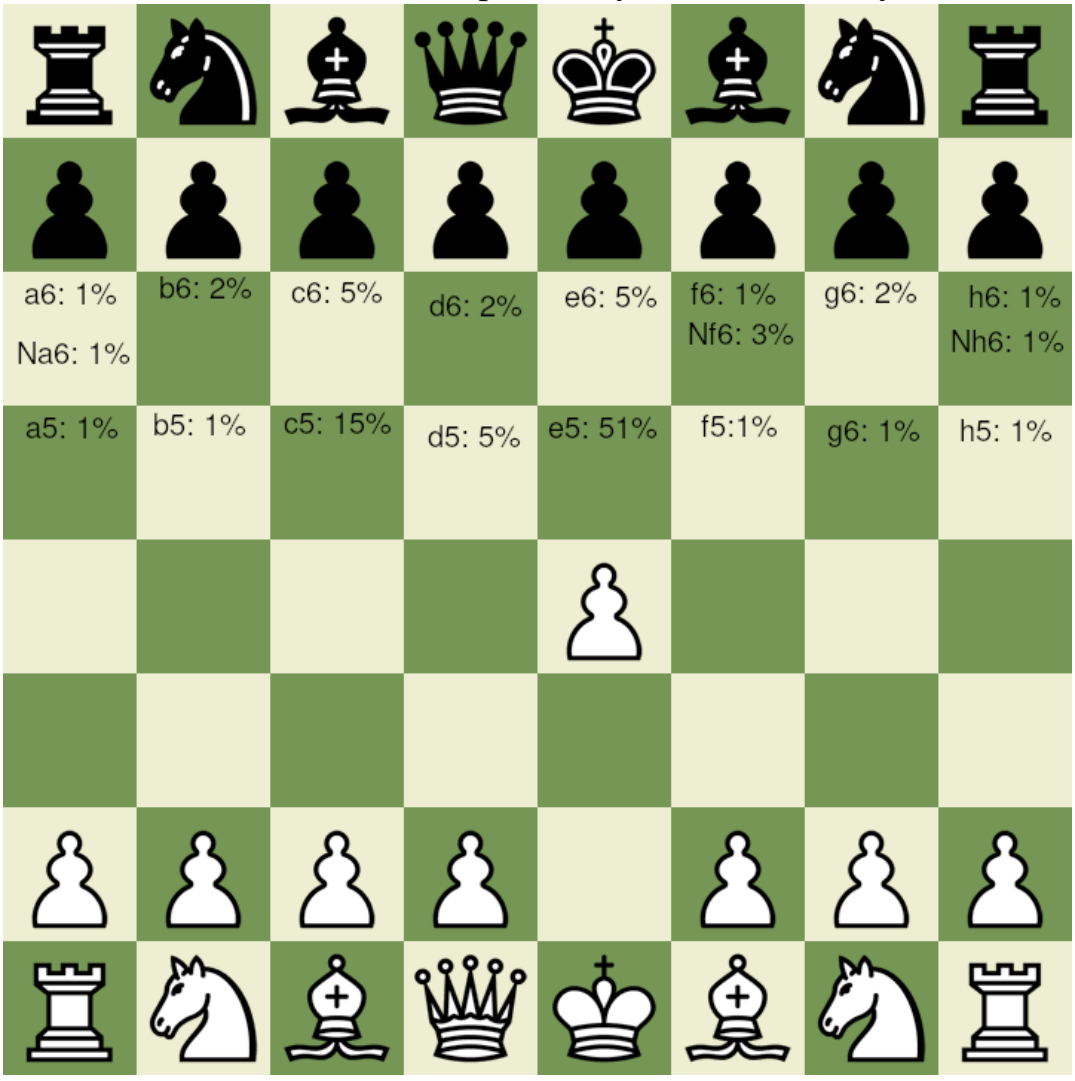
Markov Decision Process and Understanding the Bellman Equation

Today I will discuss MDP and finding the optimal value function. You should already have some understanding of bayesian probability and markov chains. MDP is critical to Reinforcement Learning as many more advanced models are based on it. All this information is based on Reinforcement Learning: An Introduction, Sutton and Barto, 2nd Edition which you can get for free here. I highly recommend it.

The Problem

An environment can be deterministic or stochastic. When it's deterministic, the action we take will result in a guaranteed state. However, when stochastic, there can be a range of possible states, each with their own probability. For example, take the chess board shown below. After we take action ‘e4’ we have the possibility to arrive in many different states, denoted by s’.



For example, the probability of the state where black plays “d5” is conditioned on our action e4 and the state, which is the board below. We can represent the chess space as markov chains, where the probability of the current state is dependent on the previous state and action. However there is a problem, we do not know what is the best move to make at each state, that

is, we do not know the value at ANY of the states. This is where reinforcement learning and the markov decision process comes in.

