# C4: Markov Decision Processes

# Markov Decision Processes

## * Decision Making & Reinforcement Learning

So far we've looked at supervised and unsupervised learning. Reinforcement learning is inspired in part by the behaviourist school of thought in psychology. It is a learning system where rewards and punishments are assigned to different scenarios with different weights to dictate prioritisation.

→ Supervised Learning : $y = f(x)$

Function approximation – Given a set of $x$ and $y$ find a function $f$ that will map some new $x$ to a proper $y$.

→ Unsupervised Learning : $f(x)$

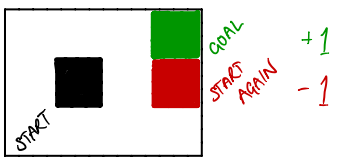
Clustering description - Given a set of $x$ find an $f$ that gives a compact description of that set of $x$

→ Reinforcement Learning : $y = f(x)$

Given $x$ and $3$ find $y$ and $f$
↑ $s$ ↑ $r$ ↑ $a$ ↑ $\pi^*$

## * Markov Decision Processes

PROBLEM




STATES : $s$

ACTIONS : $A(s)$ , $A$ e.g. U/D/L/R

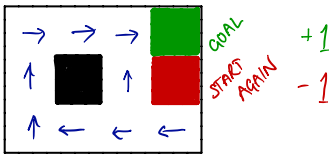
MODEL : $T(s, a, s') \sim P(s' \mid s, a)$ { Probability of moving in the direction you choose is 80%. There is a 10% chance of moving in either direction 1'lar to that

REWARD : $R(s)$ , $R(s, a)$ , $R(s, a, s')$ e.g. +1, -1 else -0.04
Temporal Credit Assignment

# SOLUTION

POLICY : $\pi(s) \rightarrow a$

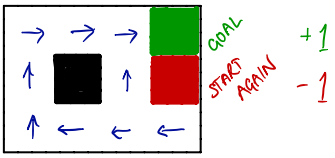
$\pi^*$ optimal policy maximises the reward you receive or expect to receive over your lifetime



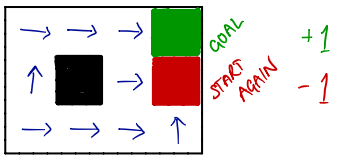

We consider the following types of problems:
→ Markovian Property - Only the present matters
→ Stationary models - Time homogeneous

The reward function matters a lot :

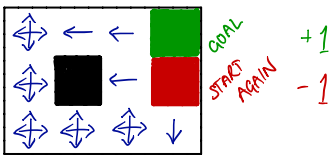
reward = -0.04




reward = -2 < 1.6284




reward = +2




* Sequences of rewards

- Stationary Assumption ⇒ Infinite time horizon
  i.e. $\pi(s) \rightarrow a$ not $\pi(s,t) \rightarrow a$

- Stationary Preferences ⇒ Utility of sequences (of states)

  A utility is a function which maps a sequence of states to a real number. It allows us to express the preference of one sequence over another.

  So, preferences being stationary means

  if $U(s_0, s_1, \dots) > U(s_0, s_1', \dots)$ then $U(s_1, s_2, \dots) > U(s_1', s_2', \dots)$

  Suppose rewards are all positive, $R(s_t) \geq 0 \ \forall \ t$, we can't just sum rewards for our utility function because the sum could be infinite:

$$U(s_0, s_1, \dots) = \sum_{t=0}^{\infty} R(s_t)$$

We have to discount them:

$$U(s_0, s_1, \dots) = \sum_{t=0}^{\infty} \gamma^t R(s_t), \quad 0 \le \gamma < 1$$

$$U(s_0, s_1, \dots) \le \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$$

## ✻ The optimal policy and Bellman's Equation

The optimal policy is given by

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \, \mathbb{E}\left( \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \right)$$

$$U^{\pi}(s) = \mathbb{E}\left( \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s \right)$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left( \sum_{s'} T(s, a, s') \, U^{\pi^*}(s') \right)$$

($T$: transition probability)

$$U^{\pi^*}(s) = R(s) + \gamma \max_a \left( \sum_{s'} T(s, a, s') \, U^{\pi^*}(s') \right)$$ Bellman's Equation

↳ $n$ non-linear equations in $n$ unknowns ($n$ = number of states $s$)

## ✻ Value Iteration

The solution of Bellman's equation can be found using the following algorithm:

