Multi-agent system college notes 2022

Multiagent systems (Vrije Universiteit Amsterdam)

Multi-agent system

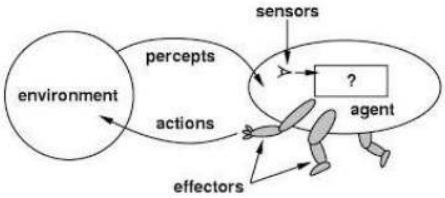
Lecture 1: Introduction course

Overview of Today's Lecture Part 1:

- Course Organisation
- Agents and Multi-Agent Systems
- Agent Types
- Environments
- Topics discussed in this course

Agents and Multi-Agent Systems

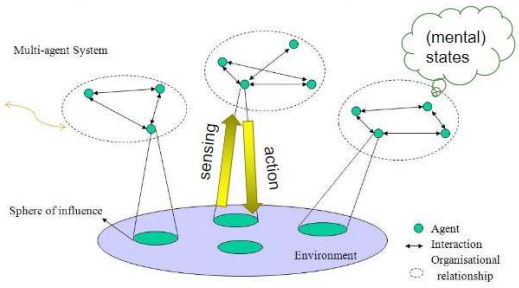
- What is an Agent? → Single agent
  - An **agent** is a computer system that is situated in some **environment**, and that is capable of **autonomous** action in this environment in order to meet its delegated **objectives**.
  - Autonomous -> not completely pre-programmed. So you don't need to pre-program all of his actions.



    - Note: autonomy is a **spectrum**!

- Multi-Agents Systems, a Definition → multiple agents. All of these agents has goals, but it tends that these agents has conflicting goals, what makes it challenging/difficult. So to be successful the agents need to learn, cooperate, coordinate, and negotiate.
  - A **Multi-Agent** System is one that consists of a number of agents that **interact (with each other and the environment)**.
  - In general, agents will have **different goals (often conflicting!)**
  - To successfully interact, they will have to **learn, cooperate, coordinate,** and **negotiate**.

- Agents and Environment
  - Multi-agent Systems (MAS)



    ➔ Green dots are agents, which can observe and act according to the observation in the environment.

➔ They can also make coalitions (see the circles), but this is exceptional. In general, they tend to be only interested in their own objectives.

- Motivations for studying MAS (1)
  - **Technological:**
    - Growth of distributed, **networked computer** systems.
      - (computers act more as **individuals** than parts).
    - **Robustness:** no single point of failure.
    - **Scalable** and **flexible**:
      - Adding new agents when needed.
      - Asynchronous, parallel processing.
    - **Development** and **reusability**
      - Components developed independently (by specialists).
  - Application: Robotics (example of technological motivation)
    - Robots as Physical Agents (Embodiment)
      - Internet of Things (IoT).
      - Swarms of drones.
      - Fleet of autonomous vehicles.
      - Physical internet.

- Motivations for studying MAS (2)
  - **Scientific:**
    - Models for **interactivity** in (human) **societies**,
      - E.g. element in economics, social sciences.
    - Models for **emergence of cooperation** (two aspects):
      - **Coordination**: cooperation among **non-antagonistic** agents.
      - **Negotiation**: coordination among **self-interested** agents. → more like a trade-off, your gain is my loss.

- Multiagent Systems
  - Typically **scientific questions** addressed:
    - How can ***cooperation*** emerge in societies of self-interested agents?
    - What actions should agents take to **optimize** their rewards/ utility?
    - How can self-interested agents **learn** from interaction with the environment and other agents to further their goals?
    - How can autonomous agents **coordinate** their activities so as to cooperatively achieve goals?

- MAS as Distributed AI (DAI) → Models that are based on social process
  - **AI: Cognitive** processes in **individuals**.
    - Inspiration: neuro-science, behaviourism, …
  - **DAI: Social** processes in **groups**.
    - Inspiration: social sciences, economics…
  - **Basic question in DAI:**
    - How and when should which agents interact (compete or collaborate) in order to **achieve** their design **objectives**?
  - **2 Approaches:**

        - **Bottom-up:** given specific **capabilities of individual** agents, what **collective behaviour will emerge**?
        - **Top-down:** Search for specific **group-level rules** (e.g. conventions, norms, etc.) that successfully constrain or **guide behaviours at individual level**.

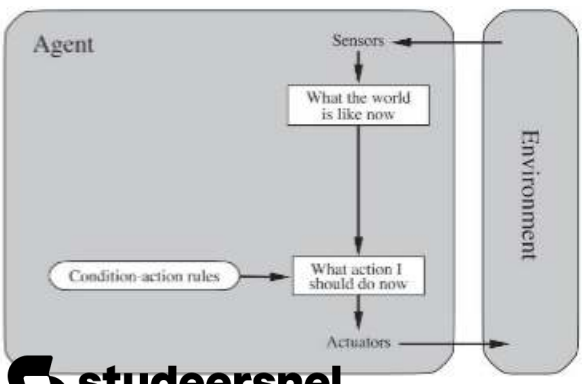
- Multiagent Systems is Interdisciplinary
    - The field of Multi-Agents Systems is influenced and inspired by *many other fields*:
        - Economics.
        - Game theory.
        - Philosophy and Logic.
        - Mathematics (e.g. optimal control).
        - Ecology.
        - Social Sciences.
    - This can be both a ***strength*** or a ***weakness***.
    - This has analogies with ***Artificial Intelligence*** itself.

Agent Types

- Intelligent Agents
    - An ***intelligent agent*** is a computer system capable of ***flexible*** autonomous action in some environment.
    - **Autonomous:** not pre-determined by designer.
    - By ***flexible***, we mean:
        1. Reactive: able to receive information from environment and respond.
        2. Pro-active: able to reason and/or learn and work towards goals.
        3. Social: able to communicate, coordinate, negotiate and cooperate.

- Simple Typology for Intelligent Agents
    - Intelligence in agents covers a **spectrum:**
        - **Reflex** agents;
            - Simple reflex agents.
            - Model-based reflex agents.
        - **Goal** based agents;
        - **Utility** based agents;
        - **Learning** based agents.

- **Type 1: Simple Reflex Agent** (no state, do not remember what has been done before, just react to the signals)
    - Reacts to environment
        - Percept → Action based on simple **if-then rules** (condition-action).
    - Properties:
        - No state: ignore history.
        - Pre-computed rules.
        - NO Partial observability



- **Type 2: Model-Based Reflex Agent**
  - Reflex agent **with state**.
  - Agent uses memory to store an **internal representation** of its world.
  - Internal **model based percept history**.
  - This internal model allows him to **handle partially observable environment**.

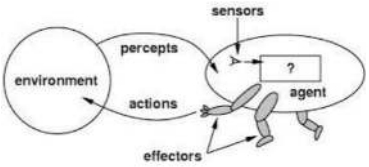
- **Type 3: Goal-Based Agent** (are more adaptive and more flexible, thinking about how to achieve their goals)
  - Goal = desired outcome.
  - Goal-based (planning) agents act by **reasoning about which actions** to achieve the goal.
  - Less efficient, but more **adaptive** and **flexible**.
  - **Search and planning:** AI subfield concerned with **finding sequences of actions** to reach goals.

- **Type 4: Utility-Based Agents** (mapping from states to number, the higher the utility the more preferred the state. Thus the number shows the happiness of the agents). Also adaptive and flexible.
  - Utility-based agents:
    - Distinguishes between **goal and non-goal states**.
  - Utility-based agents use **utility function**.
  - Utility function:
    - U(state) quantifies "happiness" (as real number).
    - Preferred world state has **higher utility** for agent.
  - Allows **rational decisions** in more situations.
    - Evaluation of the **trade-offs among conflicting** goals.
    - Evaluation of **competing** goals.

- **Type 5: Learning Agent**
  - Four essential components:
    - **Actor:** responsible for selecting action in environment.
    - **Critic:** quantifies how well the agent is doing with relation to the performance standard (e.g. utility).
    - **Learner:** responsible for making improvement by learning from interactions.
    - **Explorer:** responsible for suggesting new actions that will lead to novel and informative experiences.

Environments

- Agents act in/on environments:



- Environments – **Accessible vs. inaccessible**
  - An ***accessible*** environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state.
  - Most moderately complex environments (including, for example, the everyday physical world and the Internet) are ***inaccessible***.
  - The more accessible an environment is, the simpler it is to build agents to operate in it.

- Environments – **Deterministic vs. non-deterministic**
  - A ***deterministic*** environment is one in which any action has a **single guaranteed effect** – there **is no uncertainty** about the state that will result from performing an action.
  - The physical world can to all intents and purposes be regarded as **non-deterministic**.
  - Non-deterministic environments present greater problems for the agent designer.

- Environments – **Static vs. Dynamic**
  - A ***static*** environment is one that can be assumed to remain unchanged except by the performance of actions by the agent.
  - A ***dynamic*** environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control.
  - Other processes can interfere with the agent's actions (as in concurrent systems theory).

Topics discussed in this Course

- Main topics in this course: ("the above is not necessarily asked in the exam")
  - **Elementary Game theory**:
    - Multiple (stateless) competing agents,
    - Rational choice among multiple actions (max reward).
  - **Exploration versus Exploitation**
  - **Reinforcement Learning** (single agent RL)
    - Single agent learning to optimize reward from sequential interactions with environment;
  - **Multi-agent Reinforcement Learning (MARL)**
    - Intro and some pointers.

- Game theory: Strategic Thinking for Rational Agents
  - For example: the prisoners' dilemma.

- Exploration versus exploitation
  - You're hungry and want to dine at a restaurant. What should you do?
    - Go to your favourite place which you know you like (**exploiting** your current knowledge)?

        - Try a new place, which might be even better, or maybe… much worse (**exploration**)?

- Reinforcement learning
    - Unsupervised learning (e.g. clustering).
    - Supervised learning.
    - Reinforcement learning: how to learn from interaction?
        - Cooking a complex dish: no feedback on every step or ingredient, but feedback on final result.

Summary

- Five trends in computing
    - Ubiquity, interconnection, intelligence, delegation, human-orientation.
- An **agent** is a computer system that is capable of *autonomous* action in some *environment*, in order to achieve its delegated *objectives*.
- Agent **properties**: reactive, proactive, social.
- A **multi-agent system** is a system that consist of a number of agents, which *interact with one another and the environment*.
- Properties of Agents:
    - Live in some ***environment***.
    - ***Observe*** this environment.
    - Maintain ***knowledge*** about the environment.
    - Make ***decisions*** about what to do.
    - ***Act*** in the environment.
    - ***Communicate*** with other systems/agents:
        - ***Coordinate*** with other agents (cooperative setting).
        - ***Negotiate*** with other agents (non-cooperative setting).

Lecture 1, Part 2: Introduction to Game Theory: Basic (from this it is exam material)

- Overview
    - Game Theory: Introduction and Examples.
    - Examples of interesting games.
    - Formalising Strategic Games: Basic Concepts.
    - Analysing Games: Weak Optimality.
        - Pareto optimality.
        - Best response.
        - Eliminating dominated strategies.
        - Regret minimisation.
    - Equilibrium Concepts: Minimax and Maximin Strategies.
    - Equilibrium Concepts: Nash equilibrium.
        - ➔ Equilibrium: getting a solution, although it is not the optimal solution for each individual.

Game Theory: Introduction and Examples

- Games people play…
  - Games as entertainment or challenge
    - Board games: chess, backgammon, GO etc.
    - Card games.
    - Rock-paper-scissors, etc.
    - …
  - Games as models
    - War games, simulations, etc.
    - More generally: many **interactions in society or nature** share the same ingredients!
    - *All models are wrong, but some are useful!* (G. Box)

- Game Theory: Science of Strategic Thinking
  - Originally, tool in **economics**.
  - Game theory provides a level of abstraction appropriate to study a wide range of **socio-economic, political** and even **biological** phenomena.

- Game Theory: Different points of view
  - Philosopher John Elster (1982): *if one accepts that interaction is the essence of social life, then game theory provides solid micro foundations for the study of social structure and social change*.
  - Hargreaves-Heap and Varoufakis: "Game Theory. A critical Introduction" (1995): *Does game theory simply repeat what everyone already knows in a language that no one understands*?

- Game Theory: Science of Strategic Thinking
  - GT is the **mathematical study** of interaction among independent, self-interested agents.
  - **Self-interest:**
    - Each agent has its own interests and preferences (aims, goals);
    - Agents tend to have (partially) conflicting interests;
    - These interests are reflected in (numerical) utilities that are consistent with the preferences; if option A is preferred to B, then $u(A) > u(B)$.
    - Agents act to maximise their utility.
  - **Coalitional/Cooperative vs. non-coalitional/non-cooperative GT**.

- Ingredients of interesting games
  - Players: you against one or more opponent(s)
    - Opponent: other agents, other version of yourself, nature, lady luck, etc.
  - Rules determine which actions can be taken, and what the corresponding pay-offs or utilities are:

        - Actions and pay-offs: exogenous variables.
    - Maximize your pay-off: Everyone wants to win!
    - **Competition and collaboration**: individuals or teams (non-cooperative and cooperative GT).

- Cooperative versus Non-Cooperative Games (synonyms to coalition and non-coalition)
    - **Non-Cooperative**
        - Selfish individuals, only consider their own interest;
        - Do **not coordinate** their action in groups.
            - Coordination might happen as "accident" of selfish behaviour (emergent property).
        - Agreements need to be self-enforcing (no contracts!)
    - **Cooperative:** binding commitments ("contracts") allow groups of players to coordinate their actions.
        - **Non-transferable utility:** pay-off of each individual increases!: E.g. Stable marriage problem. // each utility must be improved. E.g. from hospital preference for certain student and vice versa, such that everybody is happy.
        - **Transferable utility:** need to find a fair way to divide the additional value (e.g. money) generated by collaboration: E.g. Shapley value.
            - ➔ Difference is that with cooperative you have binding contracts.
            - ➔ Non-transferable utility → everybody needs to be happy.

- Simultaneous vs. Sequential Games
    - **Simultaneous games:** players make their moves simultaneously, i.e. without knowing what the other players will do!
        - Rock-paper-scissors.
        - Sealed bid auctions.
    - **Sequential games:** sequence of successive moves by players who can see each other's moves:
        - Chess.
        - Card games.
        - Open cry auctions.

- Encoding utilities of actions: Matrix form
    - **Simultaneous game: two players, finite number of actions**

Normal form or payoff matrix of a 2-player, 2-strategy game

        - Pay-off values for blue and red players.

- Encoding utilities of actions: Extensive Form
    - **Sequential game: Decision tree**

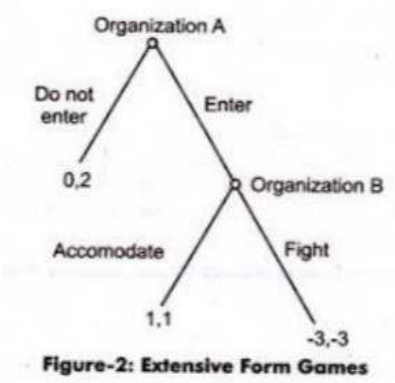


Figure-2: Extensive Form Games

- What is organization B going to do? → Accommodate or Fight?
- Thus organization A is going to enter the market and organization B is going to choose to accommodate.

- Encoding utilities of actions: General case
  - Utility is a **function of the joint action** of players:
  - Ultimatum game (one shot):
    - Player A can choose any fraction 0 x 1 for himself, and offer the rest (1 – *x*) to player B;
    - If player B accepts this offer, then that is the outcome. If he rejects it, then both get zero.

$$u_A(x) = \begin{cases} x \\ 0 \end{cases} \quad \text{and} \quad u_B(x) = \begin{cases} 1-x & \text{if B accepts} \\ 0 & \text{if B rejects} \end{cases}$$

  - **Important applicability issue: Rationality vs. Behaviour (emotion)**

Examples of interesting games

- Economics: Cournot's Duoply Model
  - Two companies make an **interchangeable product** (e.g. bottled water).
  - Without knowing the other company's strategy (i.e. **simultaneously**), both need to determine the quantity they will produce, say q_1 and q_2 respectively.
  - The unit price p of the product in the market depends on the total produced quantity $q_1 + q_2$; specifically
    $p(q_1, q_2) = \alpha - \beta(q_1 + q_2) \quad (\alpha, \beta > 0).$
  - Firm i = 1, 2 can produce the item at a unit-cost c_i.
  - What quantity q_i should each company produce in order to maximise its profit?

- Children dividing pie
  - Two kids get a pie and need to divide it among themselves.
  - Zero- (or constant) sum game: **my gain is your loss**!
  - Selfish agents, but rational (cutter knows that chooser will pick largest piece).
  - **Equilibrium Solution:** minimax and maximin.
    - Cutter: mitigating the worst result that chooser can enforce.
    - Chooser: maximizing his pay-off.
    - Satisfactory equilibrium, notwithstanding selfish behaviour.
  - **Mechanism design:** ("inverse GT")

- How to set up rules of game so that selfish individual behaviour will lead to social welfare?

- Prisoners' dilemma (RAND, 1950s)

Two suspects in a crime are put into separate cells. The police officer tells them: *Currently you're charged with trespassing which carries a jail sentence of one month. I know you were planning a robbery though, but cannot prove it – I need your testimony. If you confess and cooperate, I will drop the charges against you, but your partner will be charged to the fullest extent of the law: 12 months in jail. I'm offering the same deal to him. If you both confess, your individual testimony is less valuable, and you will get 8 months each.*

	quiet	confess
quiet	$-1, -1$	$-12, 0$
confess	$0, -12$	$-8, -8$

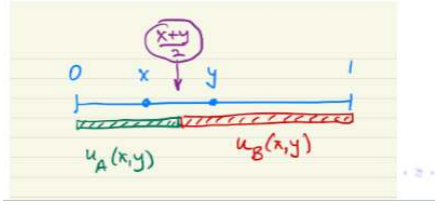
  - What should the suspects do?

- Penalty kicks
  - Penalty-kick game: soccer penalties have been studied extensively.

	Defend left	Defend right
Left	0.58, -0.58	0.95, -0.95
Right	0.93, -0.93	0.70, -0.70

- Ice cream time! (Hotelling's game)
  - Two players but continuous (infinite) action space:
    - Each player can choose any position between 0 and 1.
    - Player A chooses *x*, player B chooses *y* (0 *x* *y*);
  - Utility:

$$u_A(x,y) = x + \frac{y-x}{2} = \frac{x+y}{2}$$
$$u_B(x,y) = 1 - y + \frac{y-x}{2} = 1 - \frac{x+y}{2}$$

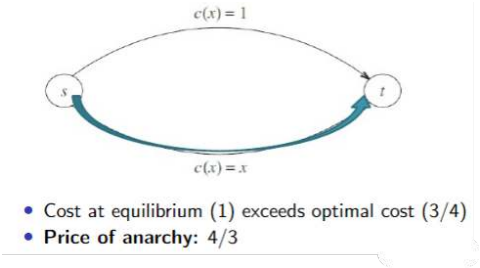


    ➔ Question where should the ice cream holders/venders be placed? answer: in the middle of *x* and *y*.

- Partnership game (handy for the homework assignment → relevant what you're there are going to do)
  - Two students work on project, and divide profits evenly (50-50).
  - Each student must decide how much effort (e.g. hours) he/she is contributing to the project. Hence, student *i* choose action s_i=amount of effort. (assume 0 s_1, s_2 4);
  - Project generates reward: $4(s_1 + s_2 + bs_1s_2)$ (with $0 < b < 1$).
  - The cost of the work to student *i* equals .
  - Pay-off for individual student:

$$u_i(s_1, s_2) = \frac{4}{2}(s_1 + s_2 + bs_1s_2) - s_i^2$$

- Selfish routing in congestion games
  - **Congestion games:**
    - Context: routing in network.
    - Single shot, *n*-player game.
    - Player chooses some resource (rout) from **set of resources**;
    - **Congestion:** Cost of resource depends on number of agents selecting this resource;
    - Pigou's example:
      - Selfish players.
      - Each player wants to minimize cost.
      - Price if anarchy might be ¾ below?

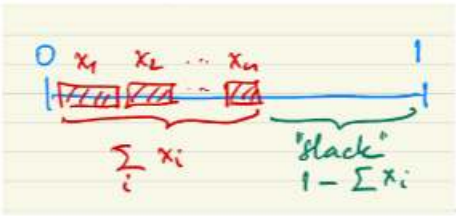


- Cost at equilibrium (1) exceeds optimal cost (3/4)
- **Price of anarchy:** 4/3

- Tragedy of the Commons
  - *N* players sharing some **common resource** (of total size 1).
    - E.g., village green, bandwidth in network, etc.
  - Each player *i* would like to have a big share (0 x_i 1)!
  - However, individual utility also depends on what others do:

$$u_i(x_i, x_{-i}) = \begin{cases} x_i\,(1 - \sum_{j=1}^{n} x_j) & \text{if } \sum_j x_j < 1 \\ 0 & \text{otherwise} \end{cases}$$

  - Is there an optimal strategy?



- Ultimatum game with impatient players
  - Kids get a box of ice-cream to share among themselves;
  - They can have all of it as long as they agree on the division;
  - If they fail to agree, their parents take away all the ice cream (conflict deal);
  - It's a hot day and the ice-cream is melting! The longer they argue, the less ice-cream there's left!

- Traveller's dilemma (Kaushik Basu, 1994)
  - Airline severely damages identical antiques purchased by two different travellers.
  - Management is willing to compensate them for the loss of the antiques, but since they have no idea about their value, they tell the two travellers to separately write

down their estimate of the value as any number between $2 and $100 without conferring with one another.
    - If both travellers write down the same number, they will be reimbursed that amount.
    - If they write different numbers, management will pay both of them the lower figure, the person with the lower number will get a $2 bonus for honesty, while the one who wrote the higher number will get a $2 penalty.

- Overview of Topics in Game Theory Course
    - Non-cooperative games
        - Matrix games (2 players, finite action sets).
        - Sequential games, e.g. bargaining.
    - Cooperative (coalitional) games;
        - Shapley value;
    - Mechanism design (inverse game theory)
        - Vickrey auction.

Formalising Strategic Games: Basic Concepts

- Game theory and strategic agents
    - **Game theory** studies **multiagent decision problems**, that is, problems in which **independent decision-makers interact**.
    - What each agent does has an **effect on the other agents** in the group (through **utility**):
    - **Assumptions:**
        - Agents have **preferences** encoded in **utility function (pay-off)**.
        - **Self-interest:** agents strive to maximize their own pay-off;
        - **Rational behaviour:** agents reason about the actions of other agents and decide rationally.

- Games: Normal-form vs. extensive-form
    - Models for games for which actions are:
        - **Sequential:** player moves after observing action of other players;
        - **Repeated:** special case of sequential game with (possibly) many iterations.

- Normal-form Games (Matrix Games)
    - **Players:**
        - Make **simultaneous moves** and receive **immediate pay-offs**;
        - **Pay-offs** are specified for the combinations of actions played.
    - **Pay-off matrix:**
        - Specifies for given action combination $a = (a_1, a_2, \ldots, a_n)$ the corresponding utility (pay-off) $u_i(a)$ for player $i = 1 \ldots n$.

- A graphical representation: matrix games
    - In the special case of *two* agents, a strategic game can be graphically represented by a **pay-off matrix**, for example:

	left	centre	right
up	1, 0	1, 2	0, 1
down	0, 3	0, 1	2, 0

- Rows correspond to actions of agent 1 and columns to actions of agent 2. Here: A_1 = {up, down), A_2 = {left, centre, right}.
- Each entry contains the pay-offs (u_1, u_2) of the two agents for each possible joint action. For example, a = (down, centre) gives (u_1, u_2) = (0, 1).

- Formal definition of normal-form game
  - A *n*-person **normal-form game** is a tuple (*N*, *A*, *u*):
    - **N** is a set of *n* **players (agents)**.
    - **A is Actions or Strategies:** $A = A_1 \times A_2 \times ... \times A_n$ where each A_i is the set of actions available to agent *I*, i.e. set of allowable moves player *i* can make. An A-element $a = (a_1, a_2, ..., a_n)$ is called an **action profile**.
    - **u is Pay-off or utility function:** where $u = (u_1, u_2, ..., u_n)$ and each u_i : is the corresponding utility function for player *i*. Notice, pay-off $u_i(a)$ for each agent depends on the *joint action* of all agents.

- Utility functions (the idea that you can rank preferences) → numerical representation of preferences
  - Von Neumann and Morgenstern, 1944: if there exists a preference relation $\succcurlyeq$ on the outcomes of a game that satisfies a number of "natural conditions"(completeness, transitivity, substitutability, decomposability, monotonicity and continuity), then there exists a function $u : \mathcal{O} \longrightarrow \mathbb{R}$ such that:
    - $u(o_1) \geq u(o_2)$ iff $o_1 \succcurlyeq o_2$
    - $u(\{(o_1 : p_1), (o_2 : p_2), \ldots, (o_n : p_n)\}) = \sum_{i=1}^{n} p_i u(o_i)$

      ➔ You sum up to utility of a certain outcome times the probability that the outcome can occur.

- Examples of competitive and cooperative (matrix) games
  - A strategic game can model a variety of situations where agents interact. These are two well-known cases:

Matching Pennies

	head	tail
head	1, −1	−1, 1
tail	−1, 1	1, −1

In a **strictly competitive** or **zero-sum** game, $\sum_i u_i(a) = 0$ for all *a* (anti-coordination game).

Going to the Movies

	action	comedy
action	1, 1	0, 0
comedy	0, 0	1, 1

In a **strictly cooperative** game, a type of **coordination** game, $u_i(a) = u_j(a)$ for all i, j, a.

➔ So with competitive games, there is a gain or a loose and cooperative you do a pay-off for e.g. to be together with a friend.

- More examples:

Chicken

	swerve	straight
swerve	0, 0	−1, 1
straight	1, −1	−5, −5

Stag Hunt

	stag	hare
stag	2, 2	0, 1
hare	1, 0	1, 1

Battles of the Sexes 1

	action	comedy
action	3, 2	0, 0
comedy	0, 0	2, 3

Battle of the Sexes 2

	action	comedy
action	3, 2	2, 1
comedy	0, 0	2, 3

- Continuous action space (action space can be discrete or continuous)
  - Hotelling's Game (ice-cream time):
    - Two players
    - Continuous (infinite) action space:
      - Each player can choose any position between 0 and 1.
      - Assume first player choses x while second player chooses y where for simplicity: $0 \leq x < y \leq 1$;
    - Utility:

$$u_1(x, y) = x + \frac{y - x}{2} = \frac{x + y}{2}$$
$$u_2(x, y) = 1 - y + \frac{y - x}{2} = 1 - \frac{x + y}{2}$$

- Strategies
  - A player's **strategy** is the **algorithm** that determines the action the player will take at **any stage of the game**.
  - **Pure strategy:** select single action and play it.
  - **Mixed strategy:** select single action according to probability distribution and play it. Rationale? Think of *matching pennies*:

	Heads	Tails
Heads	1, -1	-1 , 1
Tails	-1, 1	1, -1

    - mixed strategy: using **randomness** not to be outsmarted by opponent.
  - **Strategy profile:** s = (s_1, s_2, …, s_n), i.e. one specified strategy for each agent.

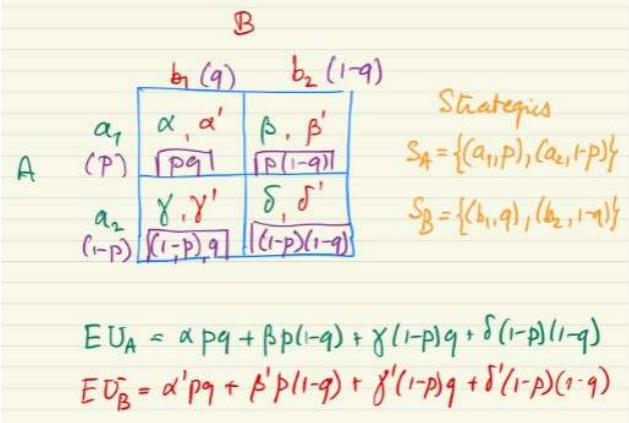
- Expected Utility for Mixed Strategies (involves the probability aspect)
  - **Pure strategy:** (Expected) utility u_i for agent *i* selecting action a_i equals $u_i\,(a_i, a_{-i})$.
  - Mixed strategy: Agent *i* plays strategy s_i which is a probability distribution over *k* possible actions:

$$s_i = \{(a_{i1}, p_{i1}), (a_{i2}, p_{i2}), \ldots, (a_{ik}, p_{ik})\} \qquad (\text{where } p_k = P(a_k))$$

  - **Expected utility** for mixed strategies:
    - Agent *i* playing mixed strategy $s_i = \{(a_{i1}, p_{i1}) \ldots (a_{in}, p_{in})\}$.
    - Agent *j* playing mixed strategy $s_j = \{(a_{j1}, p_{j1}) \ldots (a_{jm}, p_{jm})\}$.

$$EU_i(s_i, s_j) = \sum_{k=1}^{n} \sum_{\ell=1}^{m} u_i(a_{ik}, a_{j\ell}) p_{ik} p_{j\ell}$$

  - Example explained:

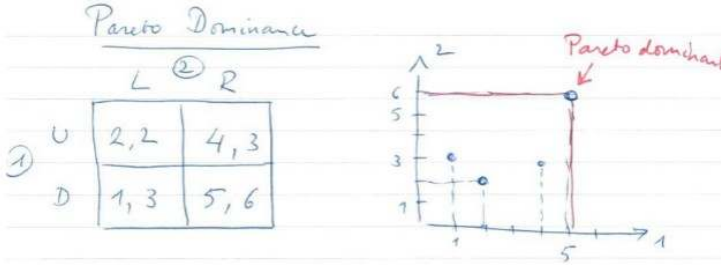


➔ You have two players (A & B). Both can do two actions (action 1 or action 2)

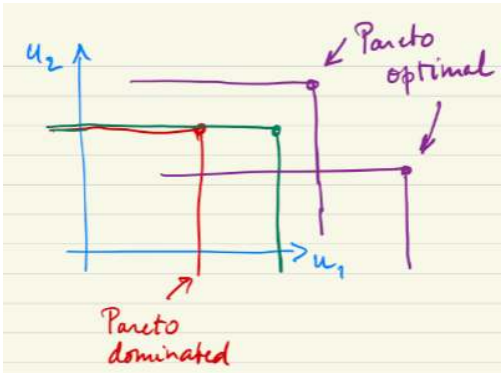
Analysing games: Solution concepts for games

- Consider point of view of a **single (self-interested) agent:**
  - Given all game information: **what strategy** should he/she adopt?
  - Complicated: depends on **actions of other agents!** *Make the most of it*…
  - **From (weak) optimality…**
    - Pareto Optimality.
    - Best Response (BR) given the actions of the other agents;
    - Iterated elimination of strictly dominated strategies (IESDS)
    - Regret minimisation.
  - … **to Equilibrium,** i.e. no incentive to deviate:
    - Maximin and minimax strategies.
    - Nash equilibrium
      - ➔ Hope for an equilibrium, as here all the players are in a sense/way happy with what is going to happen.

- Pareto optimality
  - Mean that you look at the utility of certain actions
  - Example:



- Pareto dominance and Partial Ordering



- Pareto optimality (not dominated by any other solution/action)
  - **Pareto optimality** is a **solution property** (not solution concept itself).
  - A joint action/strategy profile *a* is **Pareto dominated** by another joint action *a'* if $u_i(a') \geq u_i(a)$ for all agents *i* and $u_j(a') > u_j(a)$.
  - A joint action/strategy profile *a* is **Pareto optimal** if there is no other joint action *a'* that Pareto dominates it.
  - Pareto dominance defines a **partial ordering** over strategy profiles. (but in the front it is not an ordering, as it is a trade-off. E.g. do you want to earn more, or more vacation days? But its more an "/" ordering)

- Pareto Front

  - ➔ So overall you should choose a solution that lies on the Pareto Front.
  - ➔ E.g. K you do not want and A-H you want to choose.

- Best response (1) (best response = think of the idea if you know what the other e.g. does, you can play the best response) = if you would know what the opponent would do (s_{-i}) you can determine, with the use of the utility matrix what your best response is.
  - **Best response** from agent *i*'s point of view:
  - Let's assume that we know the strategies of all the other agents, i.e. s_{-i} is known:
  - Agent *I*'s **best response** to strategy profile s_{-i} is (a possibly mixed) strategy $s_i^* \in S_i$ such that:

$$\forall s_i \in S_i : \quad u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i}).$$

  - Of course, … in a realistic setting we don't know the strategies of other agents! **Not** solution concept, but **essential ingredient** for Nash eq.!

- Best response (2) (example → pure strategy)

Chicken

	swerve	straight
swerve	0, 0	−1, 1
straight	1, −1	−5, −5

Stag Hunt

	stag	hare
stag	2, 2	0, 1
hare	1, 0	1, 1

  - Chicken:

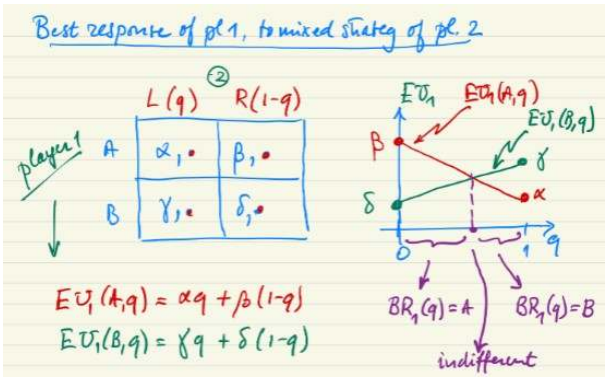
    $BR_1(a_2 = \text{swerve}) = \text{straight}$ $BR_1(a_2 = \text{straight}) = \text{swerve}$

  - Stag hunt:

    $BR_1(a_2 = \text{stag}) = \text{stag}$ $BR_1(a_2 = \text{hare}) = \text{hare}$

    ➔ So for the chicken game: if you know that the second player is gong to swerve, the best response is to go straight and if you know that player 2 is going straight the best response is to swerve.

- Best response to mixed strategy (2B)



    ➔ Best response of player 1, when player 2 plays with a mixed strategy.

    ➔ Expected utility for player 1, when it plays A, when you know player 2 plays L(q) (q is probability) = red formula.

    ➔ For the graph if the q is between 0 and 0.5, the best response is the red one. Otherwise if q is above 0.5, the best response is the green line. (the functions red and green correspond to the lines red and green in the graph).

    ➔ So if q is small the best thing you can do is A. Otherwise play B.

    ➔ Horizontal = q, the probability. Vertical = the expected utility.

- Best response (3)
  - Best response **not necessarily unique!**
  - When the best response includes two (or more) actions, then the agent must be **indifferent** among them!
  - In fact: **any mixture** of these actions would also be a best response (mixed) strategy.
  - Indeed,
    - If a_{i1} and a_{i2} are best both best response actions to s_{-i}, then $u_i(a_{i1}, s_{-i}) = u_i(a_{i2}, s_{-i}) =: u_i^*$.
    - Then, for any mixed strategy $s_i = \{(a_{i1}, p_1), (a_{i2}, p_2)\}$:

$$u_i(s_i, s_{-i}) = p_1 u_i(a_{i1}, s_{-i}) + p_2 u_i(a_{i2}, s_{-i}) = (p_1 + p_2)u_i^* \equiv u_i^*,$$

since $p_1 + p_2 = 1$.

- Best response (4): continuous state space
  - **Partnership game (two players)**
    - **Actions:** choice of individual contributions to joint project.

      $0 \leq x, y \leq 4$
    - **Utilities:** *utility = profit – cost*

      $$\begin{cases} u_1(x,y) &= 2(x+y+bxy) - x^2 \qquad (0 \leq b < 1) \\ u_2(x,y) &= 2(x+y+bxy) - y^2 \end{cases}$$

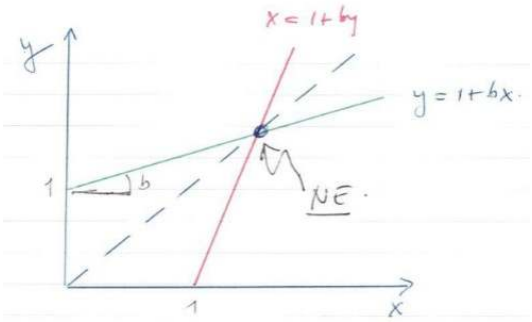
      ➔ x is the amount you put in and y is the amount that your partner puts in.

      ➔ x^2 is the costs (So -x gives also an increase → you don't want to put in too much or too less)
    - Utility is quadratic (high input is very costly):
    - **Best response:** For a given input *x* of player 1, what input *y* of player 2 maximizes the latter's utility ($u_2(x, y)$)?
    - **Finding the maximum utility $u_2(x, y)$ for given** *x*:

      $$\frac{\partial u_2}{\partial y} = 2(1+bx) - 2y = 0$$
    - **Best response solution***:*

      $y^* \equiv BR_2(x) = 1 + bx$
    - **Similarly:**

      $x^* \equiv BR_1(y) = 1 + by$

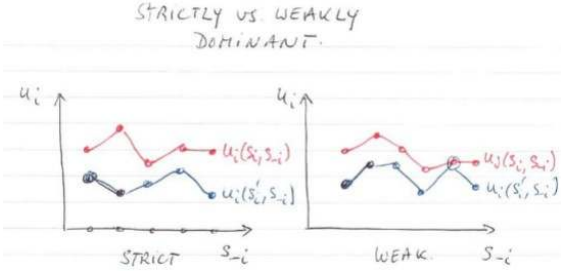
      ➔ The slope is b, So the larger b the higher the line goes.

- Domination for strategies
  - Let s_i and s_i' be two strategies for player *i* , and S_{-i} set of all strategy profiles for the other players:
    - s_i **strictly dominates** s_i' if

      $u_i(s_i, s_{-i}) > u_i(s_i', s_{-i}) \qquad \forall s_{-i} \in S_{-i}$
    - s_i **weakly dominates** s_i' if

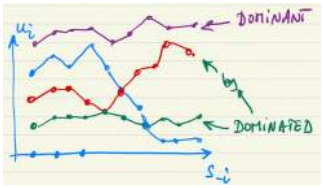
      1. $u_i(s_i, s_{-i}) \geq u_i(s_i', s_{-i}) \qquad \forall s_{-i} \in S_{-i}$, and
      2. $u_i(s_i, s_j) > u_i(s_i', s_j)$ for at least one $s_j \in S_{-i}$

- Strict vs. weak dominance



➔ Weak dominance: at least one value were it is strictly better. So 1 point must be better than the other: 

- Dominant and dominated Strategies



Red is better to choose than green (therefore red is a dominant strategy when comparing to green. But comparing red with purple gives purple as a dominant strategy)

  - (Strictly/Weakly) **Dominant Strategy:** (strictly/weakly) dominates every other strategy of the agent;
  - Strictly/Weakly) **Dominated Strategy:** is (strictly/weakly) dominated by at least one of this agent;
  - A strictly **dominated strategy will never be the best response** to anything!
  - For a **dominating strategy, we don't have to worry** what the opponents are going to do!
  - Dominance plays important role in mechanism design.

- Iterated <u>elimination</u> of strictly dominates strategies (IESDS)
  - **IESDS** (a.k.a. What <u>NOT</u> to do?) is based on the following assumptions:
    - It is **common knowledge** that all agents are rational.
    - Rational agents **never** play strictly dominated actions.
    - Hence, strictly **dominated actions** can be **eliminated**.

	left	centre	right
up	13, 3	1, 4	7, 3
middle	4, 1	3, 3	6, 2
down	−1, 9	2, 8	8, −1

What would IESDS predict in this game?

  - <u>*Centre* **strictly dominates** *right*</u>. Row player knows that column player will never play the dominated action *right*. Hence he can eliminate that action and only needs to consider the simpler game:

	left	centre
up	13, 3	1, 4
middle	4, 1	3, 3
down	-1, 9	2, 8

  - For the row player, action *middle* strictly dominates *down*; hence eliminate! We are left with the simpler game where *centre* dominates *left*:

	left	centre
up	13, 3	1, 4
middle	4, 1	**3, 3**

- Application Dominated Strategies: Prisoner's Dilemma

	Quiet	Confess
Quiet	−1, −1	−12, 0
Confess	0, −12	−8, −8

  - **Prisoner's Dilemma**
    - Quiet is a strictly dominated strategy for both players, hence can be **eliminated**.
    - Players will therefore both play **confess**, yielding pay-off (-8, -8).
    - Notice that this action profile is **Pareto dominated**!
      - ➔ **However, notice that ending up with -8, -8 is far worse than ending up with -1, -1.**
- Cournot duopoly (discrete version)
  - Unit production cost: c = 1;
  - Q_A(Q_B) = quantity produced by A (B)
  - P = Market price (per unit) : $P = 12\text{-}2(Q_A + Q_B)$
  - **Pay-off:**

$$u_A(A2, B3) = Q_A(P(Q_A, Q_B) - c) = 2(P(2, 3) - c) = 2(12 - 2 \cdot 5 - 1) = 2$$

	$B0$	$B1$	$B2$	$B3$	$B4$	$B5$
$A0$	0, 0	0, 9	0, 14	0, 15	0, 12	0, 5
$A1$	9, 0	7, 7	5, 10	3, 9	1, 4	−1, −5
$A2$	14, 0	10, 5	6, 6	2, 3	−2, −4	−2, −5
$A3$	15, 0	9, 3	3, 2	−3, −3	−3, −4	−3, −5
$A4$	12, 0	4, 1	−4, −2	−4, −3	−4, −4	−4, −5
$A5$	5, 0	−5, −1	−5, −2	−5, −3	−5, −4	−5, −5

1. A3 (B3) strictly dominates A5 (B5), eliminate A5/B5
2. A3 (B3) strictly dominates A4 (B4), eliminate A4/B4
3. A1 (B1) strictly dominates A0 (B0), eliminate A0/B0

	$B1$	$B2$	$B3$
$A1$	7, 7	5, 10	3, 9
$A2$	10, 5	6, 6	2, 3
$A3$	9, 3	3, 2	−3, −3

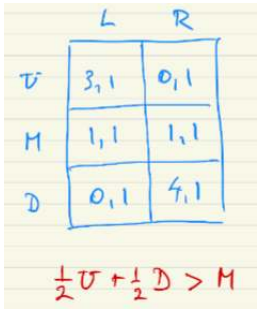
4. A2(B2) strictly dominates A3 (B3), eliminate A3/B3
5. A2(B2) strictly dominates A1 (B1), resulting in strategy profile (A2,B2) with utility (6, 6);
6. Notice: **not Pareto-optimal!** (dominated by (A1,B1), with utilitity (7, 7)

    - ➔ So if a row is dominating another row, you can eliminate it.

