Reinforcement learning

Introduction to reinforcement learning

Related to animal learning literature.

Thorndike's law, according to Barto (1992):
If an action taken by a learning system is followed by a satisfactory state of affairs, then the tendency of the system to produce that particular action is strengthened or reinforced. Otherwise, the tendency of the system to produce that action is weakened.

i.e. measure how well each decision correlates with global reinforcement signal.

RL in a nutshell (Harmon and Harmon, 1986)

Steps to riding a bike:

- Bike angled at 45 deg to right; turn left $\rightarrow$ OUCH
- Bike angled at 45 deg to right; turn right $\rightarrow$ OUCH
- (Okay, being at 45 deg a bad idea)
- Bike angled at 40 deg to right; turn left. $\rightarrow$ 45 deg to right $\rightarrow$ OUCH
- Bike angled at 40 deg to right; turn right $\rightarrow$ 35 deg to right $\rightarrow$ . . .

Concerns about reinforcement learning

- Concerns in a nutshell: credit assignment problem.

Two problems with reinforcement learning:

1. No specific signal on what the desired outputs should be.
   "It is as if each person in the United States tried to decide whether he or she had done a useful day's work by observing the gross national product [GNP] on a day by day basis" (Hinton, p220).
2. Reinforcement signal can occur long time after action. (e.g. one bad move in a game of chess, causing eventual loss of game.)
   "If ... a person wants to how their behavior affects the GNP, they need to know whether to correlate today's GNP with what they did yesterday or with what they did five years ago." (Hinton, p220).

   Need for intermediate reinforcement states.

Hinton (1989) Connectionist learning systems, *Artificial Intelligence*, 40:185-234.

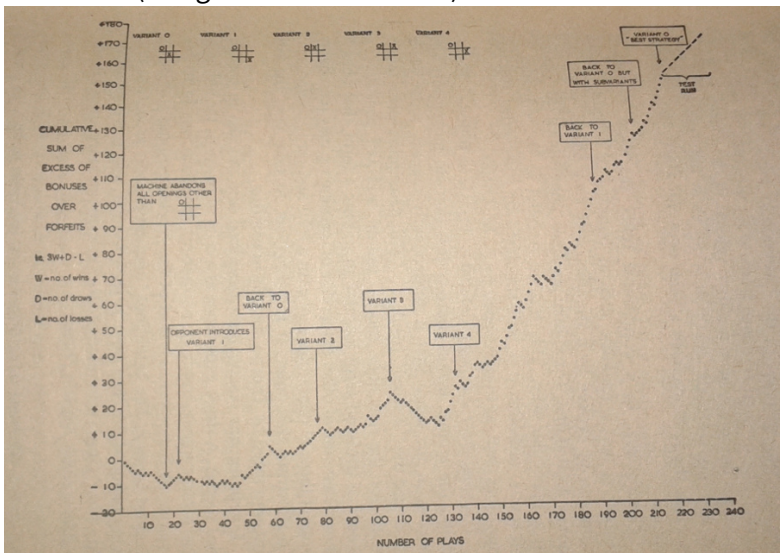
MENACE (Michie, 1963)

https://www.mscroggs.co.uk/blog/19

Machine Educable Noughts And Crosses Engine
Each coloured marble represents a position (1-9) to move to. Start each box
MENACE always moved first. Had four sets of matchboxes (totalling 304), one for each step of the game (no choice for potential fifth move).
Each matchbox contains coloured marbles (more than one of each colour); pick a marble **at random** to move to the corresponding square.
Keep matchboxes out on table; at end of game either:
win (draw): add 3 (1) of marble chosen at each stage back into corresponding box.
lose: remove the marble at each stage that was chosen.
Concedes if a mathbox is empty.

MENACE performance (Scroggs version of 1963 Fig. 4)

Menace vs Michie (220 games over 16 hours).



Recent implementation
`https://www.youtube.com/watch?v=R9c-_neaxeU`

What is going on here?

Delayed reward: reward only comes at end of the game.
Exploration vs exploitation tradeoff: we draw a marble randomly from a matchbox, rather than always taking one of the most common marbles – why?