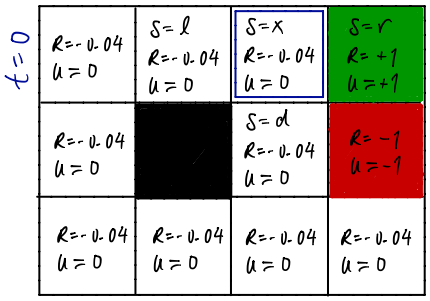
- Start with some arbitrary utilities
- Update utilities based on neighbours:

$$\hat{U}^{\pi^*}_{t+1}(s) = R(s) + \gamma \max_a \left( \sum_{s'} T(s, a, s') \, \hat{U}^{\pi^*}_{t}(s') \right)$$

(we use hats to denote the approximation; sum over neighbouring states)

- Repeat until converged

Example: Find $U^{\pi^*}_1(x)$ and $U^{\pi^*}_2(x)$




$\gamma = 1/2$

$R(s) = -0.04$

$U_0(s) = 0$

(unless marked otherwise)

Transition matrix for $s = x$:

$$T(s, a, s') = \begin{pmatrix} 0.8 & 0.1 & 0 & 0.1 \\ 0.1 & 0.8 & 0.1 & 0 \\ 0 & 0.1 & 0.8 & 0.1 \\ 0.1 & 0 & 0.1 & 0.8 \end{pmatrix} \begin{matrix} \uparrow \\ \rightarrow \\ \downarrow \\ \leftarrow \end{matrix}$$

$s'$ \ $a$: x e s w

$t=0$

| | | | |
|---|---|---|---|
| $R=-0.04$ $U=0$ | $S=l$ $R=-0.04$ $U=0$ | $S=x$ $R=-0.04$ $U=0$ | $S=r$ $R=+1$ $U=+1$ |
| $R=-0.04$ $U=0$ | | $S=d$ $R=-0.04$ $U=0$ | $R=-1$ $U=-1$ |
| $R=-0.04$ $U=0$ | $R=-0.04$ $U=0$ | $R=-0.04$ $U=0$ | $R=-0.04$ $U=0$ |

$$U_1(x) = -0.04 + \frac{1}{2} \max_a \begin{cases} \uparrow : (0.8\times 0) + (0.1\times 1) + (0.1\times 0) \\ \rightarrow : (0.8\times 1) + (0.1\times 0) + (0.1\times 0) \\ \downarrow : (0.8\times 0) + (0.1\times 1) + (0.1\times 0) \\ \leftarrow : (0.8\times 0) + (0.1\times 0) + (0.1\times 0) \end{cases}^{*} = 0.36$$

we note the maximal value is in the direction where the reward is greatest i.e. $S=r$

$$U_1(d) = -0.04 + \frac{1}{2} \max_a \begin{cases} \uparrow : (0.8\times 0) + (0.1\times -1) + (0.1\times 0) \\ \rightarrow : (0.8\times -1) + (0.1\times 0) + (0.1\times 0) \\ \downarrow : (0.8\times 0) + (0.1\times -1) + (0.1\times 0) \\ \leftarrow : (0.8\times 0) + (0.1\times 0) + (0.1\times 0) \end{cases} = -0.04 \quad {}^{*}$$

with the current zero utilities the best policy at $S=d$ is $\leftarrow$ but this will change in future iterations

we suspect we don't need to calculate this since the best policy at $S=x$ will likely be $\rightarrow$ again

$$U_1(l) = -0.04 + \frac{1}{2} \max_a \begin{cases} \uparrow : (0.8\times 0) + (0.1\times 0) + (0.1\times 0) \\ \rightarrow : (0.8\times 0) + (0.1\times 0) + (0.1\times 0) \\ \downarrow : (0.8\times 0) + (0.1\times 0) + (0.1\times 0) \\ \leftarrow : (0.8\times 0) + (0.1\times 0) + (0.1\times 0) \end{cases} = -0.04$$

