Exercise 3.1 Devise three example tasks of your own that fit into the MDP framework, identifying for each its states, actions, and rewards. Make the three examples as *different* from each other as possible. The framework is abstract and flexible and can be applied in many different ways. Stretch its limits in some way in at least one of your examples. ☑

1. Any Board (Checkers, Chess, ...) or Card (Blackjack, Poker, ...) Game
Consider any board or card game. It's irrelevant whether the transitions are stochastic (i.e. involve a dice roll) or deterministic (i.e. like checkers stone jumps). Both can be captured within the MDP framework.

On the high level a match consists of human players taking turns and choosing from the actions that the game rules allow them to do. The nature of match naturally breaks the problem into episodes terminated by the win, loss or tie. To actually train an agent capable of playing a particular game one needs to equip the game logic with and opponent AI to substitute human players.

Flip side of providing the AI as part of the environment is that the exploitative nature of RL agents can learn particular quirks and bugs of the AI that don't transfer well into matches against humans.

- **States:** Situation on the game board or the visible cards (grid with values representing the pieces or slots with agent's and dealer's cards)
- **Actions:** Move available from the current situations (pawn to B2 or hit/stand)
- **Transitions:** Determined by the game rules. AI substitutes the moves otherwise done by human opponents. AI this way becomes a part of the environment for the RL agent.
- **Rewards:** +1/0/-1 for win/tie/loss

2. Competitive Arms Race Between Two Agents
Consider the MDP as described in example 1 but with the opponent AI controlled by another RL agent. Each agent would see the environment exactly as described above but this setup doesn't require pre-programmed AI for training and learning can be achieved by competitive arms race between the agents.

From the local perspective of an agent the opponent moves can be considered as a part of the environment. And vice versa from the opponent's perspective. Because both agents are learning the environment is non-stationary and changes over time similarly to Exercise 2.11.

If the learning process is carefully tuned and controlled not to progress too quickly it should be possible to achieve optimal play from random initial behavior. It is crucial to keep the win/loss ration around 50% to prevent one of the agents learning much faster and exploiting the other one. Agent that would always lose wouldn't receive any meaningful learning signal about the correctness of it's actions and there would be no way for it to recover from this state.

3. A Supervised Learning Task (Image Classification or House Price Regression)
This is in some ways underutilizing and in some ways stretching the MDP concept. Underutilization lays in the trivial transition dynamics that "feeds" the agent sequentially with the dataset. Stretching is in actions and rewards that represent the desired output of the learner (classification labels or regressed values)

This representation allows any supervised learning problem to be framed as an MDP with the following structure:

- **States:** Input that would be provided to the learning system (images or feature vectors)
- **Actions:** Class that the learning system believes that image belongs to or a regressed value (cat, dog or house price)
- **Transitions:** Transition to next sample from the training dataset (following image or next feature vector)
- **Rewards:** Soft-max or MSE loss of the sample provided as state

Exercise 3.2 Is the MDP framework adequate to usefully represent *all* goal-directed learning tasks? Can you think of any clear exceptions? ☑

There are many tasks that doesn't fit into the framework. First, MDP as presented in this chapter is formulated for problems with discrete states, actions and rewards.

There are many naturally continuous tasks that don't easily match the discrete MDP template. Discretization unfortunately makes the problem very hard to solve for high dimensional action and state spaces because the number of available actions or states grows exponentially with the dimension of the space.

Another problem is the goal itself. In many cases it's not clear how to formulate a reward system that would yield the desired behavior of the agent. The relationship between rewards and actions is not straightforward. Simple ideas frequently fail with unintended consequences. The infamous example of this is the paperclip problem where a successful RL agent turns the entire universe into paperclips...

Many problems, especially in engineering, have two or more objectives that are in direct conflict. For example a typical requirements for a design of a mechanically stressed part is to be simultaneously lightweight and stiff. It's not clear how to mix the two criteria into a single reward value even though each of them is easily evaluated for a particular design.

