COS 511: Foundations of Machine Learning

Rob Schapire
Scribe: Kostas Tsioutsiouliklis

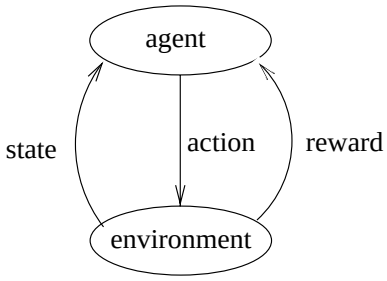
Lecture #23
April 29, 2003

1 Review and Motivation

In this lecture we will continue our study of *Reinforcement Learning*, which we started in the previous lecture.

Consider the following problem:

An agent (or robot) exists in a certain environment. The environment consists of states, and the agent moves between states in this environment. Based on the current state information the agent decides which state to move to next. Depending on the agent's action, the environment returns a new state information and some reward. The goal in this problem is to maximize the agent's reward.



The above interaction between the agent and the environment can also be shown as follows:

$$s_0 \xrightarrow[r_1]{a_0} s_1 \xrightarrow[r_2]{a_1} s_2 \xrightarrow[r_3]{a_2} \cdots$$

Here, the agent starts at state s_0 and initially executes action a_0. This gets the agent reward r_1 and takes him to state s_1. At state s_1, he executes action a_1, which gets him reward r_2 and takes him to state s_2. And so on.

We assume that the environment obeys the Markov property, that is, everything in the past can be summed up in the current state, or in other words, the future depends only on the current state. We also assume that the environment has a finite number of states. And as mentioned before, our goal is to find an optimal way to act in this environment.

2 Markov Decision Process (MDP)

Define a policy $\pi : S \to A$ to be a mapping between what has happened in the past and what has to be done at the current state. S is the set of states and A the set of possible actions. The goal is to find a policy that maximizes the reward.

2.1 Optimal policy

We define $V^{\pi}(s)$ to be the value of policy π when starting at state s. We can write:

$$V^{\pi}(s) = E_{\pi}[r_{t+1} + r_{t+2} + r_{t+3} + \ldots | s_t = s] \tag{1}$$

$$a_t = \pi(s_t) \tag{2}$$

The sum in equation 1 is typically infinite. In a more realistic setting the rewards in the future get discounted, and equation 1 becomes:

$$V^{\pi}(s) = E_{\pi}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s_t = s] \tag{3}$$

where $\gamma \in [0, 1)$ is the decay factor.

Under this setting our goal is to find a policy π, that maximizes $V^{\pi}(s)$. We will show the following:

Theorem. There exists a policy π^*, which maximizes $V^{\pi}(s)$ for all states s.

We won't proove this theorem formally, but intuitively, why should it be true? Why should there be a single policy that is optimal in all states, and not several policies each of which is optimal in only some states?

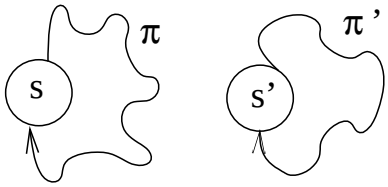
Let's assume that there is an optimal policy π for state s and a different optimal policy π' for state s':

$$V^{\pi}(s) > V^{\pi'}(s) \tag{4}$$

$$V^{\pi}(s') < V^{\pi'}(s') \tag{5}$$

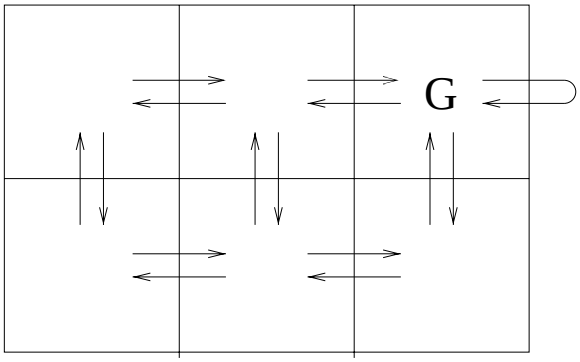
Now, let's say that the agent starts at state s and follows the optimal policy π at this state. After a while he reaches state s'. At this point, it is better to switch to policy π', which is optimal at state s'. But then we could define a new policy that would consist of the combination of π and π', and which would outperform both of them at all stages.

If s and s' are disjoint, then the following picture applies, where again, a combination of π and π' is better than either π or π' alone.

