Rewards and Value Systems: Conditioning and Reinforcement Learning

Acknowledgements:

Many thanks to P. Dayan and L. Abbott for making the figures of their book available online. Also thanks to R. Sutton and A. Barto for making figures and slides available online.
Outline

- Introduction
- Review of classical conditioning and its variations
- Rescorla Wagner rule
- Predicting future reward with temporal differences
- Instrumental conditioning
- Markov decision processes
- The general reinforcement learning problem
- Policy Iteration
- Actor Critic architecture
Resources

Dayan & Abbott

Sutton & Barto
A Taxonomy of Learning Settings

- Unsupervised
- Self-supervised
- Reinforcement
- Imitation
- Instruction
- Supervised

Increasing amount of “help” from the environment
A truly central topic in Cognitive Science!
What is Reinforcement Learning?

Central aspects:
- learning what to do: mapping situations to actions
- focus on learning through interaction with environment (infant has no explicit teacher but can interact with environment)
- evaluative feedback from environment (pleasure & pain) that must be predicted

Two key problems:
- trial and error learning leads to the exploration versus exploitation dilemma
- delayed rewards lead to the temporal credit assignment problem
Examples and Themes

Examples:
- playing chess and improving
- adaptive controller of petroleum refinery
- gazelle calf learning to run minutes after birth
- mobile robot deciding whether to continue current task or to go home to recharge batteries
- person preparing breakfast

Common Themes:
- interaction with environment
- seeking a goal
- sometimes uncertainty about state of environment
- sometimes uncertainty about outcomes of actions
Pavlov’s classic finding: (classical conditioning)
Initially, sight of food leads to dog salivating:

Food: unconditioned stimulus (US) → Unconditioned response (UR) (reward)

Sound of bell consistently precedes food. Afterwards, bell leads to salivating:

Bell: conditioned stimulus (CS) → Conditioned response (CR) (expectation of reward)
Variations of Conditioning 1

Extinction:
- Stimulus (bell) repeatedly shown without reward (food).
- Result: conditioned response (salivating) reduced.

Partial reinforcement:
- Stimulus only sometimes preceding reward
- Result: conditioned response weaker than in classical case.

Blocking (2 stimuli S1+S2):
- First: S1 associated with reward: (classical case)
- Then: S1 and S2 shown together followed by reward.
- Result: Association between S2 and reward not learned.
Variations of Conditioning 2

Inhibitory Conditioning (2 stimuli): alternate 2 types of trials:
1. S1 followed by reward.
2. S1+S2 followed by absence of reward.
Result:
   S1 becomes predictor of presence of reward,
   S2 becomes predictor of absence of reward.

To show this use for example the following 2 methods:
• train animal to predict reward based on S2. Result: learning slowed
• train animal to predict reward based on S3, then show S2+S3. Result: conditioned response weaker than for S3 alone.
Variations of Conditioning 3

Overshadowing (2 stimuli):
- Repeatedly present S1+S2 followed by reward.
- Result: often, reward prediction shared unequally between stimuli.

Example (made up):
- red light + high pitch beep precede pigeon food.
- Result: red light more effective in predicting the food than high pitch beep.

Secondary Conditioning:
- S1 preceding reward. Then, S2 preceding S1.
- Result: S2 leads to prediction of reward.
- But: if S1 following S2 showed too often: extinction will occur.
# Summary of Conditioning Findings

(incomplete, has been studied extensively for decades, many books on topic)