Exercise 3.3 Consider the problem of driving. You could define the actions in terms of the accelerator, steering wheel, and brake, that is, where your body meets the machine. Or you could define them farther out—say, where the rubber meets the road, considering your actions to be tire torques. Or you could define them farther in—say, where your brain meets your body, the actions being muscle twitches to control your limbs. Or you could go to a really high level and say that your actions are your choices of *where* to drive. What is the right level, the right place to draw the line between agent and environment? On what basis is one location of the line to be preferred over another? Is there any fundamental reason for preferring one location over another, or is it a free choice? ☑

I don't think there's one right level to look at the driving problem. All the mentioned levels play a critical role and a real world driving agent needs to be able to solve all of them at the same time.

The mentioned tasks together form a hierarchy of RL problems at different timescales and with different rewards, state, and actions. Higher level system that makes choices "where" to drive naturally receives rewards based on optimality of it's pathfinding. On a lower level, the "limb" system executes the maneuvers dictated by the choices of the "where" system and receives rewards based on how precisely it follows the planned path. At the lowest level the "tire" system is responsible to adhering to limits of traction and is penalized for large tire slip so it effectively learns the role of an electronic traction control system under acceleration or ABS under braking.

The reason why it's beneficial to break the problem into a hierarchy of tasks at different timescales is the rewards sparsity. Each system is executed at a roughly at an order of magnitude smaller time-step than the previous one. System "where" runs at ~1Hz, "limb" at ~10Hz and "tire" at ~100Hz. The sparsity of rewards determines the difficulty of the RL problem and how hard is it for an agent to solve it.

Breaking the problem into the smaller chunks naturally sidesteps this issue. It would be much harder to learn an agent operating at the 100Hz time-step that would receive rewards for correct maneuvers like turns or merges. These typically run over the range of 10s of seconds and it's not clear how to distribute the reward over 1000s of decisions taken by the "tire" agent.

Exercise 3.4 Give a table analogous to that in Example 3.3, but for $p(s', r \mid s, a)$. It should have columns for s, a, s', r, and $p(s', r \mid s, a)$, and a row for every 4-tuple for which $p(s', r \mid s, a) > 0$. ☑

Because rewards are deterministic the probability distribution. from the table wont change a lot. Columns will shuffle and change name but all entries other than already present in the table will have 0 probability.

$p(s', r \mid s, a)$ r

s	a	s'	~~$p(s' \mid s, a)$~~	~~$r(s, a, s')$~~
high	search	high	α	r_{search}
high	search	low	$1-\alpha$	r_{search}
low	search	high	$1-\beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	-

Exercise 3.5 The equations in Section 3.1 are for the continuing case and need to be modified (very slightly) to apply to episodic tasks. Show that you know the modifications needed by giving the modified version of (3.3).

$$\sum_{s' \in \mathcal{S}^+} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}^+, a \in \mathcal{A}(s).$$

Exercise 3.6 Suppose you treated pole-balancing as an episodic task but also used discounting, with all rewards zero except for -1 upon failure. What then would the return be at each time? How does this return differ from that in the discounted, continuing formulation of this task?

$G_T = 0$

$G_{T-1} = R_T + \gamma G_T = -1$

$G_{T-2} = \underbrace{R_{T-1}}_{0} + \gamma G_{T-1} = -\gamma$

$G_{T-3} = R_{T-2} + \gamma G_{T-2} = -\gamma^2$

$G_i = -\gamma^{T-i-1} \quad i \neq 0$

return at time i

Exercise 3.7 Imagine that you are designing a robot to run a maze. You decide to give it a reward of $+1$ for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes—the successive runs through the maze—so you decide to treat it as an episodic task, where the goal is to maximize expected total reward (3.7). After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?

$$\max G_0 = \max \sum_{i=0}^{T} R_i = \max \overbrace{0 + 0 + \dots}^{T-1} + 1 = 1$$

For every terminated episode agent recieves the same reward irrespectively of how long it took to escape the maze. Because solving the maze in fewer timesteps isn't rewarded there's no incentive for the agent to do it.

