Devoxx BE Q Learning Workshop

Katharine Beaumont @katharineCodes

November 2017

1 Reinforcement Learning

What is reinforcement learning? Unlike supervised or unsupervised machine learning, we are focusing on goal-directed learning from interaction. As with other types of machine learning, there are strong mathematical foundations. Unlike other types of machine learning (or at least, more so), there are strong psychological and behaviouralist foundations. From page 1-2 of *Reinforcement Learning: An Introduction* (Second Edition) by Richard S Sutton and Andrew G Barto:

> Reinforcement learning, like many topics whose names end with "ing," such as machine learning and mountaineering, is simultaneously a problem, a class of solution methods that work well on the class of problems, and the field that studies these problems and their solution methods. Reinforcement learning problems involve learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. In an essential way these are closed-loop problems because the learning system's actions influence its later inputs. Moreover, the learner is not told which actions to take, as in many forms of machine learning, but instead must discover which actions yield the most reward by trying them out. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These three characteristics— being closed-loop in an essential way, not having direct instructions as to what actions to take, and where the consequences of actions, including reward signals, play out over extended time periods—are the three most important distinguishing features of the reinforcement learning problem.

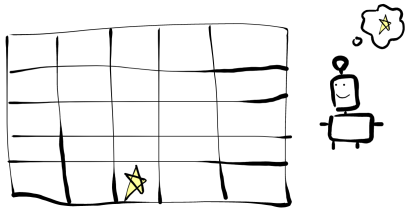
So we are looking at:

- A learner (or an agent)
- In an uncertain environment that gives rewards

- Who is not explicitly directed
- Who takes actions and learns over time

The agent needs to be able to sense the state of the environment in some way, and take actions that affect the state of the environment.

It also need to have goals, or a set of goals, that relate to the state of the environment.

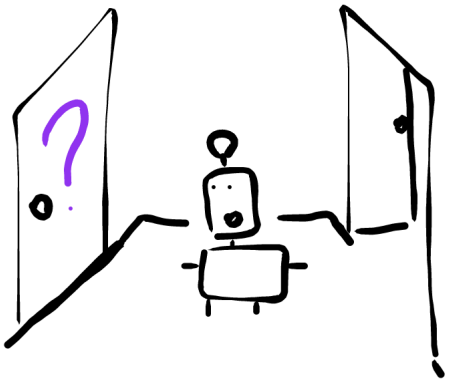


We want the agent to learn from it's experience.

1.1 Key Terms

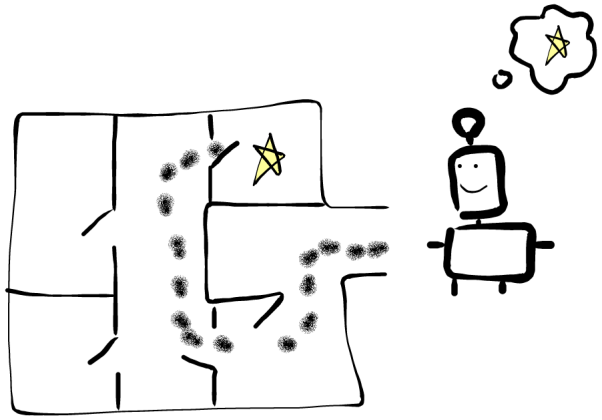
Exploration vs exploitation

In an unknown environment, the agent needs to explore in order to find the rewards. Once it has learned where the rewards are, it needs to choose actions that it has learned will lead to the rewards. This is a trade-off between exploration and exploitation. It is something that needs to be balanced in reinforcement learning problems: exploiting what an agent knows about an environment vs exploring new areas.



Policy

A policy defines an agent's way of behaving at a given time. For example, if we know there are a series of steps to take that lead to the goal in the best way (i.e. least number of steps, or maximum reward), then taking those steps is the optimum policy.



Value

The value of an action is a measure of how good it is. This is not the same as a reward. A reward is something we get from the environment, like in a Sonic the Hedgehog game, getting a gold coin. The value is how good a step we look in our journey to the reward.

2 QLearning

Image you are standing at the entrance to a maze. The walls are high and you can't see over them. You will know when you get to the centre of the maze because there is a statue signifying that you have reached it. But for now, you just have to wander blindly, not knowing which is the best path, until eventually you find the centre.

Once you've reached the centre, you start again at the entrance to the maze. Not this again! You start again, wandering through the corridors, until eventually you recognise that you're a few steps away from the end (maybe 2?) - now you know the centre of the maze, and the few steps before it.

Now, start AGAIN. This time, when you're 3 or 4 steps away from the centre, you remember - these are the 4 steps I need to take to get to the centre.