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Pre-Train</th>
<th>Train</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pavlovian</td>
<td></td>
<td>$s \rightarrow r$</td>
<td>$s \rightarrow r'$</td>
</tr>
<tr>
<td>Extinction</td>
<td>$s \rightarrow r$</td>
<td>$s \rightarrow \cdot$</td>
<td>$s \rightarrow r'$</td>
</tr>
<tr>
<td>Partial</td>
<td>$s \rightarrow r$</td>
<td>$s \rightarrow \cdot$</td>
<td>$s \rightarrow \alpha \ 'r'$</td>
</tr>
<tr>
<td>Blocking</td>
<td>$s_1 \rightarrow r$</td>
<td>$s_1 + s_2 \rightarrow r$</td>
<td>$s_1 \rightarrow r'$, $s_2 \rightarrow r'$</td>
</tr>
<tr>
<td>Inhibitory</td>
<td>$s_1 + s_2 \rightarrow \cdot$</td>
<td>$s_1 \rightarrow r$</td>
<td>$s_1 \rightarrow r'$, $s_2 \rightarrow -r'$</td>
</tr>
<tr>
<td>Overshadow</td>
<td>$s_1 + s_2 \rightarrow r$</td>
<td></td>
<td>$s_1 \rightarrow \alpha_1 r'$, $s_2 \rightarrow \alpha_2 r'$</td>
</tr>
<tr>
<td>Secondary</td>
<td>$s_1 \rightarrow r$</td>
<td>$s_2 \rightarrow s_1$</td>
<td>$s_2 \rightarrow r'$</td>
</tr>
</tbody>
</table>

Table 9.1 Classical conditioning paradigms. The columns indicate the training procedures and results, with some paradigms requiring a pre-training as well as a training period. Both training and pre-training periods consist of a moderate number of training trials. The arrows represent an association between one or two stimuli ($s$, or $s_1$ and $s_2$) and either a reward ($r$) or the absence of a reward ($\cdot$). In Partial and Inhibitory conditioning, the two types of training trials that are indicated are alternated. In the Result column, the arrows represent an association between a stimulus and the expectation of a reward ($r'$) or no reward ($\cdot'$). The factors of $\alpha$ denote a partial or weakened expectation, and the minus sign indicates the suppression of an expectation of reward.

figure taken from Dayan&Abbott
The setting: (Rescorla & Wagner, 1972):
Consider stimulus variable $u$ representing presence ($u=1$) or absence ($u=0$) of stimulus. Correspondingly, reward variable $r$ represents presence or absence of reward.

The expected reward $v$ is modeled as “stimulus times weight“:

$$v = wu$$

Learning is done by adjusting the weight to minimize error between predicted reward $v$ and actual reward $r$. 
How to change the weight?

Denote the prediction error by $\delta$(delta):

$$\delta = r - v$$

Learning rule:

$$w \leftarrow w + \epsilon \delta u,$$

where $\epsilon$ is a learning rate.

Q: Why is this useful?
A: This rule does stochastic gradient descent to minimize the expected squared error $(r - v)^2$; $w$ converges to $<r>$. R.W. rule is variant of the “delta rule” in neural networks.
**Goal:** minimize squared error $E = \delta^2$

\[
\frac{\partial}{\partial w} E = \frac{\partial}{\partial w} \delta^2 = \frac{\partial}{\partial w} (r - v)^2
= \frac{\partial}{\partial w} (r - wu)^2
= 2(r - wu)(-u)
= -2\delta u
\]

\[w \leftarrow w - \eta \frac{\partial E}{\partial w}\]

Note: in psychological terms the learning rate is a measure of *associability* of stimulus with reward.

\[w \leftarrow w + \varepsilon \delta u, \text{ where } \varepsilon = 2\eta\]
Example:

prediction error $\delta = r - v$; learning rule: $w \leftarrow w + \epsilon \delta u$
Multiple Stimuli

Essentially the same idea/learning rule:

In case of multiple stimuli: \( \nu = \mathbf{w} \cdot \mathbf{u} \)
(predicted reward = dot product of stimulus vector and weight vector)

Prediction error: \( \delta = r - \nu \)
Learning rule: \( \mathbf{w} \leftarrow \mathbf{w} + \varepsilon \delta \mathbf{u} \)

\[
\frac{\partial}{\partial w_i} \delta^2 = \frac{\partial}{\partial w_i} (r - \nu)^2 = \frac{\partial}{\partial w_i} (r - \mathbf{w} \cdot \mathbf{u})^2
\]

\[
= 2(r - \mathbf{w} \cdot \mathbf{u}) \frac{\partial}{\partial w_i} \left( - \sum_j w_j u_j \right)
\]

\[
= -2\delta u_i
\]
In how far does Rescorla Wagner rule account for variants of classical conditioning?