$t=1$

| | | | |
|---|---|---|---|
| $R=-0.04$ | $S=l$ $R=-0.04$ $U=-0.04$ | $S=x$ $R=-0.04$ $U=0.36$ | $S=r$ $R=+1$ $U=+1$ |
| $R=-0.04$ | | $S=d$ $R=-0.04$ $U=-0.04$ | $R=-1$ $U=-1$ |
| $R=-0.04$ | $R=-0.04$ | $R=-0.04$ | $R=-0.04$ |

once again, the maximal value is in the direction where the reward is greatest

$$U_2(x) = -0.04 + \frac{1}{2} \max_a \begin{cases} \uparrow : (0.8\times 0.36) + (0.1\times 1) + (0.1\times -0.04) \\ \rightarrow : (0.8\times 1) + (0.1\times -0.04) + (0.1\times 0.36) \\ \downarrow : (0.8\times -0.04) + (0.1\times -0.04) + (0.1\times 1) \\ \leftarrow : (0.8\times -0.04) + (0.1\times -0.04) + (0.1\times 0.36) \end{cases}^{*} = 0.376$$

Notes:
- Utility values propogate out from their neighbours
- We don't need to know the exact values of

$$\sum_{s'} T(s,a,s') U_t^{\pi^*}(s')$$

for each action only how they compare to each other for the neighbouring states.

Actually what we really want is the optimal policy, not the utility function. To find this, we don't need to know the exact values for the utility function, only the relative values. This leads us to our next algorithm.

* Policy Iteration

The solution of Bellman's equation can be found using the following algorithm

- Start with $\pi_0$ ← guess
- Evaluate: given $\pi_t$ calculate $U_t = U^{\pi_t}$

$$U_t(s) = R(s) + \gamma \left( \sum_{s'} T(s, \pi_t(s), s') \, U_t(s') \right)$$

↳ n linear equations in n unknowns (n = number of states s)

- Improve:

$$\pi_{t+1}(s) = \underset{a}{\operatorname{argmax}} \left( \sum_{s'} T(s, a, s') \, U_t(s') \right)$$

Notes:

- No max so our new equation is LINEAR
- More expensive to evaluate (matrix inversion - $O(n^3)$)
- Converges faster
- Makes bigger jumps because we are working in policy space rather than utility space.
- Guaranteed to converge - proof is like k-Means

* What does all this have to do with reinforcement learning

This should make it easier to think about / understand how reinforcement learning works.

In reinforcement learning you don't necessarily know what the rewards or transitions are or even the states or actions for that matter.

# Reinforcement Learning

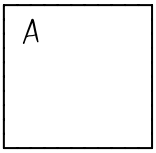
...In Markov Decision Processes

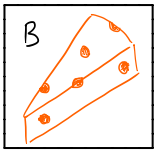
* Reinforcement Learning API

Model (T, R) → PLANNER → Policy (π) } Planning

Transitions (s, a, s', r) → LEARNER → Policy (π) } Reinforcement Learning

Brief aside: Reinforcement Learning 1930's




Animal sees stimulus (red/green light) s
takes an action (opens a door) a
receives a reward (cheese?) r

Strengthens response to the stimulus
Reward Maximisation

Transitions (s, a, s', r) → MODELER → Model (T, R)

Model (T, R) → SIMULATOR → Transitions (s, a, s', r)

TD Gammon (1992) Gerald Tesauro IBM

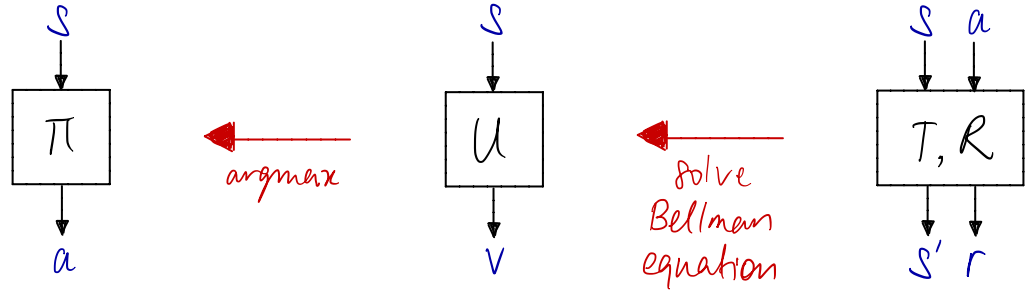
Model → SIMULATOR → Transitions → LEARNER → Policy