Notation (Stone, Chapter 10)

In state s_t; action taken a_t leads to reward r_{t+1} and new state s_{t+1}.
Policy π defines probability of choosing an action a from all possible actions given state s_t.
Reward signal R_{t+1} is immediate reward for state s_t after taking action a_t.
Return G_t cumulative total award from trial $t+1$ to end of an episode. We use a discount factor γ so

$$G_t = \sum_{\tau=0}^{T} \gamma^\tau R_{t+\tau+1} = R_{t+1} + \gamma G_{t+1}$$

state-value function $v(s_t)$ is expected return based on state s_t. Our current estimate is given by $V(s_t)$.

$$v^\pi(s) = E_\pi[G_t | s_t = s]$$

activation-value function $q^\pi(s_t, a_t)$ is expected return based on current state s_t and action a_t specified by policy π. Current estimate is given by $Q(s_t, a_t)$.

The Bellman Equation (Stone, Section 10.6)

Given

$$v^{\pi}(s) = E_{\pi}[G_t | s_t = s]$$

which is

$$v^{\pi}(s) = E_{\pi}[R_{t+1} + \gamma G_{t+1} | s_t = s]$$

'it can be shown' (under Markov assumption) that ...

$$v^{\pi}(s) = E_{\pi}[R_{t+1} + \gamma v^{\pi}(s_{t+1}) | s_t = s]$$

Learning State-Value Functions (Stone, Section 10.7)

To estimate optimal policy, we need the state-value function $v^{\pi}(s_t)$, but that is unknown. If it were known, we could do gradient descent on

$$E = 1/2 \sum_t [V(s_t) - v(s_t)]^2$$

Failing that, we can say e.g. $v^{\pi}(s_t) \approx R_{t+1} + \gamma v^{\pi}(s_{t+1})$ and proceed (eqn 10.16 onwards) to get (10.28) an online update rule:

$$\Delta V_k(s_t) = \epsilon(R_{t+1} + \gamma V_k(s_{t+1}) - V_k(s_t))$$

s.t. when epsilon term is zero, $V_k(s_t)$ is correctly predicting future value.
This is *policy evaluation*.
Note that Sutton and Barto (2018) refer to this as 'learning a guess from a guess'. It works (see their textbook).

Learning action-value functions (Stone section 10.9)

State-value function and the action-value function:

$$v^{\pi}(s) = \sum_a \pi(a|s) q^{\pi}(s, a)$$

If we choose the optimal next action as

$$a^*_{t+1} = \underset{a}{\operatorname{argmax}}\, Q(s_{t+1}, a)$$

This is greedy; to explore (rather than exploit) with some small probability we can choose a random action instead.
By a similar approach to updating V we get an online (**SARSA**) rule:

$$\Delta Q(s_t, a_t) = \epsilon(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Q learning assumes optimal action taken at next state:

$$\Delta Q(s_t, a_t) = \epsilon(R_{t+1} + \gamma Q(s_{t+1}, a^*_{t+1}) - Q(s_t, a_t))$$

This is *policy improvement*.

Generalised Policy Iteration (Stone)