(prediction: \( v = w \cdot u \), error: \( \delta = r - v \); learning: \( w \leftarrow w + \varepsilon \delta u \))

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Pre-Train</th>
<th>Train</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pavlovian</td>
<td></td>
<td>( s \rightarrow r )</td>
<td>( s \rightarrow 'r' )</td>
</tr>
<tr>
<td>Extinction</td>
<td></td>
<td>( s \rightarrow . )</td>
<td>( s \rightarrow '.' )</td>
</tr>
<tr>
<td>Partial</td>
<td></td>
<td>( s \rightarrow r ) ( s \rightarrow . )</td>
<td>( s \rightarrow \alpha 'r' )</td>
</tr>
<tr>
<td>Blocking</td>
<td>( s_1 \rightarrow r ) ( s_1 + s_2 \rightarrow r )</td>
<td>( s_1 \rightarrow 'r' ) ( s_2 \rightarrow '.' )</td>
<td></td>
</tr>
<tr>
<td>Inhibitory</td>
<td>( s_1 + s_2 \rightarrow . ) ( s_1 \rightarrow r )</td>
<td>( s_1 \rightarrow 'r' ) ( s_2 \rightarrow '-'r' )</td>
<td></td>
</tr>
<tr>
<td>Overshadow</td>
<td>( s_1 + s_2 \rightarrow r )</td>
<td>( s_1 \rightarrow \alpha_1 'r' ) ( s_2 \rightarrow \alpha_2 'r' )</td>
<td></td>
</tr>
<tr>
<td>Secondary</td>
<td>( s_1 \rightarrow r ) ( s_2 \rightarrow s_1 )</td>
<td>( s_2 \rightarrow 'r' )</td>
<td></td>
</tr>
</tbody>
</table>

Table 9.1 Classical conditioning paradigms. The columns indicate the training procedures and results, with some paradigms requiring a pre-training as well as a training period. Both training and pre-training periods consist of a moderate number of training trials. The arrows represent an association between one or two stimuli (\( s \), or \( s_1 \) and \( s_2 \)) and either a reward (\( r \)) or the absence of a reward (\( . \)). In Partial and Inhibitory conditioning, the two types of training trials that are indicated are alternated. In the Result column, the arrows represent an association between a stimulus and the expectation of a reward (\('r'\)) or no reward (\('.'\)). The factors of \( \alpha \) denote a partial or weakened expectation, and the minus sign indicates the suppression of an expectation of reward.

figure taken from Dayan&Abbott
(prediction: \( v = w \cdot u \), error: \( \delta = r - v \); learning: \( w := w + \varepsilon \delta u \))

- **Extinction, Partial Reinforcement**: o.k., since \( w \) converges to \( <r> \)
- **Blocking**: during pre-training, \( w_1 \) converges to \( r \). During training \( v = w_1u_1 + w_2u_2 = r \), hence \( \delta = 0 \) and \( w_2 \) does not grow.
- **Inhibitory Conditioning**: on S1 only trials, \( w_1 \) gets positive value. on S1+S2 trials, \( v = w_1 + w_2 \) must converge to zero, hence \( w_2 \) becoming negative.
- **Overshadow**: \( v = w_1 + w_2 \) goes to \( r \), but \( w_1 \) and \( w_2 \) may become different if there are different learning rates \( \varepsilon_i \) for them.
- **Secondary Conditioning**: R.-W.-rule predicts negative S2 weight!