Exercise 3.8 Suppose $\gamma = 0.5$ and the following sequence of rewards is received $R_1 = -1$, $R_2 = 2$, $R_3 = 6$, $R_4 = 3$, and $R_5 = 2$, with $T = 5$. What are $G_0, G_1, \dots, G_5$? Hint: Work backwards.

$G_5 = 0$

$G_4 = R_5 + \gamma G_5 = 2$

$G_3 = R_4 + \gamma G_4 = 4$

$G_2 = R_3 + \gamma G_3 = 8$

$G_1 = R_2 + \gamma G_2 = 6$

$G_0 = R_1 + \gamma G_1 = 2$

Exercise 3.9 Suppose $\gamma = 0.9$ and the reward sequence is $R_1 = 2$ followed by an infinite sequence of 7s. What are G_1 and G_0?

$$G_1 = \sum_{k=0}^{+\infty} 0.9^k \cdot 7 = \frac{7}{1-0.9} = 70$$

$$G_0 = R_1 + \gamma G_1 = 2 + 0.9 \cdot 70 = 65$$

$$G_t = \sum_{k=0}^{+\infty} \gamma^k R_{t+1+k} = R_{t+1} + \gamma G_{t+1}$$

$$= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2} + \dots$$

Exercise 3.10 Prove the second equality in (3.10).

$$S_N = \sum_{k=0}^{N} \gamma^k = 1 + \gamma^1 + \gamma^2 + \dots + \gamma^N = 1 + \gamma \underbrace{\left(1 + \gamma + \dots + \gamma^{N-1}\right)}_{\sum_{k=0}^{N-1} \gamma^k = S_{N-1}}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}.$$

We have a recurrent relation for two following members of the series:

$$S_N = 1 + \gamma S_{N-1}$$

At the limit $N \to +\infty$ two subsequent members of the series are equal:

$$S_\infty = 1 + \gamma S_\infty \Rightarrow S_\infty = \sum_{k=0}^{+\infty} \gamma^k = \frac{1}{1-\gamma}$$

Exercise 3.11 If the current state is S_t, and actions are selected according to stochastic policy π, then what is the expectation of R_{t+1} in terms of π and the four-argument function p (3.2)?

$$\mathbb{E}_\pi\left[R_{t+1} \mid S_t\right] = \sum_{\forall r} r \, p_\pi(r \mid s_t) = \sum_{\forall r \forall s' \forall a} r \, p(s', r \mid s_t, a) \, \pi(a \mid s_t)$$

Exercise 3.12 Give an equation for v_π in terms of q_π and π.

Exercise 3.13 Give an equation for q_π in terms of v_π and the four-argument p.

$$v_\pi(s) = \sum_{\forall a} \pi(a \mid s) \, q_\pi(s, a)$$

$$q_\pi(s, a) = \sum_{\forall s' \forall r} p(s', r \mid s, a) \left(r + \gamma v_\pi(s')\right)$$

Exercise 3.14 The Bellman equation (3.14) must hold for each state for the value function v_π shown in Figure 3.2 (right) of Example 3.5. Show numerically that this equation holds for the center state, valued at +0.7, with respect to its four neighboring states, valued at +2.3, +0.4, −0.4, and +0.7. (These numbers are accurate only to one decimal place.)

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{r} \sum_{s'} p(s', r \mid s, a)\left(r + \gamma v_\pi(s')\right)$$

$$0.7 = \frac{1}{4} \cdot 0.9 \left(0.7 + 2.3 + 0.4 - 0.4\right) = \frac{2.7}{4} \approx 0.68$$

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Exercise 3.15 In the gridworld example, rewards are positive for goals, negative for running into the edge of the world, and zero the rest of the time. Are the signs of these rewards important, or only the intervals between them? Prove, using (3.8), that adding a constant c to all the rewards adds a constant, v_c, to the values of all states, and thus does not affect the relative values of any states under any policies. What is v_c in terms of c and γ?

$$G_t = \sum_{k=0}^{+\infty} \gamma^k R_{k+t+1} \qquad \bar{G}_t = \sum_{k=0}^{+\infty} \gamma^k (R_{k+t+1} + c) = G_t + \sum_{k=0}^{+\infty} \gamma^k \cdot c = G_t + \frac{c}{1-\gamma}$$

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$

$$\bar{v}_\pi(s) = \mathbb{E}_\pi\left[\bar{G}_t \mid S_t = s\right] = \mathbb{E}\left[G_t + \frac{c}{1-\gamma} \mid S_t = s\right] = v_\pi(s) + \underbrace{\frac{c}{1-\gamma}}_{= \text{const} = v_c}$$

Value function will change values but the induced policy will stay the same when a constant gets added to all rewards.