Reinforcement learning based planner

Transitions → MODELER → Model → PLANNER → Policy

Model based reinforcement learning

## * Three Approaches to Reinforcement Learning




Policy Search

Value Function Based

Model Based

Direct Use
Indirect Learning

We'll focus on this

Direct Learning
Indirect Use

## * A new kind of value function

Before we had

Value function:

$$U(s) = R(s) + \gamma \max_a \left( \sum_{s'} T(s, a, s') U(s') \right)$$

Policy:

$$\pi(s) = \operatorname{argmax}_a \left( \sum_{s'} T(s, a, s') U(s') \right)$$

Our new value function - Q function:

$$Q(s, a) = R(s) + \gamma \sum_{s'} T(s, a, s') \max_{a'} Q(s', a')$$

Value for arriving in s, leaving via a, proceeding optimally thereafter

Note,

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

and

$$\pi(s) = \operatorname{argmax}_a Q(s, a)$$

In Q-learning we are evaluating the Bellman equations from data

## * Estimating Q from transitions

$$Q(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(s',a')$$

The problem here is we don't know $R$ or $T$, we only have transitions:

$\langle s, a, r, s' \rangle$:

$$\hat{Q}(s,a) \xleftarrow{\alpha_t} r + \gamma \max_{a'} \hat{Q}(s',a')$$

($\alpha_t$: learning rate; $\hat{Q}$: again we hat to denote estimate; $\max_{a'} \hat{Q}(s',a')$: utility of next state; $r + \gamma \max_{a'} \hat{Q}(s',a')$: utility of state)

Notation:
$v \xleftarrow{\alpha} x$
$v \leftarrow (1-\alpha)x + \alpha v$

Aside: $v_t \xleftarrow{\alpha_t} x_t \quad x \sim X, \quad \sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$

$v_t$ converges to $\mathbb{E}(X)$

e.g. $\alpha_t = 1/t$

$\sum_{i=1}^{t} 1/i = \ln t, \quad \sum_{t=1}^{\infty} 1/t^2 = \pi^2/6$

Back to estimating transitions...

$\langle s, a, r, s' \rangle$:

$$\hat{Q}(s,a) \xleftarrow{\alpha_t} r + \gamma \max_{a'} \hat{Q}(s',a')$$

(changing over time)

doesn't really work

$$= \mathbb{E}\left[ r + \gamma \max_{a'} \hat{Q}(s',a') \right]$$
$$= R(s) + \gamma \mathbb{E}\left[ \max_{a'} \hat{Q}(s',a') \right]$$
$$= R(s) + \gamma \sum_{s'} T(s,a,s') Q(s',a')$$

## * Q-Learning Convergence

$\hat{Q}$ starts anywhere $\quad \hat{Q}(s,a) \xleftarrow{\alpha_t} r + \gamma \max_{a'} \hat{Q}(s',a')$

then $\quad \hat{Q}(s,a) \to Q(s,a)$ soln of the Bellman eqn

if $\forall s,a$ visited infinitely often, $\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$

and $\quad s' \sim T(s,a,s'), \quad r \sim R(s)$

## ✱ Choosing Actions

Q-learning is a family of algorithms which typically vary in the following ways

1. how actions are chosen
2. how $\hat{Q}$ is initialised
3. how $\alpha_t$ decays

How could we choose actions?

→ always choose $a_0$ → won't explore / learn
→ choose randomly → won't use what's learned
→ use $\hat{Q}$ (always choose optimal action) → greedy learner, local minima
→ random restarting → too slow, we already need to visit s,a infinitely often

We want to combine learning and some randomness in order to optimise over the whole state space. To do this we use a "simulated annealing" like approach where we sometimes take a random action,

$$\hat{\pi}(s) = \begin{cases} \underset{a}{\text{argmax}}\ \hat{Q}(s,a) & \text{with probability } 1-\varepsilon \\ \text{random action} & \text{with probability } \varepsilon \end{cases}$$

## ✱ ε Greedy Exploration

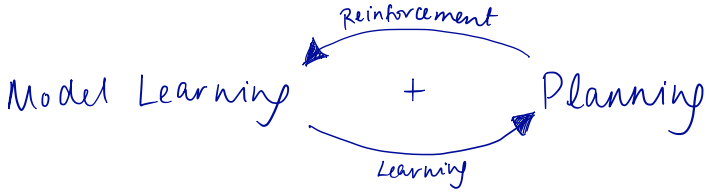
In Greedy Limit and Infinite Exploration (GLIE) ε decays,

