Statistical Approaches to Learning and Discovery

Reinforcement Learning

Zoubin Ghahramani & Teddy Seidenfeld
zoubin@cs.cmu.edu & teddy@stat.cmu.edu

CALD / CS / Statistics / Philosophy
Carnegie Mellon University
Spring 2002

Markov Decision Problems (MDPs)

States: s_t
Actions: a_t
Rewards: r_t

Average (undiscounted) future return:

$$R_t = \lim_{k \to \infty} \frac{1}{k} \sum r_{t+k+1}$$

Discounted future return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Markov property:

$$P(s_{t+1}, r_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots) = P(s_{t+1}, r_{t+1} | s_t, a_t)$$

Markov Decision Problems (MDPs)

The world is characterized by
Transition Probabilities:

$$\mathcal{P}^a_{ss'} = P(s_{t+1} = s' | s_t = s, s_t = a)$$

Expected rewards:

$$\mathcal{R}^a_{ss'} = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s')$$

The agent is characterized by
Policy:

$$\pi(s, a) = P(a_t = a | s_t = s)$$

Graphical Representation:

Value Functions

Value Function: how good is it to be in a given state? This obviously depends on the agent's policy:

$$V^{\pi}(s) = E_{\pi}(R_t | s_t = s) = E_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right)$$

State-action value function: how good is it to be in a given state and take a given action:

$$Q^{\pi}(s, a) = E_{\pi}(R_t | s_t = s, a_t = a) = E_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right)$$

The relation between the state value function and the state-action value function:

$$V^{\pi}(s) = \sum_{a} \pi(s, a) Q^{\pi}(s, a)$$

Self-Consistency of Value Functions

A fundamental property of value functions is that they satisfy a set of recursive consistency equations. V^π is the unique solution to these equations.

$$\begin{aligned}
V^\pi(s) &= E_\pi(R_t | s_t = s) \\
&= E_\pi\left(r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s \right) \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma E_\pi \left(\sum_{k=0}^{\infty} \gamma^k r_{t+l+2} \middle| s_{t+1} = s' \right) \right] \\
&= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right]
\end{aligned}$$

We can solve them using a "backup operation" from $s' \to s$ (or other means). Linear system of $N \equiv |s|$ equations in N unknowns.

$$\mathbf{v} = (I - \gamma \sum_a \mathrm{diag}(\boldsymbol{\pi}_a)\mathcal{P}^a)^{-1} (\sum_a \boldsymbol{\pi}_a \odot \mathrm{diag}(\mathcal{P}^a \mathcal{R}^{a\top}))$$

There is a similar equation for $Q^\pi(s,a)$

Optimal Policies and Values

Optimal Policy: π^* such that $V^{\pi^*}(s) \geq V^{\pi}(s)\ \forall s$. There may be more than one optimal policy.

Question: Is there always at least one optimal policy? YES

Optimal state value function: $V^*(s) = \max_\pi V^\pi(s)\ \forall s$

Optimal state-action value function: $Q^*(s,a) = \max_\pi Q^\pi(s,a)\ \forall s$. This is the expected return of action a in state s, thereafter following optimal policy.

$$Q^*(s,a) = E\left(r_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t = s, a_t = a\right)$$

Bellman Optimality Equation

$$\begin{aligned}
V^*(s) &= \max_a Q^{\pi^*}(s,a) \\
&= \max_a E_{\pi^*}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right) \\
&= \max_a E_{\pi^*}\left(r_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t = s, a_t = a \right) \\
&= \max_a \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^*(s')]
\end{aligned}$$

$$\begin{aligned}
Q^*(s,a) &= E\left(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \middle| s_t = s, a_t = a \right) \\
&= \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q(s', a')]
\end{aligned}$$

N equations in N unknowns for V^*
NA equations in NA unknowns for Q^*

Solving MDPs

Given the optimal value function, V^*, it is easy to get optimal policy π^*: be **greedy** w.r.t. V^*.