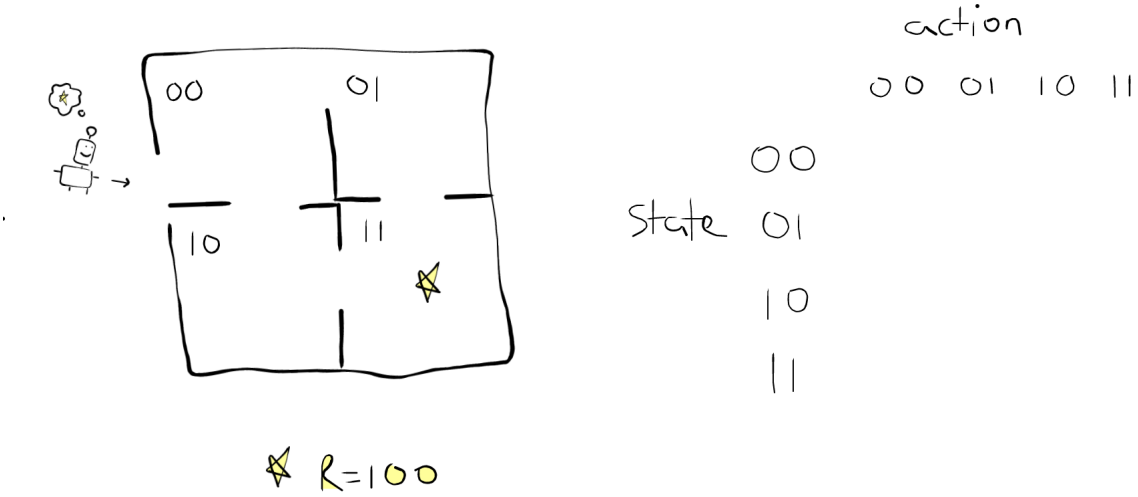
How many times do you have to do this before you remember the entire path, start to centre?

This is how Q Learning works. A *reward* signifies the centre of the maze. The agent starts out exploring randomly, until eventually it gets to the reward. It stores this value in a Q table, a memory of the reward got from the environment for moving from one room into the next: or, from taking an action in a state (action: move to the next room. State: current room). A discount factor (commonly denoted by γ) *discounts* the reward: so the room one step from the reward (or goal) has a slightly lower value in memory. The room two steps from the goal has a lower value in memory still. This memory is built up over several

iterations (or episodes). Eventually, the best path can be found by starting out, and selecting from memory the next best room.

A learning rate (commonly denoted by α) determines how fast you learn. The trade off between exploring or exploiting (randomly picking vs choosing the best action from memory) is denoted by epsilon, ϵ, which is treated as *probability explore*. So 0.1 is *explore 10% of the time*. In the beginning, the agent doesn't know anything about it's environment, so it will explore all the time. As it learns more, and builds up it's memory, this trade off will become more significant.

Example



The agent starts with an empty Q table, it has no memory of moving from any states into others (or, taking actions in a particular state, where the action is moving into another state).

We're going to denote a cell in this table as Q(S(t), A(t)). This is the memory of the value of taking a step, A, from S.

We update the Q table using this:

$$Q(S(t), A(t)) \leftarrow Q(S(t), A(t)) + \alpha[R(t+1) + \gamma max Q(S(t+1), a) - Q(S(t), A(t))].$$

When a step is taken, we increment the current Q value for taking that action, A(t), from state S(t).

We add:

- The reward we got from the environment
- To the discount, γ, times the maximum next best possible action from memory
- Minus the current memory Q(S(t), A(t))

- and we times it all by the learning rate, α

So if $\gamma = 0.7$, $\alpha = 0.5$. The agent is in 0,0. The first action chosen is "move to 0,1".

The Q table is empty, so Q(S(t), A(t)) is 0. There is no reward, so R(t+1) is 0. There is no memory of the best action to take from 0,1, so Q(S(t+1), a) is 0.

Now the agent chooses 1,1 as an action. The reward is 100, R(t+1). Again, the Q table is empty, so Q(S(t), A(t)) is 0, and Q(S(t+1), a) is 0.

α is 0.5, so we put 100 * 0.5 in the Q table, where the state S(t) is that we are in the room 0,1, and the action is 1,1.

	0,0	0,1	1,0	1,1
0,0	0	0	0	0
0,1	0	0	0	50
1,0	0	0	0	0
1,1	0	0	0	0

Once the agent moves, that is one episode, it has reached the goal.

Next episode: The agent is in 0,0. The first action chosen is "move to 0,1".

The Q table is empty for this move, so Q(S(t), A(t)) is 0. There is no reward, so R(t+1) is 0. However, in memory we can see that once we are in 0,1, the best is action is to move to 1,1, and the value for this is 50. So in our formula,

$$Q(S(t), A(t)) \leftarrow Q(S(t), A(t)) + \alpha[R(t+1) + \gamma maxQ(S(t+1), a) - Q(S(t), A(t))].$$

We plug in:

$$Q(S(t), A(t)) \leftarrow 0 + \alpha[0 + \gamma 50 - 0].$$

So we update the table:

	0,0	0,1	1,0	1,1
0,0	0	17.5	0	0
0,1	0	0	0	50
1,0	0	0	0	0
1,1	0	0	0	0

Now the agent chooses 1,1 as an action. The reward is 100, R(t+1). There are no subsequent actions, because we finish training in the goal state.

$$Q(S(t), A(t)) \leftarrow Q(S(t), A(t)) + \alpha[R(t+1) + \gamma maxQ(S(t+1), a) - Q(S(t), A(t))].$$

We plug in:

$$Q(S(t), A(t)) \leftarrow 50 + \alpha[100 + \gamma 50 - 50].$$

So the new Q value becomes: 92.5.

	0,0	0,1	1,0	1,1
0,0	0	17.5	0	0
0,1	0	0	0	92.5
1,0	0	0	0	0
1,1	0	0	0	0