- Iterated elimination of strictly dominated actions (3)
  - More challenging example:
    - Rules of the game:
      - Game played in large group (e.g. auditorium).
      - Each player picks number between 1 and 100.
      - Collect all numbers and compute the mean.
      - Winner is player whose number was closest to ½ of mean.
    - What strategy should you use when picking your number?
    - **Bounded rationality** vs. "homo economicus"! Rationality is bounded by limits to our resources (Simon, 1982):

- Cognitive capacity, available information, time, emotion, etc.

- Domination by mixed strategy
  - It is possible that the dominant strategy is mixed!

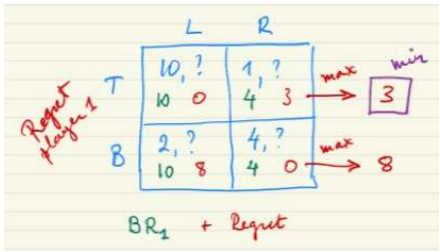


    ➔ So another strategy can still be dominated by mixing two other strategies!
- Best response and IESDS
  - Strictly dominated strategies are **never a best response;**
  - **Church-Rosser property:** Order of elimination does not matter for IESDS (strict dominance)!
  - Eliminating **weakly dominated** strategies might be too drastic!

- Regret minimisation
  - What is **my maximal possible regret** if **I take this action?**
  - **Regret:** diff. btw pay-offs **best response** and **current action**:

$$R_1(s_1, s_2) = u_1(BR_1(s_2)) - u_1(s_1, s_2)$$

    ➔ Regret = best response – current action.
  - Take action to **minimise max regret**



    ➔ If you play T, you max regret will be 3.
  - **Regret** (for agent i) is the **difference** between the **actual and maximal pay-off** for a **given action profile (s_i, s_j)**

$$R_i(s_i, s_j) = u_i(BR_i(s_j)) - u_i(s_i, s_j) = \max_{s_i'} u_i(s_i', s_j) - u_i(s_i, s_j)$$

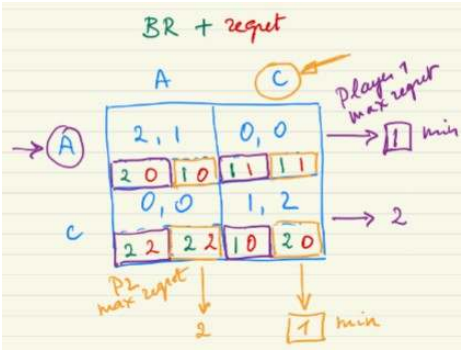
  - For **each action s_i**, there is a **maximum regret** depending on **s_j**:

$$R_i^{max}(s_i) = \max_{s_j} R_i(s_i, s_j)$$

  - **Regret minimisation (minimax regret)**: agent i picks action s_i that **minimises max regret**:

$$s_i^{rm} := \arg\min_{s_i} R_i^{max}(s_i) = \arg\min_{s_i} \max_{s_j} R_i(s_i, s_j).$$

  - **Actual solution concept**: allows an agent to choose a strategy with specific guarantees/ properties;
  - Example of regret minimisation for BoS:

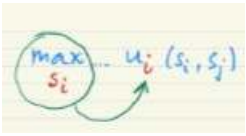


Lecture 2/3: Equilibrium Concepts: Minimax, Maximin and Nash

Equilibrium Concept: Minimax and Maximin Strategies

- **Minimax and maximin for zero-sum games.**
- **Minimax and maximin for general games.**

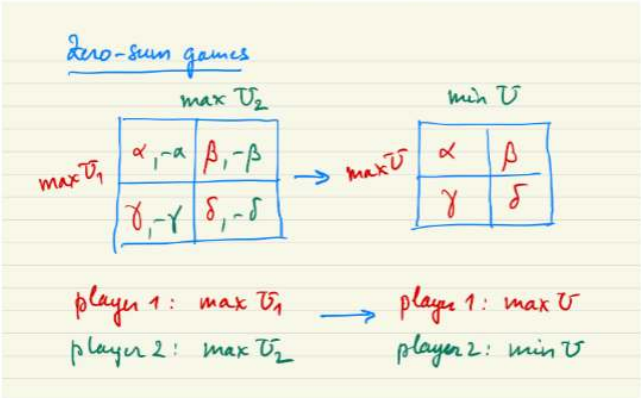
- Maximin and Minimax Strategies
  - **Actual equilibrium concepts:** allow an agent to choose a strategy with specific guarantees/properties;
  - **Equilibrium:** no incentive to (unilaterally change).
  - Focus on **2-player game**;
  - Most natural interpretation for (2-player) **zero-sum games**: maximising own pay-off = minimising opponent's pay-off.
  - **General sum games:** assume opponent is malicious.
    - … or threatening (e.g. in case of repeated game).
  - We consider **player *i*'s point of view**: (i.e. maximise over s_i).



➔ Keep in mind that the first player is trying to maximise.
➔ The maximise player is the first player, so he always try to maximise over the strategies.

- Zero-Sum Games: "my gain is your loss"

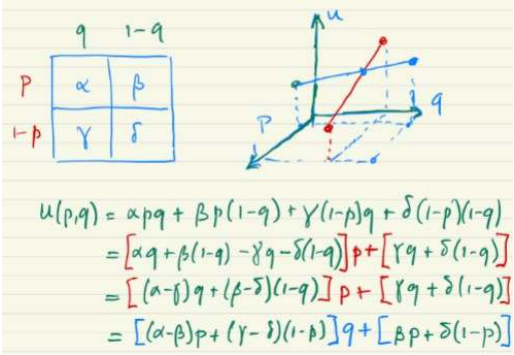
$$u_i(s_i, s_j) = -u_j(s_i, s_j)$$



➔ There is only one utility. One player is trying to maximise its utility and the other one is trying to minimise its utility. The opponent is getting what you lose, vice versa.

➔ So the pay-off matrix shows that its just the opposite for the other player.

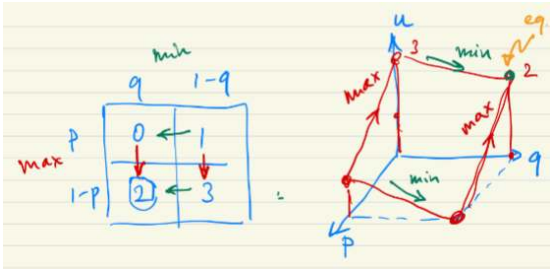
- Expected utility is bi-linear: (you need to realise that the EU I bi-linear)



$$u(p,q) = \alpha pq + \beta p(1-q) + \gamma(1-p)q + \delta(1-p)(1-q)$$
$$= [\alpha q + \beta(1-q) - \gamma q - \delta(1-q)]p + [\gamma q + \delta(1-q)]$$
$$= [(\alpha-\beta) q + (\beta-\delta)(1-q)]p + [\gamma q + \delta(1-q)]$$
$$= [(\alpha-\beta)p + (\gamma-\delta)(1-p)]q + [\beta p + \delta(1-p)]$$

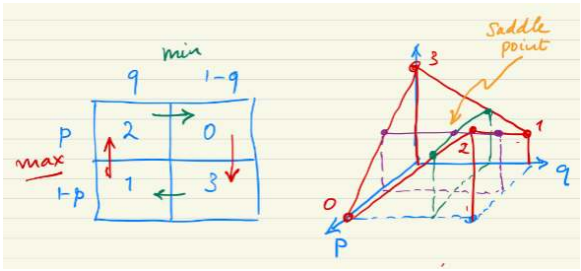
➔ So when you fix p/q you get a linear line.

- Zero-sum game: example

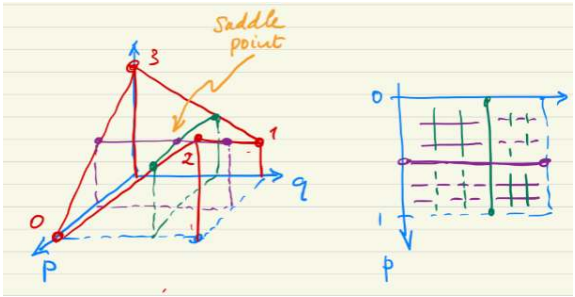


➔ The column player wants to minimize, so it goes from right to left. The row player wants to maximise, so it goes from top to bottom.

- Zero-sum game: example

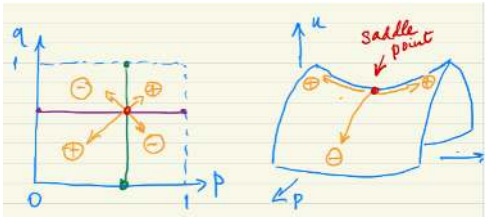


➔ Here we need to look at mixed strategies, there is not equilibrium in terms of pure strategies. (in the example above there is)



➔ You have one green and one purple horizontal line for one value of p (p*) and one value of q (q*).

- Zero-sum game: saddle geometry (The square left is the same as above picture square right)



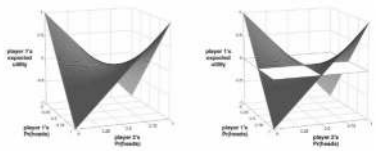
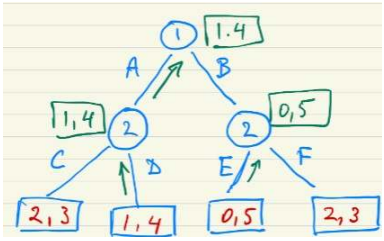
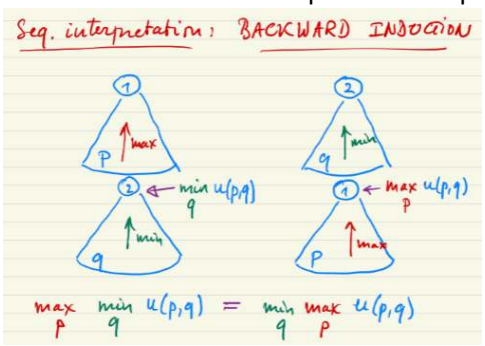


Figure 3.13: The saddle point in Matching Pennies, with and without a plane at $z = 0$.

- Aside: Backward induction in sequential game



- Backward induction in sequential interpretation



  - Equilibrium would be reaches if this condition (maxmin=minmax) is satisfied (Neumann)

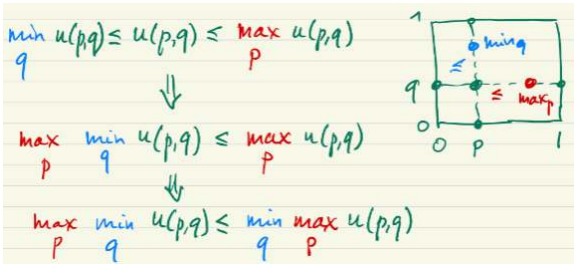
- Intuition for Minimax Thm for 2p-zerosum game
  - Equilibrium in simultaneous game, but …
    1. Suppose player 1 goes first:
       - If he picks p_0 then player 2 will try to minimise $u(p_0, q)$ resulting in $\min_q u(p_0, q)$;
       - Pl 1 knows this and will therefore pick p to maximise result:

$$\max_p \min_q u(p, q)$$

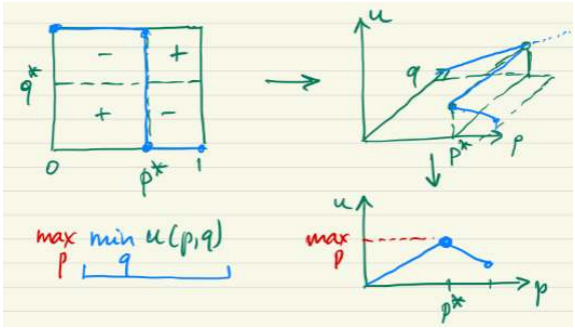
    2. Suppose player 2 goes first:
       - If he picks q_0 then player 1 will try to maximise $u(p,q_0)$ resulting in $\max_p u(p,q_0)$;
       - Pl 2 knows this and will therefore pick q to minimise result:

$$\min_q \max_p u(p, q)$$

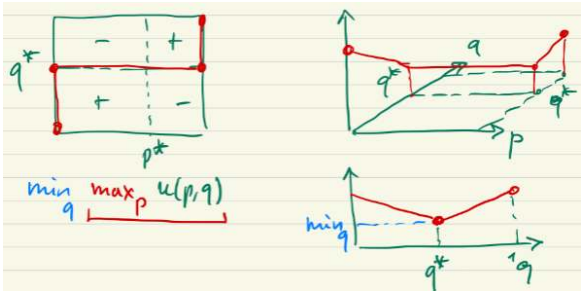
- Basic inequality



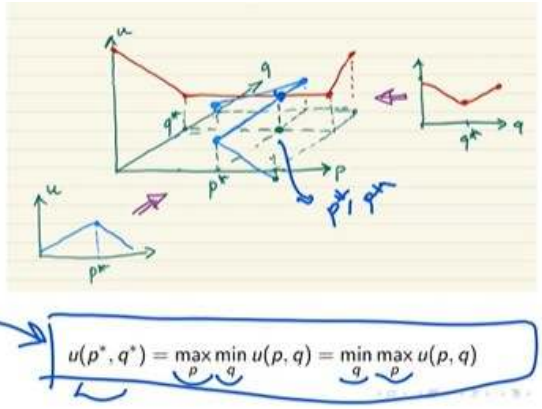
  - For **2p-zs games**, this inequality can be tuned into an **equality**!
- Zero-sum game: example



  - Fix *p*, minimise over *q*, then max over *p*.
  - Represents the max of the min.
- Zero-sum game: example



  - Fix *q*, maximise over *p*, then min over *q*.
  - Represents the min of the max.
- Combining the two above gives:

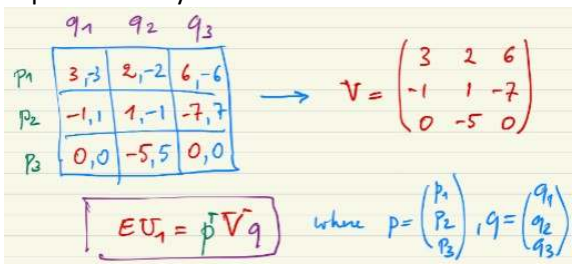


- Minimax and Maximin for Zero-sum Games
  - **Zero-sum game:**
    - "my gain is your loss!"

$$u_i(s_i, s_j) = -u_j(s_i, s_j)$$

- For (finite) **zero-sum games: minimax equals maximin.**
  - But beware! Could be result of mixed strategies!
- **Maximin strategy**
  - Maximize minimal gain for myself.
- **Minimax strategy**
  - Minimize maximal gain for opponent.
  - Minimize maximal loss for myself (zero-sum game).
- **Value of zero-sum game:** Common value of minimax and maximin.

- Matric notation for Minimax Thm (2p-zerosum game)
  - Expected utility $u(p, q) = p^T Vq$ is **bi-linear function** of *p* and *q*.



- Minimax Theorem
  - Let A be the *n* × *m* pay-off matrix for the first player in a **2-player, finite, zero-sum game** (i.e. pay-off for player 2 equals −A). Furthermore, let p = (p_1,p_2,...,p_n) and q = (q_1,q_2,...,q_n) be probability distributions representing mixed strategies for player 1 and 2, respectively. The corresponding (expected) pay-off u := EU_1 for player 1 is then given by:

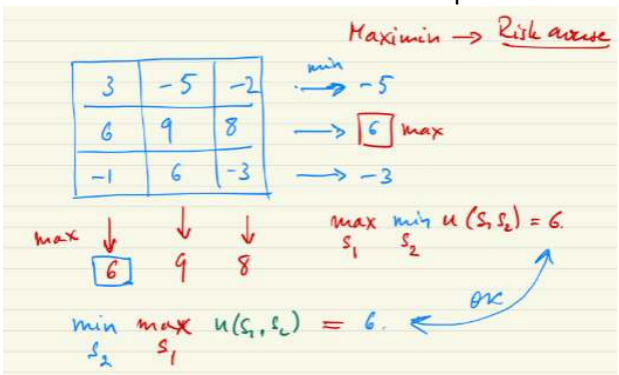
    $$u(p, q) = p^T Aq$$

  - Minimax Theorem, Von Neumann, 1928
    - Under the assumptions above, there exists probability vectors p* and q* such that:

      $$u(p^*, q^*) = \max_p \min_q u(p, q) = \min_q \max_p u(p, q)$$

      ➔ (Nash equilibrium is just an extension of this).

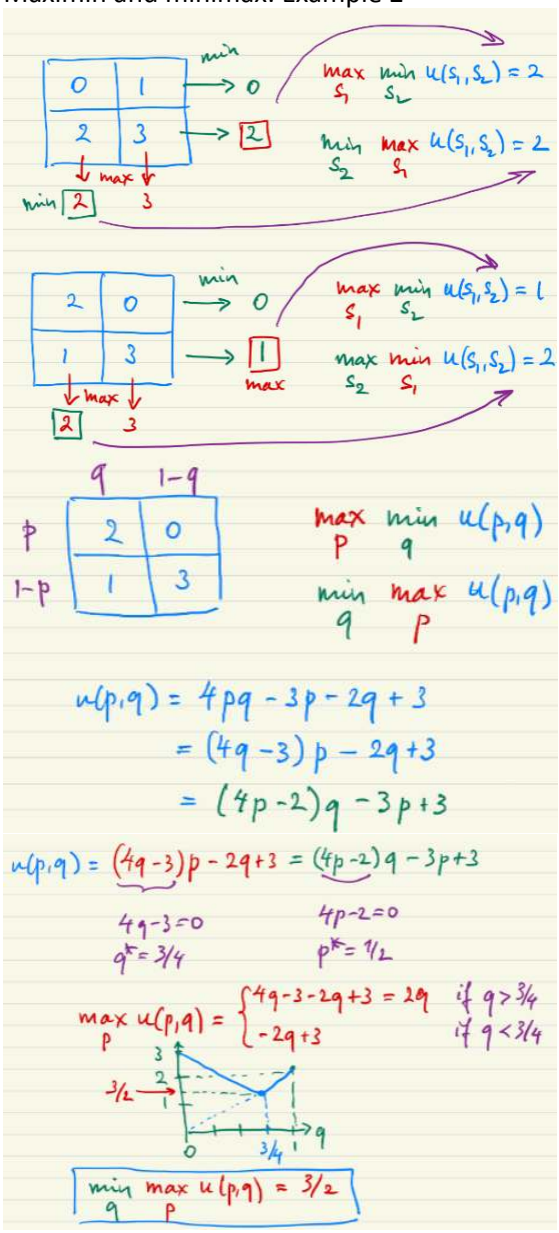
- Maximin vs minimax: Numerical example



  ➔ This is the utility matrix pf the first player. The utility of the second player will just be the opposite.

  ➔ So the first player is going to maximise each column → gives, 6, 9 and 8. So the minimum of the maximum is 6. The second player, likes to minimise all the rows, which gives -5, 6 and -3. So the maximum of the min is 6.

- Maximin and minimax: Example 2



  - ➔ If you know p*, here (1/2) and q* here (3/4). You can also plug it into the formula u(p, q): $u(p^*, q^*) = \max_p \min_q u(p, q) = \min_q \max_p u(p, q)$
  - ➔ So in all the above games there is an equilibrium. Although it might sometimes be a mixed strategy.

Maximin Value and Safety Strategy

- Maximin Value and Safety Strategy
  - **Helpful interpretation: dealing with a spy**
    - **Player i** knows that **player j is vindictive and malicious…**
    - Furthermore, **player j has spy** in **player i's camp** who informs him (JIT) of i's action (s_i);
    - **Simul → seq:** Equiv. to **i playing first**, revealing i's strategy;
    - **Player j** will then play strategy to **minimise i's pay-off**:

$$\text{resulting pay-off for i} = \min_{s_j} u_i(s_i, s_j)$$

- However, **player i** suspects he has a mole in his camp and therefore plays the action that **maximises his worst pay-off:**

$$v_i^{mami} := \max_{s_i} \min_{s_j} u_i(s_i, s_j)$$

- **Safety:** this **maximin value** for player i is **best pay-off player I can guarantee** himself (**irrespective of what opponent does**).
  - ➔ Take in mind that the other payer is malicious and therefore tries to minimise your utility.

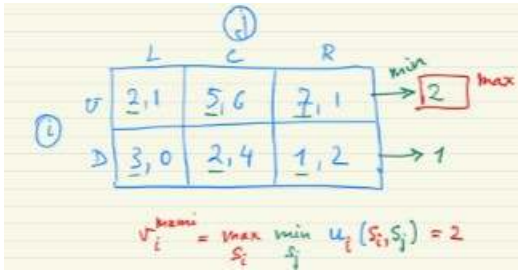
- Maximin strategy (safety strategy): Algorithm
  1. Player i computes for each of his actions the worst possible outcome:

$$s_i \longrightarrow \min_{s_j} u_i(s_i, s_j)$$

  2. Next player i chooses action s_i to maximise his minimal (worst) pay-off:

$$v_i^{mami} := \max_{s_i} \min_{s_j} u_i(s_i, s_j)$$

  - **Agent tries to maximise pay-off of worst possible outcome**



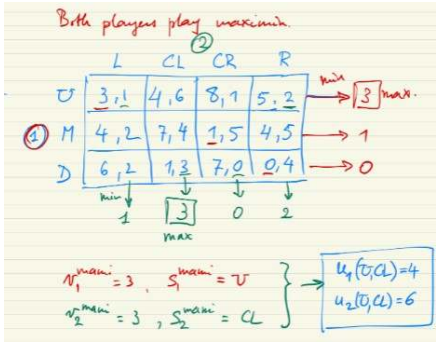
- Maximin Value and Strategy
  - **Context:** 2-player, general sum game
    - **Maximin value** (or **security level** for WORST CASE) is the guaranteed minimal pay-off for agent *i* playing strategies in S_i:

$$v_i^{mami} := \max_{s_i \in S_i} \min_{s_j \in S_j} u_i(s_i, s_j)$$

    - **Maximin or safety strategy** for agent *i* maximizes his worst pay-off:

$$s_i^{mami} = \underbrace{\arg \max_{s_i} \min_{s_j} u_i(s_i, s_j)}_{\text{security level}}$$

- Maximin (safety) value and strategy
  - **Why play maximin strategy?**
    - **Pay-off guarantee!** Highest pay-off agent *i* can guarantee himself **irrespective of** the actions taken by other agent(s).
    - **Worst case analysis:** assume that opponent is **malicious**!
  - Example: both agents play maximin-strategy



- Minimax Value against Punishment Strategy
  - **Helpful interpretation: Dealing with a spy**
    - **Player i** knows that **player j is vindictive and malicious**…
    - … but **player i has spy** in **player j camp** who informs him (JIT) of j's action (s_j);
    - **Player i** will then play his **best response** yielding utility:

      $$BR_i(s_j) \longrightarrow \max_{s_i} u_i(s_i, s_j)$$

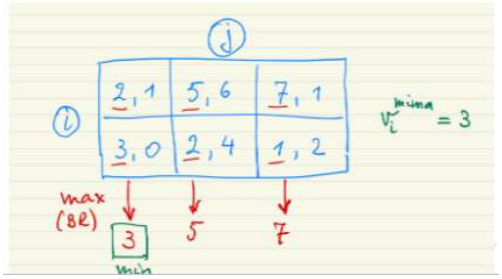
    - However, **player j** wants to **punish** player i plays the action that minimises the pay-off of i's best response, without regard for own pay-off ("**threat**");
    - This yields the **minimax value for player i**:

      $$v_i^{mima} := \min_{s_j} \max_{s_i} u_i(s_i, s_j)$$

    - This is **worst pay-off player j can force** on **player i**.

  - Minimax value for player i: Algorithm
    1. **Player i** computes best response for each of j's strategies s_j:
       $BR_i(s_j) = \max_{s_i} u_i(s_i, s_j)$;
    2. Then **player j** picks action to **minimise player I best pay-off**. This results in the **minimax value for player i**:

       $$v_i^{mima} := \min_{s_j} \max_{s_i} u_i(s_i, s_j)$$



➔ So the punishment strategy is different from the safety strategy.

- Recap: Minimax Value and Punishment Strategy
  - Minimax value for agent i :
    - Given the strategy sj of his opponent, agent i will play its best response, resulting in a pay-off:

      $$\max_{s_i} u_i(s_i, s_j)$$

    - The opponent is aware of this and wants to "punish" i by minimizing this pay-off, yielding the minimax value for player i:

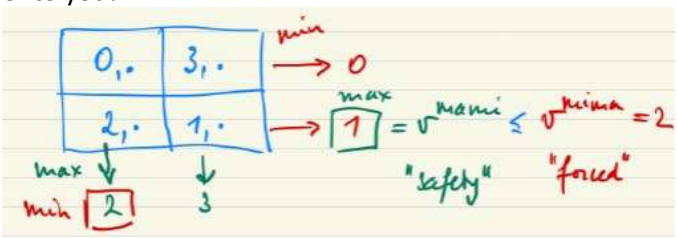
      $$v_i^{mima} := \min_{s_j} \max_{s_i} u_i(s_i, s_j)$$

  - The corresponding minimising strategy s_j^{mima} is called the **minimax strategy** for **player j**.
  - If j plays his minimax strategy s_j^{mima} , then i cannot do better than v_i^{mima} (even if i plays best response $BR_i(s_j^{mina})$.

- Minimax and Maximin value for player i.
  - In general:

$$v_i^{mami} = \max_{s_i} \min_{s_j} u_i(s_i, s_j) \quad \leq \quad \min_{s_j} \max_{s_i} u_i(s_i, s_j) = v_i^{mima}$$

  - "Your **guaranteed pay-off** is a **lower bound** for the **worst your opponent can force** onto you!"



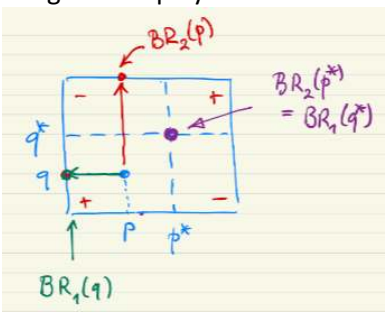
- Safety and Punishment Strategy
  - Player i 's maximin strategy is **safety strategy**
    - player i concerned about his own safety.
    - strategy yields highest guaranteed outcome for player i.
    - viable **solution algorithm**.
  - Player i 's minimax strategy is **punishment strategy**
    - i 's strategy is directed **against** player j.
    - player i tries to minimize best (i.e. maximum) pay-off for j.
    - Useful as **threat** (e.g. in repeated games);
    - **'s maximin strategy** gives rise to **j 's maximin value**.

Equilibrium Concept: Nash equilibrium

- **Nash equilibrium: Definition**
- **Nash equilibrium: Amplification**
- **Further examples of Nash equilibria**

- Nash equilibrium
  - Extension of von Neumann's minimax theorem
    1. From **two person** to ***n* person** game;
    2. From **zero sum** to **general utilities**.

- Nash equilibrium: intro
  - Reconsider an arbitrary strategy profile (*p*, *q*) in 2p-zs game;
  - Imagine one player is allowed to change his mind!



Here you assume you know what the other is doing and the question is than do you want to change and if you do not want to change that is then the Nash equilibrium (purple).

- Nash equilibrium (1)
  - A **Nash equilibrium** (NE, 1950) is a solution concept based on *conditions* instead of an *algorithm*.
  - **Mutual best response:** NE is joint strategy profile *s** such that **for each agent** *i* the strategy s_i^* is a **best response** to s^*_{-i};
  - **Formally:** A strategy profile $s^* = (s_1^*, s_2^*, \ldots, s_n^*)$ is a **strict NE** if:

$$\forall \text{agents } i,\ \forall s_i' \neq s_i^* : u(s_i^*, s^*_{-i}) > u(s_i', s^*_{-i}).$$

  - **Strict** (>) versus **weak** () NE.
  - **No Regret/ Self-enforcing:** a (strict) NE is a stable strategy profile for which no agent has an incentive to **unilaterally deviate**;

- Nash equilibrium: Computation of NE (pure strategy)
  - **Find mutual best responses:**
    - **Battle of the Sexes** (two pure strategy NEs)

	action	comedy
action	2, 1	0, 0
comedy	0, 0	1, 2

    - **Prisoner's dilemma** (single NE, not Pareto-optimal!)

	hush	confess
hush	−1, −1	−12, 0
confess	0, −12	−8, −8

      - ➔ For the battle of the sexes, the Nash Equilibrium is (action action) and (comedy, comedy). The bad thing about this is that it does not help to make a decision. Therefore, this is a weakness as it does not tell you what to do.
      - ➔ For the Prisoner's dilemma we see that there is one Nash Equilibrium (Confess, Confess).
      - ➔ Mutual best response = Nash equilibrium.
- Nash's Theorem
  - Existence of Nash Equilibrium: A **Finite strategic game** (i.e. finite number of players and actions) always has **at least one Nash equilibrium** (allowed mixed strategies).
    - A **pure** Nash equilibrium can be strict or weak;
    - A **mixed** Nash equilibrium is necessarily weak.
      - ➔ Strict theory; your utility is going down and weak it is not necessarily moving up or downwards.

- Nash equilibrium: Nash's Theorem
  - A **finite strategic game** is a game with a **finite number of agents** and a **finite number of actions**;
  - A game may have **zero, one** or **more pure-strategy NE**.
  - If there's a **single** NE: natural solution concept, but might be sub-optimal!
  - If there are **multiple NEs**: there might be no compelling reasons to pick a particular one; but…
    - **Schelling's focal points.**

- Note on multiple Nash Equilibria
  - The existence of a **unique best alternative** is **exceptional**. In general, non-cooperative game theory is plagued by a **multiplicity of equilibria**. Hence, prescriptions of **how to act without any coordination or cooperation** are in general impossible. **Non-cooperative game theory** has to confine itself to suggest what should be **excluded**

from choice. There have been refinements of the notion of a Nash equilibrium, but they do not free non-cooperative game theory from this multiplicity problem. The social environment, in which non-cooperative game theory is meaningfully applied, is one in that **any kind of binding contract or commitment** about actions is <u>unavailable</u>. Lying and cheating is possible without sanctions. So a player's announcements and **promises** in a pre-play phase are **credible** only, if they are totally **in line with his best interests**. If the latter can be determined by rational reasoning the former are superfluous.

- Computation of NE
  - **Pure NE for each agent** *i* th strategy s_i^* is a **best response** to s^*_{-i} (mutual best response);
    - Matrix games (discrete/action) space;
    - Continuous action space.
  - **Mixed NE:** make opponent indifferent (matrix games only);

- Nash equilibrium: Computation of mixed NE
  - Matching pennies (zero-sum game)
    - **No pure strategy NE ...**

	heads	tails
heads	$\underline{1}, -1$	$-1, \underline{1}$
tails	$-1, \underline{1}$	$\underline{1}, -1$

    - ...hence, at least one **mixed strategy NE!**
    - **Intuitively** this is obvious; play each action with prob $= 1/2$:

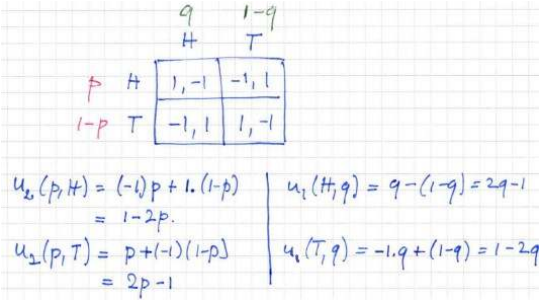
$$s_1 = s_2 = \{(H, 1/2), (T, 1/2)\}$$

    - **Expected utility (pay-off)**:

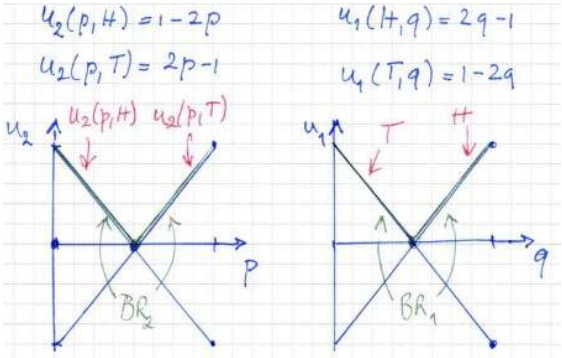
$$u_1(s_1, s_2) = \frac{1}{4}u_1(H,H) + \frac{1}{4}u_1(T,H) + \frac{1}{4}u_1(H,T) + \frac{1}{4}u_1(T,T) = 0.$$

➔ No Nash equilibrium.

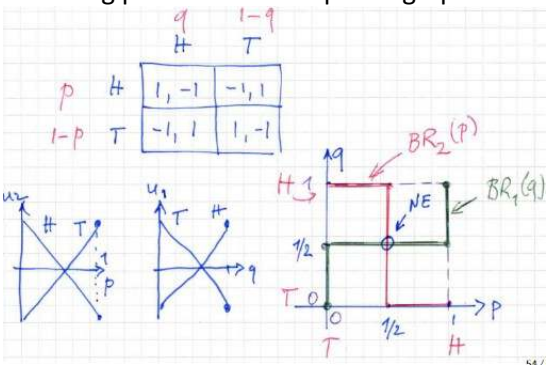
- Matching pennies: Computation of mixed NE (example computing NE)



- Matching pennies: Computation of best response



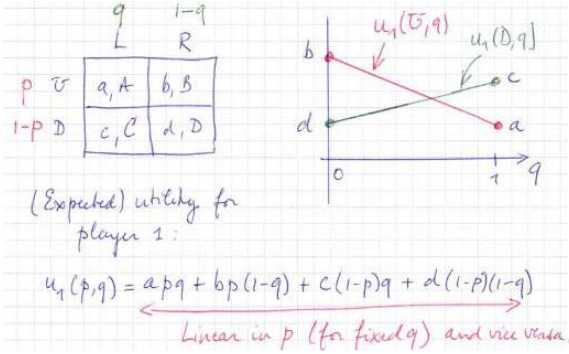
- Matching pennies: Best response graph



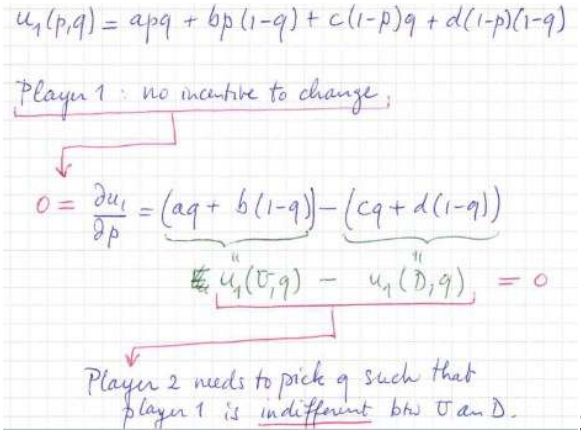
- Matching pennies: Mixed Nash Equilibrium
  - **Nash equilibrium characteristics:**
    - At the intersection point (p = 1/2 , q = 1/2), players are simultaneously playing best response to each other;
    - No player can do strictly better by unilaterally deviating:
      - If player 2 keeps playing q = 1/2 then player 1's utility u_1 = (p,q = 1/2) can be computed as follows:

$$\begin{aligned} u_1(p,1/2) &= (1\cdot p + (-1)\cdot(1-p) + (-1)\cdot p + 1\cdot(1-p))\cdot\frac{1}{2} \\ &= 0 \end{aligned}$$

      - Player 1 therefore has no incentive to change his strategy (change p).
      - Some consideration for player 2.
- Computation of mixed Nash equilibrium

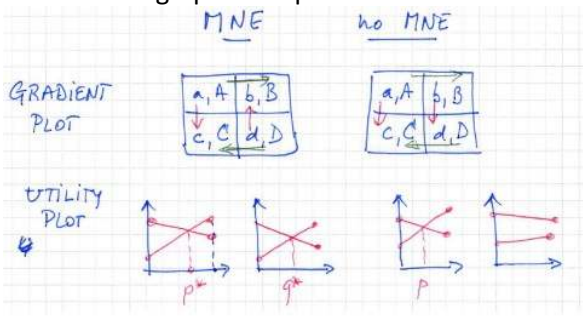


Capital letter means stochastic.



So because the = 0, you get $U_1(U, q) = U_1(D,q)$.

- Some useful graphical representations



MNE = Mixed NE. With no MNE, the graphs will not intersect. With the right graph you have one dominated strategy, it doe not matter what the opponent will do → means that you will end up with a **pure strategy**.

- Nash equilibrium: Computation of mixed equilibrium
  - Battle of the Sexes (<u>two pure strategy NEs</u>).
  - Additional NE by **mixing pure strategies**?

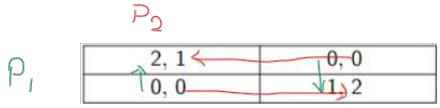
$s_1 = \{(A, p), (C, 1-p)\}$ and $s_2 = \{(A, q), (C, 1-q)\}$.

	Action (q)	Comedy ($1-q$)
Action (p)	2, 1	0, 0
Comedy ($1-p$)	0, 0	1, 2

- Determining the mixture parameters p and q
  - Ag 1 chooses p such that Ag 2 is **indifferent** btw actions A and C; if not Ag 2 would focus on the most lucrative option.

Condition for Ag 1: $u_2(s_1(p), A) = u_2(s_1(p), C)$

A = action and C = comedy



  - Battle of the Sexes (<u>mixed strategy Nes</u>)

	Action (q)	Comedy ($1-q$)
Action (p)	2, 1	0, 0
Comedy ($1-p$)	0, 0	1, 2

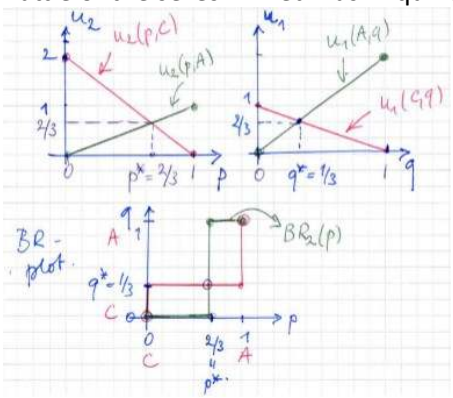
- Determining the mixture parameters p and q

$$\text{Ag 1:} \quad EU_2(s_1(p), A) = EU_2(s_1(p), C) \implies p = 2(1-p)$$

$$\text{Ag 2:} \quad EU_1(A, s_2(q)) = EU_1(C, s_2(q)) \implies 2q = (1-q)$$

$$\textbf{Conclusion:} \quad p = \frac{2}{3},\ q = \frac{1}{3} \qquad EU_1(s_1, s_2) = EU_2(s_1, s_2) = 2/3.$$

- Battle of the sexes: Mixed Nash Equilibrium



- Nash equilibrium: Computation of mixed equilibrium
  - **Prisoner's Dilemma** (does mixed strategy NE exist?)

	Quiet (q)	Confess ($1-q$)
Quiet (p)	$-1, -1$	$-12, 0$
Confess ($1-p$)	$0, -12$	$-8, -8$

  - Determining the mixture parameters p and q.

$$\text{Ag 1: } EU_2(s_1, Q) = EU_2(s_1, C) \implies -p - 12(1-p) = -8(1-p)$$

$$\text{Ag 2: } EU_1(Q, s_2) = EU_1(C, s_2) \implies -q - 12(1-q) = -8(1-q)$$

$$\textbf{Conclusion:} \quad p = q = \frac{4}{3} \quad \textbf{impossible!}$$

Because of the p and q above 1, there is no probability and therefore: no mixed strategy.

- Mixed NE for game with three actions
  - **Rock-Paper-Scissors: no PURE NE!**

		v	w	$1-(v+w)$
		R	P	S
p	R	$0,0$	$-1,1$	$1,-1$
q	P	$1,-1$	$0,0$	$-1,1$
$1-(p+q)$	S	$-1,1$	$1,-1$	$0,0$

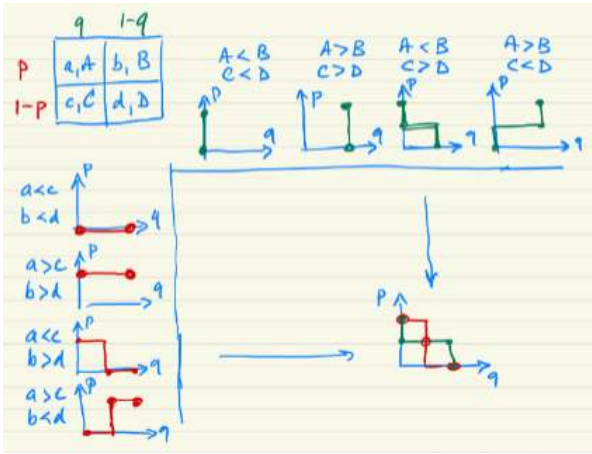
Determine p, q by insisting that player 2 is indifferent btw actions:

$$\begin{aligned} 0\cdot p + (-1)\cdot q + 1\cdot(1-p-q) &= 1\cdot p + 0\cdot q + (-1)\cdot(1-p-q) \\ 0\cdot p + (-1)\cdot q + 1\cdot(1-p-q) &= -1\cdot p + 1\cdot q + 0\cdot(1-p-q) \end{aligned}$$

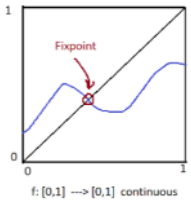
Result: $p = q = 1/3$. Determine mixing parameters u, v similarly.