- **Rescorla Wagner rule qualitatively accounts for wide range of conditioning phenomena but not secondary conditioning.**
Motivation:
- Rescorla-Wagner rule can’t capture secondary conditioning
- Animals can predict when in a trial reward occurs
- How can we model these things?

Idea:
- keep track of time during trial
- learn to predict how much reward is still to come in a trial. Define total future reward (value function):

\[
\sum_{\tau=0}^{T-t} r(t+\tau)
\]

\( t \) : current time
\( T \) : time at end of trial
\( r(t) \) : reward at time \( t \)
total future reward: $\left\langle \sum_{\tau=0}^{T-t} r(t + \tau) \right\rangle$

Prediction of total future reward:
• assume simple linear function of stimulus history:

$$\nu(t) = \sum_{\tau=0}^{t} w(\tau) u(t - \tau)$$

• note: a little unrealistic, assumes perfect memory

**Question:** how do we need to update the weights?
• problem arises because we do not yet know the total future reward at time $t$, since it depends on rewards that are yet to come!

**Solution:** “bootstrapping” (here: estimating quantities based on estimates of other quantities)
Trick: rewrite future reward in one trial this way:

\[
\sum_{\tau=0}^{T-t} r(t + \tau) = r(t) + \sum_{\tau=0}^{T-t-1} r(t + 1 + \tau)
\]

Now: \(v(t+1)\) is an estimate of the total future reward from the next time step in the same way that \(v(t)\) is an estimate of the total future reward from this time step. Thus:

\[
\sum_{\tau=0}^{T-t} r(t + \tau) \approx r(t) + v(t + 1)
\]

Now we can define an approximate prediction error that does not depend on quantities in the distant future:

\[
\delta(t) = r(t) + v(t + 1) - v(t) \quad \delta : \text{temporal difference error (TD-error)}
\]
\[ \delta(t) = r(t) + \nu(t + 1) - \nu(t) \]

**Learning rule:**

- adjusting weights so as to reduce this temporal difference error (stochastic gradient descent) leads to the temporal difference learning rule:

\[
\mathbf{w}(\tau) \rightarrow \mathbf{w}(\tau) + \varepsilon \delta(t) u(t - \tau)
\]
Example:

Setting: stimulus at $t=100$, reward around $t=200$

Model learns to correctly predict when reward will occur!
Link to neurobiology:

- dopaminergic neurons in the ventral tegmental area (VTA) seem to encode something resembling a temporal difference error (TD-error).

Examples from W. Schultz, recorded in monkey.

Question: can our new temporal difference model account for secondary conditioning?
• **So far**: only concerned with prediction of reward; didn’t consider agent’s actions. Reward usually depends on what you do! *Skinner boxes*, etc.

• **Distinguish two scenarios:**
  * Rewards follow actions immediately (**Static Action Choice**). Example: n-armed bandit problem (slot machine)
  * Rewards may be delayed (**Sequential Action Choice**). Example: playing chess

• **Goal in both cases**: choose actions to maximize rewards
**n-armed Bandits**

Non-associative case: no different states to distinguish

**Aim:** focus on exploration vs. exploitation dilemma

*n*-armed bandit problems: each round pull one of *n* arms and receive random reward from unknown distribution specific to that slot machine. How to maximize the reward over time?
**Action Values**

**Def.:** *action value* $Q^*(a) = \text{true value of action} = \text{average reward when playing } a.$

This needs to be estimated since it’s unknown. Define the estimate $Q_t(a)$ after choosing action $a$ $k_a$ times as the sample mean:

$$Q_t(a) = \frac{r_1 + r_2 + \ldots + r_{k_a}}{k_a}$$

If action $a$ tried often enough: $Q_t(a)$ converges to $Q^*(a)$

We also need an initial estimate before $a$ has been tried. Let’s choose a default value $Q_0(a)$

*But how to choose action?*
greedy and $\varepsilon$-greedy methods

greedy policy:
- always chose action $a$ whose $Q_t(a)$ is maximal
- but no effort spent on exploring seemingly inferior actions to see if they aren't better than previously thought.