Exercise 3.16 Now consider adding a constant c to all the rewards in an episodic task, such as maze running. Would this have any effect, or would it leave the task unchanged as in the continuing task above? Why or why not? Give an example.

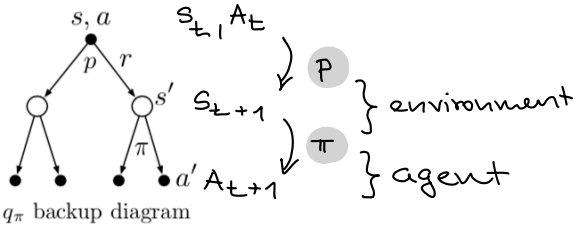
$$G_t = \sum_{k=0}^{T} R_{k+t+1} \qquad \bar{G}_t = \sum_{k=0}^{T} (R_{k+t+1} + c) = G_t + cT$$

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$

$$\bar{v}_\pi(s) = \mathbb{E}_\pi\left[\bar{G}_t \mid S_t = s\right] = v_\pi(s) + \underbrace{c\,\mathbb{E}_\pi\left[T \mid S_t = s\right]}_{\neq \text{const}}$$

Value function will change differently for each state. Magnitude of the change is proportional to the mean # timesteps to the end of the episode. Generally it takes a different # timesteps to reach the end from every state, so the implicit policy will be different. States further away from the end will receive a higher value increase which will incentivize agent to not terminate the episode. If c is high enough agent may run in a loop infinitely trying to collect infinite reward.

Exercise 3.17 What is the Bellman equation for action values, that is, for q_π? It must give the action value $q_\pi(s,a)$ in terms of the action values, $q_\pi(s',a')$, of possible successors to the state–action pair (s,a). Hint: The backup diagram to the right corresponds to this equation. Show the sequence of equations analogous to (3.14), but for action values.



q_π backup diagram

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t=s, A_t=a] =$$

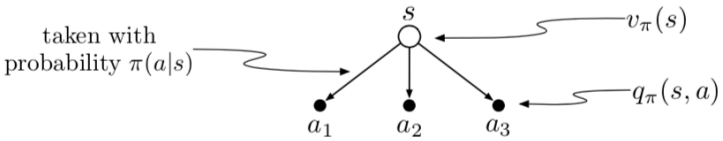
$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t=s, A_t=a]$$

$$= \sum_{\forall s' \forall r} p(s',r \mid s,a)\left(r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1}=s', A_t=a]\right)$$

$$= \sum_{\forall s' \forall r} p(s',r \mid s,a)\left(r + \gamma \sum_{\forall a'} \pi(a' \mid s')\, q(s',a')\right)$$

$$\sum_{\forall a} \pi(a \mid s')\, \mathbb{E}_\pi[G_{t+1} \mid S_{t+1}=s', A_{t+1}=a'] \qquad q(s',a')$$

Exercise 3.18 The value of a state depends on the values of the actions possible in that state and on how likely each action is to be taken under the current policy. We can think of this in terms of a small backup diagram rooted at the state and considering each possible action:

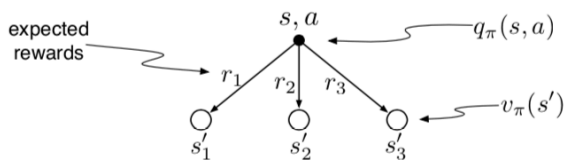


Give the equation corresponding to this intuition and diagram for the value at the root node, $v_\pi(s)$, in terms of the value at the expected leaf node, $q_\pi(s,a)$, given $S_t = s$. This equation should include an expectation conditioned on following the policy, π. Then give a second equation in which the expected value is written out explicitly in terms of $\pi(a|s)$ such that no expected value notation appears in the equation.