    - ➔ You can calculate p and q just by solving one of the equations.
    - ➔ Indifferent implies that you don't want to change.

- Nash equilibrium for 2 X 2 matrix game

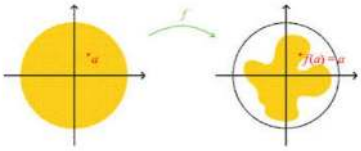


- Aside: Sperner's Lemma (1928) and fix-point theorems
  - **Definition: Fix-point:** Point a ∈ A is a fix-point for function f : A → A, iff f(a) = a.



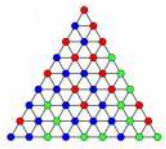
  - **Sometimes(!)**, fixpoints can be computes using function iteration.

- Nash Theorem is based on Fixed-Point Theorem
  - Brouwer's Fixed Point Thm
    - Let K ⊂ be a compact and convex, and f : K → K continuous. The f has a fix-point in K, i.e. there exists a x_0 ∈ K : $f(x_0) = x_0$.

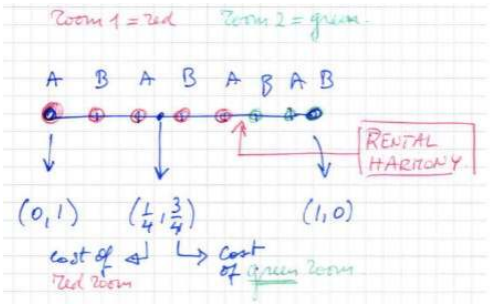


: Non-constructive existence proof!

- Aside: Sperner's Lemma (1928) and fix-point theorems
  - **If in triangle**:
    - Corner nodes have different colours.
    - Nodes on outer edges have two possible colours (determined by corners)
  - **Then** there is a sub triangle with 3 differently coloured corners.



- Sperner's lemma: Rental Harmony: Fair division of rent



- Divide interval in segments by adding points.
- Assign alternating decision makers (A and B).
- Decision maker decides which room he picks (for given rental division)
  - ➔ (0,1), means red room is now 0% of the rent and the green room is now 100% of the rent.
  - ➔ The arrow between the red and green dot represents the price point on which both roommates can agree.
  - ➔ This game you can extent by having three roommates and then you get the triangle above. Where each edge of the triangles should have a different colour to find the agreement, where everybody is happy about the price point.

- Games with NO Nash Equilibrium (So you just need to know that there are examples where there is no nash equilibrium)
  - **Dollar auction:** Sealed bid auction: highest bid gets item, 2nd highest pays this amount!
    - **So no equilibrium, it escalates**.
  - More generally: One can construct games without NE by making sure that either
    - State space is **not compact**.
    - Utility function is **not continuous**.
  - **Ex. For 2-player games with continuous state spaces:**
    - Non-compact: $u_i(x, y) = xy$ for $0 \leq x, y < 1$.
    - Non-continuous: $0 \leq x, y \leq 1$ and

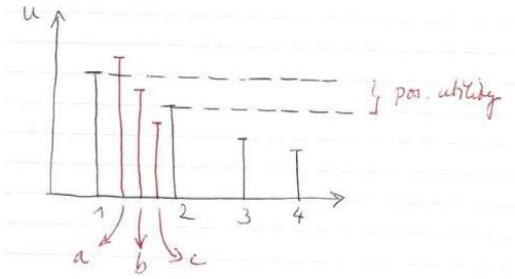
$$u_i(x,y) = \begin{cases} xy & \text{if} \quad x,y < 1 \\ 0 & \text{if} \quad x,y = 1 \end{cases}$$

- Nash equilibrium: Computational aspects
  - Finding a Nash equilibrium for 2-player zero-sum games can be done efficiently by formulating a linear program. Notice that in this case: NE = minimax = maximin.
  - Finding a Nash equilibrium is not known to be NP-complete because it is not a decision problem.
  - PPAD (polynomial parity argument, directed version) is a class describing problems for which a solution always exists.
  - Daskalakis, Goldberg, and Papadimitriou showed that finding a sample **Nash equilibrium** of a **general-sum finite game** with two or more players is **PPAD-complete** (i.e. “difficult!”).

Lecture 4

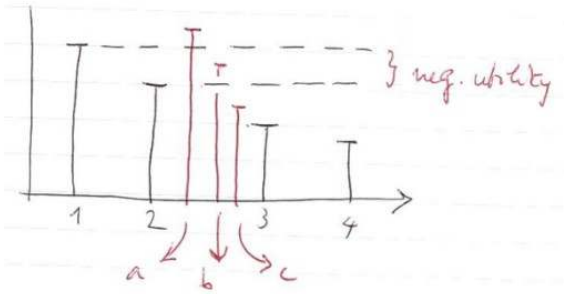
- Vickrey auction: Second Price Auction

- **Vickrey Auction**:
  - n sealed bid auction for single item;
  - highest bid wins, but pays 2nd highest price;
- **Truth-telling** is (weakly) dominates strategy;
- NE: Neither winner nor loser(s) have incentive to deviate:
- Winner:
  - Higher: still winner, same price;
  - Lower: might lose, but if still winner, still paying 2nd price;
- Loser:
  - Higher: possibly winner, but at higher price;
  - Lower: still loser;
- Example of **mechanism design**. → So even selfish agents come up with a fair solution. (dividing the cake is also a mechanism design).
- Vickrey Auction: Alternatives for winner → real world example of mechanism design.



➔ Agent 1, means he wins the auction.

- Vickrey Auction: Alternatives for runner-up



- Hawk or Dove
  - Equilibria as a function of **exogenous pay-off variables**;
  - **Exogenous variables** are imposed on the game (not by players);
  - **Strategy: Hawk or dove:**
    - Two parties are in conflict over some good of value $v > 0$;
    - Fighting over it comes at cost c.
    - **Pay-off matrix:**

	Hawk	Dove
Hawk	$\frac{v}{2} - c, \frac{v}{2} - c$	$v, 0$
Dove	$0, v$	$\frac{v}{2}, \frac{v}{2}$

    - Different outcomes depending on: $c < v/2, c = v/2, c > v/2$

- Investment game
  - Many agents (n) but only two strategies;

- **Strategies:** each agent can either not invest (N) or invest 10 euro (I).
- **Pay-offs**:
  - N: No, investment: pay-off = 0;
  - I: Invest 10 Euro:

$$\text{net pay-off (in Euro)} = \begin{cases} 5 & \text{if at least 90\% of agents invest} \\ -10 & \text{otherwise} \end{cases}$$

- **Coordination game!**

- Strategic Effects
  - **Symmetric penalty kick game:**

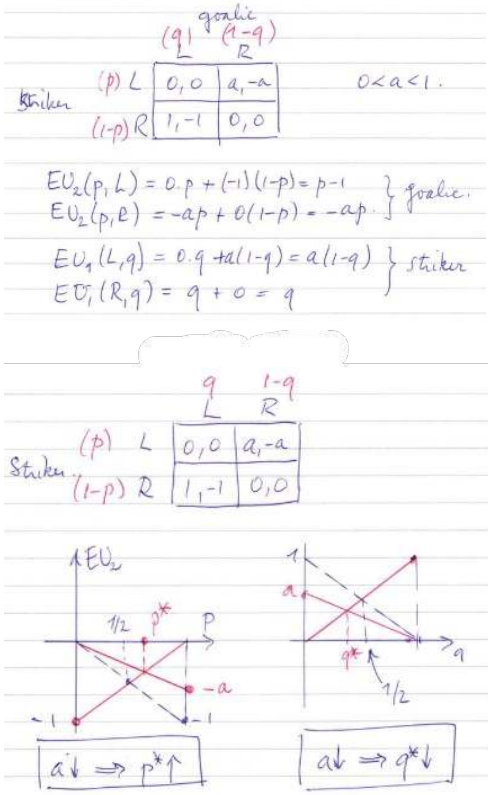
		goal keeper: left	goal keeper: right
kicker	left	0, 0	1, −1
	right	1, −1	0, 0

  - Suppose kicker has weak left kick: **direct and indirect effect!**

		goal keeper: left	goal keeper: right
kicker	left	0, 0	a, $-a$
	right	1, −1	0, 0

$0 < a < 1$

  - Strategic effect



➔ kicker has higher probability going left and the goali has higher probability jumping right. But the quality of the kicker is less (so therefore not necessarily more goals).

- Nash equilibrium: NE and IEWDS
  - Eliminating **weakly dominated** strategies might erase NEs!

	L	R
U	2,3	4,3
D	3,3	1,1

  - Two NEs: (4,3) and (3,3).
  - L **weakly** dominates R;
  - Eliminating R would result in single solution (3,3);
  - Notice that the **Pareto-optimal NE would be eliminated**.

- Focal points (Schelling)
  - Focal points
    - Some equilibria are more “natural” than others;
    - Allow coordination without communication;
  - Example: allow two persons to split 100 Euro: if they match, they can keep the money. Majority proposes 50=50.

- Other solution concepts
  - **Minimax equilibrium:** zero-sum special case Nash.
  - **Trembling-hand perfect equilibrium**: each player’s action is a best-response even if other players make small mistakes.
  - **-Nash equilibrium**: deviating benefits no agent more than .
  - **Correlated equilibrium:** agents can condition strategy on external signal: “if there is intelligent life on other planets, in a majority of them, they would have discovered correlated equilibrium before Nash equilibrium”
  - **Evolutionary stable state:** “A population is said to be in an evolutionarily stable state if its genetic composition is restored by selection after a disturbance, provided the disturbance is not too large.”

- Summary
  - Game theory studies utility-based multiagent decision making.
  - Solving a game means trying to predict its outcome.
  - Rational agents never play strictly dominated actions.
  - No agent has an incentive to **unilaterally deviate** from a Nash equilibrium.
  - There is always an NE in mixed strategies.
  - Nash equilibria may not be Pareto optimal.

Lecture 5: Sequential Games

- Sequential games
  - **Normal-form games:**
    - **Simultaneous** moves by players.
    - Central solution concept: **Nash equilibrium**.
  - **Sequential games:**
    - Players move in **succession**, observe (at least partially) **prior moves** by opponents.
    - **Perfect versus imperfect information:** what exactly is known about previous moves?
      - Players have full knowledge of all the preceding moves (**perfect information**).
      - Players might not know the complete game history till then (**imperfect information**).
    - **Model for many sequential interactions** in games, politics, economics, etc.

- Simultaneous vs. Sequential Games
  - **Simultaneous games:** player make their moves simultaneously, i.e. **without knowing** what the other players will do!
    - Rock-paper-scissors.
    - Sealed bid auctions.
    - Cournot's duopoly model.
  - **Sequential games:** Sequence of successive moves by **players who can see each other's moves** (**to some extent** – see next slide):
    - Chess
    - Card games
    - Open cry auctions
    - Stackelberg's duopoly model
    - Negotiation (Rubinstein's model)

- Extensive form representation of sequential game
  - Visualisation of temporal relationships (**game tree**).
  - **Extensive form** is finite game representation that **does not assume** that players act **simultaneously**;
  - Can be converted in normal form representation (*possibly exponentially larger!*).
  - **Game tree:** makes **temporal structure** explicit.

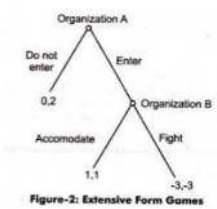


Figure-2: Extensive Form Games

- Information in Game Theory

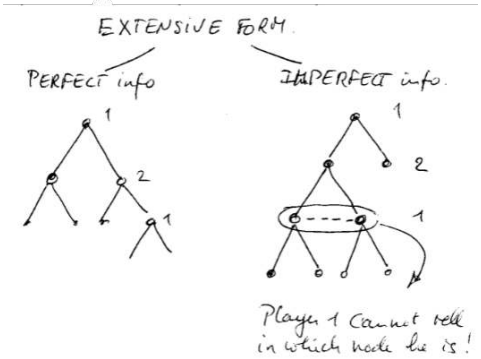
	PERFECT complete history known to all players	**IMPERFECT** unaware of actions taken by others
COMPLETE NO private info agents, actions, payoffs known	E.g. chess	Simultaneous games Information sets
INCOMPLETE private info e.g. private valuation	• Open cry auction • Different types of opponents	Sealed bid auction

- Aside: Types of knowledge
  - **Mutual knowledge** is
    - Known to all players,
    - But players do not know that others know.
    - E.g. *the elephant in the room*, solutions to homework's.
  - **Common knowledge** is
    - known to all players,
    - and all players know all others know…
    - and all players know all others know that all others know…
    - and so on…
    - e.g. In continental Europe one drives on the RHS of the road.

- Major Ideas in Non-Cooperative Game Theory

	SIMULTANEOUS (STATIC)	**SEQUENTIAL** (DYNAMIC)
Complete information NO private info	Nash equilibrium	Backwards induction, Subgame-perfect NE: *discard NE based on non-credible threats*
Incomplete information private info	Bayesian Nash eq.	Perfect Nash eq.

- Sequential games: perfect vs. imperfect information



Backward Induction for Sequential Games with Perfect Information

- Sequential Games with Perfect Information
  - **Prototype: two-player, sequential-move game:**
    - Player 1 chooses action $a_{11} \in A_1$;
    - Player 2 observes a_{11} and then chooses action $a_{21} \in A_2$;
    - …
    - Hereafter, both players receive pay-off: $u_1(a_{1n}, a_{2n})$ and $u_2(a_{1n}, a_{2n})$ respectively;
  - **Examples:**
    - Various board and card games (e.g. chess, go, etc.)
    - Stackelberg's sequential-move version of Cournot's duopoly;
    - Rubinstein's bargaining model.

- Solving Extensive Form Games using Backward Induction.

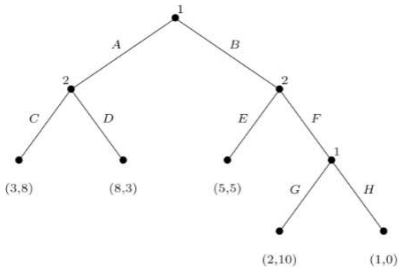
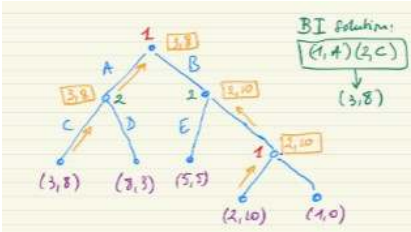
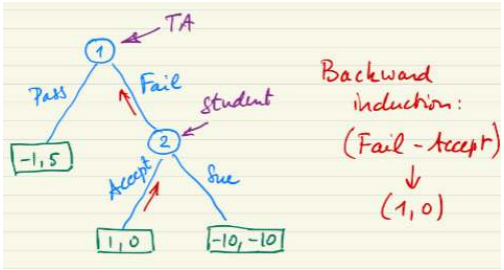


Figure 5.2: A perfect-information game in extensive form.

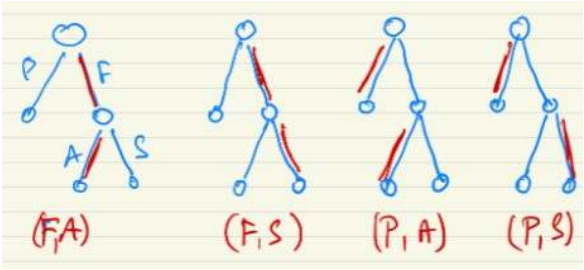
  - **Backward induction**
    - Basic assumption:
      - Players believe that all **future play will be rational**.
      - And **condition decisions** on what they expect in future.
    - Algorithm
      - **Start at leaf-nodes:** easy decision as only one player involved;
      - **Propagate** decisions and utilities **to root**.



  - Bad homework: Backward induction (another example)
    - Student is trying to bully TA into giving passing grade!



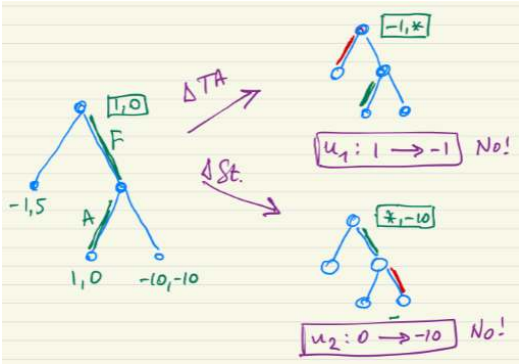
      - ➔ So backward induction: P2 chooses (1,0) then P1 chooses Fail, because 1 is higher than -1.
  - Other possible strategy profiles



➔ So the backward induction was the first on the left (F,A)

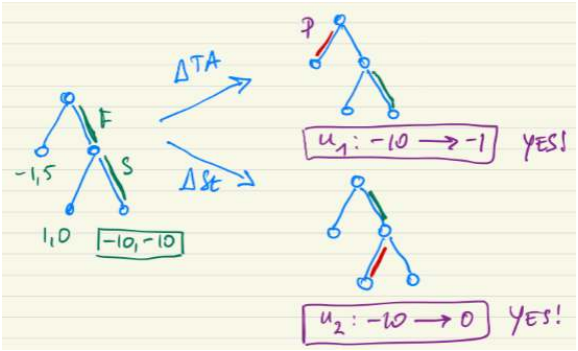
- Is **backward induction** solution (*F*, *A*) a **Nash eq.**? See next points
- Are there other **Nash eq.**?

- Backward Induction -solution(F, A) is Nash eq.

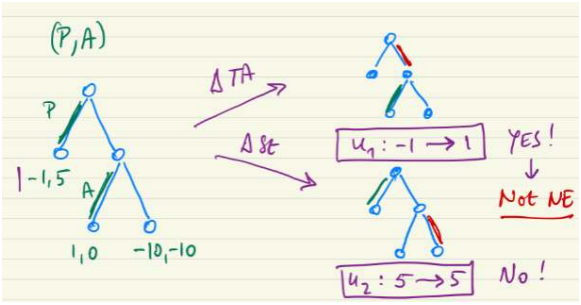


➔ So it is a NE, because P1 does not want to go from 1 to -1, and P2 does not want to go from 0 to -10.

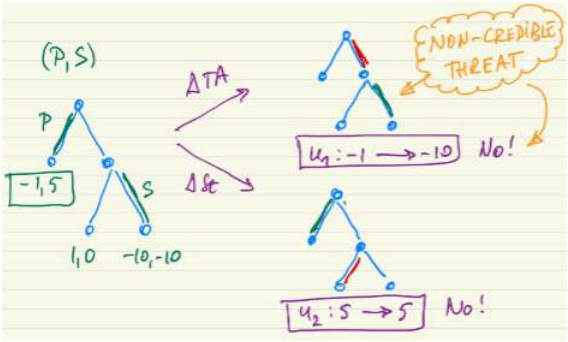
- (Fail, Sue) is not NE



- Pass-Accept is not NE

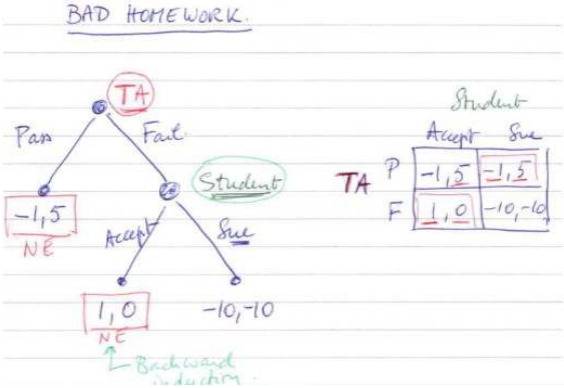


- (Pass, Sue) is NE because of non-credible threat



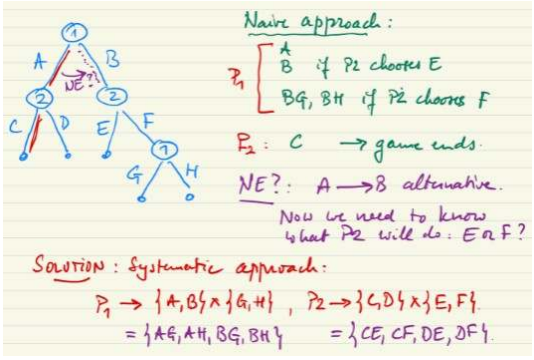
  - **(Pass, Sue): no incentive for unilateral deviation: hence NE!**
  - … but due to **non-credible treat (sue)** by student.
    - ➔ Non-credible treat does not respond to what a rational agent would do. So you basically want to exclude that.
    - ➔ You go from 5 to 5, because the TA is letting you pass. So it does not matter if you Sue or not.

- Bad homework: from extensive to normal form



  - **NE in extensive form** reappear in **normal** form.
  - **Checking NE in normal form** is easier than in **extensive** form.
    - ➔ The NE are (F,A) and (P,S).

- From Extensive for Normal Form (perfect information) so from extensive form to normal form
  - **Aim:** transform **sequential game in normal form** to use **standard methods to find NEs**.



➔ Extensive form is the tree and normal form the matrix.

- From Extensive to Normal Form (perfect information)
  - **A pure strategy** for player *i* in a (perfect information) sequential game is a **complete plan of action** specifying which action to take at **each of its decision nodes**…
  - … **irrespective of whether or not that node can be reached when playing the strategy!**
  - Mathematically: it's the **product space** of the **possible actions** in **each decision node**:
    - Node 1 has 2 decision nodes: hence $\{A, B\} \times \{G, H\} = \{(A, G), (A, H), (B, G), (B, H)\}$
    - Node 2 has 2 decision nodes: hence $\{C, D\} \times \{E, F\} = \{(C, E), (C, F), (D, E), (D, F)\}$
  - **Alterative perspective:**
    1. **Extensive form:** player "waits" till one of his nodes is reaches, then decides what to do;
    2. **Normal form:** each player makes a **complete contingent plan** in advance.
  - **Informally:**
    - It's a **complete and contingent plan** instructing an assistant playing on your behalf, what to do in **each possible situation;**
    - Suppose that your assistant misunderstood and ended up in another node, then he still needs to know what to do.
    - Allows to explore whether unilateral deviation would be advantageous (Nash criterion).

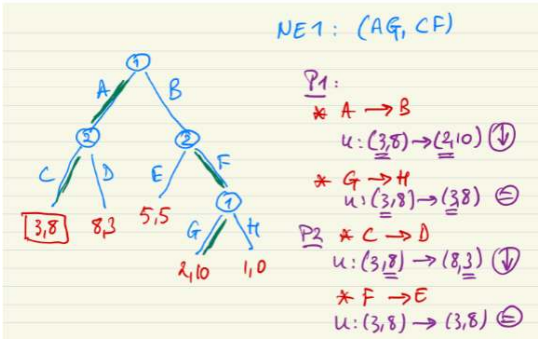
	(C,E)	(C,F)	(D,E)	(D,F)
(A,G)	3, 8	3, 8	8, 3	8, 3
(A,H)	3, 8	3, 8	8, 3	8, 3
(B,G)	5, 5	2, 10	5, 5	2, 10
(B,H)	5, 5	1, 0	5, 5	1, 0

Figure 5.3: The game from Figure 5.2 in normal form.

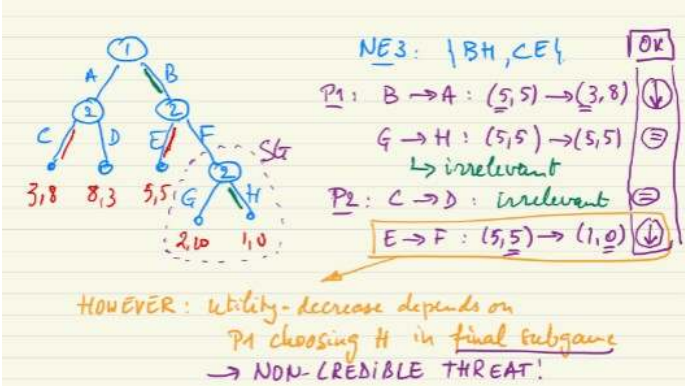
	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	3, 8	**3, 8**	8, 3	8, 3
(A, H)	3, 8	**3, 8**	8, 3	8, 3
(B, G)	5, 5	2, 10	5, 5	2, 10
(B, H)	**5, 5**	1, 0	5, 5	1, 0

Figure 5.4: Equilibria of the game from Figure 5.2.

  - Checking normal form NE (AG, CF)

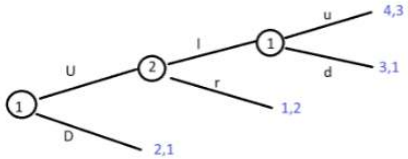


- Checking normal form NE (BH, CE) **based on non-credible threat!**



Backward Induction and Subgame-Perfect Equilibrium

- Example: Backward Induction vs. Nash Equilibria

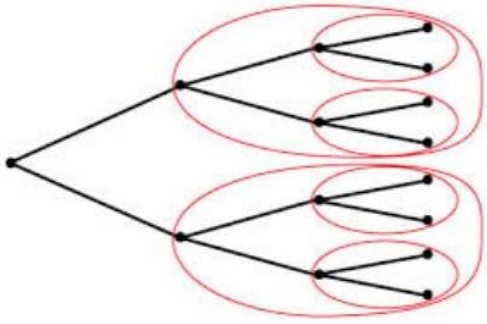


	ℓ	r	
Uu	$\underline{4,3}$	$1,2$	backward induction!
Ud	$3,1$	$1,2$	
Du	$2,1$	$\underline{2,1}$	"r non-credible threat"
Dd	$2,1$	$\underline{2,1}$	"d non-credible threat"

Non-credible threats do not constitute NE in their subgame!

➔ Non-credible treats not in sub-game

- Subgames and Subgame Perfection

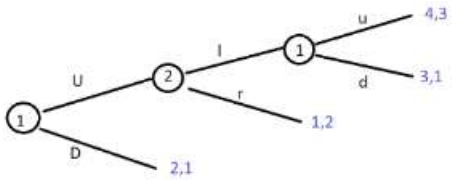


Subgame-Perfect Equilibrium (SGPE) :
induces Nash equilibrium in every subgame!

➔ Total tree is total game, the red circles are the sub-games.

- Subgame Perfect Nash Equilibrium
  - **SGPE: refinement of Nash Equilibrium**
  - Subgame-perfect equilibrium (SPGE, Selten 1965)
  - A Nash equilibrium *s* (of game *G* as a whole) is **sub-game-perfect** iff **for every subgame *G'*** of *G*, the **restriction of *s* to *G'* is also a Nash equilibrium**.
    - **SGPE rules out** Nash equilibria that rely on **non-credible threats**;
    - Put differently: SGPE is the study of **credible threats**.

- Example 2: Nash equilibria for subgames



	ℓ	r	
Uu	$\underline{4,3}$	$1,2$	backward induction!
Ud	$3,1$	$1,2$	
Du	$2,1$	$\underline{2,1}$	"r non-credible threat"
Dd	$2,1$	$\underline{2,1}$	"d non-credible threat"

$SG2:$

	ℓ	r
u	$\underline{4,3}$	$\underline{1},2$
d	$3,1$	$\underline{1},\underline{2}$

Notes

- d non-credible threat in SG1, implies r non-credible threat in SG2;
- Subgames (e.g. SG2) can have extra NE (comp. to full game)

➔ NE of the subgame (subtree player 2 + 1 → SG2) is (4,3) and (1,2).

  - *d* non-credible threat in SG1, implies *r* non-credible threat in SG2;
  - Subgames (e.g. SG2) can have extra NE (comp. to full game)
  - Game has two non-trivial subgames:
    - SG1 rooted at 2nd decision node of 1,
    - SG2 rooted at decision node of 2.
  - Normal form yields 3 NE's. Do they **induce NE in all subgames**?
    - ***NE1 = (Dd, r)*** with utility (2,1): induces action *d* in SG1 (not NE!).
    - ***NE2 = (Du, r)*** with utility (2,1): induces action (*u*, *r*) in SG2 (not NE!).
    - ***NE3 = (Uu, l)*** with utility (4, 3): induces actions *u* in SG1, (*u*, *l*) in SG2 (OK!).

➔ So take the whole game and eliminate the NE that do not induces in the subgames.

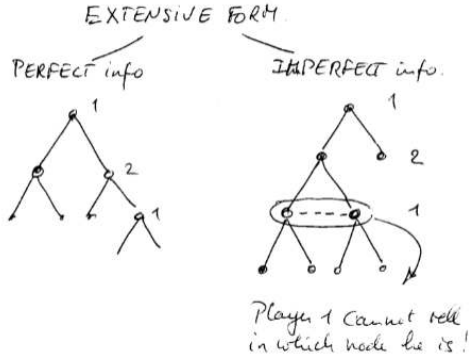
- Finding SGPE in perfect information games
  - **Two approaches:**
    - **Matrix form**
      1. Convert game-tree into matrix;
      2. Find all Nash equilibria;
      3. Eliminating the ones that depend on non-credible threats, i.e. do not induce a NE for each subgame.
    - **Backward induction (see next slide)**
      - **Works if NO simultaneous moves or infinite horizons!**

- Backward induction
  - **Algorithm to find subgame perfect equilibrium**
    - Consider each subgame of the game (in increasing order of inclusion).
    - Find the NE for the subgame;
    - Replace the subgame by a new terminal node that has the equilibrium pay-offs;
  - **Zermelo's thm** (1913)
    - With **perfect information** (one player in each iteration), a deterministic move is optimal. Hence there is a SGPE where each player uses a pure strategy.
    - **For games with imperfect information**, a SGPE may require mixed strategies.

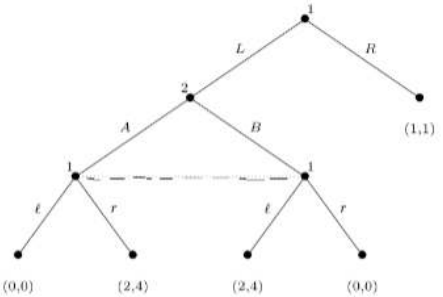
Sequential games with imperfect information

- Imperfect information and information sets
  - **Imperfect information: Intuition** Players need to act.
    - With **partial or no knowledge** of **actions taken by others,**
    - With partial recall, i.e. **limited memory** of **own past actions**.
  - An **imperfect- information game** is an extensive-form game in which each player's decision nodes are partitioned into **information sets**;
  - Intuitively, if two decision nodes are in the **same information set** then the agent **cannot distinguish** between them.

- Sequential games: perfect vs. imperfect information

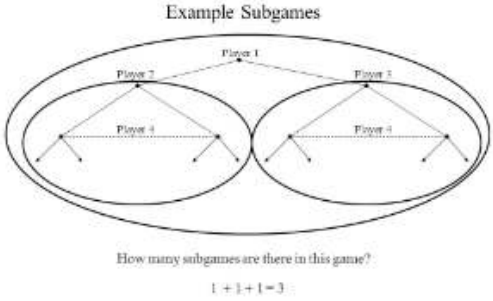


- Subgames: Condition on information set



  - **At any information set,** a player must have the **same strategies** regardless of how the player arrived there;
  - **Subgames cannot break up information sets!**
    - ➔ So here the only subgame is the subtree two. The subtrees 1 & 1 below cannot be subgames, as you then break up the information set.
- Subgame of a Sequential Game with Imperfect Information

- **Subgame definition:**
  - SG's **initial node** has **singleton information set**;
  - All **successors** are in SG;
  - Any information-set is either **completely in or out**;



- Pure strategies and induced normal form

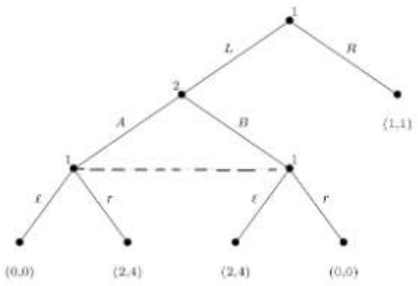


Figure 5.10: An imperfect-information game.

	A	B
$L\ell$	$0,0$	$2,4$
Lr	$2,4$	$0,0$
$R\ell$	$1,1$	$1,1$
Rr	$1,1$	$1,1$

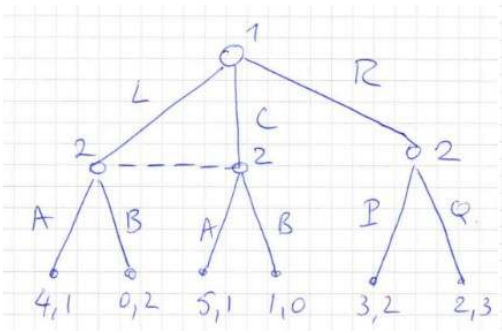
Figure 5.14: The induced normal form of the game from Figure 5.10.

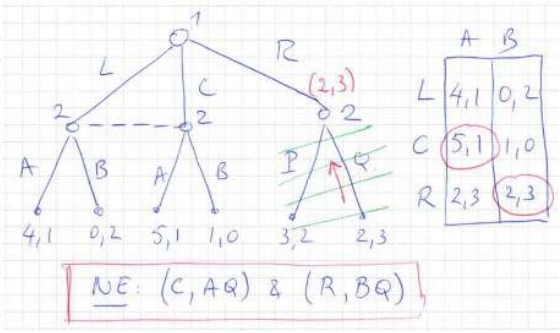
  - Pure actions are **cartesian products** over actions in **information sets.**

- Subgame Perfect Nash Equilibrium for Imperfect Information Games
  - **SGPE (aka SPE/SPNE): refinement of Nash Equilibrium:**
  - Subgame-perfect equilibrium (SPGE, Selten 1965): A Nash equilibrium *s* (of game *G* as a whole) is **subgame-perfect** iff **for every subgame *G'*** of G, the **restriction of *s* to *G'* is also a Nash equilibrium**.
    - **SGPE rules out** Nash equilibria that rely on **non-credible threats**;
    - Put differently: SGPE is the study of **credible threats**.

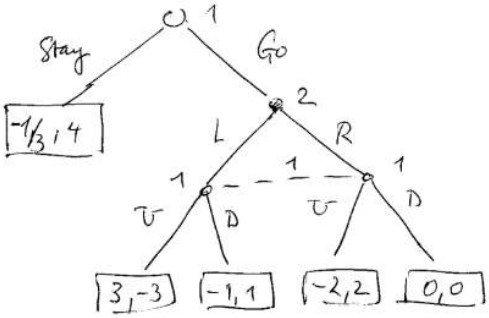
- Generalized Backwards Induction for Imperfect Information Games
  - **Systematically proceed as follows (if possible)**
    - Consider in turn each subgame of the game **in increasing order of inclusion**.
      - **Start at end of game, (no strategic interaction left).**
      - **Work backwards to beginning!**
    - **Apply BI** as far as you can;
    - Replace the subgame by a **new terminal node(s)** that has the equilibrium pay-offs (we might need to **consider different possibilities**);
    - If BI is not possible, use normal form solution techniques to find **NE(s) for the remaining game** (including **mixed** ones);

- Generalized BI: example 1

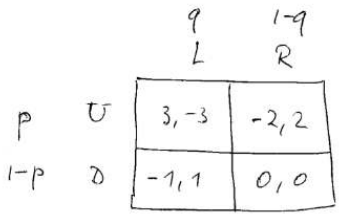




- Generalized BI: example 2

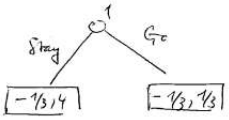


  - (cont'd)



$p^* = 1/6$ $q^* = 1/3$. no pure NE, therefore, there should be a mixed NE.

$$EU_1 = \frac{1}{6}\cdot\frac{1}{3}\cdot 3 + \frac{1}{6}\cdot\frac{2}{3}(-2) + \frac{5}{6}\cdot\frac{1}{3}(-1) + \frac{5}{6}\cdot\frac{2}{3}0$$
$$= \frac{1}{6} + \left(-\frac{2}{9}\right) + \left(\frac{-5}{18}\right) + 0$$
$$= \frac{3-4-5}{18} = -\frac{6}{18} = -1/3.$$



- Backward Induction vs. Subgame Perfection

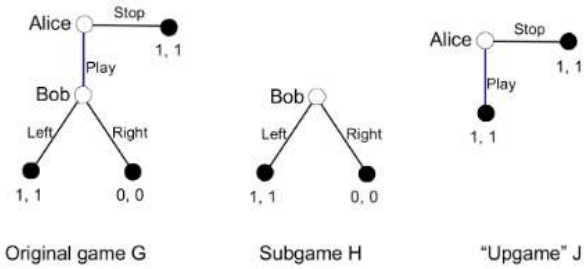


Figure 1. Backward induction versus subgame perfection.

The backward induction "upgame" J is NOT a subgame!

Ref: M. Kaminski: Generalized Backward induction, *Games* 2019, 10, 34

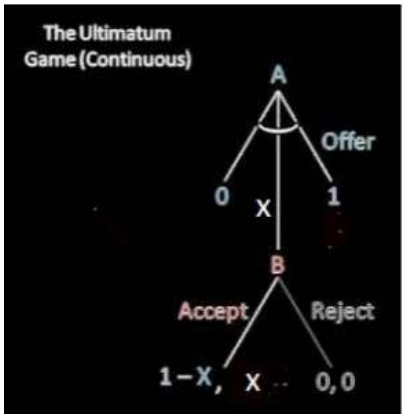
- **Finite** sequential games **with perfect information**:
  - All SPE can be found by backward pruning, ie.
    - systematic and incremental substitution of terminal subgames with Nash-eq. pay-offs.
  - All BI solutions (backward pruning) are SPE
- **Backward Induction** also works in some more general cases
  - e.g. some infinite games (e.g. Rubinstein)
- **More complex games** require **more restrictions** on the Nash eq. solution to **eliminate unreasonable solutions**.
  - E.g. sequential rationality, perfect equilibrium.

Lecture 6

Bargaining as example of sequential game

- **Ultimatum game (UG): baseline** (simplest non-trivial) model for **bargaining:** *take-it-or-leave it!*
- **Assumptions**
  - **Surplus** can be divided continuously ($0 \leq x \leq 1$)
  - **Two agents:**
    - A **proposes** split *x* versus 1-*x*, (**proposal power**).
    - B accepts or rejects;
  - **No deal (conflict deal)** is considered **worst outcome;**
  - Both agents aim to **maximize their utility**

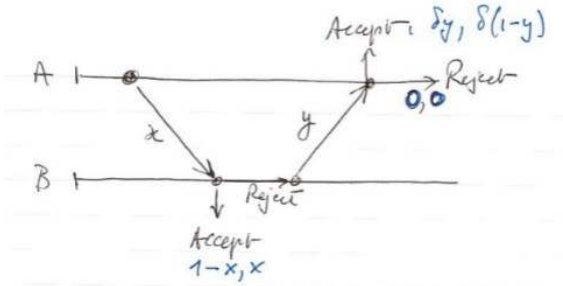
- One round ultimatum game
  - A makes single offer, B either accepts or rejects!



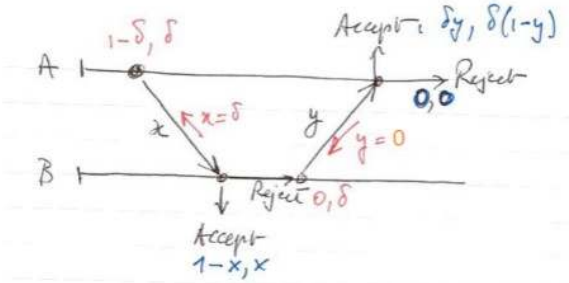
➔ The bow below A means that you can choose any value between 0 and 1.

➔ B needs to decide whether it accepts X from A, when he does then B gets X and A gets 1-X. When he rejects, both gets 0.

- Two rounds ultimatum game with impatient players / BI solution (same with the ice-cream kids example → the longer you wait the less you get)
  - **Power of counter-offer:**
    - A makes offer;
    - B either accepts, or makes counter-offer.
    - However, in each round the total is **reduced by factor δ < 1** (the ice cream is melting!)
  - A makes offer, but B can make counter-offer!

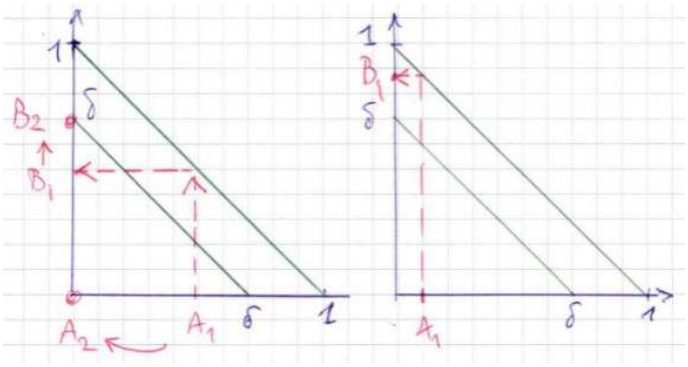


  - Use **backward induction** to find optimal solution.



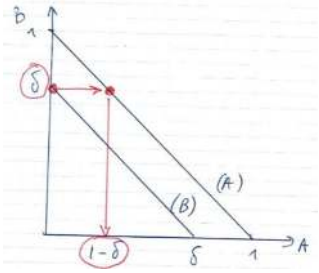
  - **Conclusion:** A offers split (1 −δ,δ) which B accepts.

- Alternative interpretation from above



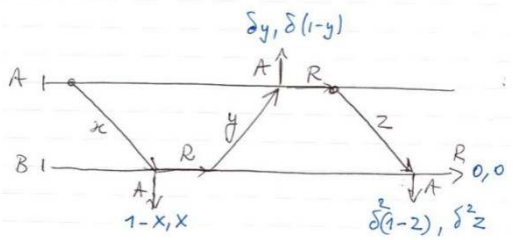
➔ You know upfront that after the second round the game ends.
➔ So in the first round the total would be 1 and in the second round the total would be delta.

