Reinforcement Learning

So far…

- Supervised learning
  - Targets given - algorithm is taught
- Unsupervised learning
  - No targets, algorithm exploits regularities
- Evolutionary learning
  - Search, exploitation and exploration

Reinforcement Learning

- Get a reward
  - Numerical signal saying how well performing
  - No information about how to improve
- Need to try out different actions to find best
  - Exploration
  - Exploitation

Psychological Motivation

- If something is good, you do it again
- If something is not good, you don't

- The Law of Effect
- Important psychological motivator
- Exactly what happens in reinforcement learning

Book

- Reinforcement Learning: An Introduction
- Richard Sutton and Andrew Barto

- Available free at:

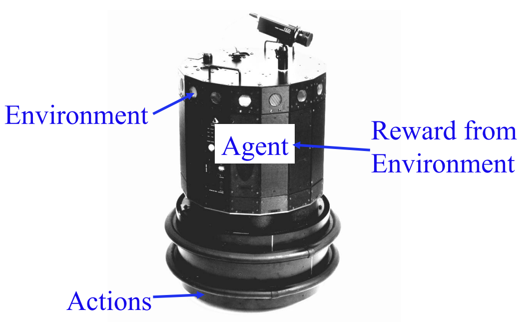
http://www-anw.cs.umass.edu/~rich/book/the-book.html

Overview

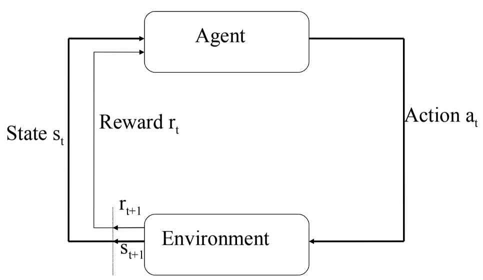
- Learn to map *states* onto *actions* to maximise some *reward* now and in the future
- Learn while interacting with environment

- States: perceptions about the environment
- Actions: things can do in current state (change environment)
- Reward: numerical value from environment

Overview



Overview



Overview

- Agent tries to work out what state it is in
- The policy decides what action to take
- The action is taken
- Reward function produces a reward based on the state and action
- Agent arrives in a new state
- Iterate

Overview

- Reinforcement learning decides what to do based on experience
  - When I've been in this state before, what reward did I get?
  - How was this reward linked to the actions?
  - Select action that maximises *expected* reward
  - Occasionally try something different

State Spaces

- Number of possible different states
- Number of possible different actions

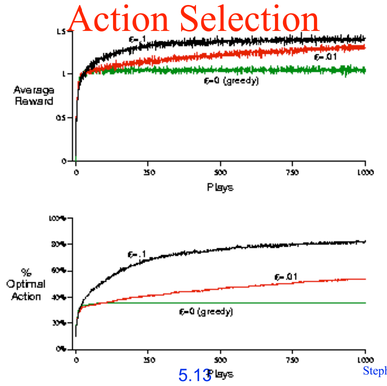
- Curse of dimensionality
- Trade-off between throwing away information, and not being computable

Action Selection

- Exploitation and Exploration
  - Greedy
  - ϵ-greedy
  - Soft-max

$$\frac{\exp(Q_t(a)/\tau)}{\sum_b \exp(Q_t(b)/\tau)}$$

Action Selection



Rewards

- Choice of action is based on expected reward
- Reward function takes current state and chosen action, and returns numerical reward
  - Positive or negative
- Based on the goal
- Crucial to good learning

Two Reward Schemes

- Robot is learning to find the centre of a maze
  - Get a reward of 50 at centre
  - Get a reward of -1 each move and +50 at centre
- This problem is *episodic*
  - Breaks into 'games'
  - Has a *terminal state*
- Not always true - continual learning

Continual Learning

- Why does this matter?
- We are predicting the rewards in the future
- This is an infinite sum, may be divergent
- Also, we don't know what will happen in the future - guesses may be wrong

Discounts

- Discount (reduce belief in) future actions
- Extra parameter

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{k-1} r_k + \ldots$$
$$= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

The Markov Property

- We are trying to predict reward *r'* for action *a* in state *s'*

$$Pr(r_t = r', s_{t+1} = s' | s_t, a_t, r_{t-1}, s_{t-1}, a_{t-1}, \ldots r_1, s_1, a_1, r_0, s_0, a_0)$$

- We can't possibly compute that
  - Need too much data
  - Probabilities will be too small
- Do we need it all?