If you have V^*, the actions that appear best after a one-step search will be optimal.

V^* turns a long-term reward into a quantity that is locally and immediately available.

Using Q^* it is even easier to get the optimal policy:

$$\pi^*(s, a) = 0 \;\; \forall a \;\; s.t. \;\; Q^*(s, a) \neq \max_{a'} Q^*(s, a')$$

Policy Improvement Theorem

Policy Evaluation $$V^{\pi}_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^{\pi}_k(s')]$$

assumes $\mathcal{P}$ known, $\mathcal{R}$ known, and a full backup (we can also sweep in place)

Policy Improvement Theorem

$$Q^{\pi}(s,\pi'(s)) \geq V^{\pi}(s)\ \forall s \ \Rightarrow\ V^{\pi'}(s) \geq V^{\pi}(s)$$

Proof:

$$\begin{aligned}
V^{\pi}(s) &\leq Q^{\pi}(s,\pi'(s)) \\
&= E_{\pi'}(r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s) \\
&\leq E_{\pi'}(r_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi'(s_{t+1})) | s_t = s) \\
&= E_{\pi'}(r_{t+1} + \gamma E_{\pi}(r_{t+2} + \gamma V^{\pi}(s_{t+2})) | s_t = s) \\
&= E_{\pi'}(r_{t+1} + \gamma r_{t+2} + \gamma^2 V^{\pi}(s_{t+2}) | s_t = s) \\
&\vdots \\
&\leq V^{\pi'}(s)
\end{aligned}$$

Policy Iteration

The policy improvement theorem suggests a way of improving policies:

$$\begin{aligned}\pi'(s) &\leftarrow \arg\max_a Q^\pi(s,a)\ \ \forall s \\ &= \arg\max_a E(r_{t+1} + \gamma V^\pi(s_{t+1})|s_t = s, a_t = a)\end{aligned}$$

This procedure converges to an optimal policy by policy improvement theorem and Bellman optimality.

$$V^{\pi'}(s) \geq \arg\max_a Q^\pi(s,a) \geq \sum_a \pi(s,a) Q^\pi(s,a) = V^\pi(s)$$

Policy Iteration: Iterates between evaluation and improvement

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \ldots \pi^*$$

Problem with Policy Iteration: Evaluation step can be really slow...

Value Iteration

Do we really need to wait until convergence?

In fact, we can improve after **one** sweep of evaluation!

$$\begin{aligned} V_{k+1}(s) &\leftarrow \max_a E(r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s, a_t = a) \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V_k(s')] \end{aligned}$$

converges: $V_k \longrightarrow V^*$. At each step we also have a policy.

Problem: it is still not feasible to update the value of every single state.
E.g. backgammon has 10^{20} states!

Bellman called this the **curse of dimensionality**

Asynchronous dynamic programming

These are in-place iterated dynamic programming algorithms that are not organized in terms of systematic sweeps over all the states.

States are backed-up in order visited or randomly.

To converge the algorithms must continue to visit every state.

Key idea in RL: We can run the DP algorithm at the same time as the agent is *actually experiencing* the MDP.

This leads to an **exploration vs exploitation tradeoff**: act so as to visit new parts of state space or exploit already visited part of state-space?

An example of a simple exploration strategy are ϵ-greedy policies:

$$\pi_\epsilon(s,a) = (1-\epsilon)\pi(s,a) + \epsilon u(a)$$

Monte Carlo and TD

Monte Carlo methods solve RL problems by averaging sample returns.

Question: how do you trade off length of sampled trajectory, vs previously estimated values?

$$
\begin{aligned}
V(s_t) &\leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \qquad (1) \\
V(s_t) &\leftarrow V(s_t) + \alpha[r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}) - V(s_t)] \\
V(s_t) &\leftarrow V(s_t) + \alpha[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V(s_{t+4}) - V(s_t)] \\
&\vdots \\
V(s_t) &\leftarrow V(s_t) + \alpha[R_t - V(s_t)] \qquad (2)
\end{aligned}
$$

Equation (1) is **Temporal Difference learning, TD(0)**. TD(λ) approximates the range eqn (1)–(2), where higher λ is closer to the full MC method.