$$v_\pi(s) = \mathbb{E}_\pi[q_\pi(S_t,a) \mid S_t=s]$$

$$v_\pi(s) = \sum_{\forall a} \pi(a \mid s)\, q_\pi(s,a)$$

Exercise 3.19 The value of an action, $q_\pi(s,a)$, depends on the expected next reward and the expected sum of the remaining rewards. Again we can think of this in terms of a small backup diagram, this one rooted at an action (state–action pair) and branching to the possible next states:

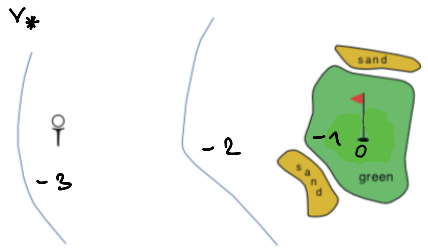


Give the equation corresponding to this intuition and diagram for the action value, $q_\pi(s,a)$, in terms of the expected next reward, R_{t+1}, and the expected next state value, $v_\pi(S_{t+1})$, given that $S_t=s$ and $A_t=a$. This equation should include an expectation but *not* one conditioned on following the policy. Then give a second equation, writing out the expected value explicitly in terms of $p(s',r \mid s,a)$ defined by (3.2), such that no expected value notation appears in the equation.

$$q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t=s, A_t=a]$$

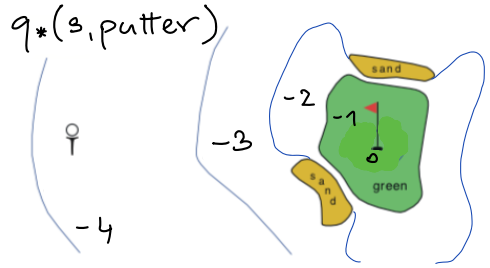
$$q_\pi(s,a) = \sum_{\forall r \forall s'} p(s',r \mid s,a)\left(r + \gamma v_\pi(s')\right)$$

Exercise 3.20 Draw or describe the optimal state-value function for the golf example.



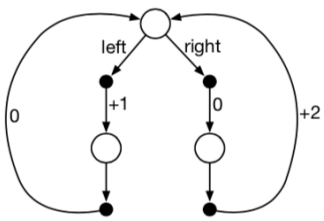
Optimal state value function is equal to $q_*(s, \text{driver})$ everywhere except for the green. There it's better to use the putter to increase the radius where the hole can be reached in one shot.

Exercise 3.21 Draw or describe the contours of the optimal action-value function for putting, $q_*(s, \texttt{putter})$, for the golf example.



Optimal action value function for putter is equal to v^* minus one extra shot for the first putter shot in region far from the green. When in region within putting range q_* is equal to $q(s, \text{putter})$ as putting is the best course of actions from this region onwards.

Exercise 3.22 Consider the continuing MDP shown on to the right. The only decision to be made is that in the top state, where two actions are available, left and right. The numbers show the rewards that are received deterministically after each action. There are exactly two deterministic policies, π_{left} and π_{right}. What policy is optimal if $\gamma = 0$? If $\gamma = 0.9$? If $\gamma = 0.5$?



$$\mathbb{E}_{\pi_{left}}[G_0 | S_{top}] = \sum_{k=0}^{+\infty} \gamma^{2k} \cdot (+1 + \gamma \cdot 0) = \frac{1}{1-\gamma^2} = \begin{cases} 1 & (\gamma = 0)^* \\ 5\,5/19 & (\gamma = 0.9) \\ 1\,1/3 & (\gamma = 0.5)^* \end{cases}$$

$$\mathbb{E}_{\pi_{right}}[G_0 | S_{top}] = \sum_{k=0}^{+\infty} \gamma^{2k} \cdot (0 + \gamma \cdot +2) = \frac{2\gamma}{1-\gamma^2} = \begin{cases} 0 & (\gamma = 0) \\ 9\,9/19 & (\gamma = 0.9)^* \\ 1\,1/3 & (\gamma = 0.5)^* \end{cases}$$

$$\underbrace{\sum_{k=0}^{+\infty} \gamma^k}_{1/(1-\gamma)} = \underbrace{\sum_{k=0}^{+\infty} \gamma^{2k}}_{S} + \underbrace{\sum_{k=0}^{+\infty} \gamma^{2k+1}}_{\gamma S} \Rightarrow S = \frac{1}{1-\gamma^2}$$