The Markov Property

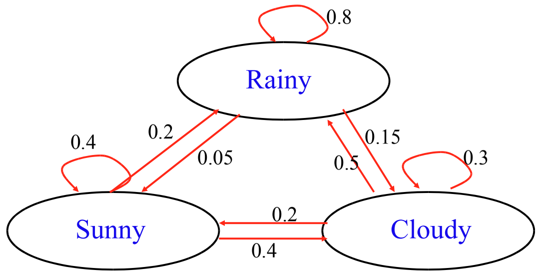
$$Pr(r_t = r', s_{t+1} = s' | s_t, a_t)$$

- Current state tells us enough
  - Example: Chess
  - Can include probability distributions
- Basis of reinforcement learning (and other algorithms)

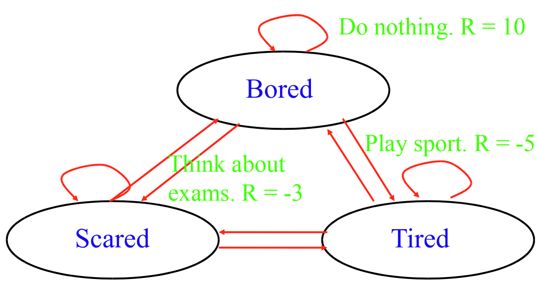
Markov Decision Processes

- Given a state, want to predict the next state and the reward for each action
- Based on experience
- Need to make a transition diagram

Markov Decision Processes



Markov Decision Processes



Goal: Rest before exam

Policy

- Soft-max and ∫-greedy are naïve policies
- Better if we can *learn* a more useful policy that is specific to the particular state

  $a_t = \pi(s_t)$
- We want the optimal policy, the one that produces the greatest rewards
- Learning policies is the crux of RL

Values

- Want to maximise expected reward in future
  - Value
- Options
  - Consider current state, average across actions
    - State-value function V(s)
  - Consider current state, look at each action
    - Action-value function Q(s, a)

Learning Policy

Aims

- Predict the value function V or Q for each policy
- Select the optimal policy
  - Value function greatest over all possible states
  - Not necessarily unique
- We're going to look at two methods:
  - Temporal Difference (TD) learning
  - Q-learning

Updating Value Estimates

- Aim is to predict value based on experience
- Could finish episode and then use

$$V(s_t) \leftarrow V(s_t) + \alpha(R_t - V(s_t))$$

Updated value
Old value
Learning parameter
Total reward so far

- Would eventually converge to true values

Temporal Differences

- What if not episodic?
- Plus, want to learn on-line

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

Current reward
Discounting parameter

- Exploit difference between current and previous time estimates

Temporal Differences

- Can use longer time predictions
- Need to keep track of the states we've visited
  - Eligibility trace
  - Don't trust updates to states haven't visited
- Trust prediction proportional to how forward in time it goes, just like with discounting
- This is the TD(λ) algorithm

SARSA

- We just used *V*(s), can also use *Q*(s, a)
- Then need (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

- These are *on-policy* methods
- Alternative: make Q approximate the optimal value independently of policy

Q-Learning

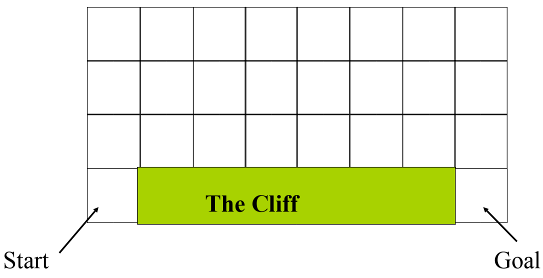
- Always look for the optimal value of Q
  - Search over all actions

$$Q(s_t, a_t) \leftarrow \alpha \left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \Big)$$

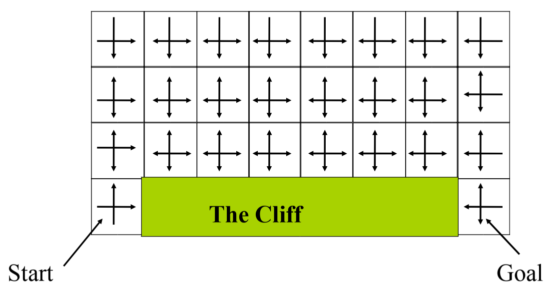
Comparisons

- Both algorithms:
  - Bootstrap
  - Are iterative
- What is the difference?
  - Q-learning always tries to follow optimal policy
  - SARSA looks at average

