The Full Reinforcement Learning Problem

So far: immediate reward after each action (n-armed bandit problem)
Now: *delayed rewards*, can be in *different states*
Example: Maze Task

![Maze Task Diagram]

Figure 9.7 The maze task. The rat enters the maze from the bottom and has to move forward. Upon reaching one of the end points (the shaded boxes), it receives the number of food pellets indicated and the trial ends. Decision points are A, B, and C.

Amount of reward after decision at second intersection depends on action taken at first intersection.
Agent and environment interact at discrete time steps: \( t = 0, 1, 2, \ldots \)

Agent observes state at step \( t \): \( s_t \in S \)

produces action at step \( t \): \( a_t \in A(s_t) \)

gets resulting reward: \( r_{t+1} \in \mathcal{R} \)

and resulting next state: \( s_{t+1} \)
The Markov Property

- By “the state” at step \(t\), we mean whatever information is available to the agent at step \(t\) about its environment.
- The state can include immediate “sensations,” highly processed sensations, and structures built up over time from sequences of sensations.
- Ideally, a state should summarize past sensations so as to retain all “essential” information, i.e., it should have the **Markov Property**:

\[
\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\} = \\
\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\}
\]

for all \(s', r\), and histories \(s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\).

“history doesn’t matter!”
Markov Decision Processes

- If a reinforcement learning task has the Markov Property, it is basically a **Markov Decision Process (MDP)**.
- If state and action sets are finite, it is a **finite MDP**.
- To define a finite MDP, you need to give:
  - **state and action sets**
  - one-step “dynamics” defined by **transition probabilities**:
    \[
    P_{ss'}^a = \Pr \left\{ s_{t+1} = s' \mid s_t = s, a_t = a \right\} \quad \text{for all } s, s' \in S, a \in A(s).
    \]
  - **reward probabilities**:
    \[
    R_{ss'}^a = E \left\{ r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s' \right\} \quad \text{for all } s, s' \in S, a \in A(s).
    \]
An Example Finite MDP

Recycling Robot

- At each step, robot has to decide whether it should (1) actively search for a can, (2) wait for someone to bring it a can, or (3) go to home base and recharge.
- Searching is better but runs down the battery; if runs out of power while searching, has to be rescued (which is bad).
- Decisions made on basis of current energy level: high, low.
- Reward = number of cans collected
Recycling Robot MDP

\[ S = \{ \text{high, low} \} \]
\[ A(\text{high}) = \{ \text{search, wait} \} \]
\[ A(\text{low}) = \{ \text{search, wait, recharge} \} \]

\[ R^{\text{search}} = \text{expected no. of cans while searching} \]
\[ R^{\text{wait}} = \text{expected no. of cans while waiting} \]

\[ R^{\text{search}} > R^{\text{wait}} \]

“transition graph”
Solving the Maze Problem

Assumptions:
• State is **fully observable** (in contrast to only **partially observable**), i.e. the rat knows exactly where it is at any time.
• Actions can have deterministic or probabilistic consequences.

Idea: maintain and improve a **stochastic policy**, determining actions at each decision point (A,B,C) using action values and softmax decision.

Actor Critic Learning:
**critic:** use temporal difference learning to predict future rewards from A,B,C if current policy is followed
**actor:** maintain and improve the policy

figure taken from Dayan&Abbott
Actor-Critic Method

Agent

Policy

Value Function

Environment

Actor

Critic

State

Action

Reward

TD error
Introduce \textit{state variable} $u$ to describe whether rat is at A,B,C. Also introduce \textit{action value vector} $Q(u)$ describing the policy (left/right). Softmax rule assigns probability of action $a$ based on action values.

Immediate reward for taking action $a$ in state $u$: $r_a(u)$

Expected future reward for starting in state $u$ and following current policy: $v(u)$ (\textit{state value function}). The rat’s estimate for this is denoted by $w(u)$.
**Policy Iteration**

**Two Observations:**
We need to estimate the values of the states, but these depend on the rat’s current policy.

We need to chose better actions, but what action is “better” depends on the values estimated above.

**Idea** (policy iteration): just iterate the two processes
- **Policy Evaluation** (critic): estimate $w(u)$ using *temporal difference learning*.
- **Policy Improvement** (actor): improve action values $Q(u)$ based on estimated state values.
Policy Evaluation

Initially, assume all action values are 0, i.e. left/right equally likely everywhere.

