Reinforcement Learning

Josh Bryan

University of Illinois at Chicago, MCS 548

December 14, 2010

Contents

What is Reinforcement Learning?

Reinforcement learning takes psychology, decision theory, and learning theory to answers "How should an agent learn to act?"

- Classical Conditioning (Pavlov's Dog)
- Utility Theory
- (Partially Observable) Markov Decision Processes
- Learns an "Agent Function"

Agent Function

- O is a set of observation symbols.
- A is a set of actions.

$$f : O^* \to A$$

Uses

- Game playing (e.g. backgammon)
- Learning tasks with delayed feedback
- Elevator Control
- Robotic Control (e.g. stick balancing, car driving)
- Simulation Based Approximation Methods (similar to fictitious play)
- Telecommunications (e.g. learning optimal routing)

What is an MDP?

Definition: Markov Decision Process

An MDP is a tuple $\langle S, A, T, R \rangle$ where:

- S is a set of states.
- A is a set of actions.
- T is a transition function.

$$T : S \times A \times S \rightarrow [0, 1]$$

- R is a reward function.

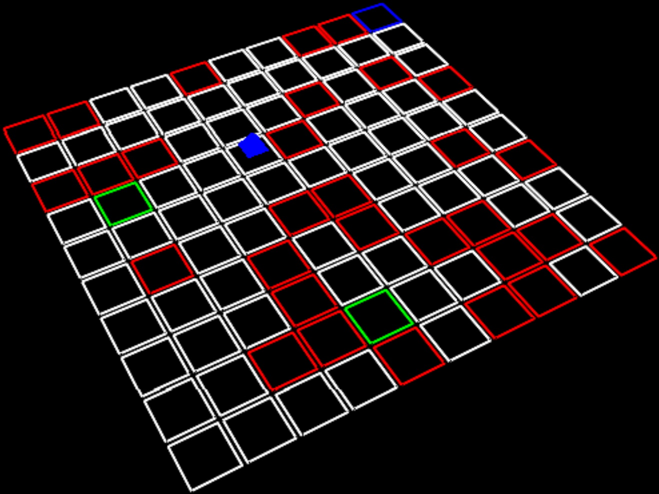
$$R : S \times A \rightarrow \mathbb{R}$$

Simple Example

As a running example, and as a demo at the end, we will be considering this environment:

- An agent is in a square 10×10 grid world. Therefore $S = \{0, \cdots, 9\}^2$.
- The set of actions are $\{up, down, left, right\}$.
- 20 of the grid squares are “bad” and give a reward of -1.
- 2 of the grid squares are “good” and give a reward of 3.
- Every action taken invokes a penalty of -0.01.

Simple Example



What is a Policy?

Definition: Policy

A policy π is a mapping from states to actions.

$$\pi : S \rightarrow A$$

Value of Policy

The value of a policy $V^{\pi}(s)$ is the **total discounted reward** obtained by starting at a state s and following the policy π.

$$\begin{aligned} V^{\pi}(s) &= E[R(s_0, \pi(s_0)) + \gamma(R(s_1, \pi(s_1)) + \gamma(\dots))|\pi, s_0 = s] \\ &= E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \middle| \pi, s_0 = s\right] \end{aligned}$$

Here, $0 \leq \gamma \leq 1$ is the discount factor. If $\gamma = 1$ this expectation may not exist unless the horizon is finite.

Alternative Value of Policy

The value of a policy $V^\pi(s)$ is the **long run average reward** obtained by starting at a state s and following the policy π.

$$V^\pi(s) \quad = \quad \lim_{n\to\infty} E\left[\frac{\sum_{t=0}^{n} R(s_n, \pi(s_n))}{n}\middle|\pi, s_0 = s\right]$$

Bellman Equations I

The solution to an MDP is an optimal policy π^* that maximizes $V^{\pi^*}(s)$ for all s:

$$\pi^* = \operatorname*{argmax}_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right]$$

There are $|A|^{|S|}$ policies. For our example, that's $4^{100} \approx 1.6 \times 10^{60}$

Bellman Equations II

Recursive Bellman Equation for π^* and V^* [1]

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} \overbrace{R(s,a)}^{\text{Immediate Reward}} + \overbrace{\gamma \sum_{s' \in S} T(s,a,s') V^*(s')}^{\text{Expected Future}}$$

$$\begin{aligned} V^*(s) &= \overbrace{R(s,\pi^*(s))}^{\text{Immediate Reward}} + \overbrace{\gamma \sum_{s' \in S} T(s,\pi^*(s),s') V^*(s')}^{\text{Expected Future}} \\ &= \max_{a \in A} R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \end{aligned}$$

[1] V^* is shorthand for V^{π^*}

What are we learning?