- Two rounds ultimatum game: "Pareto Shuffle"

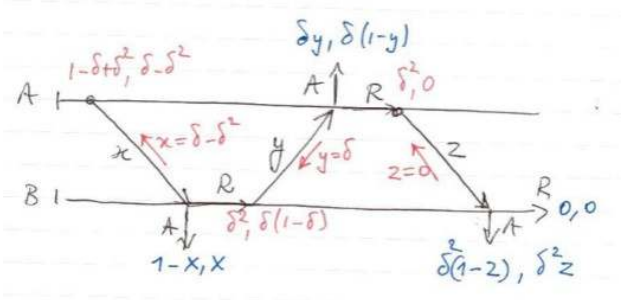


➔ So the optimal solution would be that B takes delta in the first round, because he knows that It cannot get better in the second round.

- Three rounds ultimatum game
  - Room for two counter-offers!

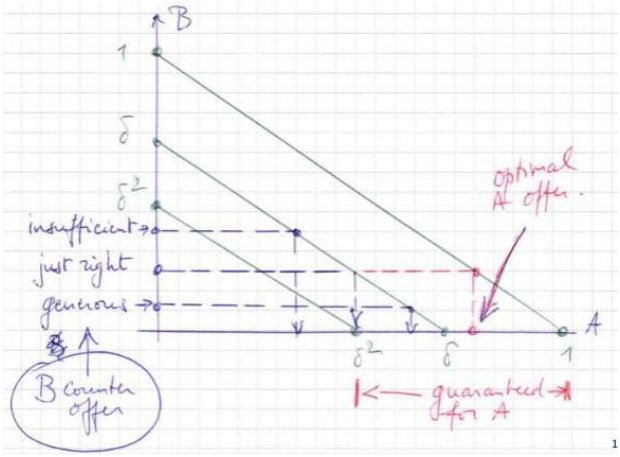


- Three rounds ultimatum game: Backwards induction

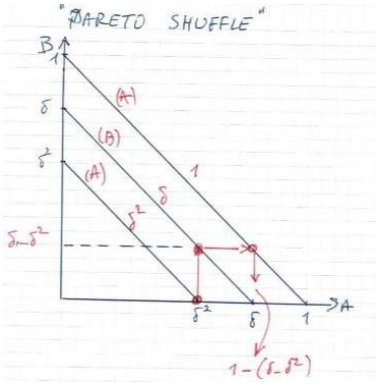


  - **Conclusion:** A proposes split $(1 - \delta + \delta^2, \delta - \delta^2)$ which B accepts.

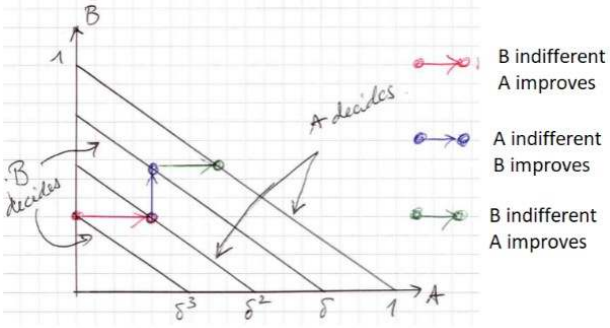
- Pareto shuffle forward



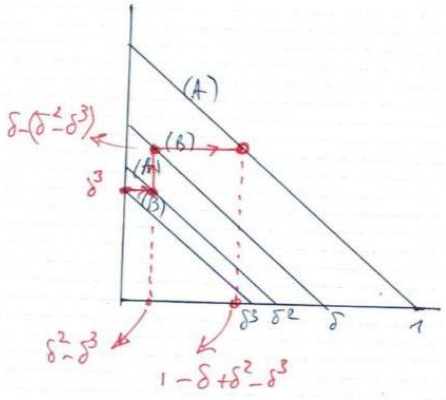
- Three rounds ultimatum game: "Pareto Shuffle"



- Four rounds ultimatum game: backward induction



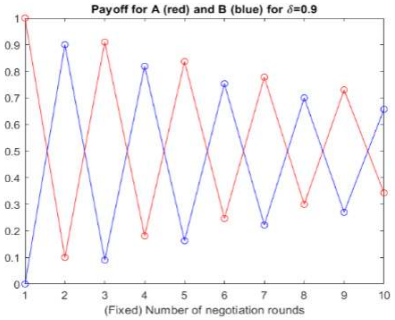
- Four rounds ultimatum game: "Pareto Shuffle"



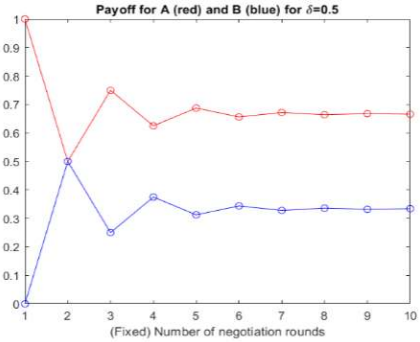
- Alternating offers bargaining
  - Generalizing to negotiation with *n* rounds;
  - When number of negotiation rounds is fixed at *n*, the equilibrium split is given in table below:

payoff for *n* rounds	$n=1$	$n=2$	$n=3$	$n=4$
for A	1	$1-\delta$	$1-\delta+\delta^2$	$1-\delta+\delta^2-\delta^3$
for B	0	δ	$\delta-\delta^2$	$\delta-\delta^2+\delta^3$

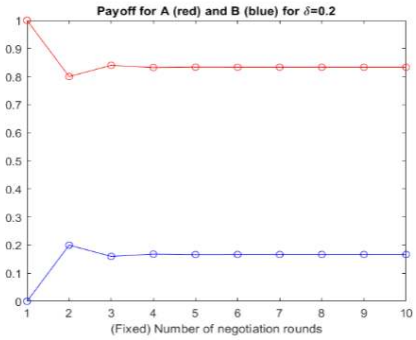
- Alternating offers bargaining (mildly impatient, δ = 0.9)



- Alternating offers bargaining (seriously impatient, δ = 0.5)



- Alternating offers bargaining (very impatient, δ =0.2)



- Mathematical aside (need to remember these equations)

$$\sum_{k=0}^{\infty} x^k = 1 + x + x^2 + x^3 + \ldots = \frac{1}{1-x} \qquad (\text{for } |x| < 1)$$

$$\sum_{k=1}^{\infty} x^k = x + x^2 + x^3 + x^4 \ldots = \frac{x}{1-x} \qquad (\text{for } |x| < 1)$$

$$1 + x + x^2 + \ldots + x^n = \frac{1}{1-x} - x^{n+1}\frac{1}{1-x}$$

$$= \frac{1-x^{n+1}}{1-x}$$

- General conclusions
  - Limit behaviour for pay-offs:

$$A(n) = 1 - \delta + \delta^2 - \ldots \pm \delta^{n-1} = \frac{1-(-\delta)^n}{1-(-\delta)}$$

$$\Longrightarrow \lim_{n\to\infty} A(n) = \frac{1}{1+\delta}$$

$$B(n) = 1 - A(n) \quad \Longrightarrow \quad \lim_{n\to\infty} B(n) = \frac{\delta}{1+\delta}$$

  - **First offer advantage:** lim A(n) ≥ lim B(n)
  - First offer advantage disappears for very patient negotiators:

$$\delta \longrightarrow 1 \quad \Longrightarrow \quad A(n) \searrow 1/2, \quad B(n) \nearrow 1/2;$$

- Rubinstein's Model: Infinite Horizon Bargaining
  - 2 agents, infinite horizon (# rounds **not fixed in advance**);
  - **Time is valuable:** discount factors for both agents (δ_1,δ_2);
  - **Optimal split:**

$$u_1 = \frac{1-\delta_2}{1-\delta_1\delta_2} \qquad \text{and} \qquad u_2 = \frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}$$

  - **Tragedy of bargaining:**
    - The more time matters, the lower the share!
    - E.g. if $\delta_2 \approx 0$, then $u_2 \approx 0$, etc.
  - Notice **first mover's advantage** for $\delta_1 = \delta_2 = \delta$:

$$u_1 = \frac{1-\delta}{1-\delta^2} = \frac{1}{1+\delta} \qquad \text{and} \qquad u_2 = \frac{\delta(1-\delta)}{1-\delta^2} = \frac{\delta}{1+\delta}$$

Repeated Games **(if you do repeated games, it can help to build a reputation)**

- Repeated Games
  - Repeat the **same** one-shot (stage) game (special case of sequential game);
  - At each stage, information about preceding games is known;
  - **Finite number** *n* (known!) of repetitions;
    - Maximize total reward:

$$R_n = \sum_{t=1}^{n} r_t$$

  - **Infinite (unlimited) number** of repetitions
    - Maximize **discounted** total reward:

$$R = \sum_{t=1}^{\infty} \delta^{t-1} r_t \qquad \text{discount factor} \quad 0 < \delta < 1$$

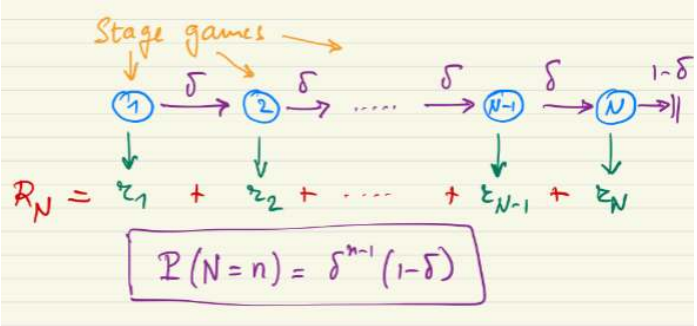
    - Equivalently: ending after **random number** of repetitions.

- Mathematical aside
  - Recall:

$$\sum_{k=0}^{\infty} x^k = 1 + x + x^2 + x^3 + \ldots = \frac{1}{1-x} \qquad (\text{for } |x| < 1)$$

$$\sum_{k=1}^{\infty} x^k = x + x^2 + x^3 + x^4 \ldots = \frac{x}{1-x} \qquad (\text{for } |x| < 1)$$

- Interpretation of discount factor



- Repeated games: interpretation of discount factor
  - After every stage-game there is a
    - Probability 1-δ that game will be ended;
    - Probability δ that game will proceed to next round.
  - Reward *R* now becomes random variable:

$$R_N = \sum_{t=1}^{N} r_t \qquad \text{where} \quad P(N=n) = \delta^{n-1}(1-\delta)$$

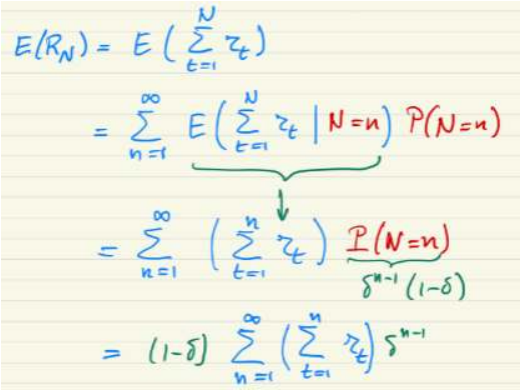
(Assuming at least one stage game is played, hence $N \geq 1$).

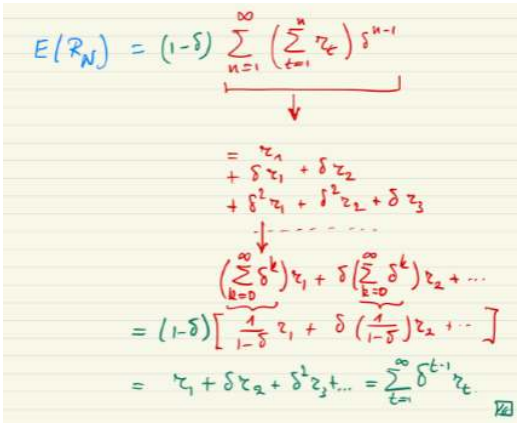
Then:

$$E(R_N) := E\left(\sum_{t=1}^{N} r_t\right) = r_1 + \delta r_2 + \delta^2 r_3 + \ldots = \sum_{t=1}^{\infty} \delta^{t-1} r_t.$$

➔ N is stochastic, just says how many rounds.

- Interpretation of discount factor(not for the exam)





- Example: Repeated Prisoner's dilemma
  - Pay-off matrix for **stage game:**

	$C(oop)$	$D(efect)$
C	3, 3	1, 4
D	4, 1	2, 2

  - For **finite number** of repetitions:
    - Single Nash eq.: Play D-D for each repetition;
    - Can be proved using **backward induction**.
  - For **infinite number** of repetitions:
    - More interesting strategies, including **cooperation**.
    - Examples: Tit-for-Tat, Grim Trigger.

- Grim Trigger Strategy for Repeated Prisoner's Dilemma
  - **Grim Trigger and Tit-for-Tat strategy**
  - **Start by cooperating …**
    - **GT:** continue cooperating, until someone defects; from then onwards, always defect!
    - **TfT:** from then onwards, copy last move of opponent.
  - If both parties play GT, is it rational to defect? Consider player 1:
    - Utility for continued **cooperation:**

$$u_1(C) = 3 + 3\delta + 3\delta^2 + \ldots = \frac{3}{1-\delta}$$

    - Utility for **defection:**

$$u_1(D) = 4 + 2\delta + 2\delta^2 + \ldots = 4 + 2\frac{\delta}{1-\delta}$$

      ➔ See the matrix for prisoner's dilemma above.
      ➔ So cooperation is more beneficial if the number is higher than for the defection.
  - Continued **cooperation is rational** if $u_1(C) > u_1(D)$.

- Grim Trigger Strategy for Repeated Prisoner's Dilemma
  - Continued **cooperation is rational** if $u_1(C) > u_1(D)$:

$$u_1(C) > u_1(D) \iff \frac{3}{1-\delta} = 4 + 2\frac{\delta}{1-\delta}$$

$$\iff \delta > 1/2.$$

  - **Interpretation:** If the players are **sufficiently patient** (i.e. future rewards are sufficiently valuable) then it's **rational to cooperate**.

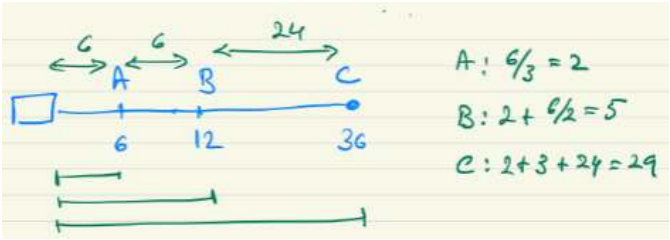
Coalitional Games

- Coalitional Game Theory
  - Basic modelling unit is **group** rather than individual agent.
  - **Transferable** vs. **non-transferable** utility.
  - **Coalitional game with transferable utility** (*N*, *v*):
    - *N* finite set of players:
    - $v : 2^N \rightarrow$ pay-off function (v (∅) = 0).
  - **Fundamental questions:**
    - Which coalitions will form?
    - How should coalition divide its pay-off among its members?

- Classes of coalitional games
  - **Super-additive game ("synergy")**
    - Game (*N* ,*v*) is super-additive iff
      $\forall S, T \subset N : S \cap T = \emptyset \Longrightarrow v(S \cup T) \geq v(S) + v(T).$
    - In particular:
      $v(S \cup i) \geq v(S) + v(i)$ for any $S \subset N \setminus \{i\}$.
  - As a consequence, for super-additive game, the **grand coalition** has the highest pay-off of all coalitional structures:
    $v(N) = v(S \cup S^c) \geq v(S) + v(S^c) \geq v(S).$
  - Therefore focus on a **fair redistribution of total pay-off** among the members of the grand coalition.

- Motivating example: Ways to allocate common benefits
  - Three friends sharing a taxi cab.
  - Three airlines investing in a new runway.

- Worked example: Fair division of taxi fare
  - Alice, Bob and Charlize share a taxi to go home;
  - The individual fares would be A(6), B(12) and C(36);
  - If they share the cab then they only have to pay the fare to the farthest destination (C=36).
  - What would be a fair way to share the fare? → **Common sense solution**

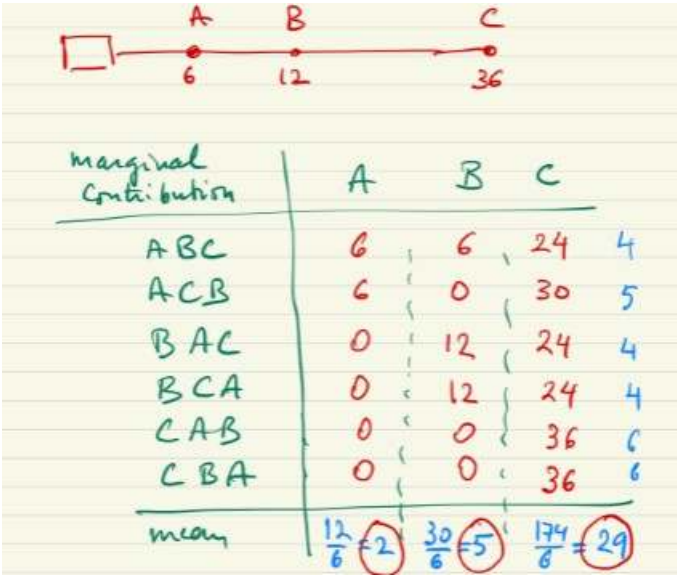


  - Consider a **sequential version** of the problem in which A, B and C arrive in random order, and pay whatever is lacking (i.e. their **marginal contribution**);
  - Permutation ACB indicates that coalition grows as follows:

$$A \rightarrow AC \rightarrow ACB$$

1. When A joins, he pays the fare to his destination: 6
2. When C joins, he pays the remainder to get to C (i.e. 36-6=24);
3. Finally, when B joins, everything is already paid for.

  - Arriving at different orders:

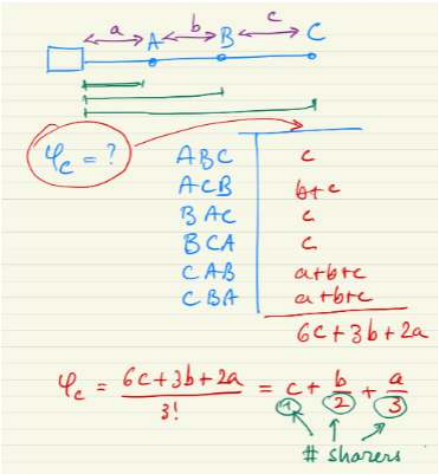


➔ Its about the first who comes, pays the ride for himself. So e.g. if B comes first he pays 12 to get home, but when A comes next, the tip is already paid by B. So A does not have to pay anything anymore and when C comes, he only need to pay the last part 24. But when C comes first, C must pay for the whole trip, therefore 36.

Lecture 7

Further explanation of lecture 6

- Shapley intuition: common sense vs. permutation solution



- Shapley value: Alternative definition
  - **Marginal contribution** only depends on what **precedes** a contributor;
  - **Marginal contribution** of player *i* to subset *S*:

$$\delta_i(S) = v(S \cup i) - v(S)$$

    - ➔ S is subset of the total group agents N.
    - ➔ i is outside the subset S, so how much value does player i give to the subset s, when adding.
  - Shapley value of player *i* : (denoting #*N* = *n*, #*S* = *s*)

$$\varphi_i(N, v) := \frac{1}{n} \sum_{S \subset N \setminus i} \binom{n-1}{s}^{-1} \delta_i(S)$$

- Shapley: amplification
  - We focus on **Shapley value** $\varphi_i(N, v)$ **for agent *i*;**
  - For any existing coalition S not including i, i.e.
    $S \subset N_i := N \setminus i$
    - Some subset that does not include i.
  - We consider the value increment due to i joining:
    $\delta_i(S) = v(S \cup i) - v(S)$
  - The size $s := \#S$ of the possible coalitions S that i joins can range between
    $0 \leq s \leq n-1.$
  - For **fixed coalition size *s*** there are $N_s := \binom{n-1}{s}$ coalitions *S* of that size. (you can pick s variations out of n-1)
  - Hence, the **mean contribution of *I* to existing coalitions *S* of size *s*** is given by:

$$\overline{\Delta}_i(s) := \frac{1}{N_s} \sum_{S:\#S=s} \delta_i(S) = \binom{n-1}{s}^{-1} \sum_{S:\#S=s} \delta_i(S).$$

- Finally, since $0 \le s \le n-1$ we compute the **average over the *n* possible choices of *s*.** This average is the **Shapley value:**

$$\begin{aligned} \varphi_i &:= \frac{1}{n}\sum_{s=0}^{n-1} \overline{\Delta}_i(s) = \frac{1}{n}\sum_{s=0}^{n-1} \binom{n-1}{s}^{-1} \sum_{S:\#S=s} \delta_i(S) \\ &= \frac{1}{n}\sum_{s=0}^{n-1} \sum_{S:\#S=s} \binom{n-1}{s}^{-1} \delta_i(S) \\ &= \frac{1}{n} \sum_{S \subset N\setminus i} \binom{n-1}{s}^{-1} \delta_i(S) \end{aligned}$$

- **Double sum** above is actually **sum over all subsets S ⊂ N \ i** .

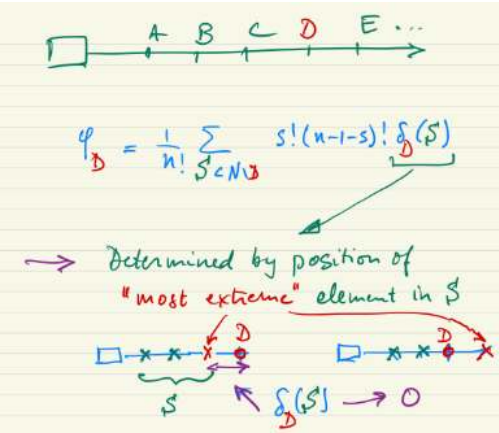
Recall:

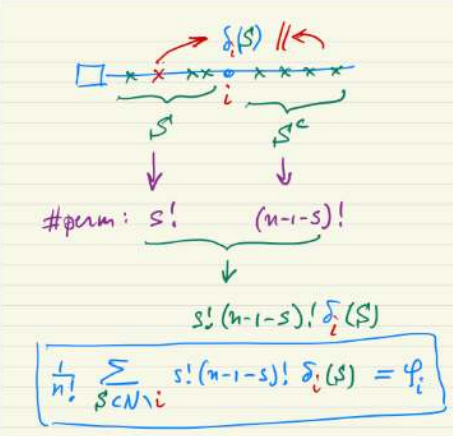
$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad \text{where} \quad n! = n(n-1)(n-2)\dots 3\cdot 2\cdot 1.$$

Hence:

$$\begin{aligned} \varphi_i &= \frac{1}{n} \sum_{S \subset N\setminus i} \binom{n-1}{s}^{-1} \delta_i(S) \\ &= \frac{1}{n} \sum_{S \subset N\setminus i} \frac{s!(n-1-s)!}{(n-1)!} \delta_i(S) \\ &= \frac{1}{n!} \sum_{S \subset N\setminus i} s!(n-1-s)!\, \delta_i(S) \end{aligned}$$

- Shapley: equivalence of definitions



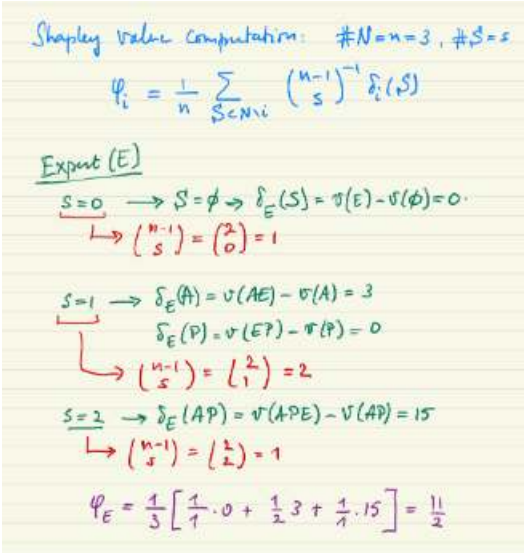


- Shapley's Axioms: Some useful terminology
  - Players *i* and *j* are **interchangeable** if their contributions to every coalition (subset) *S* is exactly the same:
    $\forall S \subset N \setminus \{i,j\}: \quad v(S \cup i) = v(S \cup j)$
  - A player *i* is a **dummy player** if the amount he contributes to any coalition is exactly the amount he's able to achieve alone:
    $\forall S \subset N \setminus \{i\}: v(S \cup i) = v(S) + v(i)$

- Shapley's Axioms
  - **Symmetry:** if *i* and *j* are **interchangeable** then:
    $\psi_i(N, v) = \psi_j(N, v).$
  - **Dummy Player:** will only get what he can achieve on his own:
    $\psi_i(N, v) = v(i).$
  - **Additivity:** consider two games $G_1 = (N, v)$, $G_2 = (N, w)$ and assume that we play G_1 with probability p and G_2 with probability $q = 1 - p$. Then
    $\psi_i(N, pv + qw) = p\psi_i(N, v) + q\psi_i(N, v_2)$

- Shapley's theorem
  - Given a coalitional game (N, v), the **Shapley values** $\varphi_i, i = 1, \ldots, n$ specifies the **unique distribution** of the total value $v(N)$ that is both:
    - **Efficient,** i.e. $\sum_i \varphi_i = v(N)$
    - **Satisfies Shapley's axioms**, i.e. *Symmetry, Dummy Player* and *Additivity*.

- Shapley value: worked example
  - An AI expert (E) developed a powerful new algorithm. However, in order to implement his ideas, he needs to create a start-up and hire a programmer (P) for 2 years. An angel investor (A) provides funding. The value that each coalition of these three stakeholders (E, P, A) can generate satisfies the following rules:
    - Without both investor and expert, no value can be generated.
    - If he has no assistance from a programmer, the expert's value equals 3, but if he can delegate the programming and focus on R&D, his value rises to 10.
    - The value created by the programmer is 5. This is in addition to the rise in value of the expert.
  - The start-up is sold to a large software company for serious money.
  - **How to split this money fairly among the three stakeholders?**
  - Shapley value: Method 1



- Shapley value: Method 2 (permutation approach/ 6 permutation approaches in this example)

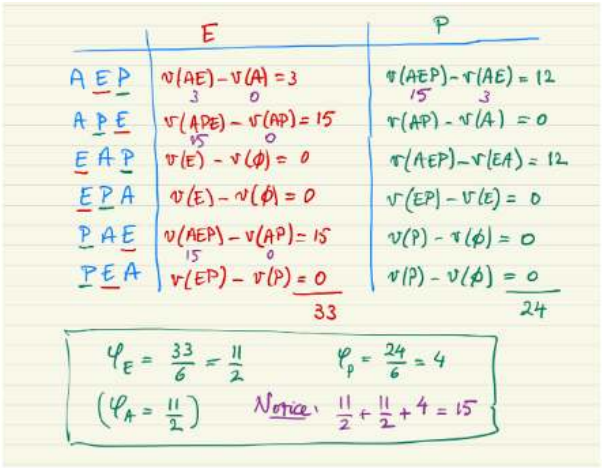


Figure: Notice distribution is efficient: $\varphi_E + \varphi_A + \varphi_P = 15 = v(N)$

- ➔ Remember divide it by n! (factorial), means in this example 3! = 6.
- ➔ So this example goes as follows: focusing on the engineer, means looking in the E column and focusing on the programmer, means looking at the P column. Regarding the E column you see e.g. that AEP means that you focus until E. EPA, you get an empty set as the engineer is the first you focus on.

Exploration versus Exploitation (first chapter of reinforcement learning part)

- Recurring themes in (sequential) decision making (vocabulary:)
  - States, actions, transitions, policy, value functions;
  - Back-up, optimisation (planning and searching).

- Sequential decision making

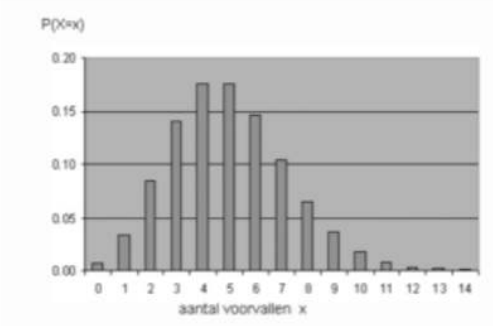
- In **sequential decision making** an agent tries to solve a sequential control problem by directly interacting with an unknown environment.
- **Learning by trial and error** Agent tries out actions to learn about their consequences.
- **Not supervised:** No examples of correct or incorrect behaviour; instead only **rewards** for actions tries.
- **Active learning:** agent has partial control over what data it will obtain for learning.
- **On-line learning:** it must maximize performance during learning, not afterwards.

Preliminaries: Recap of Probability Theory

- Stochastic Variables
  - Stochastic (or random) variables: abstract model the idea of a **randomly determined numerical outcome**;
  - Formally;

    $X : \Omega(\text{"outcomes"}) \longrightarrow \mathbb{R}$

    ➔ Important that e.g. drawing an outcome of an infinte set, is to put the ball back into the set when drawing an outcome again. E.g. if you want to draw the probability distribution from this stochastic var.
  - In an **experiment:** one draws ball (outcome ”ω”) and reads number (x) on the ball;
- Probability Distribution
  - Discrete set of outcome values:
    - $X : \Omega \longrightarrow \{x_1, x_2, x_3 \ldots\}$
    - $P(X = x_k) = p_k \qquad (p_k \geq 0)$
    - $\forall k : \; p_k \geq 0 \qquad \text{and} \qquad \sum_k p_k = 1$



  - continuous outcome values X : Ω →
    - Characterized by density function $f(x)$ such that

      $$P(a \leq X \leq b) = \int_a^b f(x)\,dx$$

    - Informally: $P(x \leq X < x + dx) = f(x)\,dx$
    - $\forall x : \; f(x) \geq 0 \qquad \text{and} \qquad \int_{-\infty}^{+\infty} f(x)\,dx = 1$

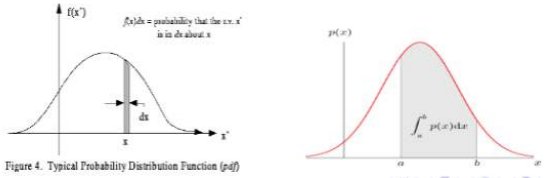


Figure 4. Typical Probability Distribution Function (pdf)

- Expectation Operator and Expected Value
  - Expectation operator (E(X)) is the mean of e.g. the column with all the points.

$$E(X) = \sum_k x_k P(X = x_k) = \sum_k x_k p_k \qquad (=: \mu)$$

$$E(X) = \int x f(x)\, dx \qquad (=: \mu)$$

$$\begin{aligned} Var(X) &= E\left((X - EX)^2\right) = E(X - \mu)^2 \\ &= \sum_k (x_k - \mu)^2 p_k \qquad \text{(discrete prob)} \\ &= \int (x - \mu)^2 f(x)\, dx \qquad \text{(continuous prob)} \end{aligned}$$

- Expectation and Variance of Linear Combination

$$E(aX + bY) = aE(X) + bE(Y)$$

$$Var(aX + bY) = a^2 Var(X) + b^2 Var(Y) + 2ab\, Cov(X, Y)$$

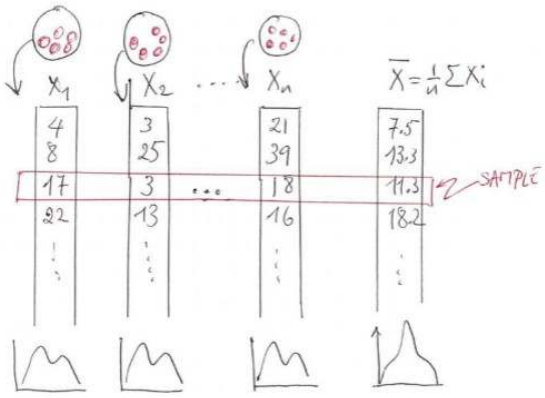
In particular:

$$E(aX) = a\,E(X) \qquad Var(aX) = a^2 Var(X)$$

If X, Y independent: $Var(X \pm Y) = Var(X) + Var(Y)$

- Independence
  - Independent events:
    $P(A \,\&\, B) = P(A)P(B)$
  - Independent (discrete) stochastic variables:
    $P(X = a \,\&\, Y = b) = P(X = a)P(Y = b)$

- Sample of independent, identically distributed (i.i.d.) random variables (rvs)
  - Lotto machines drawn below:



  - ➔ For each of these balls the histogram would be the same, because the distribution in these balls are the same → therefore it is called identically distributed, but they are independent, because you can not predict what is going to happen in the second machine.
  - ➔ If you draw the sample mean of all distributions, it would look different → it would be more peaked.
  - ➔ At the end how more variables, the more it looks like a normal distribution.

- Expectation and Variance of Sample Mean
  - Let X_1, X_2, ..., X_n be sample of size n from distribution (population) with mean μ and variance σ^2:
    - Sample mean:

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

    - Mean and variance of sample mean:

$$E(\overline{X}_n) = \mu \qquad Var(\overline{X}_n) = \frac{\sigma^2}{n}$$

    - (Asymptotic) distribution of sample mean (Central Limit Theorem, CLT):

$$\overline{X}_n \longrightarrow N(\mu, \sigma^2/n)$$

- Conditional Probability and Independence (56:56)
  - **Conditional Probability**

$$P(A \mid B) := \frac{P(A \cap B)}{P(B)}$$

  - **Independence**

$$P(A \cap B) = P(A)P(B \mid A) = P(B)P(A \mid B)$$

$$\text{If } A, B \text{ are independent:} \quad P(A \mid B) = P(A)$$

  - **Conditional independence**

$$P(A \cap B \mid C) = P(A \mid C)P(B \mid C)$$

- Bayes' Rule
  - Bayes' Rule

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

  - Law of total probability:
    - Let $B_1, B_2, \ldots, B_n$ partition of outcome space:

$$P(A) = \sum_{i=1}^{n} P(A \cap B_i) = \sum_{i=1}^{n} P(A \mid B_i)P(B_i)$$

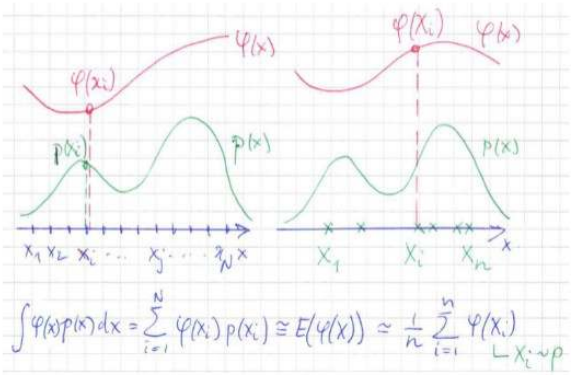
- Monte Carlo: Sample-based computation
  - Let $X_1, X_2, \ldots, X_n$ be a sample (i.i.d) from given (discrete/continuous) distribution;

$$EX = \left\{ \begin{array}{c} \sum_{k=0}^{\infty} x_k P(X = x_k) = \sum_{k=0}^{\infty} x_k p_k \\ \int x f(x) dx \end{array} \right\} \approx \frac{1}{n} \sum_{i=1}^{n} X_i$$

$$E\varphi(X) = \left\{ \begin{array}{c} \sum_{k=0}^{\infty} \varphi(x_k) p_k \\ \int \varphi(x) f(x) dx \end{array} \right\} \approx \frac{1}{n} \sum_{i=1}^{n} \varphi(X_i)$$

➔ Below is with adding the probability.

- Monte Carlo approximation → using sampling graph/ equation on the right.



➔ So there are two ways to calculate the integral. The first way by multiplying it with the probability, the second only the above graph, because the sampling already take care about how the distribution is.

- Kullback-Leibler Divergence
  - Let *p*, *q* be probability distributions (cont. or discrete).
  - **Kullback-Leibler divergence**

$$KL(p||q) := \int p(x) \log \frac{p(x)}{q(x)} dx \geq 0$$

$$KL(p||q) := \sum_k p_k \log \frac{p_k}{q_k} \geq 0$$

    - Divergence is not symmetric:

$$KL(p||q) \neq KL(q||p)$$

    - Sample $X_1, X_2, \ldots, X_n \sim p$:

$$KL(p||q) \approx \frac{1}{n} \sum_{i=1}^{n} \log \frac{p(X_i)}{q(X_i)}$$

➔ KL divergency is a way to see whether distribution are similar → if the KL divergency is 0 than the distributions are the same. Defined by the above integral.

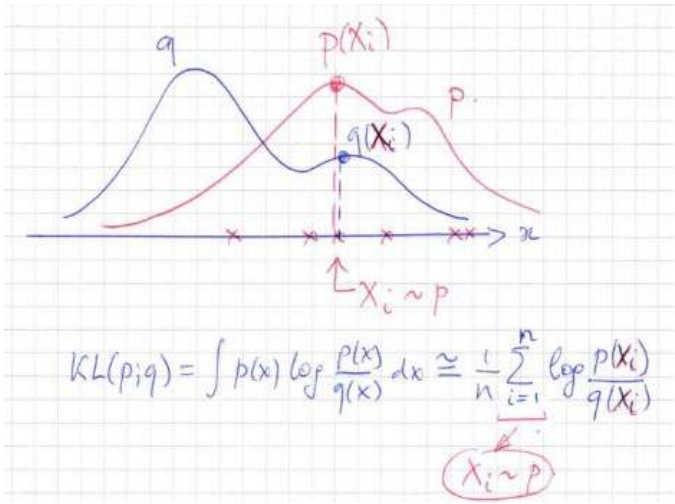
- Kullback-Leibler divergence is always positive
  - Since $X_i \sim p$ . . .
  - Typically:

$$\frac{p(X_i)}{q(X_i)} \geq 1$$

  - And hence, typically:

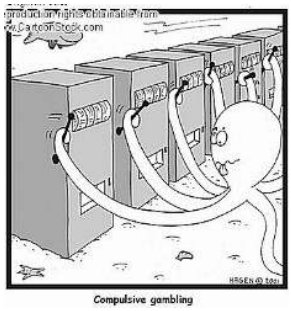
$$\log \frac{p(X_i)}{q(X_i)} \geq 0$$

- That is why KL(p||q) ≥ 0

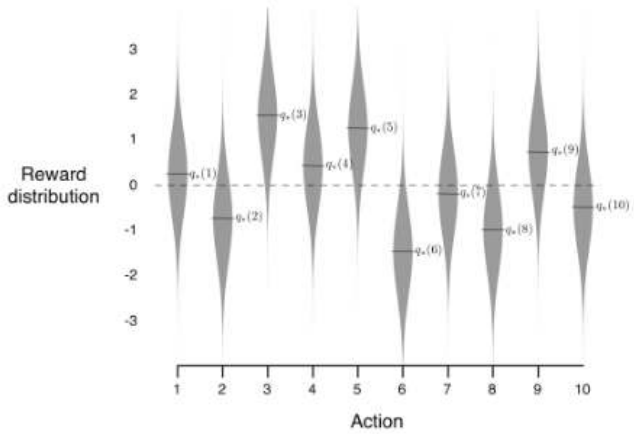


K-Armed Bandit Problem

- Prototypical problem: *K*-armed bandit problem
  - Sit before a slot machine (bandit) with many arms.
  - Each arm has an unknown stochastic pay-off.
  - Goal is to maximize cumulative pay-off over some period.



- K-armed bandit: example
  - Visualisation of the **reward distribution for each arm** (*k*=10). In this case **arm 3** would yield the **optimal** expected reward.



➔ Exploitation vs. exploration, therefore means that with exploitation you can have a stochastic value of one arm that is higher than the other, but it does not mean that its mean is higher than the other (exploration).

- Formalizing the *k*-armed bandit problem
  - There are *k* **actions** available at each timestep;
  - After action *t*, the agent receives (stochastic) reward $R_t \sim f_{at}$.
  - In a **finite-horizon** problem, the agent tries to maximize its **total reward** over *T* actions:
    $\sum_{t=1}^{T} R_t$
  - In an **infinite-horizon** problem, the agent tries to maximize its **discounted total reward**: (otherwise, it would be infinite)
    $\sum_{t=0}^{\infty} \gamma^t R_t$ where $0 < \gamma < 1$
  - The **discount factor** γ can be interpreted as the probability of the game continuing after each step.

- Exploration and exploitation
  - The agent's ability to get reward in the future depends on what it knows about the arms.
    - It must **explore** the arms in order to **learn** about them and improve its chances of getting future reward;
    - It must **use what it already knows in order to maximize** its total reward; Thus it must **exploit** by pulling the arms it expects to give the largest rewards;

- Balancing exploration and exploitation
  - The main challenge in a k -armed bandit is how to **balance the competing needs of exploration and exploitation**.
  - If the horizon is finite, exploration should decrease as the horizon gets closer.
  - If the horizon is infinite but γ < 1, exploration should decrease as the agent's uncertainty about expected rewards goes down.

- Action-value methods
  - **Formal Setup:**

- Each **arm (i.e. action a = 1,...,k)** generates **stochastic reward *R*** sampled from **probability distribution f_a** with **unknown mean *q(a)***, hence: *q(a)* := $E_{fa}(R)$.
- **Action-value:** *q(a)* can be thought of as the (average) **value** (quantity) generated by **taking action a**;
  - Compare, to *state-action value* $q(s, a)$ in RL.
- We use the **sample average $Q_t(a)$** is an **estimate of *q(a)***. Specifically, if action *a* has been chosen k_a times, yielding rewards $R_1, R_2, ..., R_{ka}$, then:

$$Q_t(a) = \frac{1}{k_a} \sum_{i=1}^{k_a} R_i$$

➔ K_a is the amount of times you pull the arm.