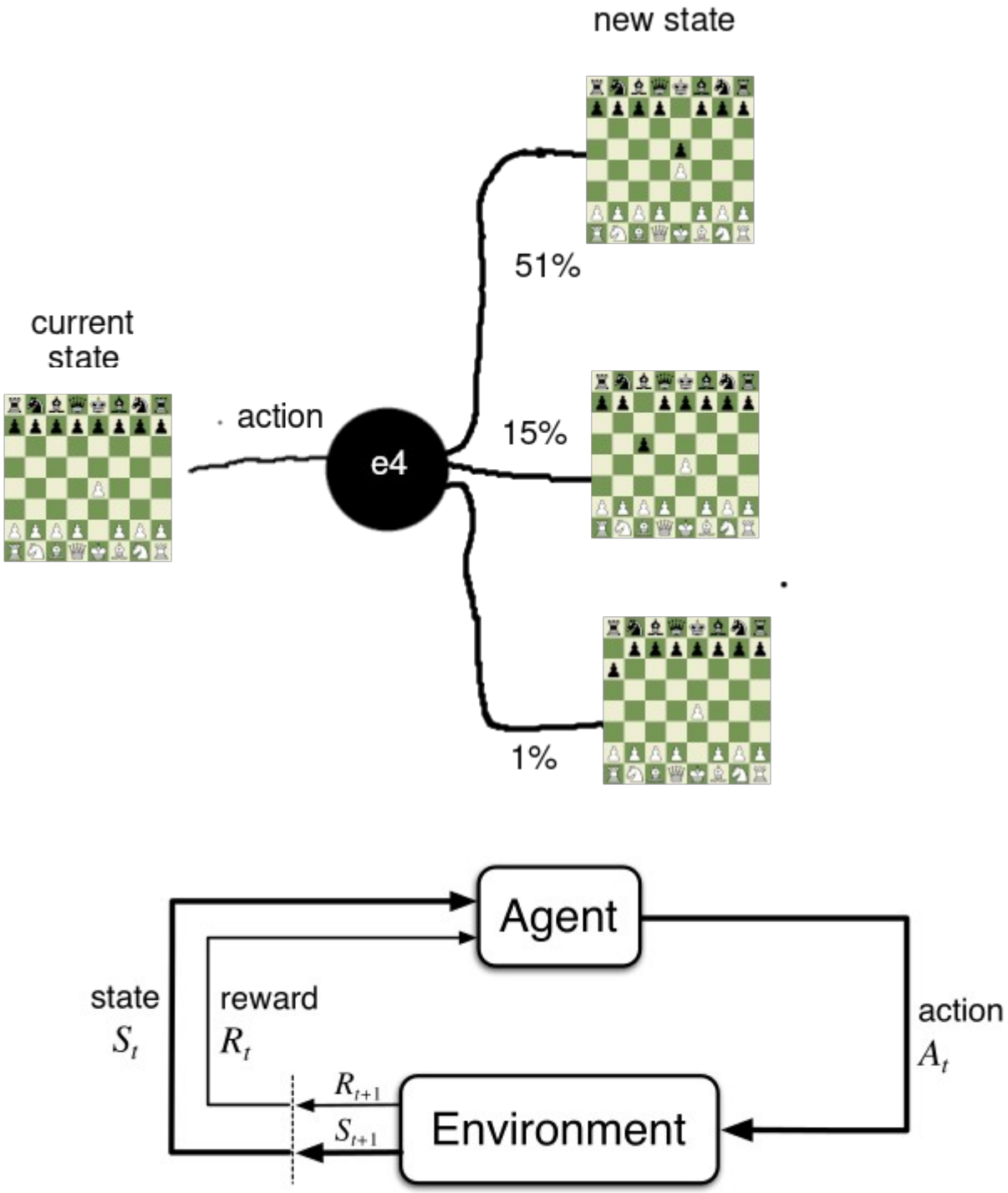


Figure 3.1: The agent–environment interaction in a Markov decision process.

The goal of the agent is quantified in terms of R, the reward. And thus must try to accumulate the maximum reward possible. In a greedy model, the agent only considers short term rewards, which might result in a lower total cumulative reward. Thus more advanced models must be formulated to try to maximize the expected reward(denoted by G_t).
At state S_t where t is the current time step, the agent makes an action A_t in the enviorment, which results in it receiving a numerical reward R_t and acquires a

representation of the next state. The history of the agent in the environment can be represented as a list of state, actions, and rewards. An episode is when we perform a series of actions. These episodes will used to update the value of state-action pairs.

$$S_t, A_t, R_t, S_{t+1}, A_{t+1}, R_{t+1}, S_{t+2}, A_{t+2}, S_{t+2} \ldots$$

We can also describe S_t and A_t as random variables dependent on previous state and actions one time iteration ago.

$$p(s', r|s, a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

Where s' is the next state. The probability of the current state and the reward is conditionally dependent on the current state in the current action. This definition allows us to compute anything about the enviorment, particularly, **the state transition probabilities,** which is used to determine $q_t(s, a)$, the estimated value of a state-action pair.

