

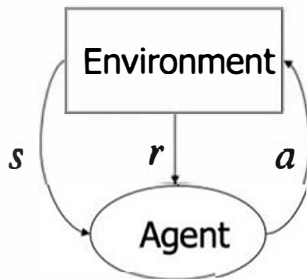
Learning with a critic

■ Who is a critic?

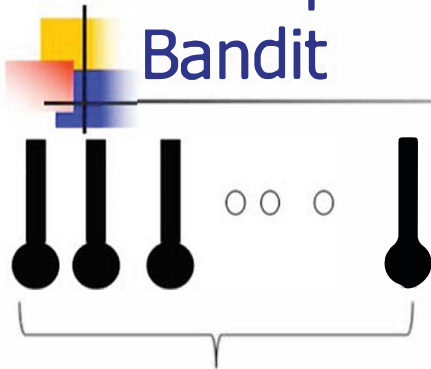
- Not a teacher
 - does not tell us what to do
- only evaluates past performances
 - feedback scarce and delayed

■ Learner (Agent)

- interacts with an environment (at some state)
 - May change the state
- gets reward / penalty sometimes (by the critic)
- tries to reach a goal (a state)
- Learns a series of actions to reach the goal.
 - Maximizes total rewards from any state.



A simple learner: K-Arm Bandit



K -levers

A simple machine
with one state

$Q(a)$: Value of action a

Initially $Q(a)=0$ for all a

Store r_a after each a , $Q(a)=r_a$

Action (a): To pull a lever to win a reward (r_a).

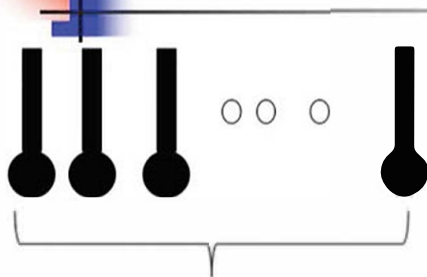
The task: to decide which lever to pull to maximize the reward

Supervised Learning: the correct class, namely, the lever leading to maximum earning labeled by a teacher.

Reinforcement learning: to try different levers and keep track of the best.

Choose a^* if $Q(a^*) = \max_a Q(a)$

Nondeterministic: K-Arm Bandit



K -levers

Action (a): To pull a lever to win a reward r with $p(r|a)$.

The task: to decide which lever to pull to maximize the reward

Reinforcement learning: to try different levers and keep track of the best.

$Q_t(a)$: Estimate of the Value of action a at time t

→ an average of all past rewards when a was chosen

Delta rule: $Q_{t+1}(a) \leftarrow Q_t(a) + \eta[r_{t+1}(a) - Q_t(a)]$ η : learning factor decreasing with time

Choose a^* if $Q(a^*) = \max_a Q(a)$



More complex environment

- Multiple states of the environment
- Action also affects the next state.
- The agent senses state (s_{t+1}) after an action (a_t).
 - may get reward (r_{t+1})
- Nondeterministic reward: $p(r|s_i, a_j)$
- To learn $Q(s_i, a_j)$: Value of taking action a_j in state s_i
- Rewards may be delayed
 - Need for immediate estimation of prospective reward

Learning a Markov Decision Process (MDP)

- An agent takes an action $a_t \in \mathbf{A}$ at state $s_t \in \mathbf{S}$ at discrete time t .
 - $p(r_{t+1} | s_t, a_t)$
 - $p(s_{t+1} | s_t, a_t)$
- Reaching a terminal state and remains there for any action with prob. 1 without any reward.
- Episode or trial: The sequence of actions from the start to the terminal state.
- Policy: mapping from the states of the environment to actions: $\pi : \mathbf{S} \rightarrow \mathbf{A}$
- Value of a policy $\pi : V^\pi(s_t) = E[r_{t+1} + r_{t+2} + \dots + r_{t+T}]$

Finite horizon

May be infinite also.