- Asymptotically:

$$Q_t(a) \longrightarrow q(a) \quad \text{as} \quad t, k_a \longrightarrow \infty.$$

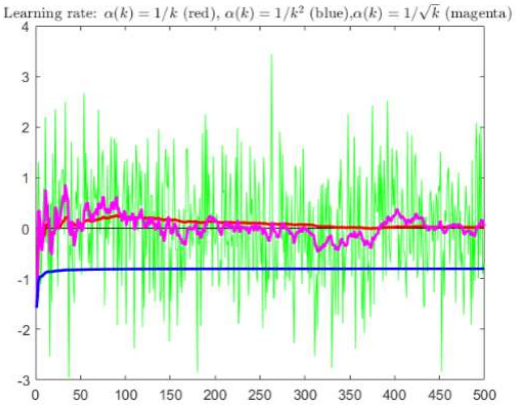
- Incremental implementation if *a* is selected at time *t* + 1:

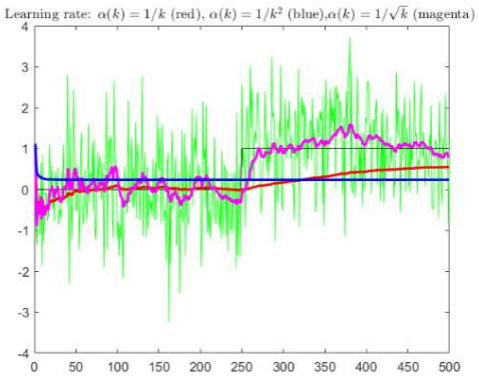
$$\begin{aligned} Q_{t+1}(a) &= \frac{k_a Q_t(a) + R_{t+1}}{k_a + 1} \\ &= \frac{k_a}{k_a + 1} Q_t(a) + \frac{1}{k_a + 1} R_{t+1} \\ &= Q_t(a) + \frac{1}{k_a + 1} \left[R_{t+1} - Q_t(a) \right] \end{aligned}$$

- Example of an **update rule** for estimates:

$$NewEst \leftarrow OldEst + LearningRate[NewData - OldEst]$$

- Note on Learning Rate





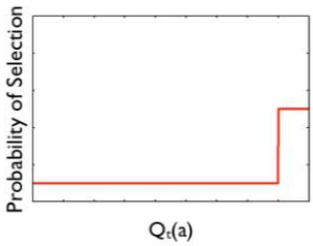
- Defining exploration vs. exploitation
  - When using action-value methods, exploration and exploitation are easy to define.
  - **Exploiting** means taking the **greedy** action:

$$a^* = \arg\max_a Q_t(a)$$

  - **Exploring** means taking any other action:

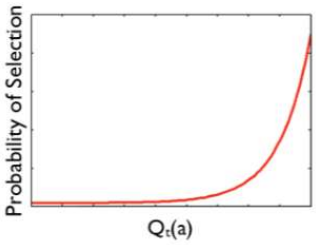
- Frequently used exploration strategies:
  1. ε-greedy
  2. Soft-max

- Epsilon-greedy exploration
  - In ε-**greedy** exploration, the agent selects a random action with probability ε, and the greedy action otherwise.



- Soft-max exploration
  - In **soft-max** exploration, the agent chooses actions according to a **Boltzmann** distribution.

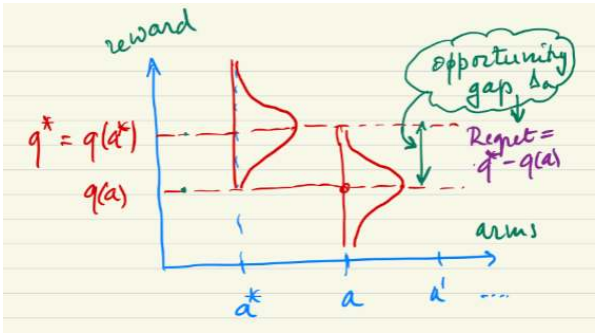
$$p(a) = \frac{e^{Q(a)/\tau}}{\sum_{a'} e^{Q(a')/\tau}}$$



➔ So the difference between the greedy and the soft-max is that the softmax, chooses e.g. if it cannot choose the best q(t), it chooses, the 2nd best one. Whereas for the greedy it chooses any other q(t).

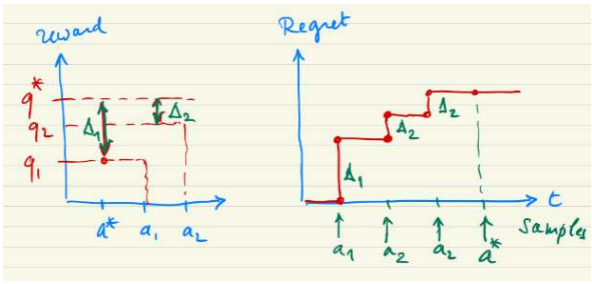
- Optimistic vs. Realistic Initialization
  - How to ensure that **every arm is sampled at least once** (preferably: at **beginning of exploration**!)
    - In **optimistic initialization**, the agent initializes its action-value estimates higher than the largest possible reward.
    - The agent always selects the **greedy action**.
    - Rewards are always disappointing, directing the agent to the **un-explored arms**.

- Quantifying Performance: Minimising Total Regret

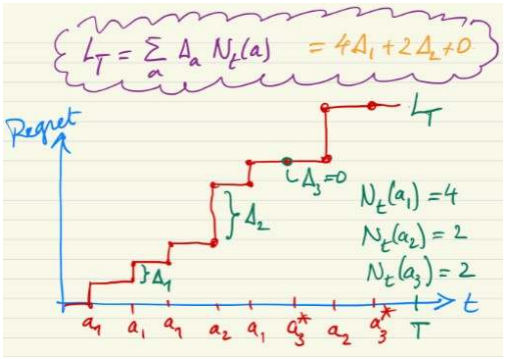


➔ A* is the best arm and q* is the best mean.

➔ Take any other arm e.g. a, the difference between these distributions (the means), is called the **opportunity gap** (what you could have had if you played the optimal one, so in that respect it is a **regret**: delta sign).



- Total regret



➔ L_T = the summation of the gap for a certain arm plus the times you picked that arm.

- Quantifying Performance: Minimising Total Regret
  - **Optimal mean reward value and opportunity gap**:

$$q^* = q(a^*) = \max_a q(a) \qquad \Delta_a := q^* - q(a)$$

  - **Regret (opportunity loss) when taking action *i***:

$$q^* - q(a_i)$$

  - **Total regret** up to time T: (important to realize is that this is the overall loss:)

$$L_T = \sum_{t=1}^{T} \Delta_{A_i} = \sum_a \Delta_a N_T(a).$$

  - **Expected total regret** (up to time T):

$$\ell_T := E(L_T) = \sum_a \Delta_a \, E N_T(a)$$

- Expected total regret for ε-greedy
  - Expected total regret:

$$\ell_t = \sum_a \Delta_a \, EN_t(a).$$

  - For ε-greedy (when we have found optimal a∗):

$$EN_t(a) = \begin{cases} (1-\epsilon)t & \text{if } a = a^*, \Delta_a = 0 \\ \frac{\epsilon}{k-1}t & \text{if } a \neq a^*, , \Delta_a > 0 \end{cases}$$

Hence: $\ell_t = (\epsilon\overline{\Delta})\, t$ where $\overline{\Delta} = \frac{1}{k-1}\sum_{a \neq a^*} \Delta_a$

  - For **constant exploration (ε) total regret grows <u>linearly</u> in *t*** (time).

- Can we do better? Lai-Robbins
  - What if we **reduce exploration ε over time**?
  - Asymptotically, **total regret** is **at least logarithmic** in number of steps:

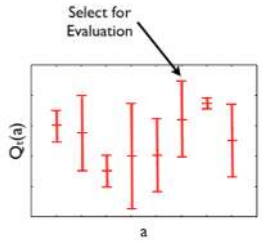
$$\ell_t \geq A \log t \qquad \text{as } t \longrightarrow \infty$$

  - Where:

$$A = \sum_{a:\Delta_a > 0} \frac{\Delta_a}{KL(f_a; f_a^*)}$$

    - KL(f ; g) Kullback-Leibler divergence;
    - What exploration schemes give rise to this performance? Regret is going to grow less harder.

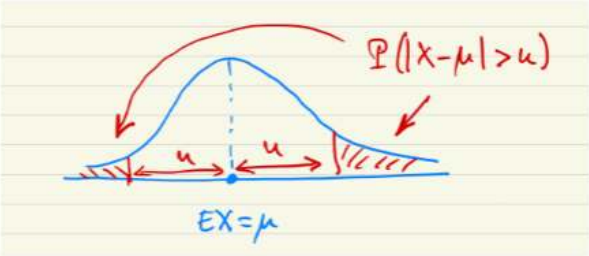
- UCB: Upper confidence bounds
  - **UCB: Optimism in the face of uncertainty!**
  - Neither ε-greedy nor soft-max consider uncertainty in action-value estimates.
  - Goal of exploration: reduce uncertainty.
  - So **focus exploration on most uncertain actions**.
  - Compute confidence intervals for each action.
  - Always take **action with highest upper bound**.



Aside: Concentration inequalities

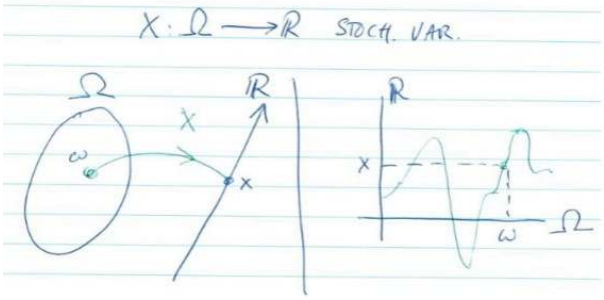
- Aside: Concentration Inequalities
  - **Concentration inequalities** provide bounds on how much a random variable *X* can deviate from some value (typically, its expected value *E(X)*):

$$P(|X - E(X)| > u) = \quad ??$$



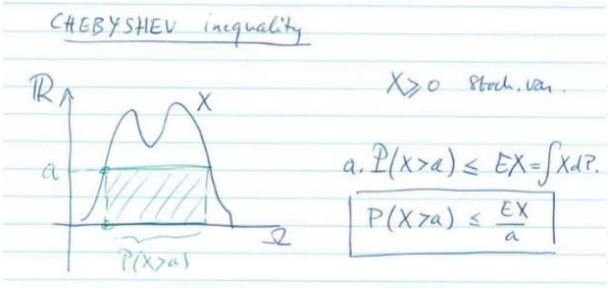
- **General but weak bounds:** Markov, Chebychev.
- For **sums (or means) of independent rv's;** stronger bounds!
  - Hoeffding, Azuma, …

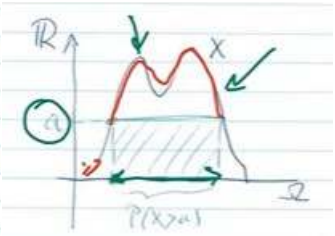
- Markov – Chebychev (1)



➔ Ω is a stochastic variable.

- Markov – Chebychev (2)

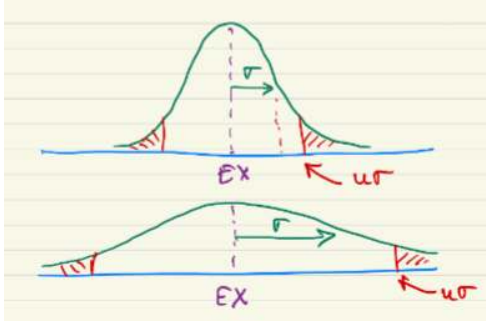




➔ The red indicates the area where the x > a. the square indicates the area of the probability x>a. Thus the square indicates P(x>a).

➔ EX is the mean. Thus p(x>a) <= EX/a

- Markov – Chebychev (3)
  - **Standard deviation σ** provides natural **yard stick (unit-length)**!
  - So it is natural to look at **standardized deviation** |X −μ|/σ, i.e. deviation |X −μ| relative to σ.



EX is the mean.

- For Y ≥ 0 we have:

$$P(Y > u) \leq \frac{EY}{u}$$

- Now take

$$Y = \frac{|X - \mu|}{\sigma} \quad \text{then} \quad EY^2 = 1$$

$$P\left(\frac{|X - \mu|}{\sigma} > u\right) = P(Y > u) = P(Y^2 > u^2) \quad \leq \frac{1}{u^2}$$

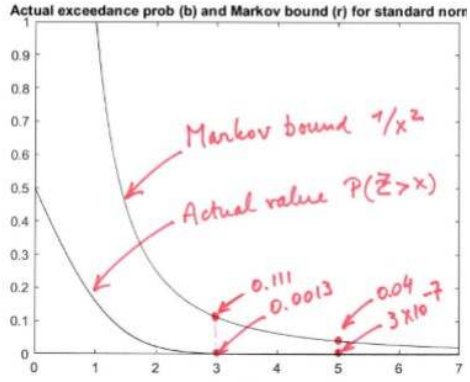
- Markov-Chebychev inequality:

$$P\left(\frac{|X - \mu|}{\sigma} > u\right) \leq \frac{EY^2}{u^2} = \frac{1}{u^2}$$

➔ So calculating the probability that a stochastic variable extend a specific value (in this case u).

- Markov – Chebychev (4)
  - The Markov – Chebychev bound is not very tight!

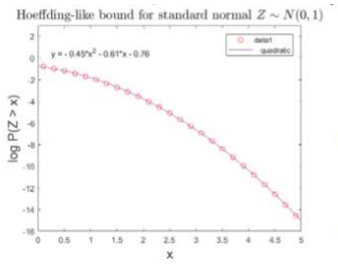
$$\text{For } Z \sim N(0,1): \quad P(Z > x) \leq 1/x^2$$



➔ The bound is $1/x^2$.

➔ So if you are question what is the probability that a stochastic variable exceeds 3 than the Markov bound tells you that it is 0.111. Therefore, the Markov bound is not very tight → does not give you very accurate information.

- Hoeffding inequality: Numerical experiment (Skipped in the lecture)
  - From **numerical experiments** for Z ~N (0,1) we **observe**



$$\log P(Z > x) \approx -\alpha x^2$$

$$\log P(\sigma Z > \sigma x) \approx -\alpha x^2$$

$$\log P(\sigma Z > u) \approx -\alpha u^2/\sigma^2$$

$$P(\sigma Z > u) \approx e^{-\alpha u^2/\sigma^2}$$

$$P\left(\frac{\sigma}{\sqrt{n}} Z > u\right) \approx e^{-\alpha n u^2/\sigma^2}$$

$$\text{Fron CLT:} \quad \overline{X}_n \sim N(\mu, \sigma^2/n) \implies P(\overline{X}_n > \mu + u) \approx e^{-nu^2/2\sigma^2}$$

- Hoeffding inequality for sum of bounded rv's

- **Hoeffding's inequality**
  - Let $X_1, X_2, \ldots, X_n$ be i.i.d. (bounded) random variables such that $c \leq X_i \leq d$ (where $L = d - c < \infty$) with mean $\mu := E(X_i)$.
  - Consider the sample mean:

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

  - Then the **exceedance probability** is **exponentially bounded**:

$$P(\overline{X}_n \geq \mu + u) = P(\mu \leq \overline{X}_n - u) \leq e^{-2nu^2/L^2}$$

    - ➔ μ is the actual mean and u is some safety criteria or so. So the probability that the sample mean exceeds the actual mean + some safety criteria.
    - ➔ **This equation is important to understand!**
    - ➔ If u is increased (means interval is larger) than the exponential is going down.
    - ➔ Increasing n means also that the exponential is going down.
  - And similarly

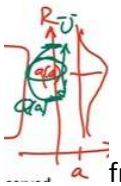
$$P(\overline{X} \leq \mu - u) = P(\mu \geq \overline{X} + u) \leq e^{-2nu^2/L^2},$$

- UCB: Applying Hoeffding to Upper Confidence Bounds
  - **Hoeffding's inequality**

$$P(\overline{X} \geq \mu + u) = P(\mu \leq \overline{X} - u) \leq e^{-2nu^2/L^2}$$
$$P(\overline{X} \leq \mu - u) = P(\mu \geq \overline{X} + u) \leq e^{-2nu^2/L^2},$$

  - **Apply to estimation of bandit reward**
    - How does **unknown real mean** $q(a)$ be estimated by **observed sample mean** $Q_t(a)$?

$$P\left(q(a) > Q_t(a) + U_t(a)\right) \leq \underbrace{e^{-2N_t(a)U_t(a)^2/L^2}}_{:=p(t)}$$



      - ➔ U_t is the following: ..... from Q_t to end/begin of distribution.
      - ➔ N is the number of times you pulled the arm. U is unknown, thus compute.
  - Hoeffding's upper bound:

$$P\left(q(a) > Q_t(a) + U_t(a)\right) \leq e^{-2N_t(a)U_t(a)^2/L^2} = p(t)$$

  - **Given exceedance probability *p(t)***, solve for **upper limit $U_t(a)$:**

$$U_t(a) = c\sqrt{\frac{-\log p(t)}{N_t(a)}}, \qquad \text{where} \quad c = L/\sqrt{2}.$$

  - Now, let **exceedance prob. go to zero** over time: e.g. $p(t) = t^{-1}$;

$$U_t(a) = c\sqrt{\frac{\log t}{N_t(a)}}.$$

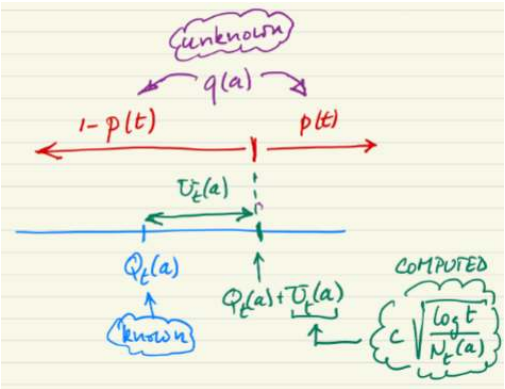
    - ➔ So u means uncertainty.

- UCB1-Algorithm: Upper confidence bounds Optimism in the face of uncertainty!
  - **UCB1-Algorithm**

$$a^*_{t+1} = \arg\max_a \left(Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right)$$

  - **DON'T** choose the arm that has performed best sofar, but…
  - **DO** choose the one that could **reasonably** perform best in the future!
  - **Optimism in the face of uncertainty!**

- UCB: Applying Hoeffding of UCB

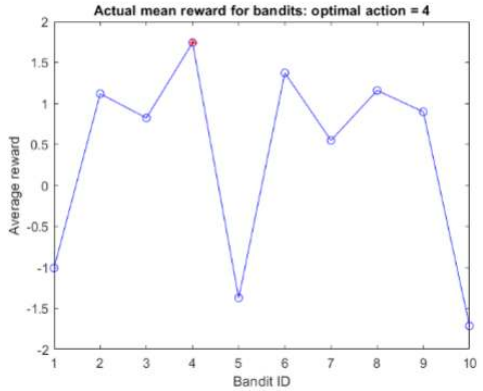


- UCB1-Algorithm

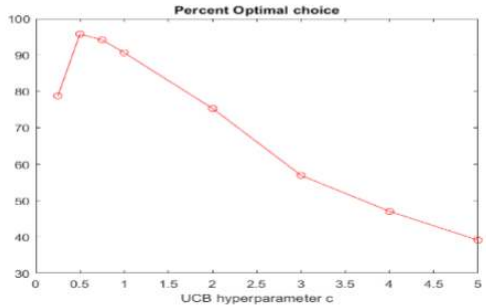
$$a^*_{t+1} = \arg\max_a \left(Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right)$$

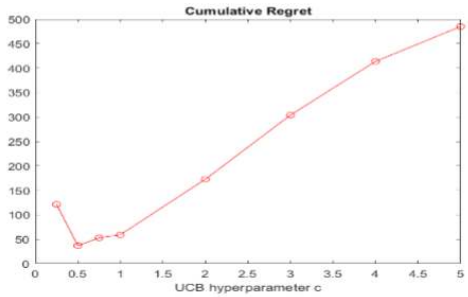
  - The parameter c is some tuneable width parameter;
  - Every time action a is sampled, $N_t(a)$ increases, hence $U_t(a)$ decreases;
  - Every time a is sampled, the UCB for the *other* actions increases (since t increases); **Every action is sampled eventually**!
  - UCB1 algorithm **achieves logarithmic asymptotic total regret**!
  - **Take home message** (B. Christian & T. Griffiths): **In the long run, optimism is the best prevention for regret!**

- UCB for k-bandit problem

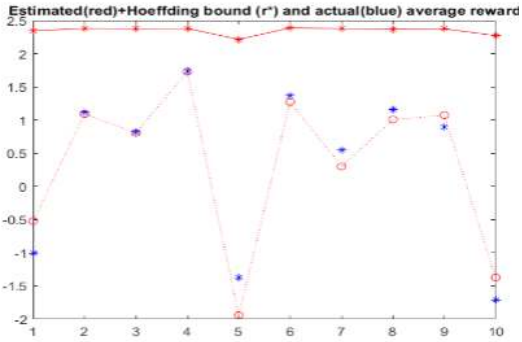


- Duration of game (number of arms pulled): T = 1000.
- Averaged over 10 experiments (same k-bandit problem).

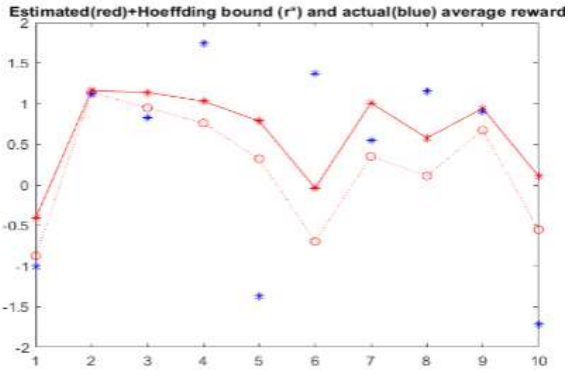




- UCB: Influence of hyper-parameter c = 5 (large)



  - C large → changes in UCB are inflated;
  - Max UCB changes more frequently, **more fickle exploration**.
- UCB: Influence of hyper-parameter: c = 0.5 (small);



  - C small → changes in UCB are tempered;
  - Max UCB changes less frequently, hence **slower exploration**.

- UCB Experiment: Evolution of Total Regret
  - UCB algorithm applied to k = 10, c = 1, averaged over 100 experiments

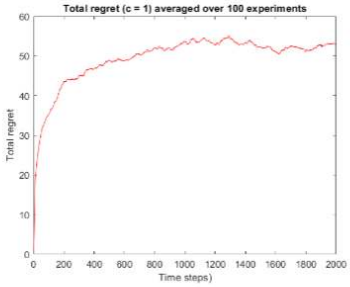


Figure: $\ell_t \sim \log(t)$ where t = number of timesteps

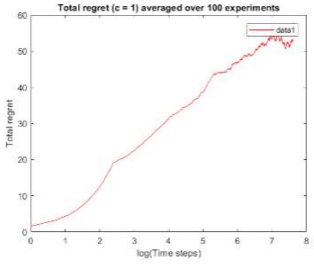
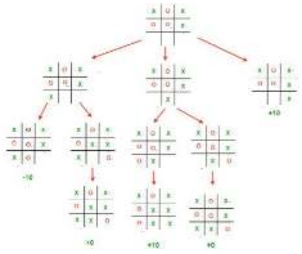


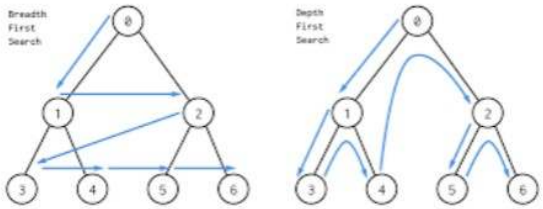
Figure: Recalibrating time-axis as $\log(t)$ yields roughly linear evolution for total regret ℓ_t (confirming $\ell_t \sim A\log(t)$).

Monte Carlo Tree Search (MCTS)

- Tree Search
  - **Tree search:** generic problem underlying many AI tasks:

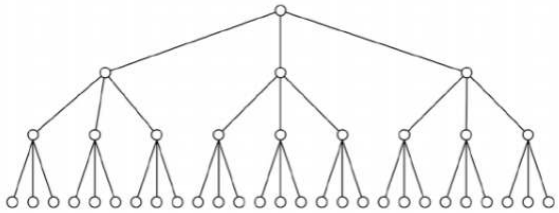


- Tree Search Algorithms
  - Uninformed search: breadth or depth first search;



  - *A** **search**: introduce heuristics that *inform search*;
  - **Minimax search** (2 player adversarial) with αβ pruning; Remove nodes outside optimal window;

- Tree Search: Branching factor and depth
  - **Difficulty:** Exponentially growth of Size of search space grows exponentially: $S \sim b^d$



- Monte Carlo Tree Search (MCTS)
  - **MCTS: main idea**
    - Decision-time planning.
    - **Prioritise** computational budget:
      - **No systematic scan** of all nodes.
      - Primarily **focus on nodes that look promising**.
    - Balance **exploration and exploitation**, e.g.:
      - ε-greedy, or UCB1 (UCT: Upper Confidence bounds for Trees).
    - MCTS is an important ingredient in many of the recent success stories in game AI (e.g. AlphaZero GO);

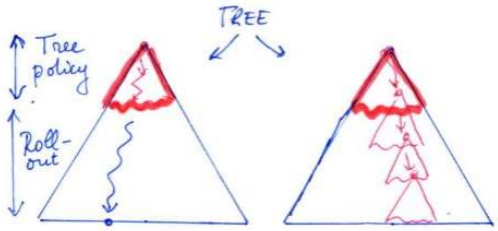
- MCTS: Tree policy vs. roll-out
- MCTS: Amplification of loop
    - **Tree traversal and node selection:**
        - Use tree policy to construct path from root to (most) promising "tree policy (aka snowcap)" leaf node;
        - **Tree policy:** always choose child node with best (but finite) UCB-value:

$$UCB(node_i) = \bar{x}_i + c\sqrt{\frac{\log N}{n_i}}$$

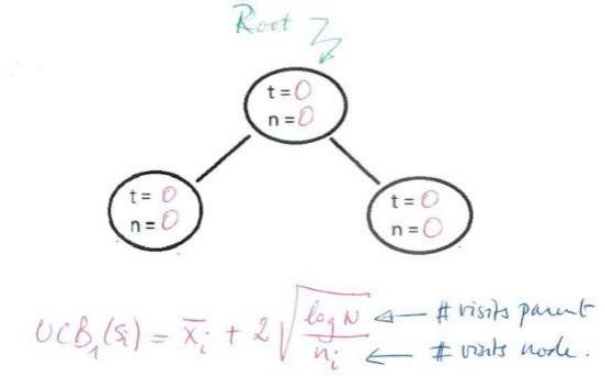
$\bar{x}_i$: mean node value; n_i: #visits of node i; N #visits parent;

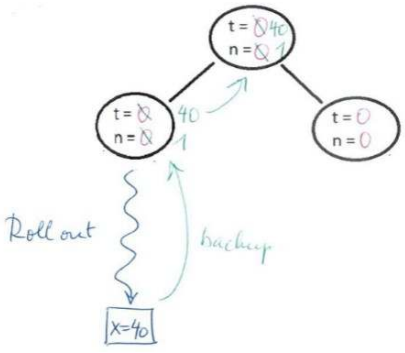
        - Proceed till **tree policy leaf node** has been reached: this node has children that haven't been explored.
    - **Expansion** of selected (tree policy) leaf node, i.e. randomly **pick unexplored child (action)**.
    - **Simulation** based on **roll-out policy**.
        - Roll-out: *random* or *fast heuristic* (roll-out states not stored!);
    - **Backup:** update values along **tree traversal** path;

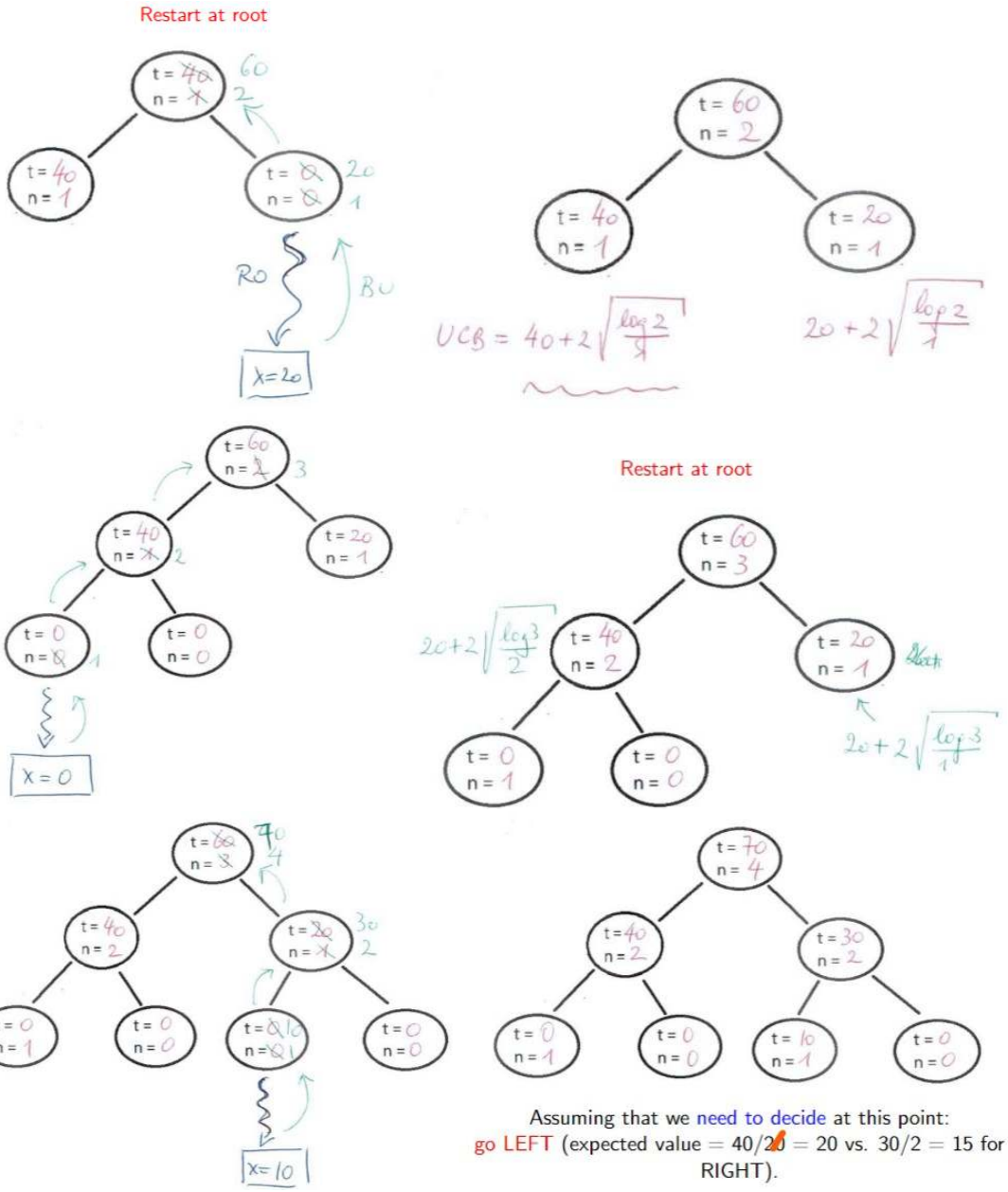
- MCTS: Tree policy vs. roll-out
    - **Repeat loop** (always **starting in same root node**) until predetermined **computation budget** has been exhausted.
    - Then choose the **best child-node as new root node** and repeat.



- MCTS: Worked Example

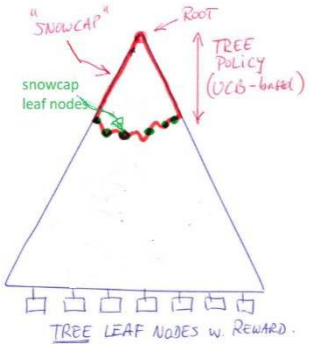




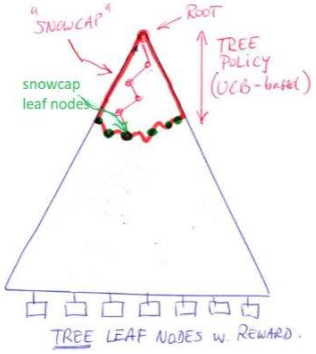


Assuming that we need to decide at this point: go LEFT (expected value = 40/2 = 20 vs. 30/2 = 15 for RIGHT).

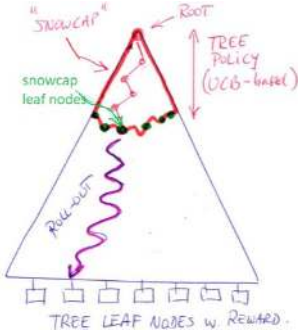
- MCTS Summary: Tree and “snow-cap” (i.e. subtree of explored nodes)



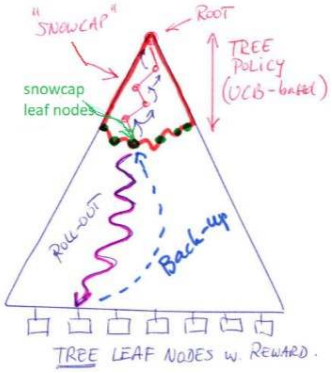
- MCTS Summary: Use **tree policy** to construct path to leaf nodes in “snow-cap edge”



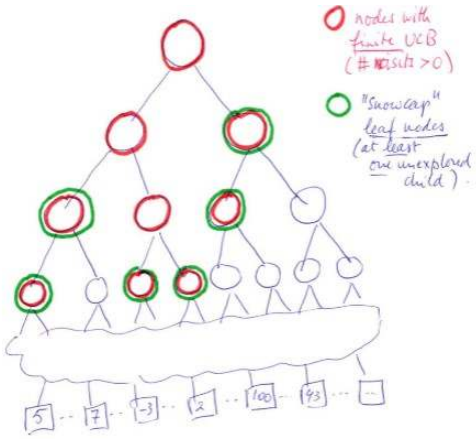
- MCTS Summary: Roll-out from “snow-cap edge” to leaf nodes (reward) in tree



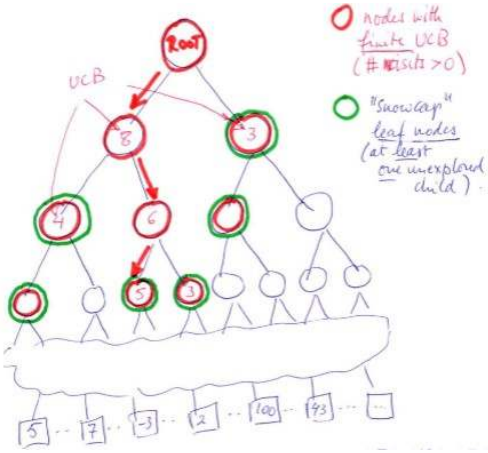
- MCTS Summary: Backup values in TP path



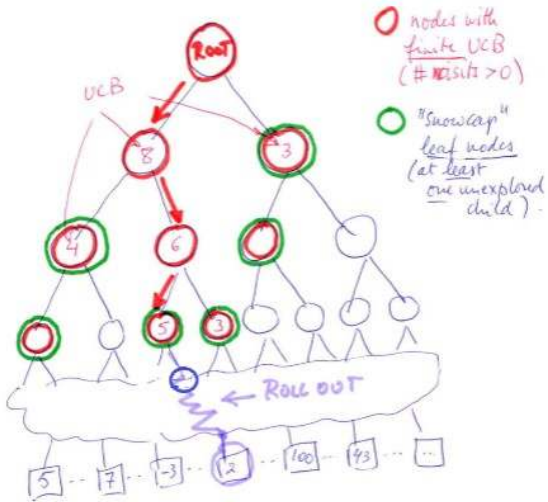
- MCTS Summary: Close-up of tree top (“snow-cap”)



- MCTS Summary: Applying **tree policy (TP)** to find path to “snow-cap leaf nodes” (using **UCB-values**)

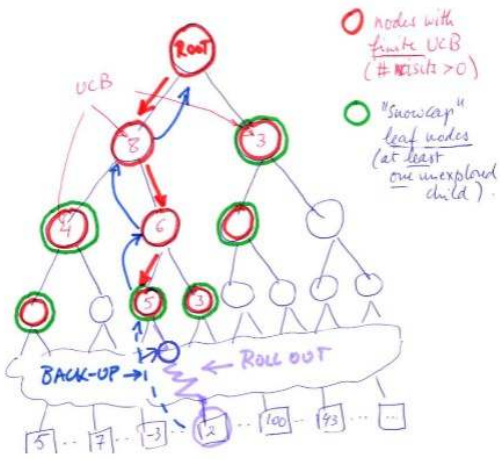


- MCTS Summary: Roll-out from “snow-cap edge”



➔ Choose the nodes with a finite UCB value. If they are none than you do a random choices and then a roll-out. Than you back-up and the UCB values change → because you have new information.

- MCTS Summary: Backup values in TP path



- **Summary**
  - **Exploitation:** Be greedy: Choose best action (according to current information).
  - **Exploration:** Try something new (use randomisation).
  - **Epsilon**-greedy: Simple but effective combination of exploration and exploitation.
  - **Upper confidence bound (UCB):**
    - Optimism in the face of uncertainty!
    - Asymptotic total regret achieves logarithmic (optimal) bound.
  - **Monte Carlo Tree Search (MCTS):** Use UCB to focus search on most promising part of the tree.

Lecture 9: Introduction to Reinforcement Learning

Part 1: MDP and Bellman Equations

- What is Reinforcement Learning (RL)
  - RL is one of **three main learning problems** studied in AI:
  - Learning by **trial and error** and **improving** over time;
  - **Natural form of learning**, ubiquitous in biology and psychology:
  - **Supervised learning** needs **teacher** (labeled data!) and RL learns on the job.
  - **RL learns on the job:** interact, explore, exploit experience!

- What makes RL attractive and interesting?
  - Natural way to learn complex skills, can we copy this?
  - RL Example 1: playing a song by ear
    - **Actions and transitions**: Moving from key to key. . .
    - **Immediate reward**: correct key or not;
    - **Cumulative reward**: Being able to play whole song.
    - Relatively easy, one gets immediate valuable feedback. . .
  - RL Example 2: cooking
    - States:
    - **Actions:** e.g. cooking, seasoning, stirring, etc.
    - **Transitions:** e.g. from raw to cooked.
    - **Immediate reward:** e.g. smell, look.
    - **Final reward:** enjoyable dinner.

        - **Problem:** what's the best sequence of actions (i.e. ***optimal policy***)?
    - RL Examples: playing chess
        - **States:** board configurations.
        - **Actions:** legal moves.
        - **Value of state/action:** very valuable to determine next move(action),
        - **Final reward:** Win(2), lose(0), draw(1).
        - **Problem:** what's the optimal action to take in each state (***policy***).

- What makes Reinforcement Learning challenging?
    1. **No Oracle**
        - Supervised learning: compare estimate to correct result (provided by oracle);
        - RL: no information on optimal action, just indicative *reward*. i.e. evaluative feedback, not prescriptive!
        - RL: only data on states/actions that have been experienced: (might be very small subset of entire state space);
    2. **Sparsity of feedback**
        - For many RL problems, **no feedback until target** is reached!
        - Initial stages: just random search!
    3. **Data generated during training**
        - Online learning: improve policies while interacting with environment.
        - No independent data generation!

          $\text{policy} \longleftrightarrow \text{data generation}$

- Reinforcement Learning: more abstract view
    - RL: Learning from (sequential) interaction: take actions to move from state to (better!?) state;
    - No labeled data, but **feedback** (reward, possibly delayed).
    - **Goal:** optimise **end-result** (i.e. long-term/cumulative reward).

- RL: even more abstract version
    - **Agent interacting with Environment:**
        - States (s), actions (a) and transitions:

          $s \xrightarrow{a} s' \quad (p(s' \mid s, a))$

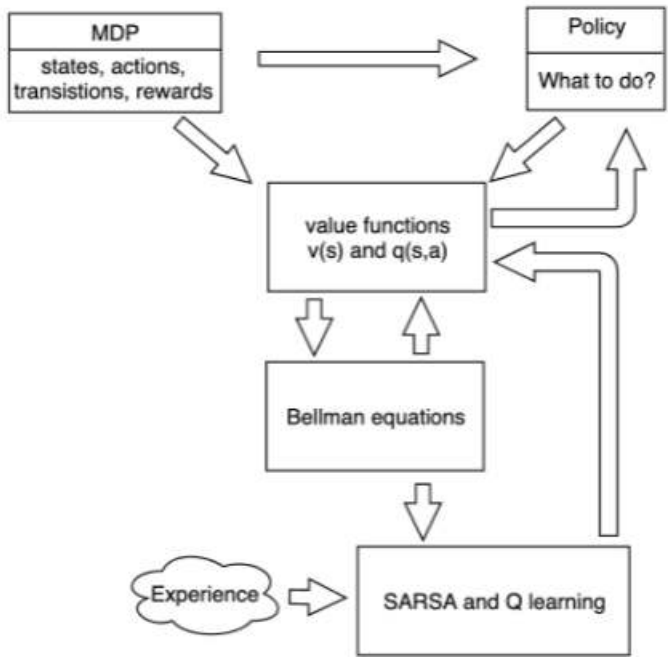
        - (immediate) reward:

          $r(s, a, s')$
    - **Example:** Taking a qualifying test (a), to pass from unqualified (s) to qualified state → new state (s'), or not!

- RL GOAL: find optimal policy!
    - **Policy π**, tells for each state (s), which action (a) to take:

      $s \xrightarrow{\pi} a$
    - **RL Goal: Given environment, determine optimal policy π∗:** what action to choose in each state to achieve best FINAL reward?

