal{S}|$).

Linear Function Approximation

- Assume value estimate is a linear function of state features.
- $$\hat{v}(s, \mathbf{w}) = \mathbf{w}^\top x(s) = \sum_{i=1}^{d} w_i x_i(s)$$
- The features, $x_i(s)$, can be non-linear functions of state variables.
  - Expressive choices for $\mathbf{x}(s)$ make linear methods more powerful than they first appear.

Linear FA Example

- What is $\mathbf{x}(s)$?
  - List of state variables: (x, v, θ, ω)
  - Any static function of the state variables.
- Suppose $\mathbf{x}(s) = (x, v, \theta, \omega)$.
  - What can you say about the value estimates as w_1 increases?



https://towardsdatascience.com/how-to-beat-the-cartpole-game-in-5-lines-5ab4e738c93f

The Prediction Objective

- As you saw in the reading, we have the following objective:

$$\overline{VE}(\mathbf{w}) = \sum_{s \in \mathcal{S}} \mu(s) \left[v_\pi(s) - \hat{v}(s, \mathbf{w}) \right]^2$$

- Note: the policy is fixed because we are just considering prediction.
- Why this objective?
- Do we ever know how well we are doing?
  - μ and v_π are unknowns.

(Stochastic) Gradient Descent

- So far we have seen how to represent value estimates when $d \ll |\mathcal{S}|$ and how to evaluate different choices of $\mathbf{w}$.
- Now, how to select $\mathbf{w}$ that minimizes prediction error.
- Assuming we visit states in proportion to μ, the following update moves us towards minimal prediction error:
  - $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(v_\pi(S_t) - \hat{v}(s, \mathbf{w}_t)) \nabla \hat{v}(S_t, \mathbf{w}_t)$
- This is the same update used for gradient-based linear regression — it's just supervised learning!

(Stochastic) Gradient Descent

- Unlike supervised learning, we don't know the targets, $v_\pi(s)$.
- Instead, we use a noisy target, U_t:
  - $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(U_t - \hat{v}(s, \mathbf{w}_t)) \nabla \hat{v}(S_t, \mathbf{w}_t)$
- Monte Carlo: $U_t \leftarrow G_t$
- TD(0): $U_t \leftarrow R_t + \gamma \hat{v}(s, \mathbf{w})$

Example 9.1

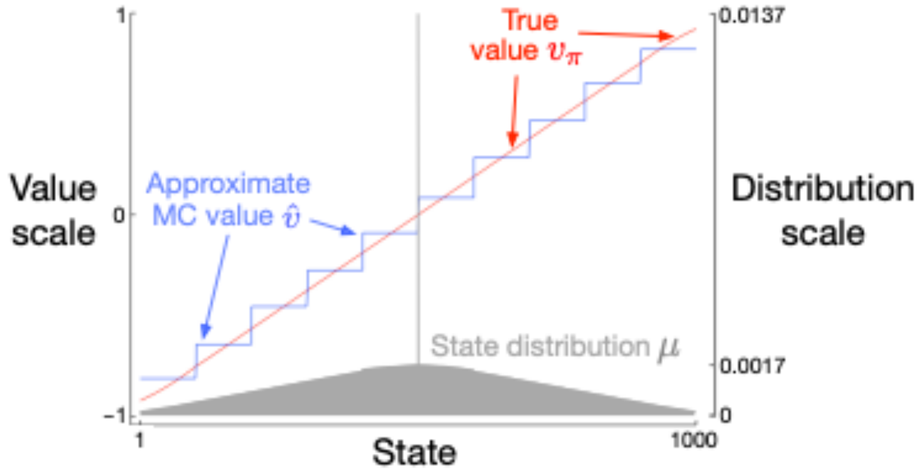


Figure 9.1: Function approximation by state aggregation on the 1000-state random walk task, using the gradient Monte Carlo algorithm (page 202).

Semi-Gradient TD

- $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(R_t + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(s, \mathbf{w}_t)) \nabla \hat{v}(S_t, \mathbf{w}_t)$
  - Why semi-gradient?
  - Why not full-gradient?
- In the linear case, $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(R_t + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(s, \mathbf{w}_t))\mathbf{x}(S_t)$
- Converges!

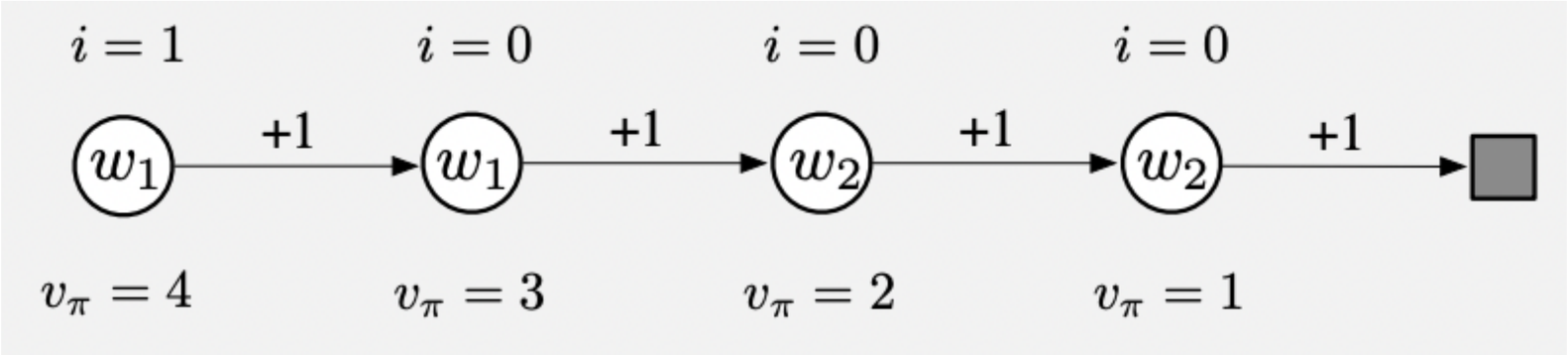
LSTD(0)

- Convergence analysis shows that Linear TD(0) converges to $\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}$.
  - $A = \mathbf{E}[\mathbf{x}_t(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top]$ and $\mathbf{b} = \mathbf{E}[R_{t+1}\mathbf{x}_t]$.
- LSTD(0) estimates A and b and then directly computes the fixed point.
  - (+) More data efficient than semi-gradient linear TD(0)
  - (-) More computation (after optimizations $O(d^2)$ vs $O(d)$ for TD(0))
- Harder to extend to deep reinforcement learning.

Interest and Emphasis

- So far, assumed we are updating states equally (same learning rate) but according to the on-policy state distribution, μ.
- We may wish to emphasize some states more.
- State interest, I_t, represents how much we care about accurate estimation in state S_t.
- Emphasis is a learned multiplier on the learning rate.
  - $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha M_t [R_t - \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})] \nabla \hat{v}(S_t, \mathbf{w})$
  - $M_t \leftarrow I_t + \gamma M_{t-1}$

Interest and Emphasis



- Interest is (1, 0, 1, 0)
- Semi-gradient 2-step TD converges to weight vector (3.5, 1.5)
- Emphatic 2-step TD converges to weight vector (4, 2)

Summary

- Function approximation allows us to represent state values when there are too many states for a look-up table.
- Approximation allows generalization but forces us to choose which states to approximate best.
- Linear function approximation is well understood theoretically and can be powerful with the right set of non-linear features.

Action Items

- Homework 3.
- Begin literature review.
- Begin reading Chapter 11.
- Midterm survey and evaluation.