Basic RL Problems

- Location of reward uncertain
- Transitions between states uncertain
- Policy constantly changing
You are here

state

action

r

reward for taking action

Reinforcement Learning Primer: Before Learning
By trying different actions from different starting points, we gradually learn the expected reward value from any starting point.

\[ Q(s,a) = r + \gamma (r + \gamma R) \]

Reinforcement Learning Primer

value

policy

\[ r + \gamma (r + \gamma (r + \gamma R)) \]

\[ r + \gamma R \]
State Space
A Monkey uses Secondary Reward

No task

Task 1

Task 2
Do dopamine neurons report an error in the prediction of reward?

- No prediction, Reward occurs
  - (No CS)
  - R

- Reward predicted, Reward occurs
  - CS
  - R

- Reward predicted, No reward occurs
  - CS
  - (No R)
Backgammon
A move in Backgammon
Backgammon played with RL and Backpropagation
Norepinephrine:
Attention, arousal, circadian rhythms

Dopamine:
Secondary reward, movement generation

Serotonin:
Sleep-wake cycle, cognitive performance, aggression

Histamine:
Energy metabolism

Norepinephrine:
Attention, arousal, circadian rhythms
Map of Temporal Discounting  
(Tanaka et al., 2004 (Kenji Doya))

- Markov decision task with delayed rewards
- Regression by values and TD errors
  - with different discounting factors $\gamma$
Task determines fixation point

Gaze distribution on litter in the ‘pickup’ condition

Gaze distribution on obstacles in the ‘avoid’ condition

Multi-tasking revealed by gaze sharing in human data

Shinoda & Hayhoe, Vision Research 2001
Scheduling - Modules - Routines

**Routines**
These behaviors contain instructions of how to use the body.

**Scheduling**
At any moment only a small subset are being used.

**Modules**
There is an enormous library of special purpose behaviors.

---

The diagram illustrates the concept of scheduling in a system, where "active" behaviors are prioritized over "inactive" ones. The "Brain" section contains a large library of special purpose behaviors, which can be "scheduled" to be used in the "Body" section.

The diagram uses visual elements to represent the process, with arrows indicating the flow of information and decision-making in the system.
Humanoid models are used to study multi-tasking
Human performance data shows unexpected regularities
Module for Litter Cleanup

1. Visual Routine
2a. Policy

2b. $V$ is value of Policy

$V(s) = \max_a Q(s,a)$

Heading from Walter's perspective
Learned Microbehaviors

Litter

Sidewalk

Obstacles
Overhead view of trajectory

Initial performance

After 100 iterations

After 150 iterations
The basic RL update for $i$-th module:

$$Q_i(s_t^{(i)}, a_t^{(i)}) \leftarrow Q_i(s_t^{(i)}, a_t^{(i)}) + \alpha \delta_{Q_i}$$

where $\delta_{Q_i}$ is given by:

$$\delta_{Q_i} = \hat{r}_t^{(i)} + \gamma Q_i(s_{t+1}^{(i)}, a_{t+1}^{(i)}) - Q_i(s_t^{(i)}, a_t^{(i)})$$
Driving Simulator
Markov Decision Processes

Problems with delayed reinforcement are well modeled as Markov decision processes (MDPs). An MDP consists of

- a set of states \( \mathcal{S} \),
- a set of actions \( \mathcal{A} \),
- a reward function \( R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \), and
- a state transition function \( T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S}) \), where a member of \( \Pi(\mathcal{S}) \) is a probability distribution over the set \( \mathcal{S} \) (i.e. it maps states to probabilities). We write \( T(s,a,s') \) for the probability of making a transition from state \( s \) to state \( s' \) using action \( a \).

The state transition function probabilistically specifies the next state of the environment as a function of its current state and the agent's action. The reward function specifies expected instantaneous reward as a function of the current state and action. The model is Markov if the state transitions are independent of any previous environment states or agent actions. There are many good references to MDP models [10, 13, 48, 90].
The basic RL algorithm w Model

We will speak of the optimal \textit{value} of a state—it is the expected infinite discounted sum of reward that the agent will gain if it starts in that state and executes the optimal policy. Using \( \pi \) as a complete decision policy, it is written

\[
V^*(s) = \max_\pi \mathbb{E} \left( \sum_{t=0}^{\infty} \gamma^t r_t \right).
\]

This optimal value function is unique and can be defined as the solution to the simultaneous equations

\[
V^*(s) = \max_a \left( R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right), \forall s \in S, \tag{1}
\]

which assert that the value of a state \( s \) is the expected instantaneous reward plus the expected discounted value of the next state, using the best available action. Given the optimal value function, we can specify the optimal policy as

\[
\pi^*(s) = \arg \max_a \left( R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right).
\]
Value Iteration

initialize $V(s)$ arbitrarily

loop until policy good enough

    loop for $s \in S$

        loop for $a \in A$

            $Q(s, a) := R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V(s')$

            $V(s) := \max_a Q(s, a)$

        end loop

    end loop

end loop
Temporal Difference Learning
Temporal difference learning [Sutton and Barto, 1998], uses the error between the current estimated values of states and the observed reward to drive learning. In a related Q-learning form, the estimate of the quality value of a state-action pair is adjusted by this error $\delta_Q$ using a learning rate $\alpha$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta_Q$$  \hspace{1cm} (3)

Two fundamental learning rules for $\delta_Q$ are 1) the original Q-learning rule [Watkins, 1989] and 2) SARSA [Rummery and Niranjan, 1994]. While Q-learning rule is an off-policy rule, i.e. it uses errors between current observations and estimates of the values for following an optimal policy, while actually following a potentially suboptimal policy during learning, SARSA$^1$ is an on-policy learning rule, i.e. the updates of the state and action values reflect the current policy derived from these value estimates. While in the general case of Q-learning, the temporal difference is:

$$\delta_Q = r_t + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$ \hspace{1cm} (4)

for the more specific case of SARSA it is:

$$\delta_Q = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t).$$  \hspace{1cm} (5)
Policy Iteration

choose an arbitrary policy $\pi'$

loop

$\pi := \pi'$

compute the value function of policy $\pi$:

solve the linear equations

$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s')V_\pi(s')$

improve the policy at each state:

$\pi'(s) := \arg \max_a \left( R(s, a) + \gamma \sum_{s' \in S} T(s, a, s')V_\pi(s') \right)$

until $\pi = \pi'$