Reinforcement Learning

Lecture 7: Function approximation

Chris G. Willcocks

Durham University

Lecture overview

Lecture covers chapters 9-11 in Sutton & Barto [1] and content in [2]

1. **Function approximation**
   - introduction
   - definition
   - challenges
2. **Incremental methods**
   - stochastic gradient descent for prediction
   - substituting $v_\pi(S_t)$
   - stochastic gradient descent for control
   - substituting $q_\pi(S_t, A_t)$
   - convergence
3. **Batch learning**
   - experience replay
   - model freezing with double Q-learning

Overview: Function approximation

Using function approximation allows us to scale RL to solve realistic problems.

1. We've seen RL finds optimal policies for arbitrary environments, if the value functions $V(s)$ and policies $Q(s, a)$ can be exactly represented in tables
2. But the real world is too large and complex for tables
3. Will RL work for function approximators?

Example: function approximation

```
Q = np.zeros([n_states, n_actions])
a_p = Q[s,:]

# action-value table is approximated:
a_p = DeepNeuralNetwork(s)
```

Function approximation definition

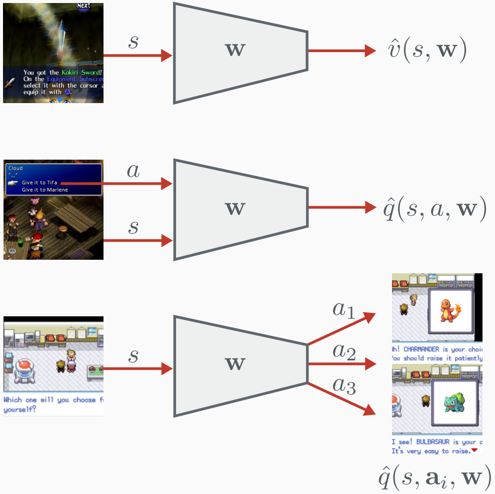
Definition: Function approximation

There are too many states/actions to fit into memory, which are too slow to process. Therefore we estimate the value function:

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

or for control we'd do:

$$\hat{q}(S, A, \mathbf{w}) \approx q_\pi(S, A),$$



Function approximation challenges

Challenges: function approximation

However this is not so straightforward:

1. Data is non-stationary
   - As you explore and discover new rewards, the value function can change drastically
2. Data is not i.i.d.
   - When playing a game, all subsequent experiences are highly correlated

Example: fog of war

Definition: incremental SGD for prediction

Incremental ways to do this, using stochastic gradient descent, to achieve incremental value function approximation.

$$
\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2}\alpha\nabla_{\mathbf{w}}(v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t))^2 \\
&= \mathbf{w}_t + \alpha(v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t))\nabla_{\mathbf{w}}\hat{v}(S_t, \mathbf{w}_t)
\end{aligned}
$$

How do we compute $v_\pi(S_t)$? We substitute it with a target.

Incremental methods stochastic gradient descent for prediction

Definition: substituting $v_\pi(S_t)$

For **MC** learning, the target is the return G_t:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{2}\alpha\nabla_{\mathbf{w}}(G_t - \hat{v}(S_t, \mathbf{w}_t))^2$$

For **TD(0)**, the target is $R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w})$:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{2}\alpha\nabla_{\mathbf{w}}(R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w}_t))^2$$

For **TD(λ)**, the target is the λ return G_t^λ:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{2}\alpha\nabla_{\mathbf{w}}(G_t^\lambda - \hat{v}(S_t, \mathbf{w}_t))^2$$

Definition: action-value function approximation for control

For control, we wish to approximate the action-value function $\hat{q}(S, A, \mathbf{w}) \approx q_\pi(S, A)$

$$
\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2}\alpha \nabla_{\mathbf{w}} (q_\pi(S_t, A_t) - \hat{q}(S_t, A_t, \mathbf{w}_t))^2 \\
&= \mathbf{w}_t + \alpha (q_\pi(S_t, A_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)) \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w}_t)
\end{aligned}
$$

Similarly we substitute $q_\pi(S_t, A_t)$ with a target.

Definition: substituting $q_\pi(S_t, A_t)$

For **MC** learning, the target is the return G_t:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{2}\alpha\nabla_{\mathbf{w}}(G_t - \hat{q}(S_t, A_t, \mathbf{w}_t))^2$$

For **TD(0)**, the target is $R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w})$:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{2}\alpha\nabla_{\mathbf{w}}(R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}_t))^2$$

For **TD(λ)**, the target is the λ return:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{2}\alpha\nabla_{\mathbf{w}}(q_t^\lambda - \hat{q}(S_t, A_t, \mathbf{w}_t))^2$$

Incremental methods convergence

Convergence of function approximation approaches

In practice, these methods 'chatter' around the near-optimal value function (there's no guarantee when you use function approximation that your improvement step really does improve the policy).

	Table	Linear	Non-linear
MC control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
Gradient Q-learning	✓	✓	✗

Table from [2].

Batch learning experience replay

Experience replay

Incremental methods have several problems:

1. They are not sample efficient
2. They have strongly correlated updates that break the i.i.d. assumptions of popular SGD algorithms
3. They may rapidly forget rare experiences that would be useful later on

Experience replay [3] and prioritised experience replay [4] address these issues by storing experiences and reusing them. These can be sampled uniformly or prioritised to replay important transitions more frequently.

Colab implementation of Experience Replay:

Colab implementation of Prioritised Experience Replay:

Definition: double Q-learning

Putting this all together, we have:

- Sample action from our ϵ-greedy policy or with GLIE
- Store $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ in the experience replay buffer $\mathcal{D}$
- Sample a random minibatch $s, a, r, s' \sim \mathcal{D}$
- Update in the gradient direction from our Q-network towards our Q-targets (with the max).
    - The targets can be frozen [5, 6]

$$\mathcal{L}_i(\mathbf{w}_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i} \left[\left(r + \gamma \max_{a'} Q(s', a'; \mathbf{w}_i^-) - Q(s, a; \mathbf{w}_i) \right)^2 \right]$$

Take Away Points

Summary

In summary:

- function approximation can work, but requires additional tricks to make it stable
- read recent papers and study implementations, such as these **DQN extensions:**
- there are many tricks, so you need to be good at reading state-of-the-art papers
- study the (Atari) benchmark results to gain intuition as to what tricks work well, and under what circumstances

References I

[1] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction (second edition). Available online. MIT press, 2018.

[2] David Silver. Reinforcement Learning lectures. https://www.davidsilver.uk/teaching/. 2015.

[3] Long-Ji Lin. “Self-improving reactive agents based on reinforcement learning, planning and teaching”. In: Machine learning 8.3-4 (1992), pp. 293–321.

[4] Tom Schaul et al. “Prioritized experience replay”. In: arXiv preprint arXiv:1511.05952 (2015).

[5] Hado Van Hasselt, Arthur Guez, and David Silver. “Deep reinforcement learning with double Q-learning”. In: arXiv preprint arXiv:1509.06461 (2015).

[6] Ziyu Wang et al. “Dueling network architectures for deep reinforcement learning”. In: International conference on machine learning. 2016, pp. 1995–2003.