$\varepsilon$-greedy policy:
- with small probability $\varepsilon$ choose action at random in order to explore, otherwise choose greedy action

Example test bed:
- 10-armed bandit task, 1000 plays total (=1 experiment)
- repeat experiment 2000 times with different normally distributed rewards for each arm
Figure 2.1  Average performance of $\epsilon$-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 tasks. All methods used sample averages as their action-value estimates.
Results of Experiments on 10-armed bandit test bed:

- greedy method tends to perform poorly in the long run
- higher $\varepsilon$ leads to faster learning
- higher $\varepsilon$ also leads to less exploitation since optimal action only chosen with probability $(1-\varepsilon)$ and inferior actions chosen otherwise

- this is an instance of the famous:

"Exploration Exploitation Dilemma"
Role of Initial Conditions:

- Note: if initial action value estimates $Q_0(a)$ are much too low, greedy strategy won’t explore at all.
- Explanation: first action tried will have higher action value than all others and no other will ever be tried!

- Conversely, if initial action values are too high, greedy strategy will have initial phase of exploration.
- Explanation: each action tried will now have lower action value than action that hasn’t been tried yet.
- Such optimistic initial values can be quite beneficial!
Illustration of benefit of optimistic initial conditions:

Figure 2.4 The effect of optimistic initial action-value estimates on the 10-armed testbed.

Figure taken from Sutton&Barto
Softmax Action Selection

Drawback of $\varepsilon$-greedy method: very bad actions are explored as frequently as very good ones

Idea: “better” actions should be explored more often

$$p(a) = \frac{\exp(Q_t(a)/\tau)}{\sum_{b=1}^{n} \exp(Q_t(b)/\tau)}$$

*Boltzmann* or *Gibbs* distribution, $\tau$: “temperature”

Note: $\Sigma p(a) = 1$

Limiting cases:

- $\tau \to 0$: choose action with highest $Q_t(a)$ (greedy policy)
- $\tau \to \infty$: choose all actions with same probability
- sometimes the inverse temperature $\beta = 1/\tau$ is used
Incremental Implementation of action value estimation

• **Problem:** if we’re computing action values like this we need to store all old $r$ values. Is this necessary?

$$Q_t(a) = \frac{r_1 + r_2 + \ldots + r_{k_a}}{k_a}$$

$$Q_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} r_i = \frac{1}{k+1} (r_{k+1} + kQ_k)$$

$$= Q_k + \frac{1}{k+1} (r_{k+1} - Q_k)$$

$$= \text{oldEstimate + stepsize(target - oldEstimate)}$$

don’t need to store old $r$ values anymore!
Non-stationary problems

- **Problem**: what if rewards are changing over time?
- **Idea**: want to forget about old $r$ values. Can do so by leaving step size constant $\alpha$ instead of $1/(k+1)$

\[
Q_{k+1} = \text{oldEstimate} + \text{stepsize}(\text{target} - \text{oldEstimate})
\]
\[
= Q_k + \alpha(r_{k+1} - Q_k)
\]
\[
= \alpha r_{k+1} + (1 - \alpha)Q_k
\]

- can also be seen as weighted average of all previous rewards, with exponentially smaller weights for rewards far back in time:

\[
Q_k = (1 - \alpha)^k Q_0 + \sum_{i=1}^{k} \alpha(1 - \alpha)^{k-i} r_i
\]
Example: Modeling Bee Foraging

Setting:
• bee chooses to fly to blue or yellow flowers (2 actions);
• probabilistic amount of nectar in flowers
• bee wants to maximize nectar volume (reward)