$$\hat{Q} \to Q \quad \text{and} \quad \hat{\pi} \to \pi^*$$

learn — use

Exploration - Exploitation Dilemma

Fundamental trade-off in Reinforcement Learning

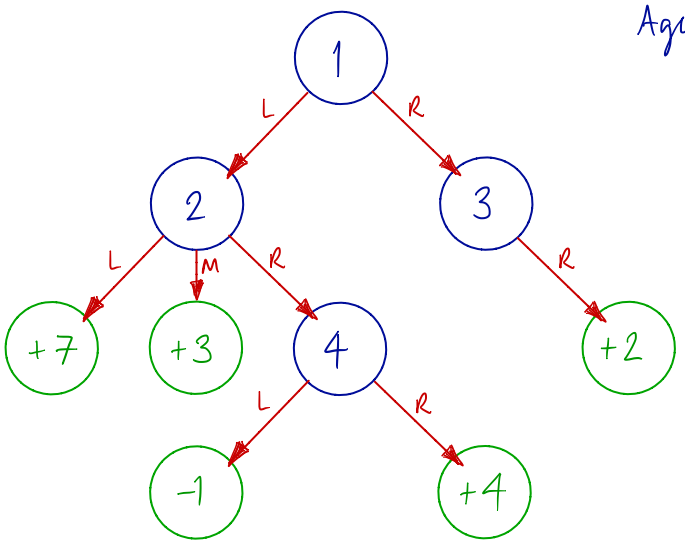


# Game Theory

* What is game theory?
  - Mathematics of conflict
  - Single agent → multiple agents
  - Economics, politics, sociology, biology, ...
  - Increasingly a part of machine learning and artificial intelligence

* A simple game

Two player (agents A and B), zero sum (really this means const. sum), finite, deterministic game of perfect information




Agent Choice
A
B
A

Four states in blue

Leaves show the score for A in green. The score for B is minus the score for A.

| B ② | L | M | R |
|---|---|---|---|
| A ③ | R | R | R |
| ① ④ | | | |
| L L | 7 | 3 | -1 |
| L R | 7 | 3 | 4 |
| R L | 2 | 2 | 2 |
| R R | 2 | 2 | 2 |

← Matrix form of the game

Tells us everything we need to know about the game.

What's the optimal strategy for each agent?

For agent A, this is given by the row with the best worst case scenario for A
For agent B, this is given by the column with the best worst case scenario for B or equivalently the worst best case scenario for A.

It turns out that for these types of games, the score or reward from the optimal strategy for both players is the same.

Fundamental Result: Von Neumann

don't need the game to be deterministic

In a

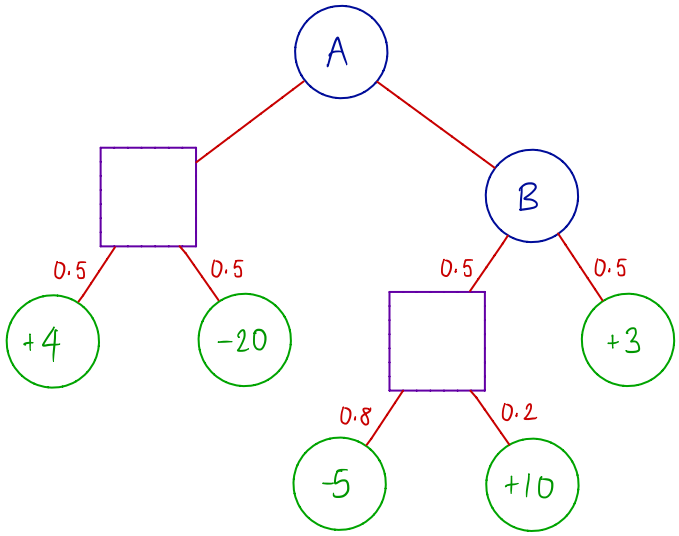
Two player, zero sum, finite, ~~deterministic~~ game of perfect information,

Minimax ≡ Maximin

and

There always exists an optimal pure strategy for each player.