True value of each state can be found by inspection:
\[ v(B) = \frac{1}{2}(5+0)=2.5; \]
\[ v(C) = \frac{1}{2}(2+0)=1; \]
\[ v(A) = \frac{1}{2}(v(B)+v(C))=1.75. \]

These values can be learned with temporal difference learning rule:

\[ w(u) \rightarrow w(u) + \varepsilon \delta \quad \text{with} \quad \delta = r_a(u) + v(u') - v(u) \]

where \( u' \) is the state that results from taking action \( a \) in state \( u \).
 Policy Evaluation Example

\[ w(u) \rightarrow w(u) + \varepsilon \delta \quad \text{with} \]
\[ \delta = r_a(u) + v(u') - v(u) \]

Figure 9.8 Policy evaluation. The thin lines show the course of learning of the weights \( w(A), \ w(B), \) and \( w(C) \) over trials through the maze in figure 9.7, using a random unbiased policy \( \mathbf{m}(u) = 0 \). Here \( \varepsilon = 0.5 \), so learning is fast but noisy. The dashed lines show the correct weight values from equation 9.23. The thick lines are running averages of the weight values.

figures taken from Dayan&Abbott
VTA neurons fire for unexpected reward: seem to represent the prediction error $\delta$.

Figure 9.3 Activity of dopaminergic neurons in the VTA for a monkey performing reaction time tasks. (A) Activity of a dopamine cell accumulated over 20 trials showing the spikes time-locked to a stimulus (left panels) or to the reward (right panels) at the times marked 0. The top row is for early trials before the behavior is fully established. The bottom row is for late trials, when the monkey expects the reward on the basis of the stimulus. (B) Activity of a dopamine neuron with and without an expected reward delivery in a similar task. The top row shows the normal behavior of the cell when the reward is delivered. The bottom row shows the result of not delivering an expected reward. The basal firing rate of dopamine cells is rather low, but the inhibition at the time the reward would have been given is evident. (A adapted from Mirenowicz & Schultz, 1994; B adapted from Schultz, 1998.)

Figure taken from Dayan&Abbott
Policy Improvement

Using so-called direct actor rule:

\[ Q_{a'}(u) \rightarrow Q_{a'}(u) + \varepsilon (\delta_{aa'} - p(a';u))\delta \]

where

\[ \delta = r_a(u) + v(u') - v(u) \]

and \( p(a';u) \) is the softmax probability of choosing action \( a' \) in state \( u \) as determined by \( Q_{a'}(u) \).

**Example:** consider starting out from random policy and assume state value estimates \( w(u) \) are accurate. Consider \( u=A \), leads to

\[ \delta = 0 + v(B) - v(A) = 0.75 \] for left turn

\[ \delta = 0 + v(C) - v(A) = -0.75 \] for right turn

rat will increase probability of going left in A
Figure 9.9 Actor-critic learning. The three curves show $P[L; u]$ for the three starting locations $u = A$, $B$, and $C$ in the maze of figure 9.7. These rapidly converge to their optimal values, representing left turns at $A$ and $C$ and a right turn at $B$. Here, $\epsilon = 0.5$ and $\beta = 1$. 

figures taken from Dayan&Abbott
Some further Ideas

- introduction of a *state vector* $u$
- *discounting* of future rewards: put more emphasis on rewards in the near future than rewards that are far away
- only partial knowledge of state
- probabilistic outcomes of actions
- continuous state and action spaces
- function approximation techniques
- …
Some Notable RL Applications

- **TD-Gammon**: Tesauro
  - world’s best backgammon program
- **Elevator Control**: Crites & Barto
  - high performance down-peak elevator controller
- **Inventory Management**: Van Roy, Bertsekas, Lee&Tsitsiklis
  - 10–15% improvement over industry standard methods
- **Dynamic Channel Assignment**: Singh & Bertsekas, Nie & Haykin
  - high performance assignment of radio channels to mobile telephone calls
TD-Gammon

Tesauro, 1992–1995

Start with a random network
Play very many games against self
Learn a value function from this simulated experience

This produces arguably the best player in the world
Elevator Dispatching

Crites and Barto, 1996

10 floors, 4 elevator cars

STATES: button states; positions, directions, and motion states of cars; passengers in cars & in halls

ACTIONS: stop at, or go by, next floor

REWARDS: roughly, –1 per time step for each person waiting

Conservatively about $10^{22}$ states
Performance Comparison

- Average Waiting and System Times
- % Waiting >1 minute
- Average Squared Waiting Time

Dispatcher