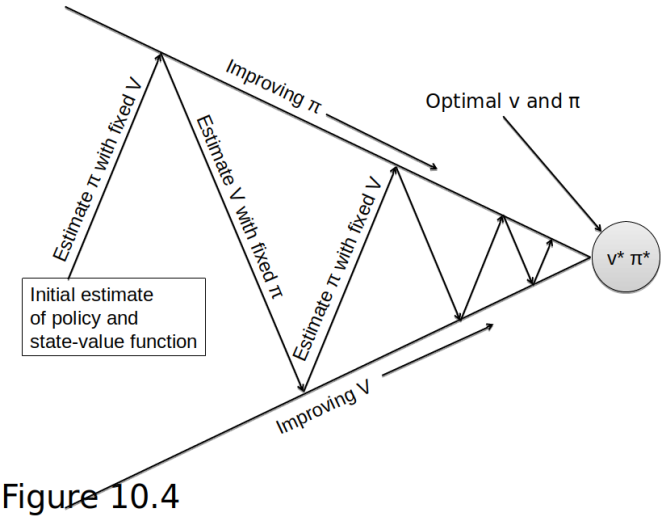
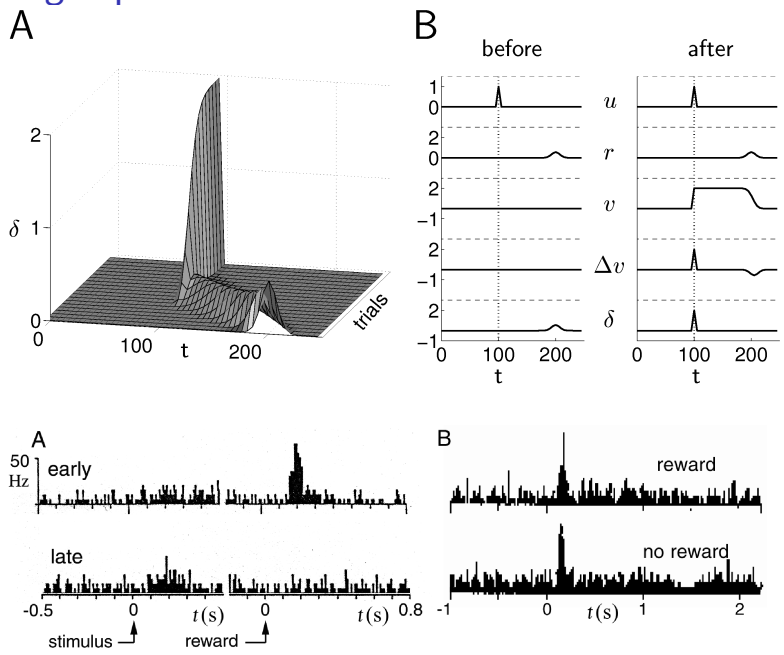


Figure 10.4

Learning to predict a reward

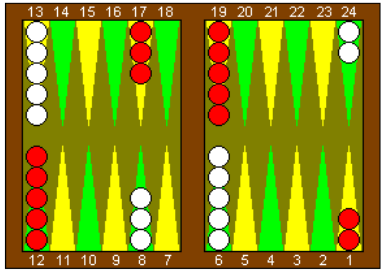


Model-based methods

These state-space approaches require estimation of transition probabilities; prohibitive for large state spaces.
Try instead to have a model, e.g. a neural network to estimate policy, or value-state.
Actor-critics architectures: actor performs current policy, whereas critic evaluates the policy. Early application: Barto et al (1983) cart pole balancing problem.

Real-world application of TD learning (Tesauro 1995)

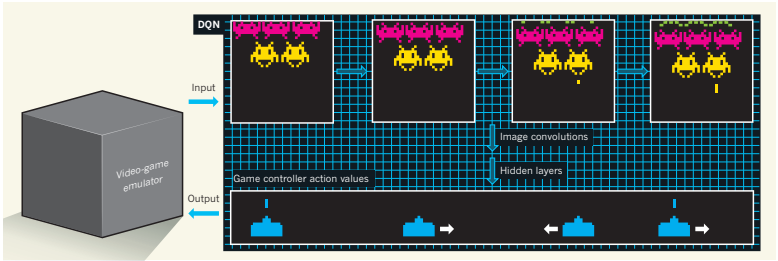
`http://www.research.ibm.com/massive/tdl.html`



Backgammon is perfect since it a noisy system (through dice-rolling); neural network takes state representation (198 units) and outputs 1 value, $V(t)$, i.e. chance of winning in that state.
After dice-rolled, each of ≈ 20 next states evaluated; pick state with maximal $V(t)$. Network trains against itself, with ultimate reward of $r = 1$ if player wins.
As of 1995, extremely close to equaling the world's best human players. Has taught experts new opening moves (e.g. splitting (24-23) rather than slotting (6-5)).

Recent developments

Mnih et al (2015) Human-level control through deep reinforcement learning. Nature 518:529–533.



System played better than professional human on 49 Atari 2600 games.
`https://www.youtube.com/watch?v=V1eYniJ0Rnk`

AlphaGo, AlphaGo Zero (Stone, p167)

- Go has about 10^{170} board positions. Much harder than chess.
- AlphaGo beat 2nd-ranked human player (Lee Sedol) in March 2016. Trained with human games.
- AlphaGo Zero started from scratch, with self-play. In 2017, AlphaGo Zero beat AlphaGo 100-0.
- AlphaZero is a generalised version of AlphaGo Zero.
- These machines are now teaching humans new moves (esp ones that could not be understood as good moves).