- We want to learn π^*.

What are we learning?

- We want to learn π^*.
- We will do this by learning a Q function:

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')$$

What are we learning?

- We want to learn π^*.
- We will do this by learning a Q function:

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')$$

- If we learn $Q(s,a)$ then

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} Q(s,a)$$

What are we given?

- We are given A and S and a sequence $\langle s_0, a_0, r_0, s_1, a_1, r_1, \ldots \rangle$ where $s \in S$ is observable and sampled by the environment according to T, $a \in A$ is chosen by the agent, and $r = R(s, a)$.

What are we given?

- We are given A and S and a sequence $\langle s_0, a_0, r_0, s_1, a_1, r_1, \ldots \rangle$ where $s \in S$ is observable and sampled by the environment according to T, $a \in A$ is chosen by the agent, and $r = R(s, a)$.
- We are not given T or R.

What are we given?

- We are given A and S and a sequence $\langle s_0, a_0, r_0, s_1, a_1, r_1, \ldots \rangle$ where $s \in S$ is observable and sampled by the environment according to T, $a \in A$ is chosen by the agent, and $r = R(s, a)$.
- We are not given T or R.
- We are not going to learn T or R.

Model Free vs. Model Based Learning

Model Free

The agent learns to act in specific environment, but does not know or learn a model of that environment.

Model/Knowledge Based

The agent learns a model of the environment and uses the model to compute how to act in the environment.

Q Recursion

We can rewrite the Q function as:

$$Q(s,a) = R(s,a) + \gamma E[Q(s',a)]$$

Q Approximation and Update

We maintain an approximation $\hat{Q}$ of Q:

$$\hat{Q}_{t+1}(s,a) \leftarrow \hat{Q}_t(s,a) + \alpha_t(s,a)\Big[\overbrace{R(s,a) + \gamma \max_{a' \in A} \hat{Q}_t(s',a')}^{\text{New Estimate}} - \overbrace{\hat{Q}_t(s,a)}^{\text{Old Estimate}}\Big]$$

Where $\alpha_t(s,a)$ is the learning rate. If $\alpha = 0$ then no learning occurs. If $\alpha = 1$ than $\hat{Q}$ is completely overwritten at each transition.

Q Approximation and Update

We maintain an approximation $\hat{Q}$ of Q:

$$\hat{Q}_{t+1}(s,a) \leftarrow \hat{Q}_t(s,a) + \alpha_t(s,a)\Big[\overbrace{R(s,a) + \gamma \max_{a' \in A} \hat{Q}_t(s',a')}^{\text{New Estimate}} - \overbrace{\hat{Q}_t(s,a)}^{\text{Old Estimate}}\Big]$$

Where $\alpha_t(s,a)$ is the learning rate. If $\alpha = 0$ then no learning occurs. If $\alpha = 1$ than $\hat{Q}$ is completely overwritten at each transition.

This is equivalent to:

$$\hat{Q}_{t+1}(s,a) \leftarrow (1 - \alpha_t(s,a))\hat{Q}_t(s,a) + \alpha_t(s,a)\left[R(s,a) + \gamma \max_{a' \in A} \hat{Q}_t(s',a')\right]$$

Learning Rate Schedule

Requirements on Schedule

- $\alpha_t(s, a)$ must decay over time to ensure convergence.

Learning Rate Schedule

Requirements on Schedule

- $\alpha_t(s, a)$ must decay over time to ensure convergence.
- $\sum_i^\infty \alpha_{t(i,s,a)}(s, a) = \infty$ where $t(i, s, a)$ is the time state s and action a is visited for the i'th time.