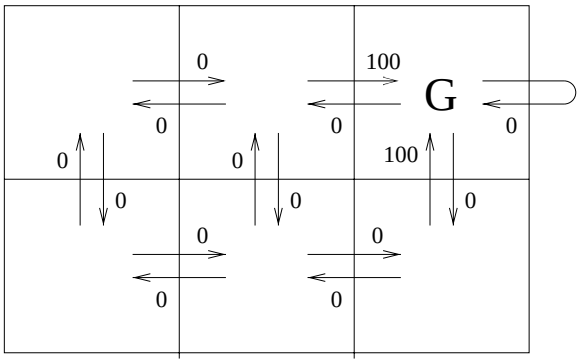


Notation: in the rest of these notes we will often use the notation V^* instead of V^{π^*}, when π^* is easily implied.

2.2 Example

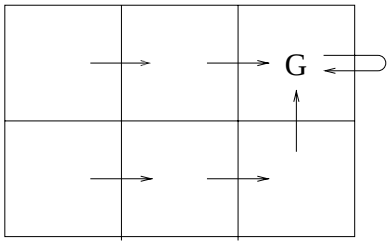


Consider a robot that moves in the above grid, let's say deterministically. If it moves to the goal state G, it gets reward = 100. If it moves anywhere else it gets reward = 0. So, the picture becomes:



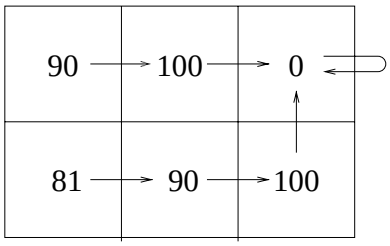
Let's also assume that the decay factor $\gamma = 0.9$.

Question: In the above setting, what is an optimal policy π^*?
Answer: To move towards the goal state:



Question: What is V^* in this case?

Answer:



Note that in the above example, we don't have control of where the robot starts.

2.3 Evaluating a policy

Before we address the issue of finding a policy, we look at how to evaluate a known policy π at a given state s_t.

We know from equation 3 that

$$\begin{aligned} V^\pi(s_t) &= E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma r_{t+3} + \ldots | s_t] && (6) \\ &= E_\pi[r_{t+1}|s_t] + \gamma E_\pi[r_{t+2} + \gamma r_{t+3} + \ldots | s_t] && (7) \\ &= E_\pi[r_{t+1} + \gamma V^\pi(s_{t+1})|s_t] && (8) \end{aligned}$$

In the last lecture we saw that for the *next state function*:

$$P(s'|s,a) = Pr[s_{t+1} = s'|s_t = s, a_t = a] \quad (9)$$

and for the *expected reward*:

$$r(s,a) = E[r_{t+1}|s_t = s, a_t = a] \quad (10)$$

So, equation 8 becomes:

$$\boxed{V^\pi(s) = r(s,a) + \gamma \textstyle\sum_{s'} P(s'|s,a)V^\pi(s')} \quad (11)$$

Equation 11 is also known as the Bellman equation. It is a linear equation with n unknowns, where n is the number of states, and it has a unique solution.

Another way to solve this equation is iteratively:

Choose a V_0 and apply the Bellman equation iteratively so that:

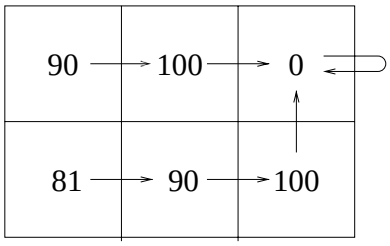
$$V_{k+1}(s) = r(s,a) + \gamma \sum_{s'} P(s'|s,a)V_k(s') \quad (12)$$

One can show that this will converge to the true value function.

Unfortunately, neither solution is satisfying. For a simpler approach, we can experiment with the environment to find an approximation to the solution.

3 Finding an optimal policy

Question: If one knew V^* how could he find an (optimal) policy? For example, if the robot is in the upper left square of the picture below, the best action is to go to the right because that state has the highest value among candidates.