Observation: real bees learn to fly to “better” flower in single session (~40 flowers)
• when bee chooses blue/yellow flower: reward is drawn from distributions \( p(r_b) \) or \( p(r_y) \)
• assume model bee has *stochastic policy*: chooses to fly to blue or yellow flower with \( p(b) \) or \( p(y) \), respectively.
• let’s assume: \( p(b), p(y) \) follow *softmax decision rule*:

\[
p(b) = \frac{\exp(\beta Q_b)}{\exp(\beta Q_b) + \exp(\beta Q_y)} , \quad p(y) = \frac{\exp(\beta Q_y)}{\exp(\beta Q_b) + \exp(\beta Q_y)}
\]

**Note**: in this two-action case we can write

\[
p(b) = \sigma(\beta (Q_b - Q_y)) , \text{ where } \sigma(x) = 1/1 + \exp(-x)
\]
is the standard sigmoid function.

• To adjust action values use: \( Q_b \rightarrow Q_b + \alpha \delta \) , where
\[
\delta = r_b - Q_b \text{ (compare slide 35)}
\]
Example: $\alpha=0.1$: constant updating of action values

Rewards for blue and yellow swap

$\beta=1$: more exploration

$\beta=50$: less exploration

Figure taken from Dayan & Abbott
Reinforcement Comparison
(an alternative to action value methods)

Intuitive idea:
• big rewards should make preceding action more likely, while small rewards should make preceding action less likely
• define “big” and “small” reward with respect to a running average of (more or less recent) past rewards.

Implementation:
• keep track of action preferences: \( p_t(a) \)
• use softmax to compute probabilities of choosing actions:

\[
\pi_t(a) = \frac{\exp(p_t(a))}{\sum_{b=1}^{n} \exp(p_t(b))}
\]
action preference update: (big rewards should make preceding action more likely)

\[ p_{t+1}(a_t) = p_t(a_t) + \gamma [r_t - \bar{r}_t], \quad 0 < \gamma \leq 1 \]

average reward update:

\[ \bar{r}_{t+1} = \bar{r}_t + \alpha (r_t - \bar{r}_t), \quad 0 < \alpha \leq 1 \]

Evaluation:

- reinforcement comparison can be very effective, sometimes performing even better than action value methods
- action value method has clear rationale: estimate average reward for each action
- reinforcement comparison: make actions with above average reward more likely
- the “direct actor” method in the Dayan and Abbott chapter can be seen as variant of this reinforcement comparison.
The Full Reinforcement Learning Problem

- **So far:** immediate reward after each action (\(n\)-armed bandit problem)
- **Now:** *delayed rewards*, world can be in *different states*
- **Example:** maze task: amount of reward after decision at second intersection depends on action taken at first intersection.

![Diagram of a maze with reward values at intersections A, B, and C.](figure taken from Dayan & Abbott)
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: $r_{t+1} \in \mathbb{R}$

and resulting next state: $s_{t+1}$

\[ \ldots \quad s_t \quad a_t \quad r_{t+1} \quad s_{t+1} \quad a_{t+1} \quad r_{t+2} \quad s_{t+2} \quad a_{t+2} \quad r_{t+3} \quad s_{t+3} \quad a_{t+3} \quad \ldots \]
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:

\[ P_{ss'}^a = \Pr \{ s_{t+1} = s' \mid s_t = s, a_t = a \} \quad \text{for all } s, s' \in S, a \in A(s). \]

• reward probabilities:

\[ R_{ss'}^a = E \{ r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s' \} \quad \text{for all } s, s' \in S, a \in A(s). \]
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 or low.
- reward = number of cans collected
- Define the following state and action sets:

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

"transition graph"

\[ R_{\text{search}} = \text{expected no. of cans while searching} \]
\[ R_{\text{wait}} = \text{expected no. of cans while waiting} \]
\[ R_{\text{search}} > R_{\text{wait}} \]
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 have deterministic consequences (in contrast to probabilistic)

**Idea:** maintain and improve a **stochastic policy** which determines the action at each decision point \((A,B,C)\) using action values and softmax decision rule