Learning a policy

- Given $p(r_{t+1} | s_t, a_t)$ and $p(a_{t+1} | s_t, a_t)$ (Model)
- **Policy**: mapping from the states to actions: $\pi : S \rightarrow A$
- **Value of a policy π** :
 - $V^\pi(s_t) = E[r_{t+1} + r_{t+2} + \dots + r_{t+T}]$
- **Discounted value with infinite horizon**:
 - $V^\pi(s_t) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots]$
- **Optimal policy π^*** : $V^*(s_t) = \max_\pi [V^\pi(s_t)]$
- As an alternative: Learn $Q(s_t, a_t)$: Value at state with action
- $V^*(s_t) = \max_a [Q(s_t, a_t)] = \max_a E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots]$
 - $= \max_a E[r_{t+1} + \gamma V^\pi(s_{t+1})]$



Learning optimal policy

$$V^*(s_t) = \max_{a_t} \left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) V^*(s_{t+1}) \right)$$

$$Q^*(s_t, a_t) = \max_{a_t} \left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \max_{a_{t+1}} (Q^*(s_{t+1}, a_{t+1})) \right)$$

- $\pi^*(s_t)$: Choose a^* providing $V^*(s_t)$

Or

- $\pi^*(s_t)$: Choose a^* if $Q^*(s_t, a^*) = \max_a Q^*(s_t, a)$



Model based learning

- Model: $p(r_{t+1} | s_t, a_t)$ and $p(a_{t+1} | s_t, a_t)$ known.
- Directly solve for the optimal value function and policy
 - using dynamic programming

$$V^*(s) = \max_a \left(E[r|s, a] + \gamma \sum_{s'} P(s'|s, a) V^*(s') \right)$$

$$\pi^*(s) = \operatorname{argmax}_a \left(E[r|s, a] + \gamma \sum_{s'} P(s'|s, a) V^*(s') \right)$$

- Two approaches
 - Value iteration algorithm
 - Policy iteration algorithm



Value iteration algorithm

Initialize $V(s)$ to arbitrary values

Repeat

For all $s \in S$

For all $a \in A$

$$Q(s, a) = \max_a \left(E[r|s, a] + \gamma \sum_{s'} P(s'|s, a) V(s') \right)$$

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

$$\pi^*(s) = \operatorname{argmax}_a \left(E[r|s, a] + \gamma \sum_{s'} P(s'|s, a) V(s') \right)$$

Until $V(s)$ converge



Policy iteration algorithm

Update policy directly from intermediate values

Initialize π to arbitrary values

Repeat

$$\pi \leftarrow \pi'$$

Compute values using π

$$V^\pi(s) = \max_a \left(E[r|s, \pi(s)] + \gamma \sum_{s'} P(s'|s, \pi(s)) V^\pi(s') \right)$$

Improve the policy at each stage.

$$\pi'(s) = \operatorname{argmax}_a \left(E[r|s, a] + \gamma \sum_{s'} P(s'|s, a) V(s') \right)$$

Until $\pi = \pi'$



Temporal difference algorithm

- No prior knowledge of model
 - No $p(r_{t+1} | s_t, a_t)$ and $p(a_{t+1} | s_t, a_t)$
- requires exploration of the environment to query the model
 - to see the value of the next state and reward
- use this information to update the value of the current state
- called **temporal difference algorithms**
 - examines the difference between current estimate of the value and the discounted value of the next state and the reward received



Deterministic Environment

- For any state-action (s_t, a_t) power a single reward (r_{t+1}) and state transition (s_{t+1}) possible.

$$Q(s_t, a_t) = (r_{t+1} + \gamma \max_{a_{t+1}} (Q(s_{t+1}, a_{t+1})))$$

- Update at every exploration
 - Add immediate reward with discounted estimate of the next state-action pair.
- Later updates more reliable
- Converge when all pairs are stable (little changes with iteration).



Nondeterministic environment

- varying reward or next state for a state-action pair
- keep a running average of values
 - Q-Learning

Delta rule: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta [r_{t+1} + \gamma \max_a \{Q(s_{t+1}, a)\} - Q(s_t, a_t)]$

- Choose next action randomly (ϵ -Greedy sampling)
 - Sample an action uniformly with a prob. ϵ initially.
 - Actions providing higher values would have higher probability
 - Softmax over Q-values
 - using a Temperature Variable (T)
 - At every iteration T increases favoring higher Q values

$$p(a|s) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_b e^{\frac{Q(s,b)}{T}}}$$



Large number of states and actions

- Not feasible through tabular search.
- Use regression to predict Q values given current value, reward and next state.
 - Requires supervisory information or labels.



Summary

- RL: Learning with a critic
 - Different from supervised learning
 - labels or end results directly not available.
 - An agent observes state (s) changes on actions (a), and may get reward (r) from a critic.
 - May be nondeterministic: $p(r|s,a)$, $p(s'|s,a)$
- To determine a policy: $\pi : S \rightarrow A$
 - For reaching goal from a state
- Model based learning:
 - Value iteration and Policy iteration
- Temporal difference algorithms

Deterministic
Nondeterministic

Q-Learning