TD(λ) has been proven to converge

These are general methods for controlling the **bias-variance tradeoff**.

SARSA and Q Learning

Definitions: *on-policy* methods evaluate or improve the current policy used for control. *Off-policy* methods evaluate or improve one policy, while acting using another (behavior) policy.

In off-policy methods, the behavior policy must have non-zero probability for each state-action the evaluated policy does.

SARSA: on-policy greedy control

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Q Learning: off-policy greedy control

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Converges if $\forall a, s$ are visited and updated infinitely often.

We can also combine the bias-variance ideas with Q and SARSA, to get $Q(\lambda)$ and SARSA(λ).

Function Approximation

For very large or continuous state spaces it is hopeless to store a table with all the state values or state-action values.
It makes sense to use **function approximation**

$$V(s) = f_\theta(s)$$

e.g. basis function representation:

$$V(s) = \sum_i \theta_i \phi_i(s)$$

Similarly for Q.
This should hopefully lead to good **generalization**

Gradient descent methods:

$$\theta(t+1) = \theta(t) + \alpha[v_t - V_t(s_t)]\frac{\partial V_t(s_t)}{\partial \theta}$$

See chapter 8 of Sutton and Barto.

Optimal Control

Optimal Control: The engineering field of optimal control covers exactly the same topics as RL, except the state is usually assumed to be continuous.

The **Hamilton-Jacobi-Bellman** optimality conditions are the continuous state generalization of the Bellman equations.

A typical elementary problem in optimal control is the linear quadratic Gaussian control **LQG** problem. Here the cost function is quadratic in states and actions, and the system is a linear-Gaussian state-space model.

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \epsilon_t$$

For this model the optimal policy can be computed from the estimated state. It's a linear feedback controller:

$$\mathbf{u}_t = L\hat{\mathbf{x}}_t$$

Influence Diagrams

You can extend the framework of directed acyclic probabilistic graphical models (a.k.a. Bayesian networks) to include **decision nodes** and **value nodes**. These are called **influence diagrams**.

Solving an influence diagram corresponds to finding the settings of th decision nodes that maximize the expectation of the value node.

It is possible to convert the problem of solving an influence diagram into the problem of doing inference in a (usually multiply connected) graphical model (Shachter and Peot, 1992). Exact solutions can be computationally intractable.

Like other graphical models, influence diagrams can contain both observed and **hidden** variables...

POMDPs

POMDP = Partially-observable Markov decision problem.

The agent does not observe the full state of the environment.

What is the optimal policy?

- If the agent has the correct model of the world, it turns out that the optimal policy is a (piece-wise linear) function of the **belief state**.

  Unfortunately, the belief state can grow exponentially complex.

- Equivalently, we can view the optimal policy as being a function of the entire sequence of past actions and observations (this is the usual way the policy in influence diagrams is represented).

  Again, unfortunately, the set of possible such sequences grows exponentially.

Efficient methods for approximately solving POMDPs is an active area of research.

Some References for Reinforcement Learning

- Kaelbling, L.P., Littman, M.L. and Moore, A.W. (1996) Reinforcement Learning: A Survey. Journal of Aritificial Intelligence Research 4:237-285.

- Sutton, R.S. and Barto, A.G. (2000) Reinforcement Learning: An Introduction. MIT Press.

- Bertsekas, D.P. and Tsitsiklis, J.N. (1996) Neuro-Dynamic Programming. Athena Scientific.

- Bryson, A.E. and Ho, Y.-C. (1975) Applied Optimal Control. Hemisphere Publishing Corp. Washington DC.