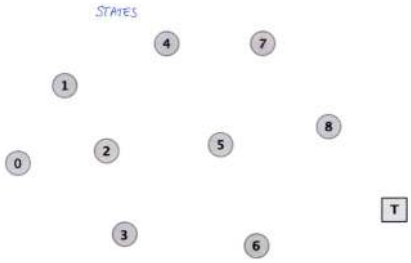
- Overview



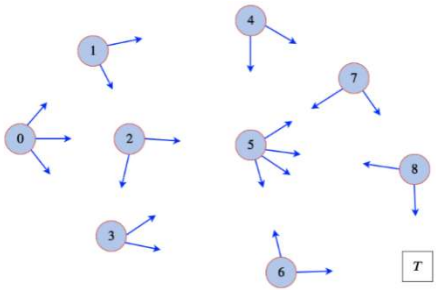
- MDP = Markov Decision Processes
- Value functions give interesting information about where to go next.
- SARSA and Q learning: functions of the Bellman equations turned into algorithms.
- Experience is fed into the SARSA and Q learning algorithms.
- With the value functions we can compute the policy.

- Markov Decision Process (MDP)
    - **Mathematical formalism** to capture the **essential characteristics** of the RL problem:
        - MDP = states, actions, transitions, rewards, (optimally: discount factor).

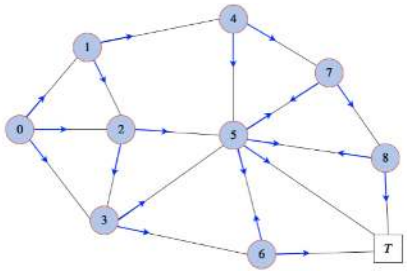
- Markov Decision Process (MDP) States
    - **States** capture the **information that is available** to the agent:



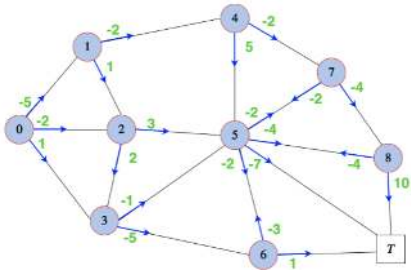
- Markov Decision Process (MDP) States + Actions:



- Markov Decision Process (MDP) States + Actions + Transitions:



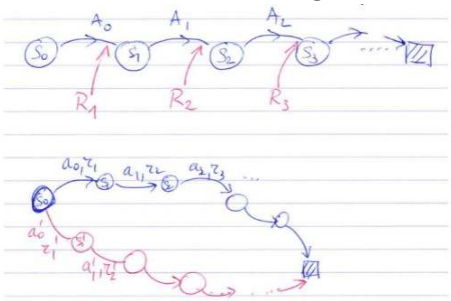
- Markov Decision Process (MDP) States + Actions + Transitions + Rewards:



➔ We need to solve: which path gives you the highest reward.

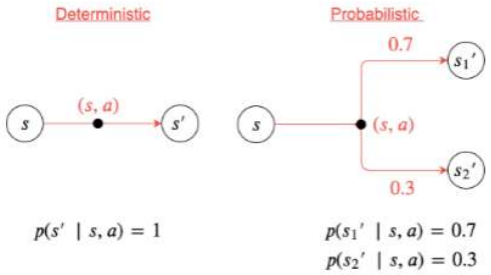
- Markov Decision Process (MDP)
  - Markov decision processes (MDP) provide a **formal model** for a **sequential decision problem**;
  - A **finite MDP** (*S*, *A*, *P*, *R*, *γ*) consists of: (is a tuple)
    - **Discrete time** t = 0,1,2,...
    - A discrete set of **states** $s \in S$ (captures relevant **information**).
    - A discrete set of **actions** $a \in A(s)$ for each *s*.
    - A **transition function** $p(s' \mid s, a)$: probability of transitioning to state *s*' when taking action *a* at state *s*. → actually a probability.
      - E.g. prob of passing exam $(s \xrightarrow{a} s')$ when studying (*a*);
    - A **reward function** $r(s, a, s') = E[R \mid s, a, s']$: expected reward when taking action *a* at state *s* and transitioning to *s*'.
      - Reward might depend on new state *s* ´ (e.g. *s* ´ = *pass* or *s*' = *fail*)
    - A planning horizon *H* or **discount factor** *γ*;
      - How important are future rewards?
      - short-sighted 0 ← *γ* →1 farsighted.

- Stochastic actions and return: interpretation → Capital letters are used to indicate stochastic events: interested in the **long-term reward → So the summation!**



Long-term return: $G_0(s_0) = R_1 + R_2 + \ldots + R_T = \sum_{t=1}^{T} R_t$

- Detailed Definition and Notation (1)

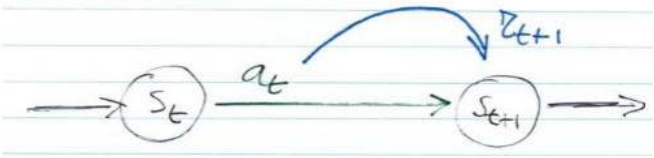


➔ So you have deterministic (p = 1) and probabilistic states!

- **Transition probability** to state *s´* when taking action *a* in state *s* :

$$p(s' \mid s, a) := P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

- Deterministic versus probabilistic transitions



  - **Expected immediate reward** when transitioning from state *s* to state *s'* under action *a*:

$$r(s, a, s') := E(R_{t+1} = r \mid S_t = s, A_t = a, S_{t+1} = s')$$

- The Markov property
  - **Markov property: informally**
    - The present (state) has all the information necessary to predict the future: <u>no need to keep track of the past</u>.
    - **"The future is independent of the past, given the present".**

$$P(S_{t+1}, R_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0) = P(S_{t+1}, R_{t+1} | s_t, a_t)$$

If reward R_{t+1} does NOT depend on successor state S_{t+1}:

$$P(S_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0) = P(S_{t+1} | s_t, a_t)$$

$$P(R_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0) = P(R_{t+1} | s_t, a_t)$$

- Joint versus marginal distribution

$$P(S_{t+1}, R_{t+1} \mid \ldots) = P(S_{t+1} \mid R_{t+1}, \ldots) \cdot P(R_{t+1} \mid \ldots)$$

  - **Dependence versus independence**
    - Action = hard work.
    - S_{t+1} = success or fail.
    - Reward R_{t+1}, could be:
      - **Dependent on next state:** e.g. bonus if you succeed;
      - **Independent of the next state:** energy expenditure for action.

- Markov Property
  - MP simplifies the mathematics and makes it tractable;
  - MP is sufficiently powerful as states can be expanded to **include all the information necessary to predict future;**
    - E.g.: to estimate the speed of a vehicle, we need at least two positions!

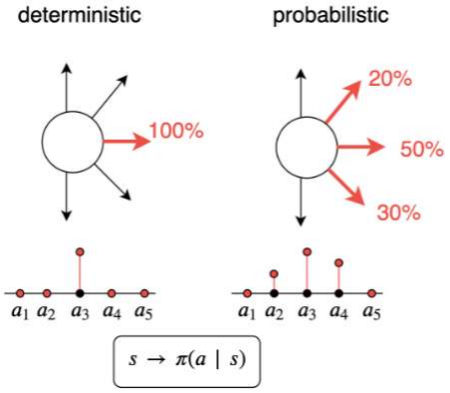
- Example of Markov game
  - Game of chess.
  - The current state of the board provides all information needed to determine next (optimal) move;
  - No need to know the history (i.e. the path leading up to the current state).

Policies, Value Functions and the Bellman Equation

- Policies for a MDP

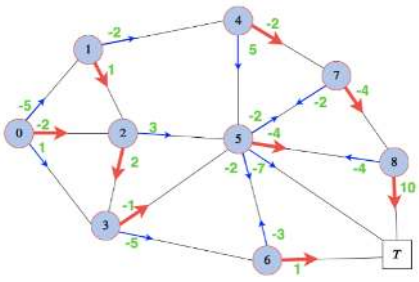
  - **Policy** specifies what actions should be taken in a given state.
  - Hence, a **policy π maps states into actions;**
    - **Deterministic policy:** a = π(s)
    - **Stochastic policy:** $\pi(a|s) := P(A_{t+1} = a | S_t = s)$
  - **Important:** policies are <u>not</u> part of the MDP!

- Deterministic vs. Probabilistic Policy

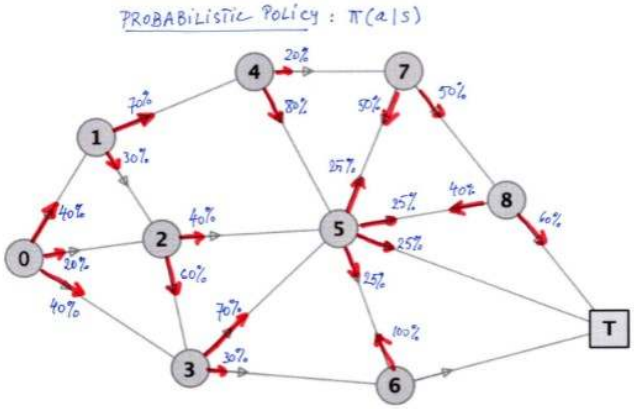


➔ So the policy is basically telling you a distribution over the actions.

- MDP + Policy (deterministic)



- MDP + Policy (Probabilistic/Stochastic)



- Return and Rewards
  - The goal of the agent is to maximize the expected (**long-term**) **return**, a (discounted) sum over the immediate rewards received:

$$G_0 = R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots = \sum_{k=1}^{\infty} \gamma^{k-1} R_k$$

  - Or more generally:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k}$$

  - **Episodic tasks** can be interpreted as infinite-horizon, if we represent episode termination as transition to an **absorbing state** with self-transitions and zero reward.

- Long-term (cumulative) return: some remarks

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k}$$

  - **Remarks**
    - Notice that *G* depends on both the **starting state** (*s*) and the **policy applied** (π):

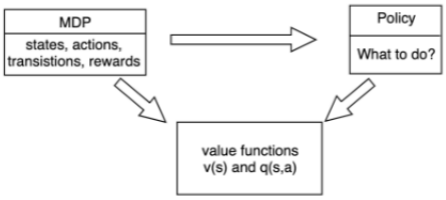
$$G \equiv G_\pi(s)$$

    - **Recursion relation:**

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

- Long-term reward

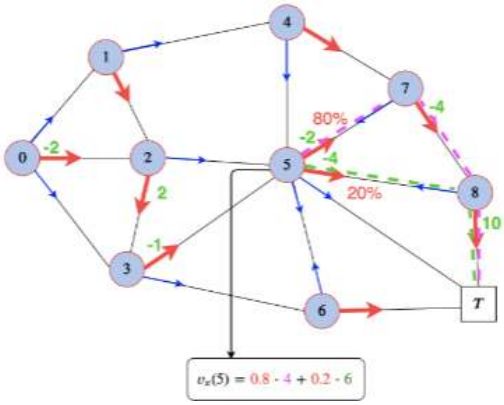
- Pole balancing
- Overview:



Now lets introduce the value functions.

- MDP + Policy → state value function $v_\pi(s)$
  - $v_\pi(s)$: **expected value** of state *s* when actions are specified by π:

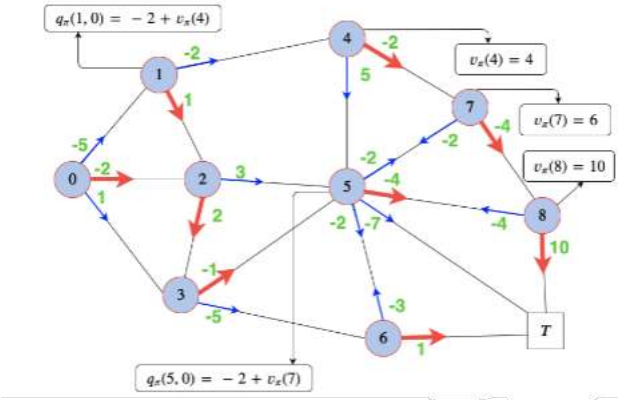
$$v_\pi(s) = E_\pi\left(\sum_{t=1}^{T} R_t \mid s_0 = s\right)$$



Take the average of the paths you've rolled-out when you're at a certain state. First count all the rewards of each path with it's probability.

- MDP + Policy → state-action value function $q_\pi(s, a)$:

$$q_\pi(s, a) = E_\pi\left(\sum_{t=1}^{T} R_t \mid s_0 = s,\ a_0 = a\right)$$



- Value function: tools for reasoning about future reward

- The **state value function** $v_\pi(s)$ assigns to each state *s* the **expected total return** when **starting** in the state *s* and **applying policy** π;

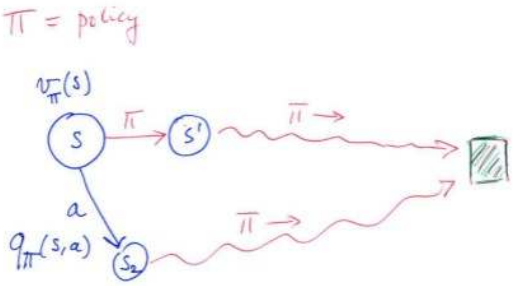
$$v_\pi(s) := E_\pi\left[G_0 \mid S_0 = s\right] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{k+1} | S_0 = s\right]$$

- The **state-action value function** $q_\pi(s, a)$ of a policy π is:

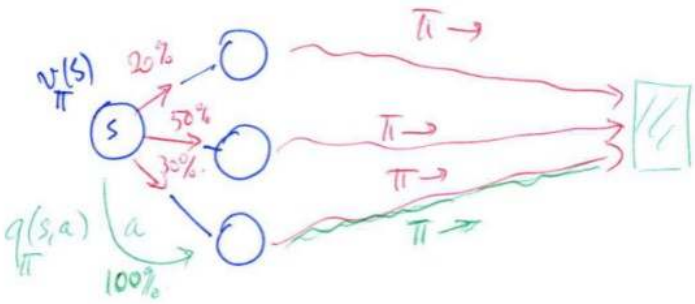
$$q_\pi(s, a) := E_\pi\left[G_0 \mid S_0 = s, A_0 = a\right] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{k+1} \mid S_0 = s, A_0 = a\right]$$

- ***q*(*s*, *a*) state space typically much larger than v (s)-state space!** Hence, more difficult to learn.

- Difference between value functions $v_\pi(s)$ and $q_\pi(s, a)$



  - $v_\pi(s)$: policy π dictates **every action along the path**;
  - $q_\pi(s, a)$: **<u>first action *a* is taken independently of the policy π</u>**, from then onward, π dictates the remaining actions taken along the rest of the path.
    ➔ That's the difference!
  - Similar interpretation for stochastic policy:



- Relationship between value and value-action function
  - Notice that *v*(*s*) is a **weighted mean** of *q* (*s*, *a*), where the weights are determined by the policy π:

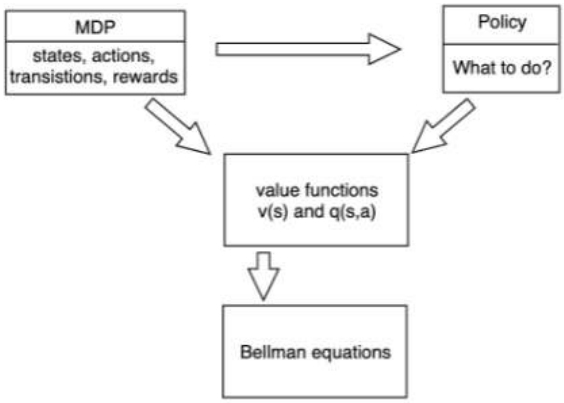
$$v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$$

  - Indeed:

$$\begin{aligned} v_\pi(s) &= E_\pi\left[G_0 \mid S_0 = s\right] \\ &= \sum_a E_\pi\left[G_0 \mid S_0 = s, A_0 = a\right] \pi(a \mid s) \\ &= \sum_a q_\pi(s, a) \pi(a \mid s) \end{aligned}$$

Bellman equations

- Overview



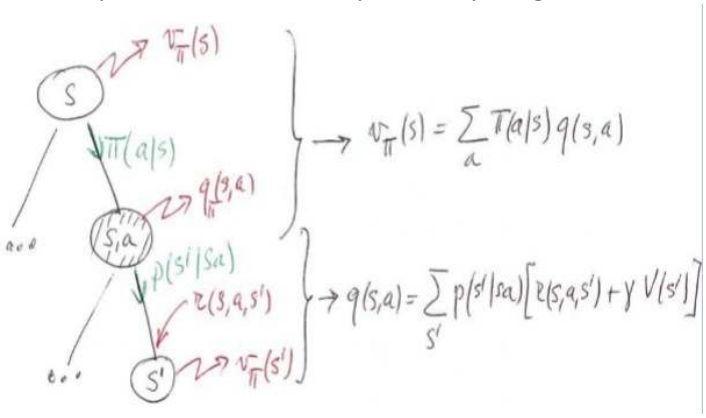
- Bellman equation for value functions
  - **Value function $v_\pi(s)$:**

$$v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$$

  - **Value function $q_\pi(s, a)$:**

$$\begin{aligned} q_\pi(s,a) &= E_\pi\left[G_0 \mid S_0 = s, A_0 = a\right] \\ &= \sum_{s'} E_\pi\left[G_0 \mid S_0 = s, A_0 = a, S_1 = s'\right] p(s' \mid s, a) \\ &= \sum_{s'} E_\pi\left[R_1 + \gamma G_1 \mid S_0 = s, A_0 = a, S_1 = s'\right] p(s' \mid s, a) \\ &= \sum_{s'} \left[r(s,a,s') + \gamma E_\pi(G_1 \mid S_1 = s')\right] p(s' \mid s, a) \\ &= \sum_{s'} \left[r(s,a,s') + \gamma E_\pi G_0(s')\right] p(s' \mid s, a) \\ &= \sum_{s'} p(s' \mid s, a)\left[r(s,a,s') + \gamma v_\pi(s')\right] \end{aligned}$$

- Bellman equations (schematically): Back-up Diagram → **Good for memorising eqs!**



  - ➔ Make sure you understand this diagram.
  - ➔ The state-action (s, a), has a certain value which is q(s, a), which is calculated by the probability times the reward going from s to s' + the new v(s') value and the *y* is the discount factor

- Bellman equation: Summary

- **Back-up**

$$v_\pi(s) = \sum_a \pi(a \mid s) q(s,a)$$

$$q(s,a) = \sum_{s'} p(s' \mid s,a) \left[r(s,a,s') + \gamma v(s') \right]$$

- **Combined into recursion equations:**

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s,a) \left[r(s,a,s') + \gamma v_\pi(s') \right]$$

$$q_\pi(s,a) = \sum_{s'} p(s' \mid s,a) \left[r(s,a,s') + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s',a') \right]$$

- The definition of v_π can be rewritten recursively by making use of the transition model, yielding the Bellman equation:

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s,a) \left[r(s,a,s') + \gamma v_\pi(s') \right]$$

- This is a set of **linear equations**, one for each state, the solution of which defines the value of π.
- A similar recursive relation holds for Q-values:

$$q_\pi(s,a) = \sum_{s'} p(s' \mid s,a) \left[r(s,a,s') + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s',a') \right]$$

- Matrix form of Bellman equation (writing in simpler form)

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s,a) \left[r(s,a,s') + \gamma v_\pi(s') \right]$$

  - We can rewrite this as:

$$v_\pi(s) = \gamma \sum_{s'} \underbrace{\left(\sum_a \pi(a \mid s) p(s' \mid s,a) \right)}_{P_\pi(s,s')} v_\pi(s') + \sum_a \underbrace{\left(\sum_{s'} p(s' \mid s,a) r(s,a,s') \right)}_{R(s,a)} \pi(a \mid s)$$

  - **Under policy π**:
    - square matrix $P_\pi(s, s')$ is **transition probability** s → s ′:

$$P_\pi(s,s') := \sum_a \pi(a \mid s) p(s' \mid s,a)$$

    - R(*s*, *a*) is the **expected (immediate) reward** when taking action *a* in state *s*:

$$R(s,a) = \sum_{s'} p(s' \mid s,a) r(s,a,s')$$

    - $r_\pi(s)$ is the **expected (immediate) reward** in state *s*:

$$\mathbf{r}_\pi(s) = \sum_a \pi(a \mid s) R(s,a)$$

- Matrix form of Bellman equation
  - **Bellman in matrix form**:

$$\mathbf{v}_\pi = \gamma P_\pi \mathbf{v}_\pi + \mathbf{r}_\pi \qquad \text{or again} \qquad (I - \gamma P_\pi)\mathbf{v}_\pi = \mathbf{r}_\pi$$

  - Solving for the value function v_π:

$$\mathbf{v}_\pi = (I - \gamma P_\pi)^{-1}\mathbf{r}_\pi$$

  - Notice that:

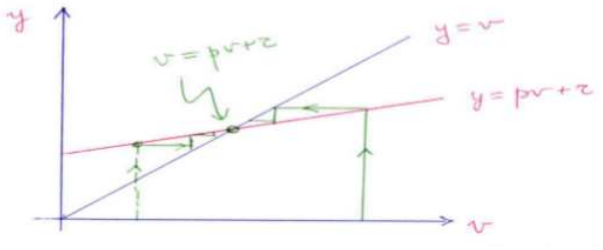
$$\mathbf{v} = (I - \gamma P)^{-1}\mathbf{r} = (I + \gamma P + \gamma^2 P^2 + \ldots)\mathbf{r}$$

- Solving Bellman equations: Iteration to Fix-Point
  - Matrix form of **Bellman equation** correspond to fix-point:

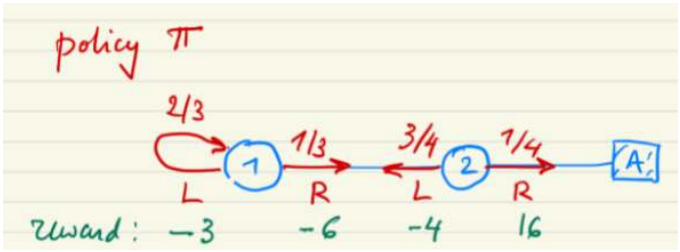
$$\mathbf{v}_\pi = \gamma P_\pi \mathbf{v}_\pi + \mathbf{r}_\pi \quad \Longrightarrow \quad \mathbf{v}_\pi = (I - \gamma P_\pi)^{-1}\mathbf{r}_\pi$$

  - **Iterative solution (fix-point solution):** update rule:

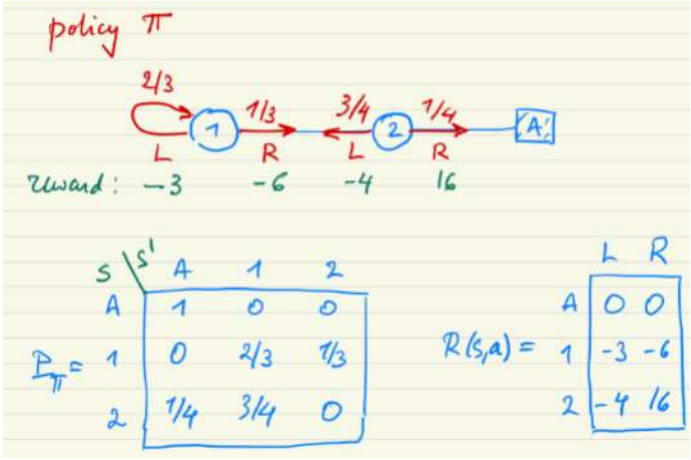
$$\mathbf{v}^{k+1} = \gamma P \mathbf{v}^k + \mathbf{r}$$



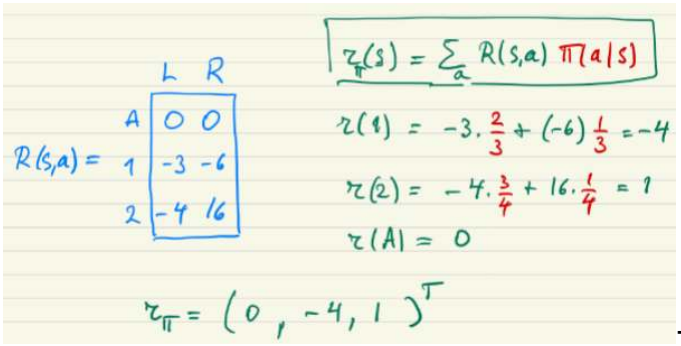
- Solving Bellman equations: Worked example



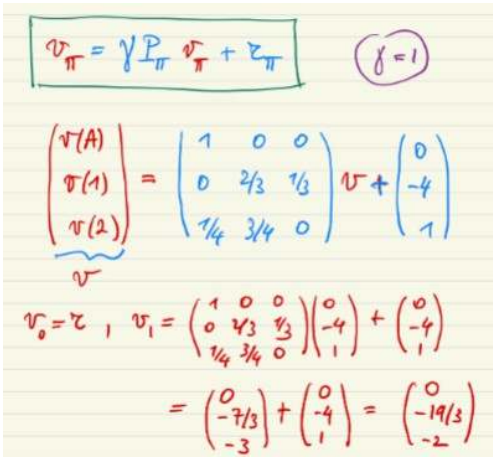
➔ 3 states MDP (1, 2 and absorbing state).
➔ Two actions in each state (L, R).
➔ So now it must be obvious that $v_\pi(1)$ is lower than $v_\pi(2)$, as $v_\pi(2)$ is closer to the absorbing state *A* with reward 16.

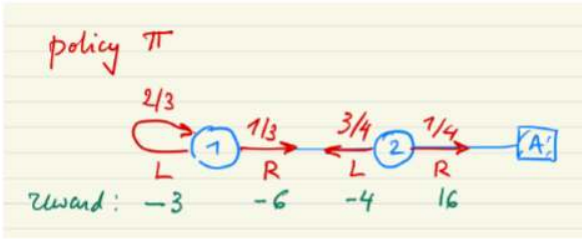


➔ R(s, a) is immediate reward.



→ is the r_π matrix.

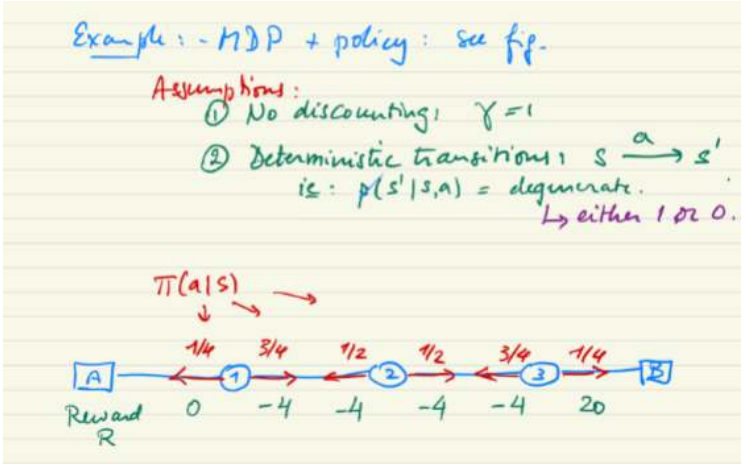


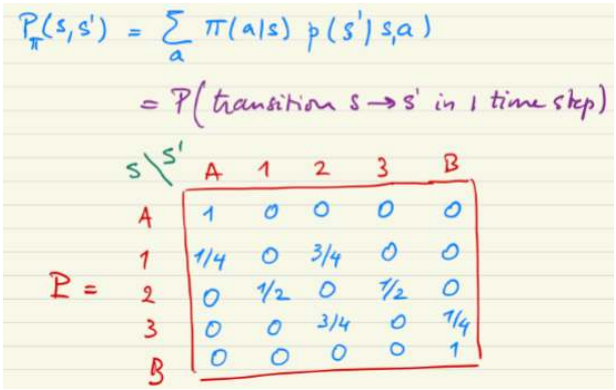


$$
\begin{aligned}
v_2 &= (0, -6.3333, -2.0000) \\
v_3 &= (0, -8.8889, -3.7500) \\
\dots &= \dots \\
v_\infty &= (0, -38.7671, -27.7992)
\end{aligned}
$$

Lecture 10

- Example: MDP + value functions





$$R(s,a) = \sum_{s'} p(s'|s,a)\, r(s,a,s')$$

= expected immediate reward when taking action a in state s.

$R =$

	L	R
A	0	0
1	0	−4
2	−4	−4
3	−4	20
B	0	0

$$r(s) = \sum_a R(s,a)\,\pi(a|s)$$

$$r = \begin{bmatrix} r(A) \\ r(1) \\ r(2) \\ r(3) \\ r(B) \end{bmatrix} = \text{diag}\left(\underbrace{\begin{bmatrix} 0 & 0 \\ 0 & -4 \\ -4 & -4 \\ -4 & 20 \\ 0 & 0 \end{bmatrix}}_{R(s,a)} \underbrace{\begin{array}{c|ccccc} & A & 1 & 2 & 3 & B \\ \hline L & * & 1/4 & 1/2 & 3/4 & * \\ R & * & 3/4 & 1/2 & 1/4 & * \end{array}}_{\pi(a|s)} \right)$$

$$r(s) = \sum_a R(s,a)\,\pi(a|s)$$

= expected immediate return in s

$$r = \begin{bmatrix} 0 & \left(\frac{1}{4}\cdot 0 + \frac{3}{4}(-4)\right) = -3 & -4 & \frac{3}{4}(-4) + \frac{1}{4}20 = 2 & 0 \end{bmatrix}$$

A 1 2 3 B

Bellman eqs in matrix form:

$$v = \begin{bmatrix} v(A) \\ v(1) \\ v(2) \\ v(3) \\ v(B) \end{bmatrix} = \begin{array}{c} \begin{array}{ccccc} A & 1 & 2 & 3 & B \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 3/4 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 3/4 & 0 & 1/4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} v + \underbrace{\begin{bmatrix} 0 \\ -3 \\ -4 \\ 2 \\ 0 \end{bmatrix}}_{r''}$$

$$v = P \cdot v + r''$$

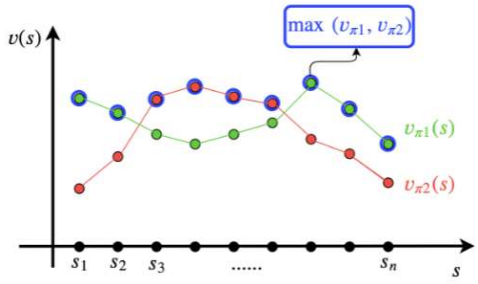
Example: $v(2) = \frac{1}{2}v(1) + \frac{1}{2}v(3) + (-4)$

$v(3) = \frac{3}{4}v(2) + \frac{1}{4}\underbrace{v(B)}_{=0} + 2$

Bellman equations for optimality

- Optimal value functions
  - The optimal value functions are defined by the **pointwise maximum over policies**:

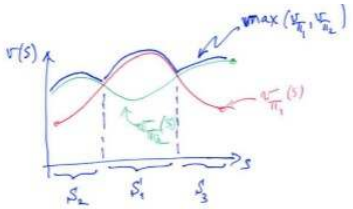
$$\forall s, a: \quad v^*(s) := \max_\pi v_\pi(s) \quad \text{and} \quad q^*(s,a) := \max_\pi q(s,a)$$



- Optimal policy π*
  - There exists an **optimal policy π*** such that its value functions corresponds to the optimal value functions:

$$v_{\pi^*}(s) = v^*(s) \quad \text{and} \quad q_{\pi^*}(s,a) = q^*(s,a)$$

  - **Intuition:**



$$\pi^*(s) = \begin{cases} \pi_1(s) & \text{if } s \in S_2 \\ \pi_2(s) & \text{if } s \in S_1 \cup S_3 \end{cases}$$

- Optimal value functions
  - Value functions define a partial ordering over policies:

$$\pi \succ \pi' \Rightarrow v_{\pi(s)} \geq v_{\pi'}(s), \forall s \in S$$

  - There can be multiple optimal policies but they all share the same **optimal state-value function**:

$$v^*(s) = \max_\pi v_\pi(s), \quad \forall s \in S$$

  - They also share the same **optimal action-value function**:

$$q^*(s,a) = \max_\pi q_\pi(s,a), \quad \forall s \in S, a \in A$$

- Optimal value functions: from weighted mean to max
  - **State-value function**
    - General:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

    - Optimal:

$$v^*(s) = v_{\pi^*}(s) = \max_a q_{\pi^*}(s,a) = \max_a q^*(s,a)$$

$$v^*(s) = \max_a q^*(s,a)$$

- **State-action value function**
  - General:

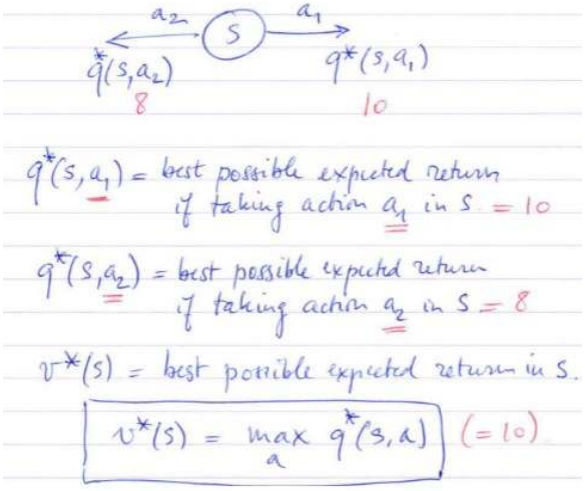
$$q_\pi(s,a) = \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma v_\pi(s))$$

  - Optimal:

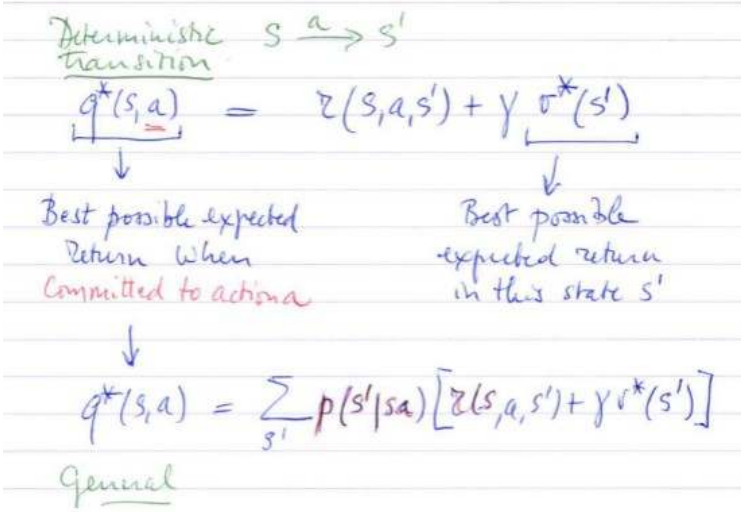
$$q_{\pi^*}(s,a) = \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma v_{\pi^*}(s))$$

$$q^*(s,a) = \sum_{s'} p(s' \mid s,a)(r(s,a,s') + \gamma v^*(s))$$

- Bellman Optimality Equations

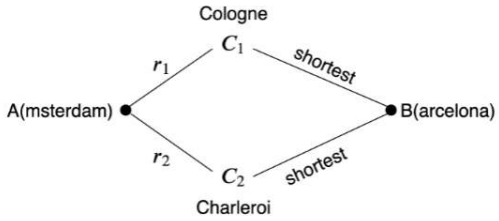


→ State-value function.



→ State-action function.

- Bellman Optimality equation: Travel Distance Analogy



$$d^*(A,B) = \min_{C_i} \{r_i + d^*(C_i,B)\}$$

- Bellman optimality conditions (for deterministic transitions)
  - Travel Analogy: Shortest distance paths:

    $$d^*(A,B) = \min_{C_i} \{r_i + d^*(C_i,B)\}$$

  - **Deterministic transitions:** $s \xrightarrow{a} s_a$

    $$v^*(s) = \max_a \{r(s,a,s_a) + v^*(s_a)\}$$

$$q^*(s,a) = r(s,a,s_a) + v^*(s) = r(s,a,s_a) + \max_{a'} q^*(s_a,a')$$

- Bellman optimality equations (General form)
  - **Optimal state value function**

    $$v^*(s) = \max_a q^*(s,a)$$

  - **Optimal state-action value function**

    $$q^*(s,a) = \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma v^*(s')\right]$$

  - **Combined**

    $$v^*(s) = \max_{a \in A} \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma v^*(s')\right]$$

    $$q^*(s,a) = \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma \max_{a' \in A} q^*(s',a')\right]$$

- Back-up Diagram for Bellman Optimality Equations
  - **Optimize over actions!**

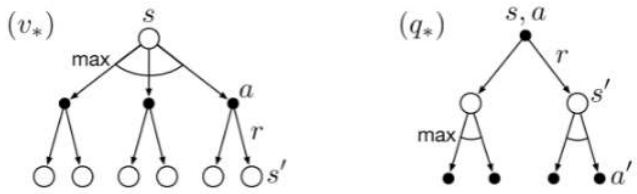


Figure 3.5: Backup diagrams for v_* and q_*

➔ Left-hand side: black dots are the $q^*(s, a)$.

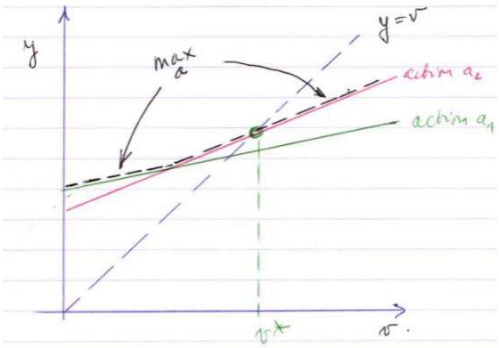
- Bellman optimality equation in matrix form

$$
\begin{aligned}
v^*(s) &= \max_{a \in A} \sum_{s'} p(s' \mid s, a) \left[r(s, a, s') + \gamma v^*(s') \right] \\
&= \max_a \left(R(s, a) + \gamma \sum_{s'} \underbrace{p(s' \mid s, a)}_{T_a(s,s')} v^*(s') \right) \\
&= \max_a \left(R(s, a) + \gamma \sum_{s'} T_a(s, s') v^*(s') \right)
\end{aligned}
$$

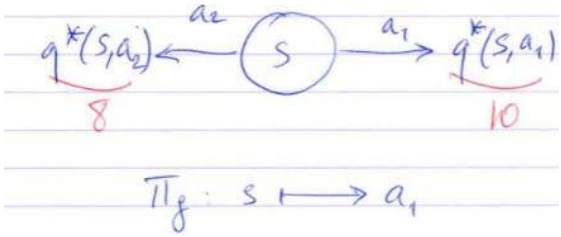
- Or in matrix notation:

$$\mathbf{v}^* = \max_a \left(R_a + \gamma T_a \mathbf{v}^* \right)$$

- Bellman optimality equation



- Importance of v^* and q^*
  - Markov property: optimal decision only depends on current state.
  - **Greedification:** in state s pick action a with highest $q^*(s, a)$.



- Bellman Optimality Conditions
  - ***v*∗ and *q*∗ satisfy a set of (non-linear) equations** analogous to Bellman equations for v_π and q_π;
  - These non-linear equations **do not refer** explicitly to the optimal policy;
  - As a consequence we can find the **optimal policy π∗** by
    1. First solving these equations to find $v^*(s)$ and $q^*(s, a)$) and
    2. Then use **greedification** to determine the corresponding optimal policy π∗:

$$\forall s: \quad a_{opt} = \arg\max_a q^*(s, a)$$

- Example: Bellman optimality conditions
  - MDP

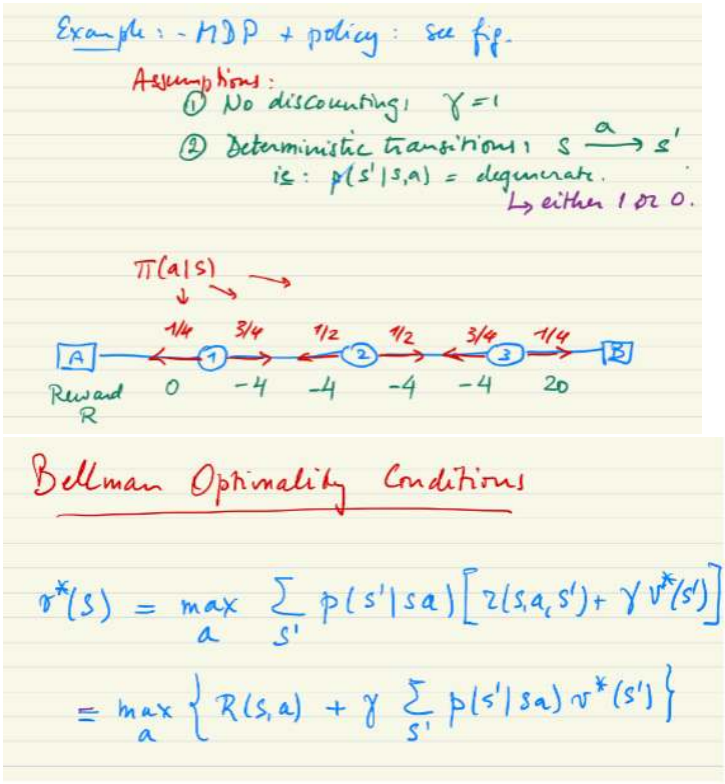
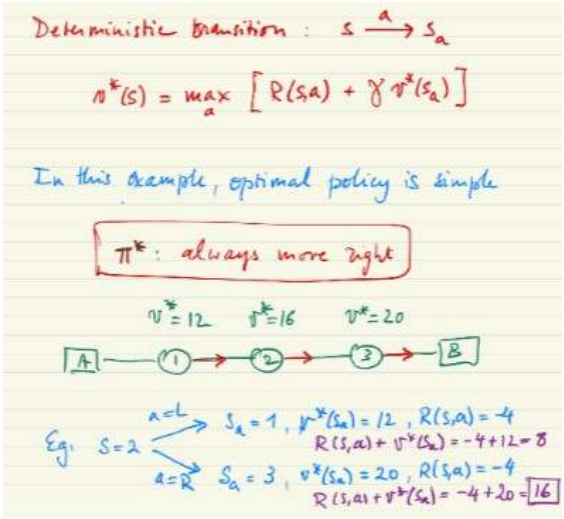
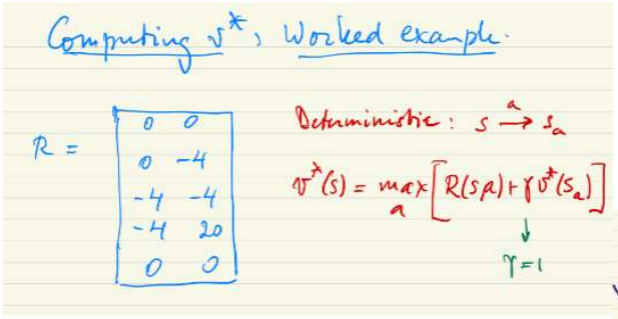


Figure: General expression for v^*

- Bellman optimality conditions: Deterministic transition





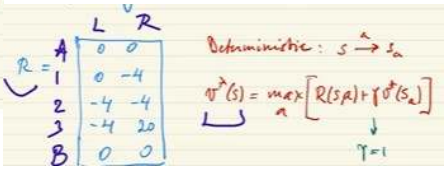
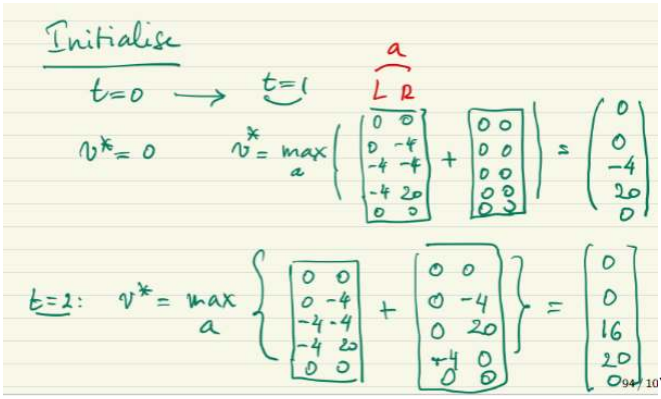


Figure: $R(s,a)$ and $v^*(s)$ for deterministic transitions

- Bellman optimality condition: Computing v^* using iteration



You take the max of both matrix, when summed up.