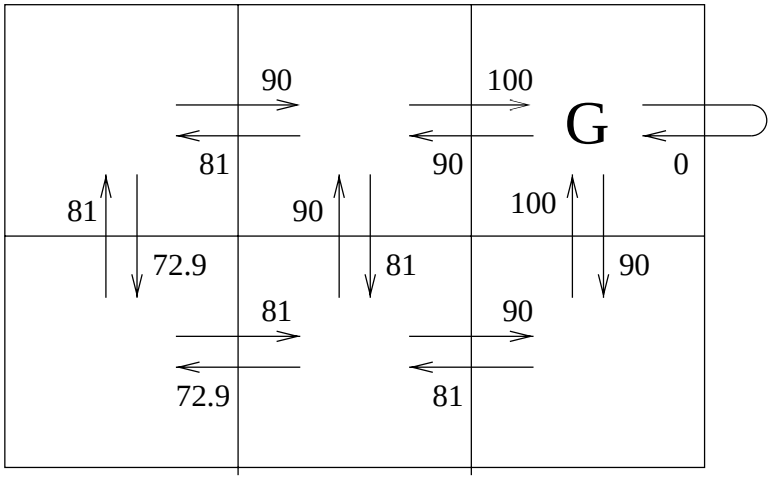


Define the *state action value function* Q as follows:

$Q^*(s,a)$ = value of starting at state s, executing action a and from there on following optimal policy π^*. That is:

$$Q^*(s,a) = E[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \tag{13}$$

Let's draw Q^* for the grid example:



In the above picture, if the robot is in the upper middle square, the right thing to do is to go to the right, because this gives maximum Q^* value.

We can also write equation 13 in terms of the next state and reward functions:

$$Q^*(s,a) = r(s,a) + \gamma \sum s' P(s'|s,a) V^*(s') \tag{14}$$

Intuitively, if one knows what Q^* is, he also knows what the optimal policy is:

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{15}$$

Thus, our strategy for finding the optimal policy is to find (or approximate) the Q^* function.

Question: How can $V^*(s)$ be written in terms of Q^*?

Answer: $V^*(s) = \max_a Q^*(s,a)$. Notice that the Bellman optimality equation is no longer linear in V^*.

Question: How can we compute V^*?

Answer: Same as before when we iterated the Bellman equation. This method is called "value iteration", and is similar to "policy iteration", which is another method for calculating V^*. In "policy iteration" we start with a policy π_0 and modify it until it converges to the optimal value. In "value iteration" we start from V_0 and iterate.

3.1 Policy iteration

Policy iteration algorithm:

1 start with π_0
2 on round k
3 compute V^{π_k} (by iteration or exact solution or by experimenting with the environment (see next lecture))
4 greedify:

$$\pi_{k+1}(s) = \arg\max_a Q^{\pi_k}(s,a)$$

$$\pi_0 \longrightarrow V^{\pi_0} \longrightarrow \pi_1 \longrightarrow V^{\pi_1} \longrightarrow \pi_2 \longrightarrow \cdots \longrightarrow \pi^*$$

Even though the above algorithm is greedy, we can prove that $V^{\pi_{k+1}}(s) \geq V^{\pi_k}(s)$ and unless π_k is optimal, the above inequality will be strict in at least one state. The algorithm stops when the value function stops changing. Therefore, the algorithm will converge because there is only a finite number of policies and no cycles possible. One can also prove that if there is equality at all states s, then π_k is optimal.

Question: In the worst case, how many policies can there be?

Answer: At most $|A|^{|S|}$. It is an open problem to find the number of iterations required by the algorithm to converge to π^*. The best known lower bound is linear and the best upper bound is still exponential.

4 Approximating the solution

So far we assumed that we know everything about the environment of the MDP, that is, both the r and p functions. To solve the problem in the probabilistic case, we can approximate the probabilities and build the complete model of the environment.

In a model-free approach, one can estimate π^* without ever making an MDP. If one can estimate Q^*, then π^* can also be found. We assume that the environment is deterministic, i.e. the reward is a deterministic function of action that is taken:

$$Pr[r_{t+1} = r(s,a) | s_t = s, a_t = a] = 1 \tag{16}$$

This is also shown in the following diagram:

$$s_t \xrightarrow[r(s_t, a_t)]{a_t} s_{t+1} = \delta(s_t, a_t)$$

Then, Q^* can be written more simply as follows:

$$\begin{aligned} Q^*(s,a) &= r(s,a) + \gamma V^*(\delta(s,a)) && (17) \\ &= r(s,a) + \gamma \max_{a'} Q^*(\delta(s,a), a') && (18) \end{aligned}$$

4.1 Q-learning

The basic idea of Q-learning is to maintain an estimate $\hat{Q}(s,a)$ of $Q^*(s,a)$.