For example:

The State Transition Probability

$$SxSxA \rightarrow probability$$
$$p(s'|s, a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in R} p(s', r|s, a)$$

The expected rewards for state-action pairs

$$SxSxA \rightarrow reward$$
$$r(s, a) = E[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in R} r \sum_{s' \in S} p(s', r|s, a)$$

Usually all four parameters are used, however, sometimes these formulas are useful. Later on, the probability of each state occurring will be used to determine the value of a state.

Earlier I mentioned we need to find G, the expected return, so that we can determine the return, the cumulative rewards of an action-state pair. To do so, we need to model G in a way that is useful with our knowledge of the environment.

$$G_t = R_{t+1} + \alpha R_{t+2} + \alpha^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \alpha^k R_{t+k+1}$$

Where α is the **discount rate,** an arbitrary constant between 0 and 1 that determines how much the agent should consider future rewards. At 0 it is **myopic** and only considers immediate rewards, while at 1 it is **farsighted,** weighing immediate and future rewards equally.

We can represent G_t in terms of future expected rewards as follows:

$$G_t = R_{t+1} + \alpha R_{t+2} + \alpha^2 R_{t+3} \ldots$$
$$G_t = R_{t+1} + \alpha(R_{t+2} + \alpha R_{t+3} \ldots)$$
$$G_t = R_{t+1} + \alpha G_{t+1}$$

We can recursively calculate G.

Important: In reinforcement learning, we can acquire thousands to millions episodes on the environment, resulting in a distribution of values for . G_t **The value of the current state and action can be seen as the expected value of all G_t's found through episodic exploration.** That is, as we experience more episodes, will converge to the true value of the state-action pair. Think of it like a science experiment: as we take more samples, our estimation becomes better. Here, as we acquire more episodes, the expected value of state should converge to its true value. Convergence is absolutely necessary. If our model doesn't converge, then it will not be effective. Given that, if we have a policy , π then we can estimate how well(value) the policy π does on state s(this can be described as $v_\pi(s)$)

The expected value of of s, given π is the expected value of all episodes will start from s.

Bellman Equation Explained

The Bellman Equation is necessary to optimize our decision process in reinforcement learning. The equation is a composed of state probabilities and rewards. Note that $\pi(a|s)$ is the conditional probability of action a and state s given π .

$$v_\pi(s) = E_\pi[G_t|S_t = s]$$
$$v_\pi(s) = E_\pi[R_{t+1} + \alpha G_{t+1}|S_t = s]$$
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \alpha E_\pi[Gt+1|S_{t+1} = s']]$$
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \alpha v_\pi(s')], \; for all \; s \in S$$

This is the Bellman Equation for π . We take an action a based on our policy, then, we are presented with several states-reward pairs s',r. We take the expected value of possible next states and the current states reward to get the expected value for state s **and weigh each state s' expected reward by its probability of occuring.** Important Note: this is for a stochastic policy, which is why we need $\pi(a|s)$ and $p(s', r|s, a)$, otherwise in a deterministic policy the value of the state will just be the expected value of episodes.

Understanding that the value of a state is the convergence of the expected value of episodes weighted by their probabilities is essential to fully understand Markov Decision Process, Monte Carlo, Temporal Difference, Q Learning, and other Reinforcement Learning models.

Optimal Value Function

Before we continue, we need to know the difference between function q and v. Function "q" describes the value of a state-action pair while v describes the value of a state. For problems that are dependent on both its state and action will need to utilize the q function to describe the model.

After we determine the estimated values for each state-action pair, we will want to execute the best possibly action:

$$v_*(s) = \max_\pi v_\pi(s)$$
$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

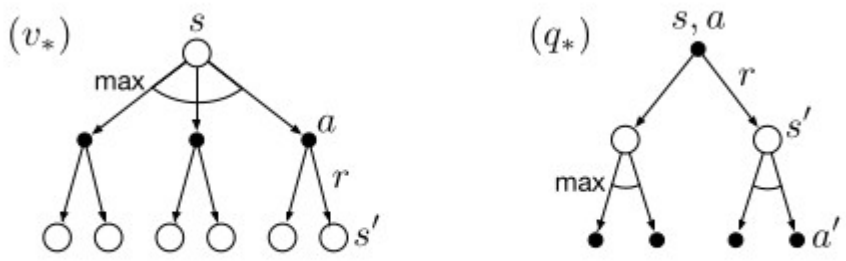


Figure 3.4: Backup diagrams for v_* and q_*

Black circles represent actions and white circles represent states.

Summary

MDP is used in stochastic spaces and utilizes expected value over episodes to estimate the value of a given state action pair. **MDP** use the Bellman Equation to estimate the value of a given state.

Earlier I referenced chess to describe the intuition behind MDP. However, with environments with such a large amount of possibilities, MDP would be a poor model due to the computation power needed to explore all the different paths. MDP is better suited for problems with a much smaller space, that are dependent on history, and the probabilities of each state are known, and where its goal can be quantified. Indeed, MDP is not an end all solution to dynamic programming. However, the theory behind MDP and Bellman Equation are critical to other models that I will explain in another blog, in particular, Monte Carlo and Temporal Difference.