- Example: Bellman optimality conditions

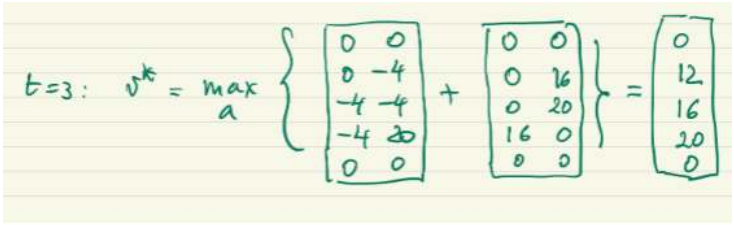


Figure: $t = 3$

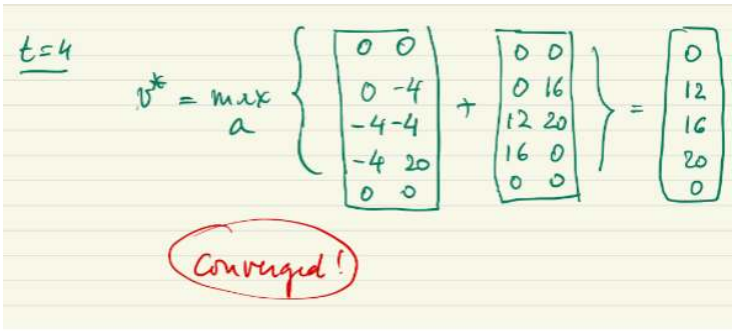


Figure: Convergence for $t = 4$

- For known MDP: compute $q^*(s, a)$ based on $v^*(s)$

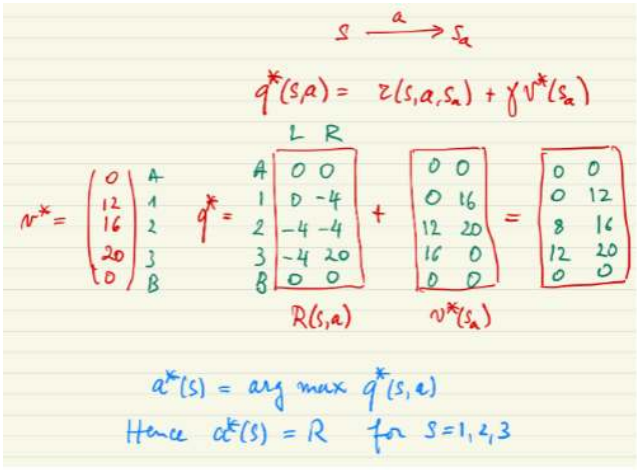


Figure: Computing $q^*(s, a)$ from $v^*(s)$

- Bellman optimality equations

- If **MDP is fully known**, it suffices to compute v^*, as q^* can then be derived:
  - **Optimal state-action value function:**

$$q^*(s,a) = \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma v^*(s')\right]$$

  - **Recall: Optimal state-value function:**

$$v^*(s) = \max_a q^*(s,a)$$

- Model-based vs model-free
  - **Model-based (PLANNING):** the MDP = (*S, A, P, R, γ*) is **completely specified**;
    - Solve the Bellman equations (DP)!
  - **Model-free (LEARNING):** only **direct experience**, i.e. **only sample paths** (states, actions and rewards) are given. Put differently, only **experience-based information** is given!
    - Random search but Bellman equations allow to propagate values!

- **Summary**
  - Bellman equations: General (**take the mean over the actions**) versus Optimal (**take the max of the actions**)
    - **State-value functions:**
      - General:

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma v_\pi(s')\right]$$

      - Optimal:

$$v^*(s) = \max_a \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma v^*(s')\right]$$

    - **State-action value functions:** General → averaging over the policy (same as state-value function)
      - General:

$$q_\pi(s,a) = \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s',a')\right]$$

      - Optimal:

$$q^*(s,a) = \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma \max_{a'} q^*(s',a')\right]$$

  - Taxonomy of RL problems

	Prediction *Estimation:* *Given π, what is v?*	**(Optimal) Control** *Optimisation:* *What is optimal π?*
model-based (MDP given)	Policy evaluation using Dyn. Programming (DP)	Policy improvement (+ Policy evaluation) = Policy iteration
model-free (MDP unknown)	Monte Carlo (MC) Temporal Diff^{ing} (TD) = "impatient MC" *bootstrapping!*	Generalized Policy Iteration *"simultaneous"*

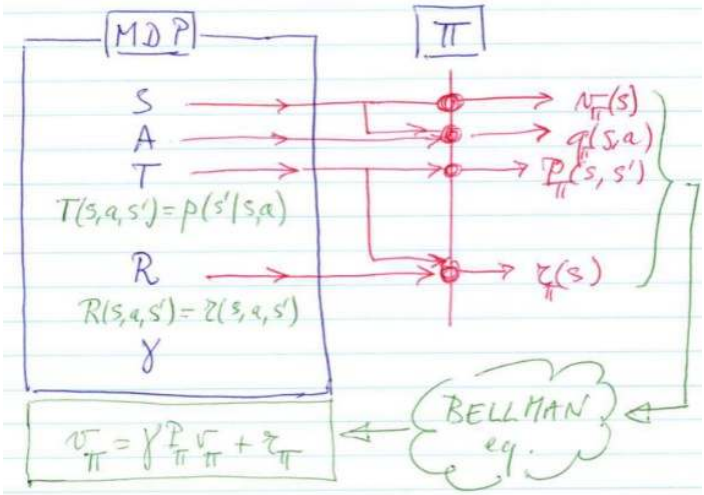
➔ Two types of problems: prediction and control (which is an optimisation problem)

Lecture 11: Introduction to Reinforcement Learning

Part 2: Model-based Prediction and Control (→ PLANNING)

Reinforcement Learning: Markov Decision Process (MDP) repetition of previous lectures

- Markov decision processes (MDP)
  - Markov decision process (MDP) provide a formal model for a sequential decision problem;
  - A **finite MDP** (*S, A, P, R, γ*) consists of:
    - **Discrete time** t = 0,1,2,...
    - A discrete set of **states** $s \in S$ (captures relevant **information**).
    - A discrete set of **actions** $a \in A(s)$ for each *s*.
    - A **transition function** $p(s' \mid s, a)$: probability of transitioning to state *s'* when taking action *a* at state *s*. → actually a probability.
    - A **reward function** $r(s, a, s') = E[R \mid s, a, s']$: expected reward when taking action *a* at state *s* and transitioning to *s'*.
    - A planning horizon *H* or **discount factor** *γ*;
      - How important are future rewards?
      - shortsighted 0 ← γ →1 farsighted
- MDP – Policy – Bellman

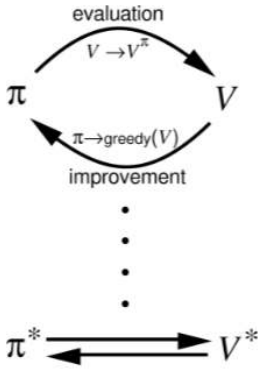


Model-based: Prediction an Control

- Model-based: prediction and control

- **Dynamic Programming (DP):** Collection of algorithms that can be used to **compute optimal policy** given a **completely specified model** for the environment (MDP);
- **Policy evaluation:** given a policy π compute value functions $v_\pi(s)$ and $q_\pi(s, a)$;
- **Policy improvement:** given a policy π and corresponding value function v_π, can we find a better policy π´ such that $v_{\pi'} \geq v_\pi$?
- **Policy iteration:** iteratively alternate between policy evaluation and improvement to find an optimal policy.

- Policy evaluation, improvement and iteration



➔ So given a policy function I can evaluate them with the use of the Bellman equation and then with the *V* I can us greedification to improve the policy function.

- Policy evaluation (1)
  - Rather than estimating value of each state independently, **use Bellman equation to exploit the relationship between states**.
  - Initial value function v_0 is chosen arbitrarily.
  - **Policy evaluation = evaluate value function under the policy** update rule:

$$v_{k+1}(s) \leftarrow \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a)\Big[r(s, a, s') + \gamma v_k(s')\Big]$$

  - Apply to every state in each **sweep** of the state space.
  - Repeat over many sweeps.
  - Converges to the fixed point $v^k = v_\pi$.

- Policy evaluation (2): Algorithm

Input π, the policy to be evaluated;

Initialize $v(s) = 0$, for all $s \in \mathcal{S}$

Repeat:

$\Delta \leftarrow 0$;

for each $s \in S$: # single sweep over all states

$v \leftarrow v(s)$

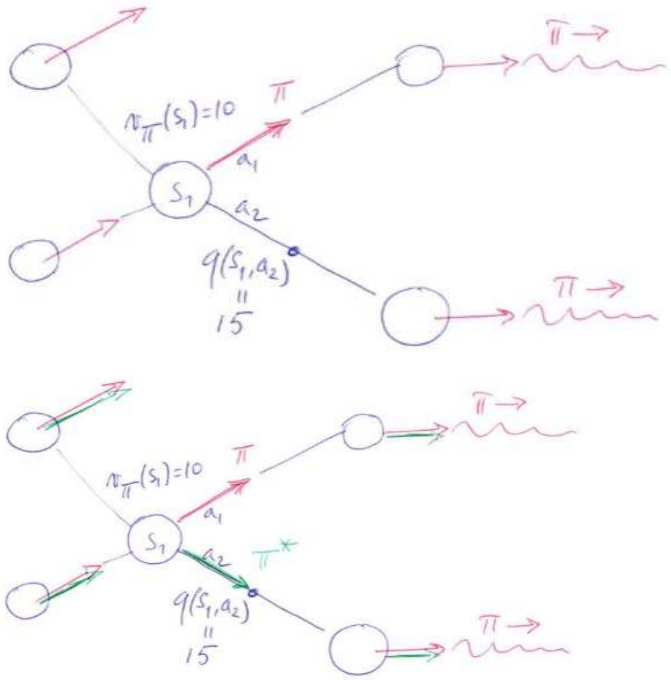
$v(s) \leftarrow \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a)(r(s, a, s') + \gamma v(s'))$

$\Delta \leftarrow \max(\Delta, |v - v(s)|)$

until $\Delta \leq$ small positive number;

Output: $v \approx v_\pi$

- Policy improvement = changing your policy to an action that gives more value. (higher *q*)



- Policy improvement (1) (complicated way to explain it)
  - Policy evaluation yields v_π, the true value of π.
  - Use this to incrementally improve the policy by considering whether for some state *s* there is a better action a π(s).
  - Is **choosing** *a* **in** *s* and **then using** π better than using π, i.e.,
    $$q_\pi(s,a) = \sum_{s'} p(s' \mid s,a)\Big[r(s,a,s') + \gamma v_\pi(s')\Big] > v_\pi(s)?$$
  - If so, then the **policy improvement theorem** tells us that changing π to take *a* in *s* will increase its value:
    $$\forall s \in S, q_\pi(s,\pi'(s)) \geq v_\pi(s) \quad \Rightarrow \quad \forall s \in S, v_{\pi'}(s) \geq v_\pi(s)$$
  - In our case, π = π′ except that π′(s) = a π(*s*).

- Policy improvement (2)
  - Applying to all states yields the greedy policy w.r.t. v_π:
    $$\pi'(s) \leftarrow \arg\max_a q_\pi(s,a)$$
    $$v_{\pi'}(s) = \max_a \sum_{s'} p(s' \mid s,a)\Big[r(s,a,s') + \gamma v_\pi(s')\Big]$$
  - If π = π′, then $v_\pi = v_{\pi'}$ and for all s ∈ S :
    $$v_{\pi'}(s) = \max_{a \in A} \sum_{s'} p(s' \mid s,a)\Big[r(s,a,s') + \gamma v'_\pi(s')\Big]$$
  - This is equivalent to the Bellman optimality equation, implying that $v_\pi = v_{\pi'} = v^*$ and π = π′ = π*.

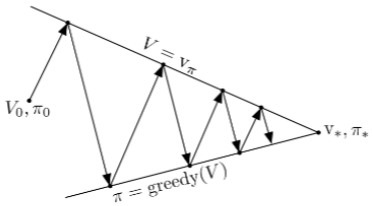
- Policy iteration (1)
  - **Policy iteration = policy evaluation + policy improvement**.

$$\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*,$$

- Policy improvement makes result of policy evaluation obsolete.
- Return to policy evaluation to compute v_{π}'.
- Converges to the fixed point $v_{\pi} = v^*$.
  - ➔ So optimise policy function until it converges → nothing changes anymore.

- Policy iteration (2): geometric analogy

A geometric metaphor for convergence of GPI:



Compare to **EM-algorithm** in ML.

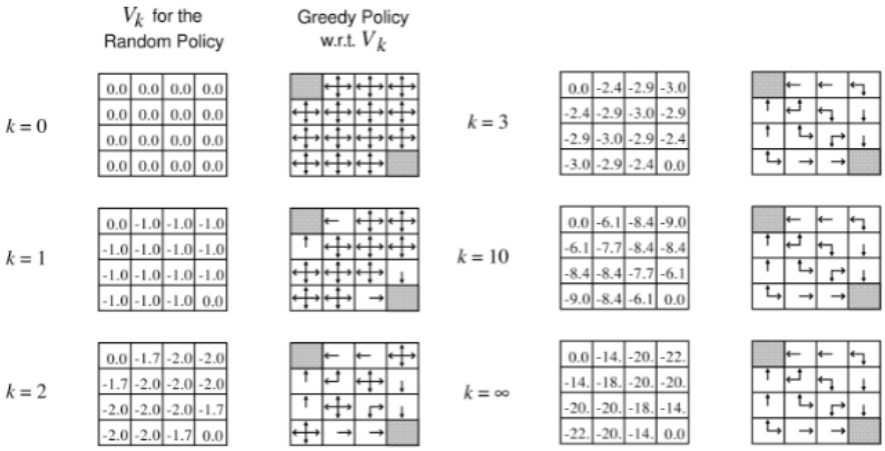
- Policy iteration (3): Algorithm (for deterministic policy)

1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $b \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s')\right]$
   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop; else go to 2

- Stopping policy evaluation early (more technical details, value functions can still change even though the grid/actions stay the same)



- Value iteration
  - Compute **optimal *v* first*** (iteratively), then derive **optimal policy π***:

    $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v^* \rightarrow \pi^*$
  - **Value iteration:**

    $$q_{k+1}(s,a) \leftarrow \sum_{s'} p(s' \mid s,a)\Big[r(s,a,s') + \gamma v_k(s')\Big]$$

    $$v_{k+1}(s) \leftarrow \max_a q_{k+1}(s,a),$$
  - Turns **Bellman optimality equation** into an **update rule**:

    $$v_{k+1}(s) \leftarrow \max_a \sum_{s'} p(s' \mid s,a)\Big[r(s,a,s') + \gamma v_k(s')\Big]$$

- Efficiency of dynamic programming (not talked about in class)
  - An MDP has $|A|^{|S|}$ deterministic policies.
  - But the worst-case computational complexity of dynamic programming is polynomial in |S| and |A|.
  - MDP planning can also be done with **linear programming**, which has better wors-case guarantees, but is impractical for large MDPs.
  - In very large MDPs, where even doing one sweep is infeasible, **asynchronous dynamic programming** must be used.
  - Convergence in the limit is guaranteed as long as every state is backed up infinitely often.

- **Summary of terminology**
  - **Value iteration** algorithms search for optimal value function v* from which policy is deduced:

    $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v^* \rightarrow \pi^*$
  - **Policy iteration** algorithms evaluate the policy π by computing the (corresponding) value function v_π and uses v_π to improve the policy: va

$$\pi_1 \longrightarrow v_1 \longrightarrow \pi_2 \longrightarrow v_2 \longrightarrow \ldots \longrightarrow \pi^*$$

- **Policy search** algorithms use optimisation techniques to directly search for an optimal policy:

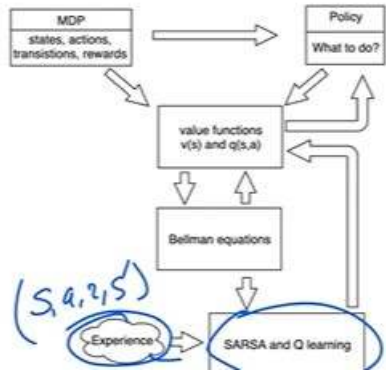
$$\pi_1 \longrightarrow \pi_2 \longrightarrow \ldots \longrightarrow \pi^*$$

Part 3: Model-free Methods: Monte Carlo, SARSA and Q-learning

Model-based versus Model-free

- Solving model-free RL problems
  - The agent has **no prior knowledge** about states, actions, rewards, transitions!
  - By **acting** in the world, the agent **gains experience**. An **experience** can be expressed as a 4-tuple: (*s, a, r, s´*).
  - Over time the agent collects a **list of experiences** that he uses **to find better policies** . . . HOW?
    - **Directly:** policy improvement.
    - **Indirectly:** estimate value functions, improve policy using greedification.

- Overview



- Definition of greedification
  - For a given **action-value function q(s, a)**, a corresponding **greedy policy** is a deterministic policy that picks (one of the) actions that **maximise the action value**:

$$\pi_g(s) = a^* := \arg\max_a q(s, a).$$

- Greedification results in policy improvement
  - To **improve the policy**, apply successive **iteration steps**:

    $$\pi_k \xrightarrow{eval} v_k, q_k \xrightarrow{greedify} \pi_{k+1}$$

  - Greedification implies **deterministic action choice** at each *s*:

    $$\pi_{k+1}(s) = a^* \quad \text{iff} \quad q_k(s, a^*) = \max_a q_k(s, a)$$

  - Hence **value function increases**, i.e. **policy has been improved**!

    $$\begin{aligned} v_{k+1}(s) &= q_k(s, a^*) \qquad (\pi_{k+1} \text{ is deterministic at } s) \\ &= \max_a q_k(s, a) \\ &\geq v_k(s) \qquad (= \textstyle\sum_{a'} \pi_k(a' \mid s) q_k(s, a')) \end{aligned}$$

- Greedification requires *q*(*s*, *a*)! model-free vs. model-based
  - **Greedification:**

    $$\pi(s) := \arg\max_a q(s, a)$$

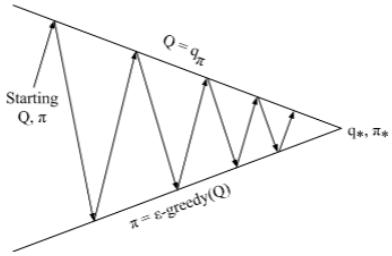
  - **Model-based:** (a.k.a. planning, search)
    - Suffices to **estimate value function *v*(*s*):**
    - ***q*(*s*, *a*) computed** with Bellman's **one-step look-ahead** (i.e. use back-up diagram):

      $$q(s, a) = \sum_{s'} p(s' \mid s, a) \left[r(s, a, s') + \gamma v(s') \right]$$

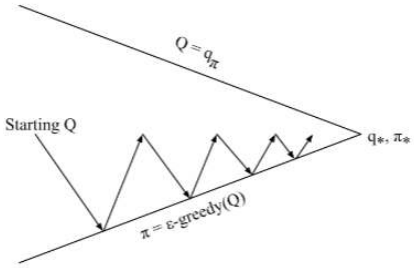
      ➔ **So in model-based it is sufficient to just look at v(s), but in model-free it is necessary to focus on q(s, a)!!!**
  - **Model-free:** (a.k.a. (optimal) control)
    - ***q*(*s*, *a*)** needs to be **estimated from collected experiences**;
    - algorithms shift **focus** from *v*(*s*) to *q*(*s*, *a*);

- Conclusion: for model-free RL
  - In contrast with to model-based, **model-free** RL is distinct:
    - **Focus on *q*(*s*, *a*)** rather than *v*(*s*).
    - Make sure **all state-action pairs (*s*, *a*) are sampled:** importance of **exploration**!

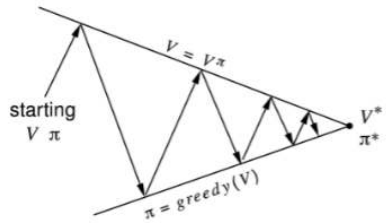
- Policy Iteration (PI) for Model-free RL (4:50)
  - **Goal: find optimal policy π∗ (aka optimal control)**
    - **PI**: Combine **policy evaluation** with **policy improvement**.
    - Introduce **exploration** in policy (e.g. π = ε-greedy(Q));



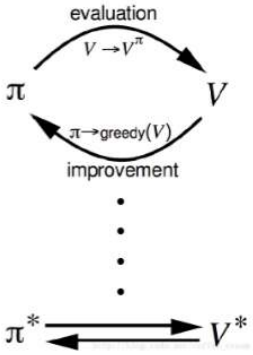
- Generalized Policy Iteration (GPI)
  - **Impatient greedification:** no insistence on making Q-function fully consistent with current policy π; → So you only do a couple of steps.
    - Hence Q ≈ q_π, but not necessarily Q = q_π;
  - **Ensure exploration**: use π = ε-greedy(Q), not π = greedy(Q);
  - Works if both processes **continue updating all states;**



- Generalized Policy Iteration (GPI) schematically



Policy evaluation Estimate v_π
Any policy evaluation algorithm
Policy improvement Generate $\pi' \geq \pi$
Any policy improvement algorithm



➔ So with Policy Iteration is mean the cycle of evaluating a policy and then improve it.

➔ Generalized means you can use any method to evaluate and any method to improve.

Monte Carlo for Model-free Policy Evaluation

- Monte Carlo methods
  - **Monte Carlo methods** are versatile statistical techniques for estimating properties of complex systems via **random sampling**;
  - Example 1: if X ∼p(x) and some function, then for any (sufficiently large) i.i.d. sample: X_1, X_2, ..., X_n:

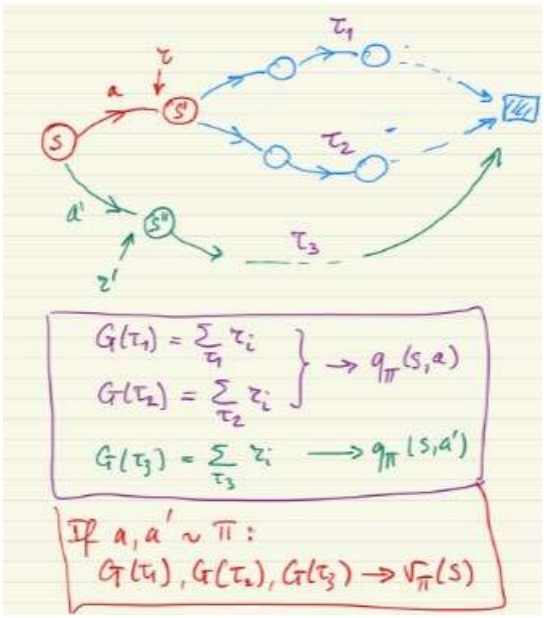
$$E(\varphi(X)) = \int \varphi(x)\, p(x) dx \quad \approx \quad \frac{1}{n} \sum_{X_i \sim p} \varphi(X_i)$$

  - Example 2: **Kullback-Leibler**

$$D_{KL}(f;g) = \int f(x) \log \frac{g(x)}{f(x)}\, dx \approx \frac{1}{n} \sum_{i=1}^{n} \log \frac{g(x^i)}{f(x^i)}$$

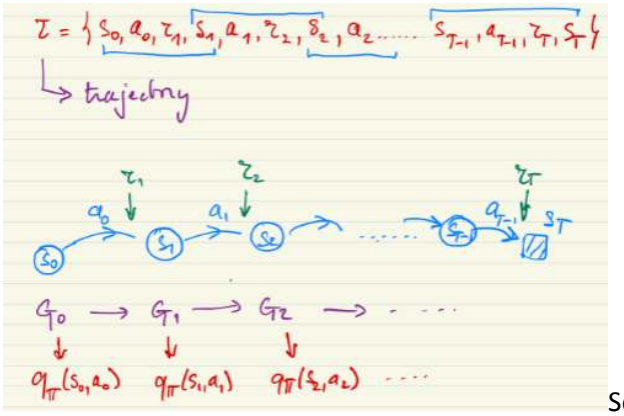
- Model-free **policy evaluation**: MC-based estimation of v_π and $q_\pi(s, a)$
  - Pick a **starting state** *s* (at **random** or systematic **sweep**).
  - **Estimation of $v_\pi(s)$**
    - Use **policy π** to generate the **initial and all subsequent actions** (till terminal state).
    - Compute total return $r_{tot} = r_1 + r_2 + ... + r_T$ along path:
    - r_{tot} yields **one sample value** for $v_\pi(s)$.
  - **Estimation of $q_\pi(s, a)$**
    - Apply **action** *a* in state *s* , observe reward r_1 and new state *s´*.
    - Use **policy π** to generate **all subsequent actions** (from s´ till terminal state).
    - Compute total return $r_{tot} = r_1 + r_2 + ... + r_T$ along path:
    - r_{tot} yields **one sample value** for $q_\pi(s, a)$.

- Monte-Carlo for model-free **policy evaluation**
  - Monte Carlo: using **sampled episodes to estimate *q*(*s*, *a*)!**



→ just sum all the rewards of each path. All of them can be used to compute the value function.

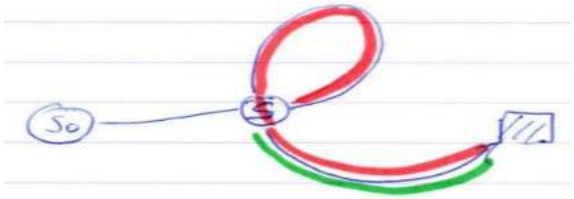
- MC estimation of *q*(*s*, *a*) along trajectory



So e.g. G_0 is the sum of all the rewards till ST and give you a q(s, a). And G_2 gives you an q(s,a) for all the summed rewards from state s_2 to ST.

➔ This is the Monte Carlo way of doing this.

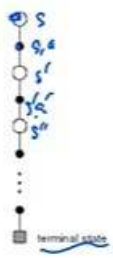
- MC policy evaluation: **first** versus **every visit** estimate
  - **First-visit MC:** average returns only for the first time *s* is visited in an episode.
  - **Every-visit MC:** average returns for every time *s* is visited in an episode:
    - **More sample efficient:** more samples per episode;
    - Samples **no longer independent**. (correlation between the values)
  - Both **converge asymptotically**.



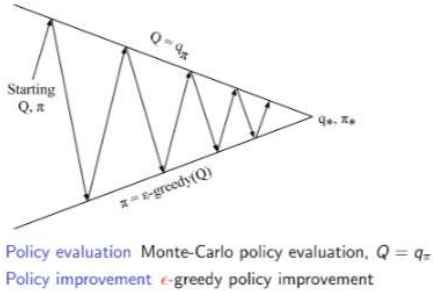
Both give same results for v(s)

- Monte-Carlo estimation *q*-values
  - Can learn q_π by averaging returns obtained when following π after taking action *a* in state *s*.
  - Converges asymptotically if **every (*s*, *a*) visited infinitely often**. So you must make sure that you sample each state-action pair.
    - Requires **explicit exploration** of actions **not favoured by π**.
    - **Possible solutions:**
      - **Exploring starts**: every (*s*, *a*) has a non-zero probability of being the starting pair.
      - **Soft policies**: $\pi(a|s) > 0$ for all (s, a). E.g. ε-greedy.

- Monte-Carlo backup diagram
  - Unlike dynamic programming, MC has only one choice at each state (i.e. the one actually taken!)
  - Unlike dynamic programming, entire episode included: **MC does not bootstrap**.
  - **Bellman equations** are NOT used!



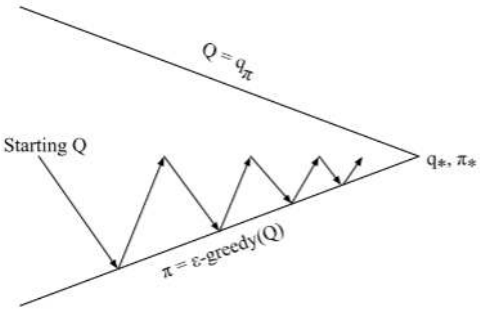
- Model-free MC: from evaluation to control
  - **Control: Find optimal policy π*:**
    - Combine **MC-based policy evaluation** with **improvement** (e.g. greedification).
    - Introduce **exploration** in policy.
      - (e.g. ε-greedy)



Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$
Policy improvement ϵ-greedy policy improvement

➔ The control is the **q*, π***

- Monte Carlo for Model-free Control (2)



Every episode:
Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$
Policy improvement ϵ-greedy policy improvement

- Monte-Carlo for model-free estimation and control
  - MC provides one way to perform **model-free reinforcement learning** (RL): finding optimal policies without an explicit model for the MDP;
  - MC for RL learns from **complete sample returns** in episodic tasks:

- Computes **value functions** using **direct sampling** rather than Bellman equations;

- Convergence condition (MC →gives v, q values): Greedy in the Limit with Infinite Exploration (GLIE) → Not something to concern about just technical details!
  - **Def (GLIE)**
    - All state-action pairs are explored infinitely often:

      $$\lim_{t\to\infty} N_t(s,a) = +\infty.$$

    - The policy converges to a greedy policy:

      $$\lim_{t\to\infty} \pi(a\,|\,s) = \begin{cases} 1 & \text{if} \quad a = \arg\max_{a'} q(s,a') \\ 0 & \text{otherwise} \end{cases}$$

  - **Example:** ε-greedy is GLIE iff ε ↓ 0.

- GLIE Monte Carlo Model-free Control (skipped)
  - Sample kth episode using π: $\{S_1, A_1, R_2, ..., S_T\} \sim \pi$
  - For each state S_t and action A_t in the episode,

    $$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

    $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\left(G_t - Q(S_t, A_t)\right)$$

  - Improve policy based on new action-value function

    $$\epsilon \leftarrow 1/k$$

    $$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

Theorem

GLIE Monte-Carlo control converges to the optimal action-value function, $Q(s,a) \to q_*(s,a)$

- Control based on MC + exploring starts: Algorithm (skipped)

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
  Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Temporal Difference Methods for Model-free Prediction

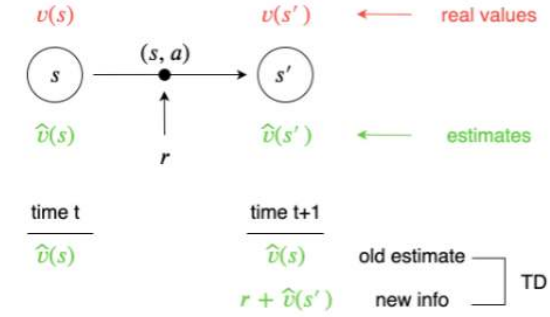
- Solving model-free RL using **Temporal Differencing**
  - By **acting in the world**, the agent gains experience.

        - An experience can be expressed as a 4-tuple: (s, a, r, s ′).
    - Over time the agent collects a list of experiences that he uses to find value functions (and corresponding policy) … HOW?
    - **Key observations:**
        - Bellman eqs. link values in neighbouring states along paths!
        - This backing-up improves learning efficiency!
        - Basis for RL algo's (SARSA, Q-learning, etc).

        Temporal Differencing (TD): Use Bellman eqs to propagate values!

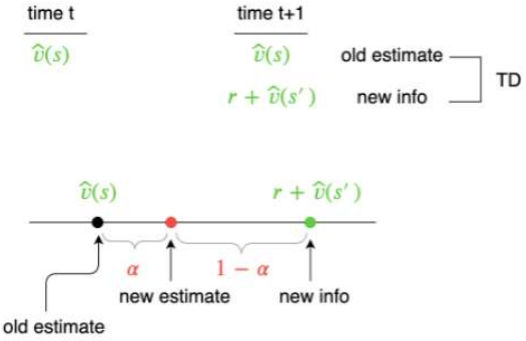
        ➔ So with MC you just collect all the rewards and with TD you use the Bellman equations.

- Temporal-differencing: Exploiting Bellman equations



The 1-step reward r is new information gained from experience.

- Temporal differencing for v : Exploiting Bellman eqs.



$$\hat{v}_{\text{new}}(s) = (1 - \alpha)\hat{v}_{old}(s) + \alpha\big(r + \hat{v}(s')\big)$$

alpha is the learning rate.

- Temporal Differencing (TD) versus Monte Carlo (MC)
    - Given learning rate α:
        - **MC update rule:**

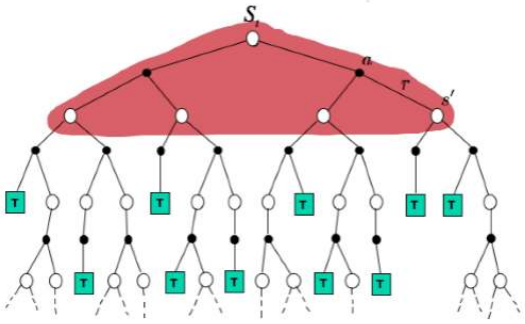
$$v_\pi^{new}(S_t) \leftarrow (1-\alpha)v_\pi^{old}(S_t) + \alpha G_t(S_t)$$
$$v_\pi^{new}(S_t) \leftarrow v_\pi^{old}(S_t) + \alpha\left[G_t(S_t) - v_\pi^{old}(S_t)\right]$$

- **TD update rule:** uses a different <u>update target</u>:

$$v_\pi^{new}(S_t) \leftarrow (1-\alpha)v_\pi^{old}(S_t) + \alpha[R_{t+1} + \gamma v_\pi(S_{t+1})]$$
$$v_\pi^{new}(S_t) \leftarrow v_\pi^{old}(S_t) + \alpha[R_{t+1} + \gamma v_\pi(S_{t+1}) - v_\pi^{old}(S_t)]$$

  - TD is a **bootstrapping methods:** bases updates on existing estimates, like DP.

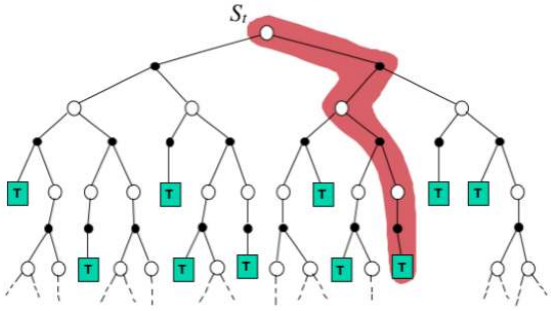
- Dynamic Programming (DP): Backup diagram



Model-based

- Monte-Carlo (MC): Backup diagram

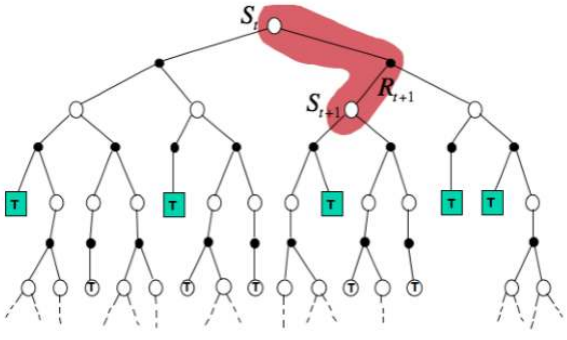
$$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$$



Model-free

- Temporal Difference (TD): Backup diagram

$$V(S_t) \leftarrow V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



the $v(s_t)$ after the arrow is the old one. / Model-free with Bellman equations.

- Temporal-difference (TD) methods
  - **DP** exploits Bellman equation but **requires model**.
  - **MC** doesn't require model but **doesn't exploit Bellman equation**.

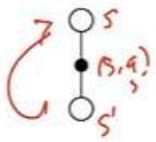
- **TD methods** can get the best of both worlds: **exploit Bellman equation without requiring a model**.
- TD is therefore **core algorithm** for **model-free RL**.

- Policy evaluation: Estimating v_π using TD(0)

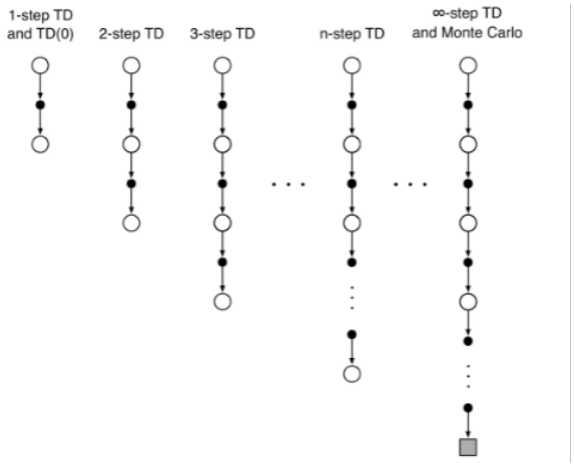
Initialize $V(s)$ arbitrarily, π to the policy to be evaluated
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    $a \leftarrow$ action given by π for s
    Take action a; observe reward, r, and next state, s'
    $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$
    $s \leftarrow s'$
  until s is terminal

episode is just a number of steps. From s to s' is one step.

- TD(0) backup diagram
  - Sampling: unlike DP but like MC, only one choice at each state.
  - **Bootstrapping:** like DP but unlike MC, use **estimate** from next state.
    - ➔ Propagate information back to *s*.



- **<u>Advantages</u> of TD prediction methods**
  - TD methods require only experience, not a model.
  - TD, but not MC, methods can be **fully incremental**.
  - Learn **before knowing the final outcome:**
    - **Efficient:** less memory and peak computation.
    - Learn from **incomplete sequences**.
  - Both MC and TD converge but TD tends to be faster.
- n-step TD: Spectrum between TD(0) and Monte Carlo (skipped)



- Issues when addressing model-free RL problems
  1. **We need to compute** *q*(*s*, *a*) **rather than** *v*(*s*)

- To **improve policy**, for every s, need to know $\max_a q(s,a)$:

  construct $\pi : s \mapsto a_{max}$ where $q(s, a_{max}) = \max_{a'} q(s, a')$

  - **Model-based:** $q(s,a)$ can be computed from $v(s)$:

    $$q(s,a) = r(s,a,s') + v(s')$$

  - **Model-free:** $q(s,a)$ needs to be estimated explicitly!

  Hence: SARSA and Q-learning focus on $q(s,a)$

2. **Keep exploring!**
   - Balance exploration versus exploitation
   - E.g. ε-greedy, soft-max, etc.

SARSA and Q-learning for Model-free Control

- Apply TD methods for estimating $q(s, a)$
  - **TD methods** exploit correlations between **successor** states:
  - **State value function** $v_\pi(s)$

    $$v_\pi(S_t) \longleftarrow v_\pi(S_t) + \alpha(\underbrace{R_{t+1} + \gamma v_\pi(S_{t+1})}_{\text{new info}} - v_\pi(S_t))$$

  - **State value function** $q_\pi(s, a)$:
    - **SARSA: policy evaluation:**

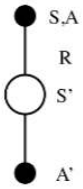
      $q_\pi(s,a) \leftarrow ??$

    - **Q-learning: optimal** state-action value:

      $q^*(s,a) \leftarrow ??$

      ➔ So SARSA is like TD but then computing the $q(s, a)$ instead of the v(s)

- SARSA: TD estimation of Q-values



  - **SARSA:** in state s, taken action a, transition to state s', use policy π to select next action a'.
  - **Bellman:** Two estimates for state-action value:

    $q_\pi(s,a)$ and $r(s,a,s') + q_\pi(s',a')$

    ➔ So you now have two estimates: one is the original $q(s, a)$ and the other is from going to new s' with certain action that give a reward see above.

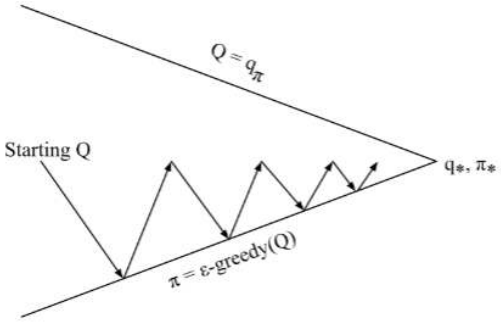
- SARSA: TD for action values
  - **TD:** bootstrap update for **value function** v_π

$$v_\pi(s) \longleftarrow v_\pi(s) + \alpha(\underbrace{r(s,a,s') + \gamma v_\pi(s')}_{\text{new info}} - v_\pi(s))$$

- **SARSA:** bootstrap update for **action value function** q_π

$$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha(\underbrace{r(s,a,s') + q_\pi(s',a')}_{\text{new info}} - q_\pi(s,a))$$

- SARSA model-free control: Schematically



Every time-step:

Policy evaluation Sarsa, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

- SARSA: Pseudo-code for model-free control

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

- SARSA convergence (not exam material)

Theorem

Sarsa converges to the optimal action-value function, $Q(s,a) \to q_*(s,a)$, under the following conditions:

- *GLIE sequence of policies $\pi_t(a|s)$*
- *Robbins-Monro sequence of step-sizes α_t*

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$
$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

alpha overtime should decrease.