Q-learning algorithm:

1 assume initially $\hat{Q}(s,a) = 0$
2 repeat:
3 observe k
4 choose action a
5 get reward r
6 observe k
7 update:
8 $\hat{Q}(s,a) = r + \gamma \max_{a'} \hat{Q}(s',a')$

Ideally, in the above algorithm we would have set:

$$\hat{Q}(s,a) = r + \gamma \max_{a'} Q^*(s',a') \quad (19)$$

but Q^* is unknown and thus we update it approximately with $\hat{Q}$.

So far, we didn't say anything as to how s and a are chosen. In practice, one would try to maximize the reward, but the best choice of s and a is a more general problem of exploration vs. exploitation.

4.2 Convergence of Q-learning

We will next show that Q-learning converges to Q^*.

Theorem. Assume that each (s,a) pair is visited infinitely often. Then $\hat{Q}(s,a)$ converges to Q^*, that is, $\hat{Q}(s,a) \to Q^*$. Or to be more precise,

$$\Delta_m = \max_{s,a} |\hat{Q}_m(s,a) - Q^*(s,a)| \to 0 \quad (20)$$

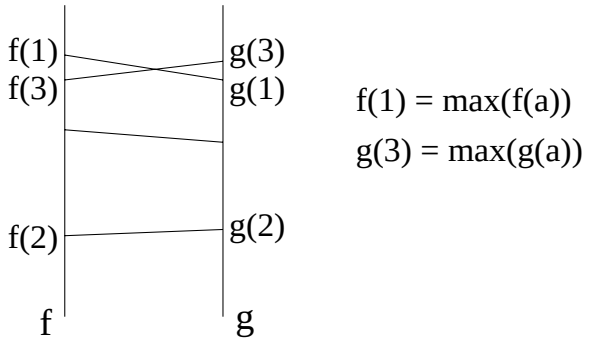
as m goes to infinity, where m is the iteration number.

Proof.
Suppose that (s,a) is visited at iteration m. Then:

$$
\begin{aligned}
|\hat{Q}_{m+1}(s,a) - Q^*(s,a)| &= |(r(s,a) + \gamma \max_{a'} \hat{Q}(s',a') \\
&\quad -(r(s,a) + \gamma \max_{a'} Q^*(s',a'))| \qquad (21)\\
&= |\gamma \max_{a'} \hat{Q}_n(s',a') - \max_{a'} Q^*(s',a')| \qquad (22)\\
&\leq \gamma \max_{a'} |\hat{Q}_n(s',a') - Q^*(s',a')| \qquad (23)
\end{aligned}
$$

The last inequality is true, because in general, let a function $f(\cdot)$ correspond to $\hat{Q}_m$ and a function $g(\cdot)$ correspond to Q^*. Then:

$$|f(a) - g(a)| \leq \epsilon \Rightarrow |\max_a f(a) - \max_a g(a)| \leq \epsilon, \forall a \qquad (24)$$



And now, as can also be seen in the picture above, the maxima for f and g have to be within distance ϵ:

$$\max_a |f(a) - g(a)| \leq \epsilon \qquad (25)$$

Thus, by the definition of Δ_m and equation 23 we get:

$$|\hat{Q}_{m+1}(s,a) - Q^*(s,a)| \leq \gamma \Delta_m \qquad (26)$$

Initially,

$$\Delta_0 = \max_{s,a} |Q^*(s,a)| \qquad (27)$$

After all (s,a) pairs are visited:

$$\Delta_m \leq \gamma \Delta_0 \qquad (28)$$

If this happens k times:

$$\Delta_m \leq \gamma^k \Delta_0 \to 0 \qquad (29)$$

■

Question: What needs to be done when rewards are stochastic?

Answer: Merge old estimate with new estimate, where

$$\hat{Q}(s,a) = r + \gamma \max_{a'} \hat{Q}(s',a') \tag{30}$$

The function becomes now

$$\hat{Q}(s,a) = (1-\alpha)\hat{Q}(s,a) + \alpha[r + \gamma \max_{a'} \hat{Q}(s',a')] \tag{31}$$

A typical value for α is:

$$\alpha = \frac{1}{(\#\text{ times } (s,a) \text{ visited})} \tag{32}$$

It can be proved similarly that this also converges.