* A simple non-deterministic game




Our fundamental theorem still holds (Von Neumann)
Again, all we need is the matrix form of the game

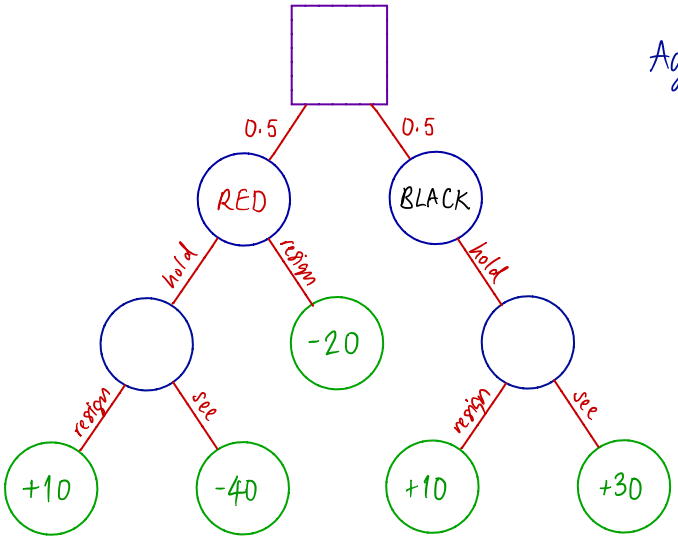
| A \ B | L | R |
|---|---|---|
| L | -8 | -8 |
| R | -2 | 3 |

* A simple non-deterministic game of hidden information

Two player, zero sum, finite, non-deterministic game of hidden information.

Mini Poker:
- A is dealt a card, red or black 50%
- A may resign if red: -20¢ for A
  else A holds
  - B resigns: +10¢
  - B sees:
    - if red: -40¢
    - if black: +30¢

# Mini Poker Game Tree




Agent Choice

A } Always hold if Black
Either resign or hold if red

B } B doesn't know if the card is red or Black and can choose to resign or see the card dealt

Matrix form of the game:

| A \ B | resign | see |
|---|---|---|
| resign | -5 | +5 |
| hold | +10 | -5 |

↑ refers to the strategy in the case A is dealt a red card only

Von Neumann does not hold when there is hidden information.
i.e. Minimax ≠ Maximin

← Pure strategy we need a mixed strategy

## * Mixed strategy

Here we have a distribution over the strategies.

Example: Suppose that A takes the following mixed strategy

If the card is red then p of the time A holds and 1-p of the time A resigns.

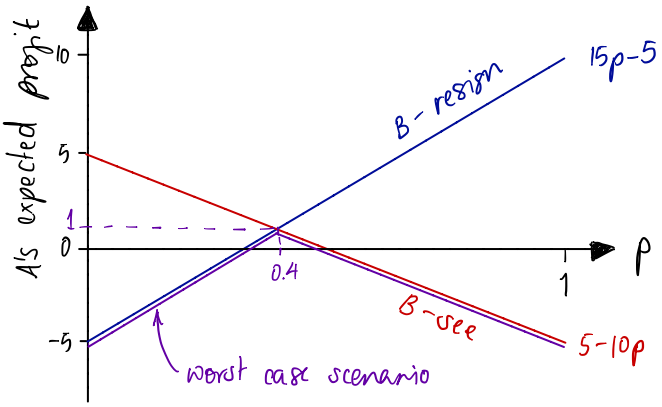
If B always resigns, A's expected profit is

$$10p - 5(1-p) = 15p - 5$$

If B always holds, A's expected profit is

$$-5p + 5(1-p) = 5 - 10p$$

Graphically...

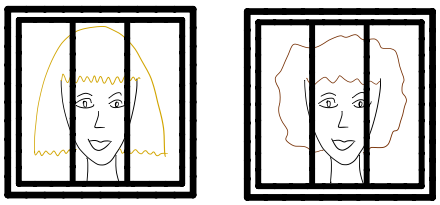



The best, worst case scenario is in this case where the lines intersect, at $p = 0.4$. A's expected profit with this strategy is 1.

Note that if B employs a mixed strategy the result is the same.

* Two player non-zero sum non-deterministic game of hidden information

Prisoners Dilemma:



A B

Two prisoners, A and B, are arrested for their involvement in a crime. Each prisoner has the option of either snitching or not.

| (A, B) | don't snitch | snitch |
|---|---|---|
| don't snitch | (-1, -1) | (-9, 0) |
| snitch | (0, -9) | (-6, -6) |

The matrix shows the jail time they get in each case.

Note, the matrix is symmetric.

If we think collectively and combine the jail times it's clear that if neither prisoner snitches the combined jail time is minimised.

If each player simply aims to minimise their own jail time, it's always better to snitch. The problem is this 'selfish' strategy, when employed by both prisoners leads to the worst possible combined jail time. The snitching strategy dominates.