**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
Actor-Critic Method

Agent

- Policy
  - Actor
  - Critic
- Value Function
  - state
  - action
  - reward
- Environment

- TD error
Formal Setup

- **state variable** $u$ (is rat is at A, B, or C?)
- **action value vector** $Q(u)$ describing 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: $v(u)$ (**state value function**)
- The rat estimates this with weights $w(u)$

![Diagram](figure taken from Dayan&Abbott)
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 $\mu(u)$ using temporal difference learning.
  - Policy Improvement (actor): improve action values $Q(u)$ based on estimated state values.
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:

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

where $u'$ is the state that results from taking action $a$ in state $u$. 

figure taken from Dayan&Abbott
$w(u) \rightarrow w(u) + \varepsilon \delta$ with $\varepsilon=0.5$ and $\delta = r_a(u) + v(u') - v(u)$

figures taken from Dayan&Abbott
Policy Improvement
(using a 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 \( v(u) \) are accurate. Consider \( u=A \), leads to

\[ \delta = 0 + v(B) - v(A) = 0.75 \quad \text{for left turn} \]
\[ \delta = 0 + v(C) - v(A) = -0.75 \quad \text{for right turn} \]

rat will increase probability of going left in A
Policy Improvement Example

\[ P[L; u] \]

\( u = A \)

\( u = B \)

\( u = C \)

\( \epsilon = 0.5 \) 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
• environment models and planning
• ...

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

Dispatcher

% Waiting >1 minute

Dispatcher

Average Squared Waiting Time

Dispatcher
Critique: Reinforcement Learning

• (general) framework for learning how to behave, i.e., how to map sensory states to actions

• very active field at the intersection of machine learning, neural networks, control theory, psychology, neuroscience

• essentially trial and error learning; has a reputation of being very slow

• in the form presented here it does not address where sensory representations and action representations come from
Midbrain Structures

- Substantia nigra
- Periaqueductal gray matter
- Cerebral aqueduct
- Superior colliculus

- Ventral
- Dorsal

- Red nucleus
- Reticular formation

- Superior colliculus (receives visual input)
- Inferior colliculus (receives auditory input)

- Tectum

- Cerebellum

- Tegmentum
dopaminergic neurons can signal unexpected rewards in a way that qualitatively resembles a temporal difference error
• Projections of dopaminergic neurons in the substantia nigra and ventral tegmental area:
  • striatum (part of the basal ganglia)
  • nucleus accumbens
  • frontal cortex
  • ...

• Further lines of evidence for involvement of dopamine in reward processing
  • some addictive drugs prolong the influence of dopamine on its target neurons
  • pathways associated with dopamine are among the best targets for electrical self-stimulation

• Evidence for dopamine modulating plasticity
Basal ganglia

- Basal ganglia
- Thalamus
- Corpus callosum
- Lateral ventricle
- Caudate nucleus
- Putamen
- Globus pallidus
- Subthalamic nucleus
- Substantia nigra

Basal ganglia
Representation of Action-Specific Reward Values in the Striatum

Kazuyuki Samejima,1*† Yasumasa Ueda,2 Kenji Doya,1,3 Minoru Kimura2*

---

[Diagram showing the representation of action-specific reward values in the striatum, with time, delay, and reward stages depicted.]
two blocks where "left" is better

monkey learns to favor action with higher probability of the big reward

long bar: large reward
short bar: small reward
\(x\): errors w/o reward

running average of \(P_L\)
Activity in three striatal projection neurons after monkey has learned to mostly choose better option in this block:

- Neuron's activity during delay reflects something similar to value of choosing “left”.
- Neuron's activity reflects something similar to negative value of choosing “right”.
- Neuron's activity reflects something similar to difference of values between choosing “left” or “right”.
Analysis of 142 out of 504 neurons from two monkeys:

Open symbols: neuron has significant regression coefficient with choice, reaction time, or movement time. Crosses: neuron is only correlated with such behavioral measures but not significantly correlated with Q-values.