Exercise 3.23 Give the Bellman equation for q_* for the recycling robot.

$$q_*(h, w) = r_{wait} + 1 \cdot \gamma \max_a q_*(h, a)$$

$$q_*(h, s) = r_{search} + \alpha \cdot \gamma \max_a q_*(h, a) + (1-\alpha) \cdot \gamma \max_a q_*(l, a)$$

$$q_*(h, re) = \emptyset$$

$$q_*(l, w) = r_{wait} + 1 \cdot \gamma \max_a q_*(l, a)$$
$$q_*(l, s) = r_{search} + \beta \cdot \gamma \max_a q_*(l, a) + (1-\beta) \cdot \gamma \max_a q_*(h, a)$$
$$q_*(l, re) = 0 + 1 \cdot \gamma \max_a q_*(h, a)$$

Exercise 3.24 Figure 3.5 gives the optimal value of the best state of the gridworld as 24.4, to one decimal place. Use your knowledge of the optimal policy and (3.8) to express this value symbolically, and then to compute it to three decimal places. ☑

$$v_*(A) = \sum_{k=0}^{+\infty} \gamma^{5k} \underbrace{\left(\gamma^0 10 + \gamma^1 \cdot 0 + \gamma^2 \cdot 0 + \gamma^3 \cdot 0 + \gamma^4 \cdot 0\right)}_{\text{Reward for completing one } A \to A' \to A \text{ cycle}} = \frac{10}{1-\gamma^5} = 24.419$$

Exercise 3.25 Give an equation for v_* in terms of q_*. ☑

$$v_*(s) = \max_a q_*(s, a)$$

Exercise 3.26 Give an equation for q_* in terms of v_* and the four-argument p. ☑

$$q_*(s, a) = \max_a \sum_{\forall s' \forall r} p(r, s' \mid s, a)\left(r + \gamma v_*(s')\right)$$

Exercise 3.27 Give an equation for π_* in terms of q_*. ☑

$$\pi_*(a \mid s) = \begin{cases} 1 & \text{if } a = \arg\max_a q_*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

Exercise 3.28 Give an equation for π_* in terms of v_* and the four-argument p. ☑

$$\pi_*(a \mid s) = \begin{cases} 1 & \text{if } a = \arg\max_a \sum_{\forall s' \forall r} p(r, s' \mid s, a)\left(r + \gamma v_*(s')\right) \\ 0 & \text{otherwise} \end{cases}$$

Exercise 3.29 Rewrite the four Bellman equations for the four value functions (v_π, v_*, q_π, and q_*) in terms of the three argument function p (3.4) and the two-argument function r (3.5). ☑

$$v_\pi(s) = \sum_{\forall a} \pi(a \mid s) \sum_{\forall s'} p(s' \mid s, a)\left(r(s', a) + \gamma v_\pi(s')\right)$$
$$v_*(s) = \max_{\forall a} \sum_{\forall s'} p(s' \mid s, a)\left(r(s', a) + \gamma v_*(s')\right)$$
$$q_\pi(s, a) = r(s, a) + \gamma \sum_{\forall s'} p(s' \mid s, a) \sum_{\forall a'} \pi(a' \mid s') q_\pi(s', a')$$
$$q_*(s, a) = r(s, a) + \gamma \sum_{\forall s'} p(s' \mid s, a) \max_{a'} \pi(a' \mid s') q_*(s', a')$$

$$r(s, a) = \sum_{\forall s' \forall r} r\, p(s', r \mid s, a)$$
$$p(s' \mid s, a) = \sum_{\forall r} p(s', r \mid s, a)$$