## * The Nash Equilibrium (as in John Nash of A Beautiful Mind)

Suppose we have $n$ players which can each choose a strategy from their set of all possible strategies,

$$S_1, S_2, \ldots, S_n$$

The collective strategy

$$s_1^*, s_2^*, \ldots, s_n^* \quad \text{where} \quad s_1^* \in S_1, \; s_2^* \in S_2, \; \ldots, \; s_n^* \in S_n$$

is a Nash Equilibrium if and only if

$$\forall i \quad s_i^* = \underset{s_i}{\operatorname{argmax}} \; u(s_1^*, \ldots, s_i, \ldots, s_n^*)$$

That is to say, assuming all other players stick with the strategy they chose, each player has chosen the best possible strategy and has no motivation to switch.

Example:

| (A, B) | don't snitch | snitch |
|---|---|---|
| don't snitch | (-1, -1) | (-9, 0) |
| snitch | (0, -9) | (-6, -6) |

Nash Equilibrium

## * Some results

In the $n$-player pure strategy game, if elimination of strictly dominated strategies eliminates all but one combination, that combination is the unique Nash Equilibrium

Any Nash Equilibrium will survive elimination of strictly dominated strategies

If $n$ is finite & $\forall i$ $S_i$ is finite $\exists$ at least one (possibly mixed) Nash Equilibrium

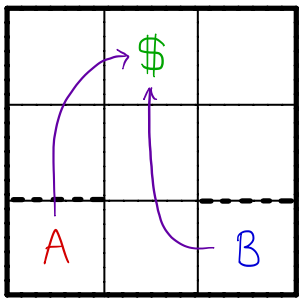
## * Final notes

Utility isn't just the reward - Mechanism Design

# * Stochastic games and multiagent reinforcement learning

MDP : RL :: STOCHASTIC GAME : MULTI-AGENT RL

Example :




- Actions : N, E, S, W, X
- First to reach goal gets $100
- Both arrive both win
- Semi-wall (50% go through)
- Coin flip if collide

Two Nash Equilibria in this case (one is shown)

Similarly to MDPs we have ... (Shapley)

| | | |
|---|---|---|
| $S$ : | states | $s$ |
| $A_i$ : | actions for player $i$ | $a, b \quad a \in A_1, \quad b \in A_2$ |
| $T$ : | transitions | $T(s, (a,b), s')$ |
| $R_i$ : | rewards for player $i$ | $R_1(s, (a,b)), \; R_2(s, (a,b))$ |
| $\gamma$ : | discount | $\gamma$ |

# * Zero sum stochastic games

Bellman Equation for stochastic games:

$$Q^*(s,(a,b)) = R_i(s,(a,b)) + \gamma \sum_{s'} T(s,(a,b),s') \text{ minimax } Q^*(s,(a,b))$$

$\langle s, (a,b), (r_1, r_2), s' \rangle$ : minimax - Q

$$Q_i(s,(a,b)) \xleftarrow{\alpha} r_i + \gamma \min_{a',b'} \text{minimax } Q_i(s',(a',b'))$$

- Value iteration works
- Minimax Q converges
- Unique solution to $Q^*$
- Policies for the two players can be computed independently
- Update efficient (polynomial time)
- Q function sufficient to specify policy

* General sum stochastic games

$$Q^*(s,(a,b)) = R_i(s,(a,b)) + \gamma \sum_{s'} T(s,(a,b),s') \overset{\text{Nash}}{\cancel{\text{minimax}}} Q^*(s,(a,b))$$

$\langle s,(a,b),(r_1,r_2),s' \rangle$ : ~~Minimax~~ Nash - Q

$$Q_i(s,(a,b)) \overset{\alpha}{\leftarrow} r_i + \gamma \overset{\text{Nash}}{\underset{a',b'}{\cancel{\text{minimax}}}} Q_i(s',(a',b'))$$

- Value iteration ~~works~~ doesn't work
- ~~Minimax~~ Nash Q doesn't converge
- Unique solution to $Q^*$ does not exist
- Policies for the two players can't be computed independently
- Update inefficient (~~polynomial time~~) P=PPAD
- Q function insufficient to specify policy

Oh dear !!!

* Some ideas for general sum stochastic games

- repeated stochastic games (folk theorem)
- cheap talk → correlated equilibria
- cognitive hierarchy → best responses
- side payments → coco values