Easy as Falling Off a Cliff



Easy as Falling Off a Cliff



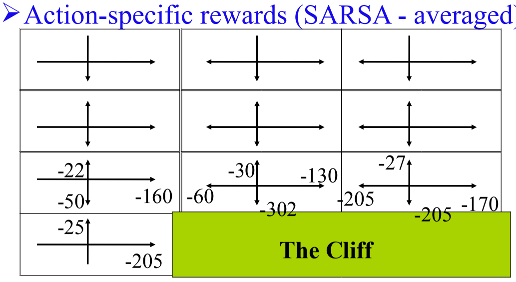
Easy as Falling Off a Cliff

- Average rewards over all actions

-28	-27	-25
-30	-35	-32
-120	-310	-210
-201	The Cliff	

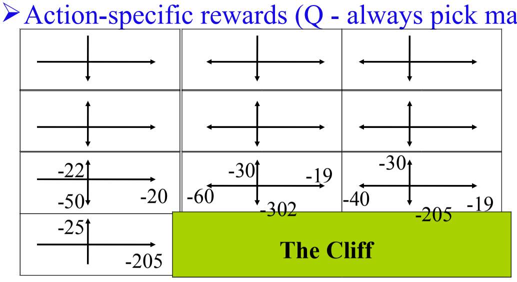
Easy as Falling Off a Cliff

- Action-specific rewards (SARSA - averaged)



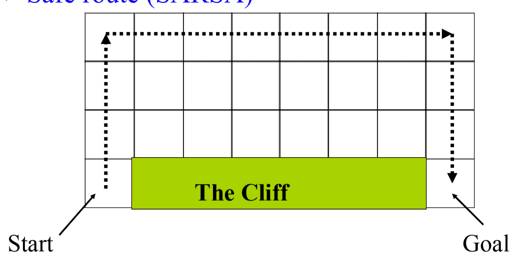
Easy as Falling Off a Cliff

- Action-specific rewards (Q - always pick max)



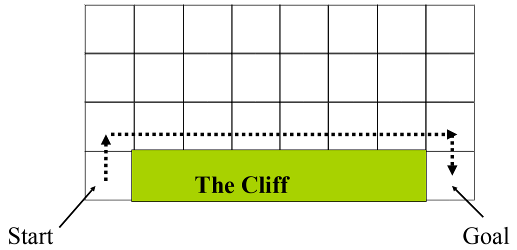
Easy as Falling Off a Cliff

- Safe route (SARSA)



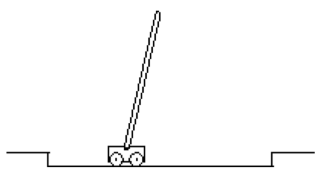
Easy as Falling Off a Cliff

- Optimal (dangerous) route (Q-learning)



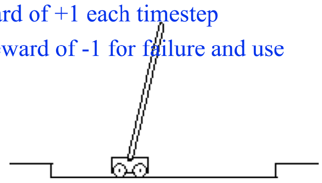
Reinforcement Learning Examples

- Pole Balancing
  - Apply forces to balance the pole
  - Don’t run out of track



Reinforcement Learning Examples

- Episodic or Continuous?
- Both equivalent
  - Episodic - reward of +1 each timestep
  - Continuous - reward of -1 for failure and use discounting



Reinforcement Learning Examples

- Box Pushing
  - Learn to clear boxes to the side of a room using Q-learning
  - Mahadevan and Connell

Reinforcement Learning Examples

- 8 sonar sensors, 4 to the front and 2 to each side, which were quantised to return NEAR or FAR
- An infra-red sensor to say if there was something directly in front of the robot (BUMP)
- A STUCK signal if the robot could not drive forward
- This made 18 bits of information (=262,144 states)

Reinforcement Learning Examples

- Motor responses could be:
  - Forward
  - Left
  - Hard left
  - Right
  - Hard right

Reinforcement Learning Examples

- They had the robot learn three different behaviours:
  - Find boxes (reward 3 if goes forward when front sensors say NEAR, -1 if a NEAR sensor goes off)
  - Push boxes (reward 1 if goes forward and BUMP stays on, -3 if BUMP goes off)
  - Unwedge if stuck (reward 1 if STUCK is turned off, -3 if stays STUCK)

Reinforcement Learning Examples

- Quite a lot of training (approximately 2000 times steps, about 2 hours)
- Results are comparable to hand-crafted code

Cautionary Notes

- States are often noisy
  - Not really observable
  - Can be partially observable
  - Or hidden
    - ✓ Hidden Markov Models well known
    - ✓ E.g., speech recognition

Cautionary Notes

- Reinforcement learning is search
  - Can be very slow
  - No guarantee of convergence to global optima
  - Can get stuck in flat regions
- Reward function is super critical