Learning Rate Schedule

Requirements on Schedule

- $\alpha_t(s,a)$ must decay over time to ensure convergence.
- $\sum_i^\infty \alpha_{t(i,s,a)}(s,a) = \infty$ where $t(i,s,a)$ is the time state s and action a is visited for the i'th time.
- $\sum_i^\infty \alpha_{t(i,s,a)}(s,a)^2 < \infty$

Learning Rate Schedule

Requirements on Schedule

- $\alpha_t(s,a)$ must decay over time to ensure convergence.
- $\sum_i^\infty \alpha_{t(i,s,a)}(s,a) = \infty$ where $t(i,s,a)$ is the time state s and action a is visited for the i'th time.
- $\sum_i^\infty \alpha_{t(i,s,a)}(s,a)^2 < \infty$

A practical solution

$$\alpha_t(s,a) = \frac{1}{1 + visits_t(s,a)}$$

where $visits(s,a)$ is the number of times action a has been taken in state s.

Learning Policy: Exploration vs Exploitation I

Exploration: Random Policy

- Choose $a \in A$ at Random in each time step.
- $\hat{Q}$ is guaranteed to converge to the true Q.

Exploitation: Locally Optimal Policy

- Choose $a = \mathrm{argmax}_{a \in A} \hat{Q}(s, a)$ in each state s.
- $\hat{Q}$ is only guaranteed to find a local optimum.

Learning Policy: Exploration vs Exploitation II

Hybrid: Logit Quantal Response or Simulated Annealing

- Choose a according to this probability distribution:

$$P(a) = \frac{e^{\beta \hat{Q}(s,a)}}{\sum_{a' \in A} e^{\beta \hat{Q}(s,a')}}$$

- If $\beta = 0$ this is equivalent to the random policy.
- As $\beta \to \infty$, this becomes the exploitative policy.
- β may be increased over time (similar to simulated annealing).

How can we improve Q learning? I

On Policy Learning

Q-learning learns the Q function for the optimal policy, not the policy actually being followed. Why would we want to learn a policy other than the optimal?

- Sometimes exploration may be “too dangerous”.
- Policy may not be entirely under agent’s control (e.g. multiagent settings).
- On Policy Learning allows easier application of eligibility traces.

How can we improve Q learning? II

Update more than one Q value at a time.

Q-learning only updates a single Q value at a time, so convergence can take a very long time if rewards are delayed. A clever method for updating multiple Q values simultaneously involves eligibility traces.

s, a, r, s', a'

For each tuple $\langle s, a, r, s', a' \rangle$, we update $\hat{Q}$ as follows:

$$\hat{Q}_{t+1}(s,a) \leftarrow \hat{Q}_t(s,a) + \alpha_t(s,a)\left[R(s,a) + \gamma \hat{Q}_t(s',a') - \hat{Q}_t(s,a)\right]$$

Note that the only difference is that we use the actual action chosen for the discounted future rather than the maximum valued action.

Atomic Bread Crumbs

The key idea is to leave exponentially decaying traces ($e(s, a)$) identifying actions and states on the path to the current state.

Update Algorithm

1. Initialize $e(s, a) \leftarrow 0, \forall s \in S, a \in A$.
2. For each $\langle s, a, r, s', a' \rangle$ tuple received:
    1. $e(\hat{s}, \hat{a}) \leftarrow \gamma \lambda e(\hat{s}, \hat{a}) \forall \hat{s} \in S, \hat{a} \in A$
    2. $e(s, a) \leftarrow e(s, a) + 1$
    3. $\delta \leftarrow R(s, a) + \gamma \hat{Q}_t(s', a') - \hat{Q}_t(s, a)$
    4. $\forall \hat{s} \in S, \hat{a} \in A$

$$\hat{Q}_{t+1}(\hat{s}, \hat{a}) \leftarrow \hat{Q}_t(\hat{s}, \hat{a}) + e(\hat{s}, \hat{a}) \alpha_t(\hat{s}, \hat{a}) \delta$$

Demo

...and now for a demo comparing Q and SARSA-λ learning!

Further extensions

- Partial observability.
- Continuous action, state, and observation spaces.
- Q function approximators (e.g. neural networks)
- Exploitation vs. Exploration
- Hybrid model free / model based learning (e.g. feudal architectures)
- Alternative optimality criteria.

Sources I

- L. P. Kaelbling, M. L. Littman, and A. W. Moore.
  Reinforcement learning: A survey.
  Journal of Artificial Intelligence, 4(1):237–285, 1996.
- Tom Mitchell.
  Machine Learning.
  McGraw Hill, 1997.
- S. Russell and P. Norvig.
  Artificial Intelligence: A Modern Approach (Third Edition).
  Prentice Hall, 2010.
- C J C H Watkins.
  Learning from Delayed Rewards.
  PhD thesis, Cambridge University, 1989.