- Applying TD to model-free RL problems
  - **Recall:** model-free, hence focus on $q(s, a)$ and use Bellman!

1. **SARSA**: Evaluating a given policy π
   - $q_\pi(s,a) = \sum_{s'} p(s' \mid s,a)\Big[r(s,a,s') + \sum_{a'} \pi(a' \mid s') q_\pi(s',a')\Big]$
   - **Sample:** $q_\pi(s,a) = r(s,a,s') + q_\pi(s',a')$
   - SARSA update rule (sample-based):

$$q_\pi(s,a) \leftarrow (1-\alpha) q_\pi(s,a) + \alpha(r(s,a,s') + q_\pi(s',a'))$$

2. **Q-learning:** Estimating the **optimal** value function $q^*(s,a)$
   - $q^*(s,a) = \sum_{s'} p(s' \mid s,a)\Big[r(s,a,s') + \max_{a'} q^*(s',a')\Big]$
   - **Sample:** $q^*(s,a) = r(s,a,s') + \max_{a'} q^*(s',a')$
   - Q-learning update rule (sample-based):

$$q^*(s,a) \leftarrow (1-\alpha) q^*(s,a) + \alpha(r(s,a,s') + \max_{a'} q^*(s',a'))$$

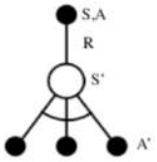
➔ Note that the first bullet-point under SARSA is the Bellman equation and see how it is transformed to the SARSA update rule.

➔ In Q-learning you try to compute q^* immediately: youd do that by using the Bellman equation for q^*.

➔ So it's exactly the same but with Q-learning you try to estimate q^* directly.

➔ So if the $q^*(s, a)$ is equal to $r(s, a, s')$ + max… (see above fifth bullet-point) than there is no updating anymore and then that is q^*.

- Q-learning: Bootstrap with **best** action, not actual action → another way to write the above



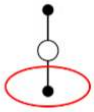
$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_{a'} Q(S',a') - Q(S,A)\right)$$

Convergence when TD-error vanishes:
Recall: optimality equation for q^*:

$$q^*(s,a) = \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \gamma \max_{a'} q^*(s',a')\right]$$

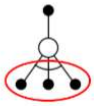
- SARSA vs. Q-learning: On-policy vs. Off-Policy

Sarsa:



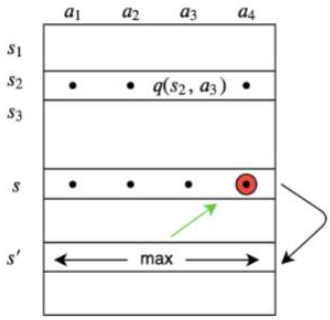
$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)]$$

Q-learning:

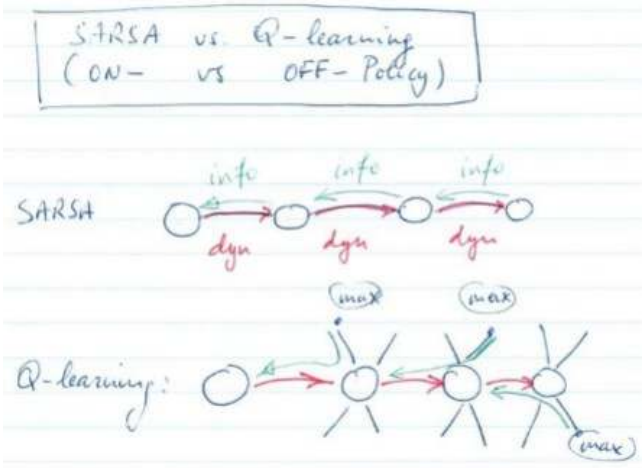


$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right]$$

- Q-learning



- SARSA (on-policy) vs. Q-learning (off-policy)



  - **Q-learning:** dynamics policy different from **backup** policy.

- ON-policy vs. OFF-policy
  - **ON-policy**:
    - Info gathered to improve policy, depends on that policy;
      Feedback loop: policy ⟷ data
    - **Sample inefficient:** experiences (s, a, r, s´, a´) cannot be re-used when policy changes since a´ depends on actual policy!
  - **OFF-policy**
    - **Improved sample efficiency:** can re-use all samples for training;
    - E.g. in Q-learning: use **data generated by dynamics policy π** to learn value function for optimal policy π*.

Lecture 12

- **Recall:**
  - In model-free it is important to directly focus on $q(s, a)$. One of the key insides is TD that exploit the Bellman equations. Alternatives are the MC, with roll-outs. SARSA and Q-learning are based on using the Bellman equations to optimise the search.

- Applying TD to model-free RL problems → **These two slides are IMPORTANT!!! Make sure you understand and know all the formula's and facts.**
  - **SARSA: Evaluating $q_\pi(s, a)$ for a given policy π**

- $q_\pi(s,a) = \sum_{s'} p(s' \mid s,a)\left[r(s,a,s') + \sum_{a'} \pi(a' \mid s') q_\pi(s',a')\right]$
- **Sample:** we expect: $q_\pi(s,a) \approx r(s,a,s') + q_\pi(s',a')$
- **TD error:** $\delta = r(s,a,s') + q_\pi(s',a') - q_\pi(s,a)$
- SARSA update rule (sample-based):

$$q_\pi(s,a) \leftarrow (1-\alpha) q_\pi(s,a) + \alpha(r(s,a,s') + q_\pi(s',a'))$$

$$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha(r(s,a,s') + q_\pi(s',a') - q_\pi(s,a))$$

- **ON Policy:** TD error depends on a' (generated by policy π).
- SARSA is evaluation method (followed by improvement);

➔ ON policy as the TD error is influenced by the policy itself.

➔ **These are the important facts you should know about SARSA!!!**

- **Q-learning: Estimate optimal value function q*(s, a) directly!**
  - $q^*(s,a) = \sum_{s'} p(s' \mid s,a)\left[(s,a,s') + \max_{a'} q^*(s',a')\right]$
  - **Sample:** we expect $q^*(s,a) \approx r(s,a,s') + \max_{a'} q^*(s',a')$
  - **TD error:** $\delta_Q = r(s,a,s') + \max_{a'} q^*(s',a') - q^*(s,a)$
  - Q-learning update rule (sample-based):

$$q^*(s,a) \leftarrow (1-\alpha) q^*(s,a) + \alpha(r(s,a,s') + \max_{a'} q^*(s',a'))$$

$$q^*(s,a) \leftarrow q^*(s,a) + \alpha(r(s,a,s') + \max_{a'} q^*(s',a') - q^*(s,a))$$

  - **OFF Policy:** TD error does NOT depend on policy.

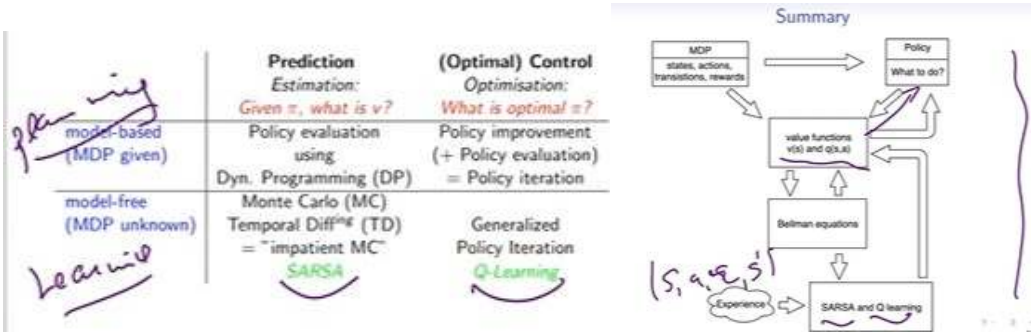
- Will QL learn the optimal policy
  - Potential problems:
    - Hypothesis space created by network does not contain optimal policy.
    - Nonconvex optimisation – multiple suboptimal min.
    - Might be slow, beyond computation limit.
    - But reaching optimum is possible.

- Q-learning ALGO: off-policy TD control
  - Make TD off-policy: bootstrap with best action, not actual action:

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

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$;
  until s is terminal

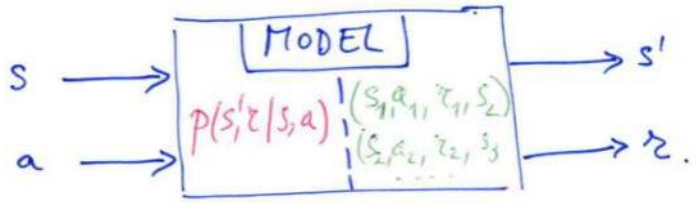
- Summary: Model-based versus Model-free



➔ Its not that planning and learning are completely different things, you can combine them. Which is explained now:

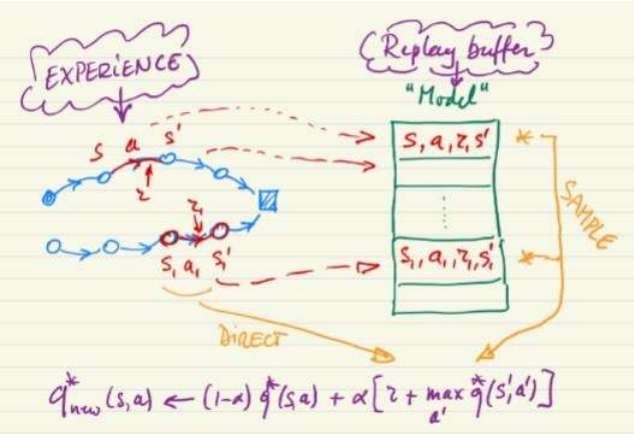
Integrating planning and learning

- Dyna-Q; Integrating planning and learning
  - **Model:** tells agents what will happen next…
    - **Model-based:** planning
    - **Model-free:** learning
  - **Distributional vs. Sample-based Model:**



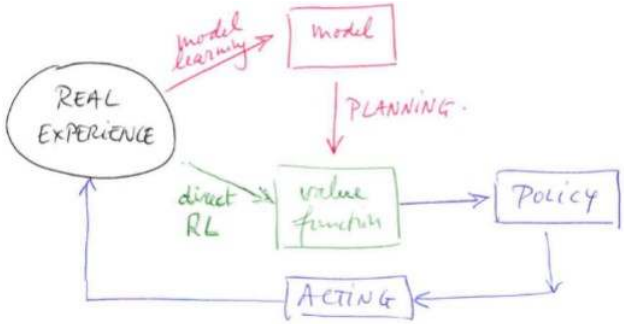
- Model-based learning
  - Until now: **Model = fully specified MDP**!
  - More generally: **anything that helps the agent to plan:**
  - More **accurate models** are **more effective** (myths vs. science):
    - **Folklore** and weather saying: "*A wet and windy May fills the barn with corn and hay.*"
    - Meteorological models running on **supercomputer**.

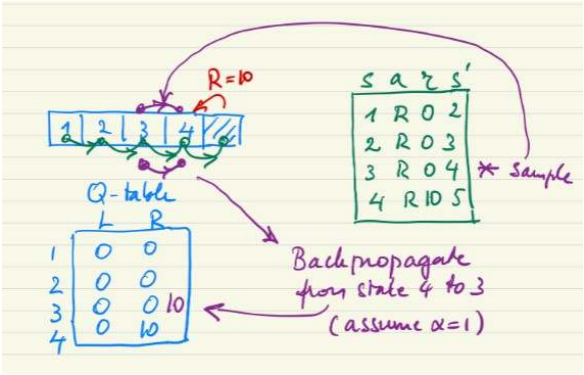
- Dyna-Q; Integrating **planning** and **learning**



So you store every experience in a database and then you sample from this database and do as if this is new experience and use this sample to update the *q** equation.

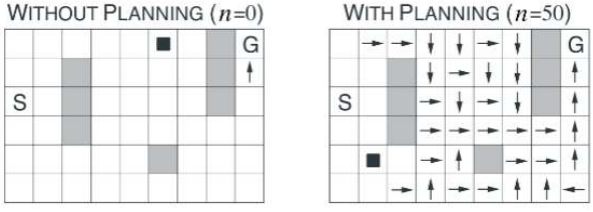
- **Real experience** can be used in **two ways**:





So purple indicates the updated information. That is a back propagated value. So you update the value by choosing the maximised value of state *s'*.

- DYNA-Q: Maze example
  - Greedy policy midway through 2nd episode:



➔ So here you can see by back propagating the value it save the information and after *n* = 50 times you get a sort of an idea how to travel. → gain a lot of efficiency.

- DYNA-Q: Algorithm

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Execute action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

- Prioritized Sweeping
  - **Uniform sampling** from experience database: **wasteful!** ;e.g. above (2nd) example, most samples will give zero zero.
  - Idea: **work "backwards" from "goal state"**. → called prioritize sweeping.
  - More generally: **backward focussing:** work backwards from any state the value of which has changed!
  - … but prioritize: Most significant changes first!

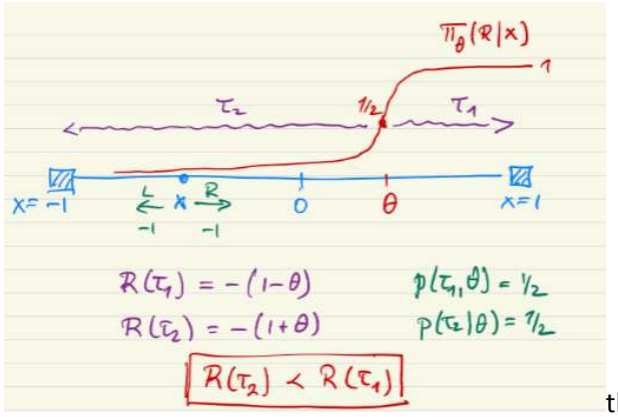
Part 4: Policy gradient algorithm and Deep Q-Learning

- **Policy gradient algorithms**
  - **Policy gradient algorithms**
    - Optimise policy directly,
    - NOT via value function (indirectly). So the above methods used the value functions to optimise the policy.
  - **Ingredients:**
    1. Parametrised policy $\pi_\theta(a|s)$ (θ to be determined).
    2. Objective function $J(\theta)$ (expected reward of a path) to be maximised.
    3. Update rule: $\theta_{new} \leftarrow \theta_{old}$, specifically gradient ascent:
$$\theta_{new} \leftarrow \theta_{old} + \nabla_\theta J(\theta_{old})$$

- Policy gradient: Objective Function
  - **Trajectory** (episodic):
$$\tau = \{s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots\ldots, s_{T-1}, a_{T-1}, r_T, s_T\}$$
  - **Cumulative return along trajectory** → you're interested in the reward after long path.
$$R(\tau) := \sum_{t=1}^{T} \gamma^{t-1} r_t \qquad \text{or} \qquad R_s(\tau) := \sum_{t=s}^{T} \gamma^{t-s} r_t$$
  - **Objective function J(θ):** Quantifying policy performance
$$J(\theta) := \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$$
  - **Goal:**
$$\max_\theta J(\theta)$$

➔ So the max reward, by comparing all the trajectories.

- Policy gradient: Example(1)
  - Absorbing states at x = ±1, absorption reward = 0.



the policy depends on the parameter theta. So the optimal value of theta would be 0. As being positive it would say go to the right and negative go to the left. Because with every step it takes you an energy of -1. So to optimise your energy-level you should be at 0. Reward is -1 times the distance (in this case 1 + theta, for T2, as you can see in the picture). Its also just what your policies says: it tells you to go left or right.

- Policy gradient theorem
  - **Goal: maximise objective function**

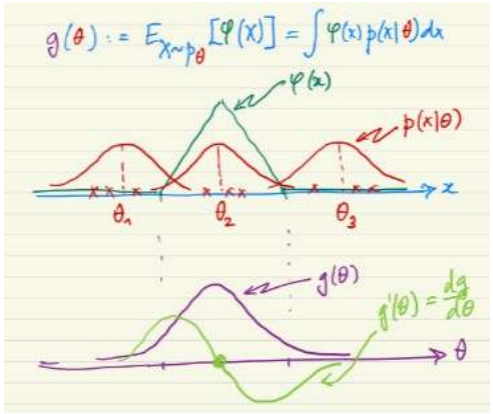
$$J(\theta) := \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \int R(\tau)p(\tau\,|\,\theta)\,d\tau$$

  - In **abstract terms** (to simplify notation): X is a stochastic variable.

$$g(\theta) := \mathbb{E}_{X\sim p_\theta}[\phi(X)] = \int \phi(x)p(x\,|\,\theta)\,dx$$

  - Need to **compute derivate** (to optimise):

$$\frac{d}{d\theta}g(\theta) = \frac{d}{d\theta}\int \phi(x)p(x\,|\,\theta)\,dx = \int \phi(x)\frac{d}{d\theta}(p(x\,|\,\theta))\,dx$$



Optimal value $\theta^* = \theta_2$

The figures are the most important. See optimal theta. Gradient is the derivative. Max and gradient the same.

$$\frac{d}{d\theta} g(\theta) = \int \phi(x) \frac{d}{d\theta} (p(x \mid \theta))\, dx$$

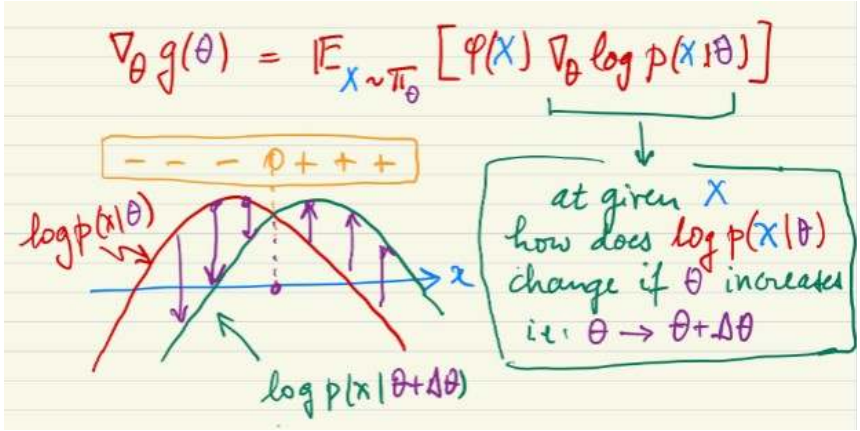
$$= \int \phi(x) \left[\frac{\frac{d}{d\theta} (p(x \mid \theta))}{p(x \mid \theta)} \right] p(x \mid \theta)\, dx$$

$$= \int \phi(x) \left[\frac{\frac{d}{d\theta} (p(x \mid \theta))}{p(x \mid \theta)} \right] p(x \mid \theta)\, dx$$

$$= \int \phi(x) \left[\frac{d}{d\theta} (\log p(x \mid \theta)) \right] p(x \mid \theta)\, dx$$

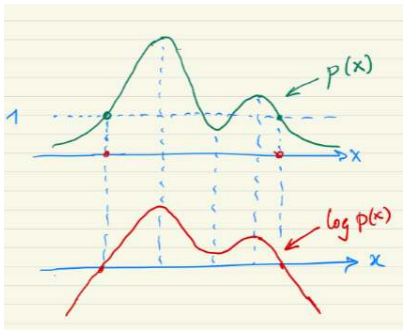
$$= \mathbb{E}_{X \sim p_\theta} \left[\phi(X) \frac{d}{d\theta} (\log p(x \mid \theta)) \right]$$

$$\boxed{\frac{d}{d\theta} \mathbb{E}_{p_\theta} [\phi(X)] = \mathbb{E}_{p_\theta} \left[\phi(X) \frac{d}{d\theta} (\log p(X \mid \theta)) \right]}$$

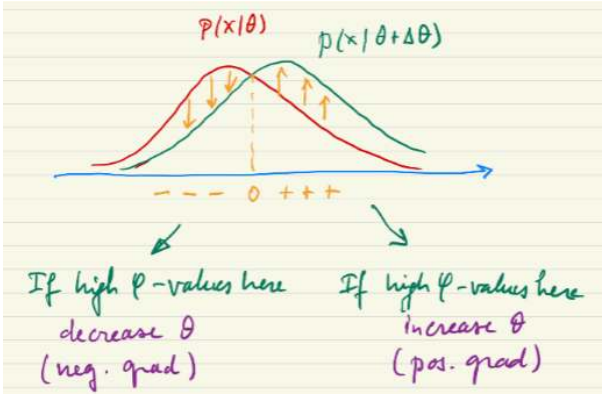


The gradient tells you how a function changes when theta is changed.

- *P*(x) and log *p*(x) have same local extremes, increasing and decreasing behaviour.
- $\text{sgn}(\nabla_\theta p(x|\theta)) = \text{sgn}(\nabla_\theta \log p(x|\theta))$



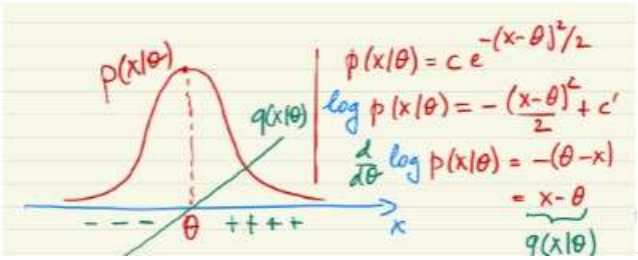
- Policy gradient theorem



Lecture 13

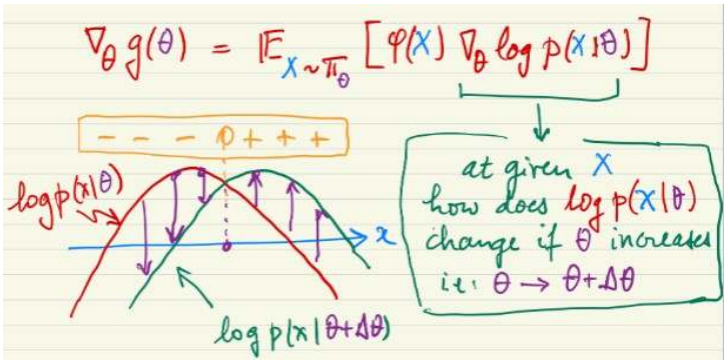
- Policy gradient theorem

$$\begin{aligned}\frac{d}{d\theta}g(\theta) &= \frac{d}{d\theta}\mathbb{E}_{X\sim p_\theta}[\phi(X)] \\ &= \mathbb{E}_{X\sim p_\theta}\left[\phi(X)\frac{d}{d\theta}(\log p(X\mid\theta))\right] \\ &\approx \frac{1}{n}\sum_{X_i\sim p_\theta}\phi(X_i)\frac{d}{d\theta}(\log p(X_i\mid\theta)) \\ &= \frac{1}{n}\sum_{X_i\sim p_\theta}\phi(X_i)q(X_i\mid\theta)\end{aligned}$$

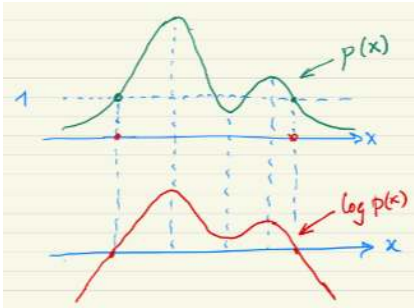


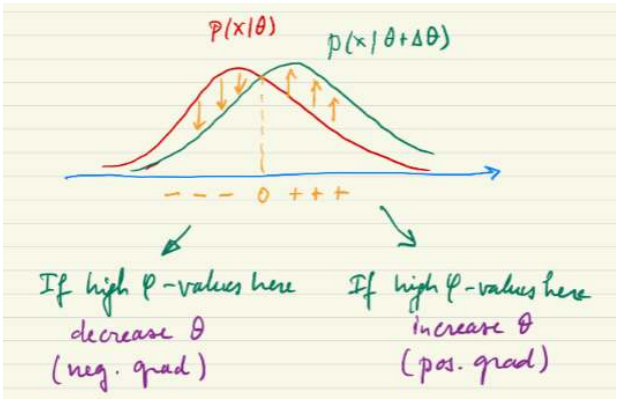
So from the function you first take the log and then the derivative. Which give you the function q(x|theta).

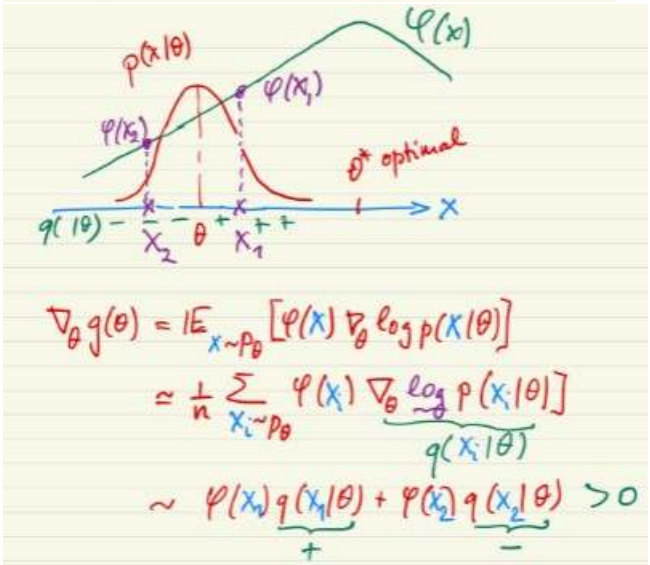
- Policy gradient theorem



  - *P*(x) vs log *p*(x)
    - P(x) and log p(x) have same local extremes, increasing and decreasing behaviour.
    - sgn(∇θp(x|θ)) = sgn(∇θ log p(x|θ)).



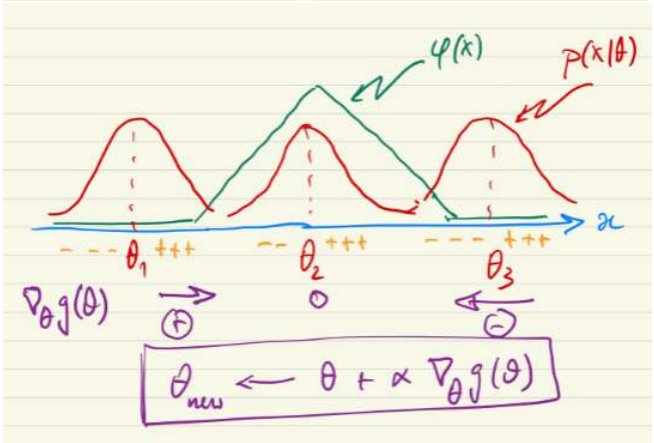




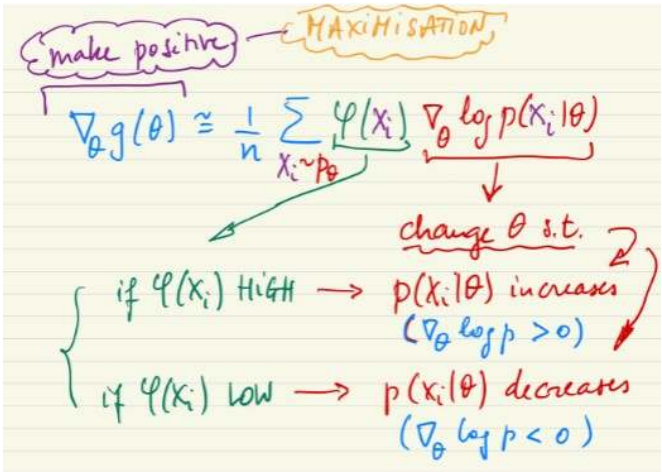
$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim p_\theta}\left[\varphi(x)\,\nabla_\theta \log p(x|\theta)\right]$$

$$\approx \frac{1}{n}\sum_{x_i\sim p_\theta} \varphi(x_i)\,\nabla_\theta \log p(x_i|\theta)$$

$$\sim \varphi(x_1)\,q(x_1|\theta) + \varphi(x_2)\,q(x_2|\theta) > 0$$



So the left you need to increase theta and the right you need to decrease theta to be like the green graph function.



➔ So if a given value x are high than you need to increase theta and if a given value x is low than you need to decrease theta. Also the same as above picture but than written differently.

- Policy gradient theorem
  - General result

$$\nabla_\theta \mathbb{E}_{X \sim p_\theta}[\phi(X)] = \mathbb{E}_{X \sim p_\theta}[\phi(X) \nabla_\theta \log p(X \mid \theta)]$$

  - Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau) \nabla_\theta \log p(\tau \mid \theta)]$$

- If $R(\tau)$ high, change θ to make τ MORE likely, i.e. $p(\tau \mid \theta) \uparrow$
- If $R(\tau)$ low, change θ to make τ LESS likely, i.e. $p(\tau \mid \theta) \downarrow$

$$p(\tau \mid \theta) = \prod_{t \geq 0} p(s_{t+1} \mid s_t, a_t) \pi_\theta(a_t \mid s_t)$$

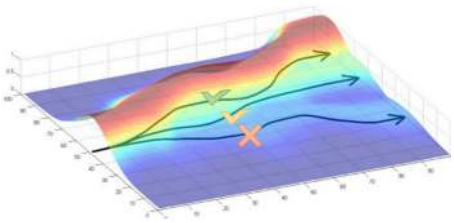
$$\log p(\tau \mid \theta) = \sum_{t \geq 0} [\log p(s_{t+1} \mid s_t, a_t) + \log \pi_\theta(a_t \mid s_t)]$$

$$\nabla_\theta \log p(\tau \mid \theta) = \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

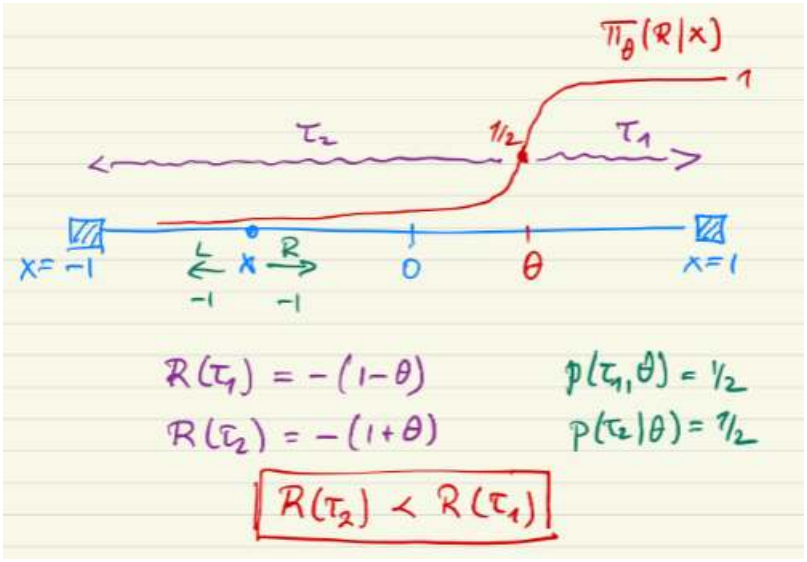
Policy Gradient Theorem

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau) \nabla_\theta \log p(\tau \mid \theta)] \\ &= \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \end{aligned}$$

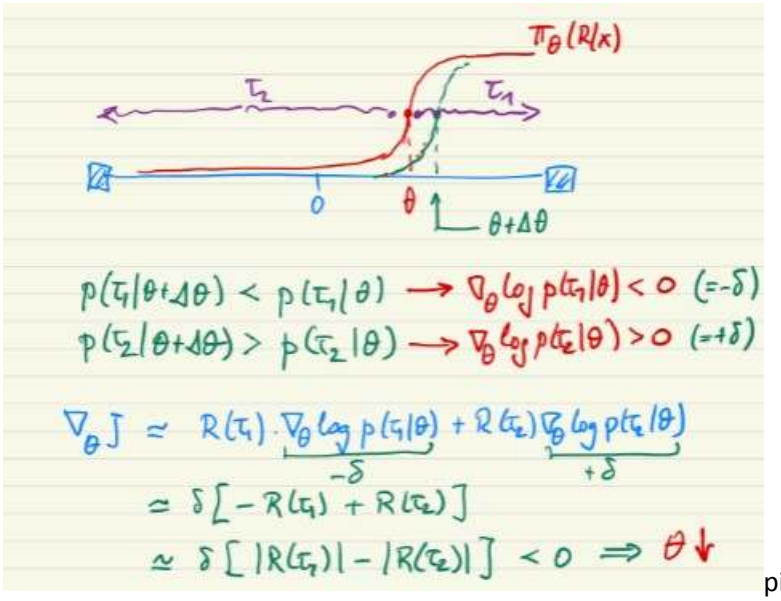
- Policy gradient illustration



- Policy gradient: Example(1)



- Policy gradient: Example(2)



pie theta (R|x) is the policy.

- ➔ If you increase theta the curve shifts to the right, then it becomes the green one. Probability will then be lower for path T1 and is thus less likely. So then if the theta shifts to the green one than the derivative for T1 will be smaller and for T2 bigger. Gradient is negative due to the energy that it costs?
- ➔ It the exam it would e.g. asks with given the formula and figure. Why do we go to the left.
- ➔ So also positioning theta at 0 would be the optimal policy as the paths are of both lengths → because you want to or the left or the right absorbing state. So in the picture above it is not optimal because the probability is the same, but T2 has a smaller path, thus higher reward.
- ➔ So again from the red theta (assume your starting point, focus on T1), you have probability to go ½ left and ½ right. So assume you go from state s to state s' at the right hand side than the next probability of going left or right would be almost 1, as you can see on the right

hand the line goes horizontal. (the blue line indicates the steps). Thus from then on you can assume that the next steps to the right are all with a probability of 1. So if you know increase theta (green curve) the probability will be smaller: say ¼ if you want to move to the right. Then you get the following probability trajectory: ¼ * ½ * 1 * 1 etc. So your probability will become smaller.

- REINFORCE algorithm; example of policy gradient algorithm
  - Does exactly what has been explained above:

1. Initialise learning rate α
2. Initialise parameter θ of policy π_θ
3. **for** episode $= 1 \ldots$ NR_EPISODES:
4. Sample trajectory $\tau = \{s_0, a_0, r_1, s_1, \ldots, r_T, s_T\}$
5. Set $\nabla_\theta J(\theta) = 0$
6. # add gradient contributions along trajectory
7. **for** $t = 0, 1, \ldots, T$:
8. $R_t(\tau) = \sum_{u=t}^{T} \gamma^{u-t} r_u$,
9. $\nabla_\theta J(\theta) = \nabla_\theta J(\theta) + R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)$
10. # update policy parameter
11. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

- Improve REINFORCE algorithm
  - Policy gradient estimate has **high variance** (trajectories might be very different!)

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

  - **Variance reduction** by introducing **action-independent baseline:**

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{T} (R_t(\tau) - b(s_t)) \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

  - Example: **Actor-Critic algorithm:**

$$R_t(\tau) = q_{\pi_\theta}(s_t, a_t) \qquad \text{and} \qquad b(s_t) = v_{\pi_\theta}(s_t)$$

Actor-Critic: Combining value-and policy-based learning

- Advantage Actor-Critic (A2C)
  - **Actor-Critic** combines **value-based** and **policy-based** learning.
  - **Actor-Critic** algorithms therefore have two components that are **learned jointly:**
    - **Actor:** learns a parametrised policy.
    - **Critic:** learns value function to evaluate state-action pairs;
  - **Advantage function:** select action based on how it performs relative to other actions in that state:

$$a_\pi(s, a) := q_\pi(s, a) - v_\pi(s)$$

- Quick recap
  - **Policy gradient along trajectory τ**

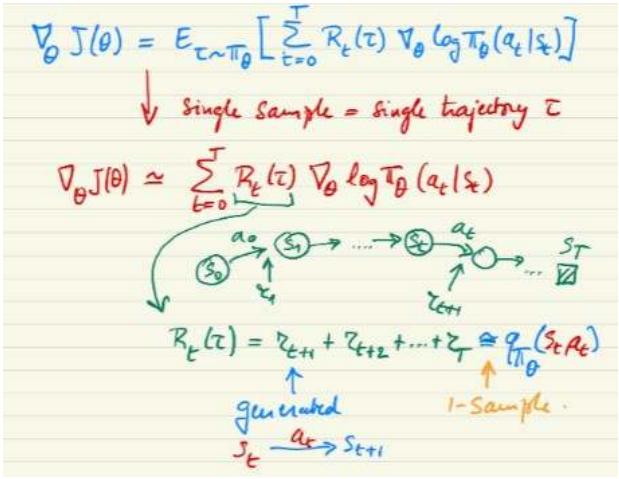
$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

  - Restricting the trajectory to the part starting at s_t

$$R_t(\tau) = R(s_t, a_t, \ldots, s_T) \implies \mathbb{E}_{\tau\sim\pi_\theta}[R_t(\tau)] = q_{\pi_\theta}(s_t, a_t);$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} q_{\pi_\theta}(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

- Policy Gradient via Monte Carlo Sampling



So Rt is actually an estimation of *q*(s, a) under a certain policy.

- Policy gradient: Quick Recap

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \quad \text{(MC)}$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} q_{\pi_\theta}(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \quad \text{(value fct)}$$

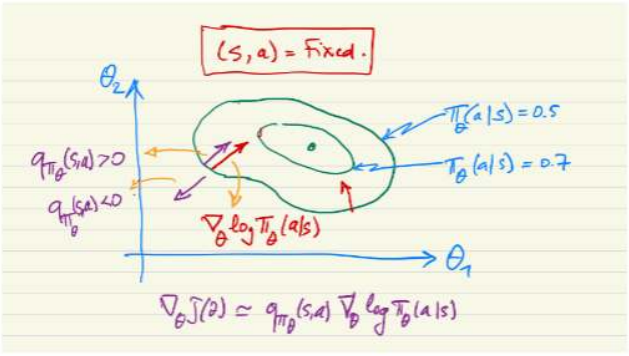
For N sampled paths $\tau_i = \{s_{i,0}, a_{i,0}, s_{i,1}, r_{i,1}, \ldots\}$:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T} R_t(\tau_i)\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\right]$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T} q_{\pi_\theta}(s_{i,t}, a_{i,t})\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})\right]$$

So rewriting of Rt to *q*.

- Gradient Policy theorem



Changing θ in the direction of $\nabla_\theta \log \pi_\theta(a \mid s)$ makes it more likely that action a will be chosen by policy π_θ. That is a good thing if $q(s, a)$ positive/large!

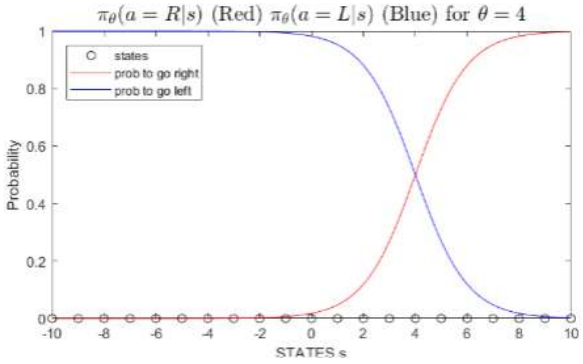
- Actor-Critic Methods

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \underbrace{\pi_\theta(a_t \mid s_t)}_{\text{ACTOR}} \underbrace{q_{\pi_\theta}(s_t, a_t)}_{\text{CRITIC}} \right]$$

  - **Critic** estimates the value function (could be action-value *q* or state-value *v* function).
  - **Actor** updates the policy distribution in the direction suggested by the critic.

- Worked example

$$\pi_\theta(a = R \mid s) = \frac{e^{s-\theta}}{1 + e^{s-\theta}} \qquad \pi_\theta(a = L \mid s) = \frac{1}{1 + e^{s-\theta}}$$



- Worked example, computation

$$\sigma(x) = \frac{e^x}{1+e^x} \Rightarrow \pi_\theta(a=R|s) = \sigma(s-\theta)$$

$$\frac{d}{dx}\log\sigma(x) = \frac{\sigma'(x)}{\sigma(x)} = \frac{\frac{e^x(1+e^x)-(e^x)^2}{(1+e^x)^2}}{\frac{e^x}{1+e^x}}$$

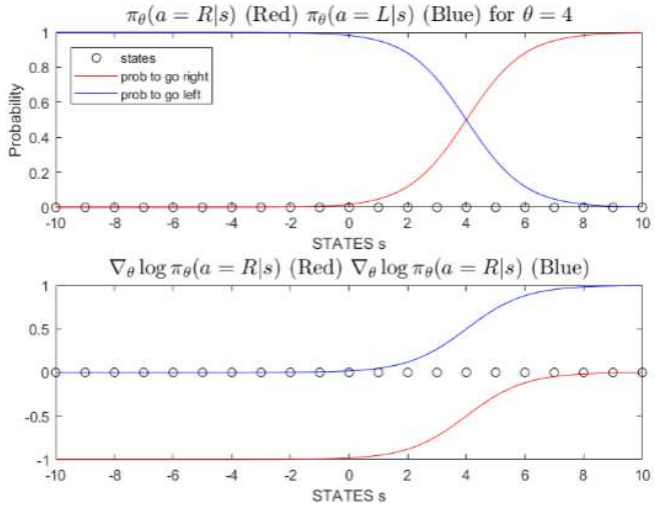
$$= \frac{1}{1+e^x}$$

$$\frac{d}{d\theta}\log\pi_\theta(a=R|s) = \frac{d}{dx}\log\sigma(x)\cdot\underbrace{\frac{dx}{d\theta}}_{=-1}$$

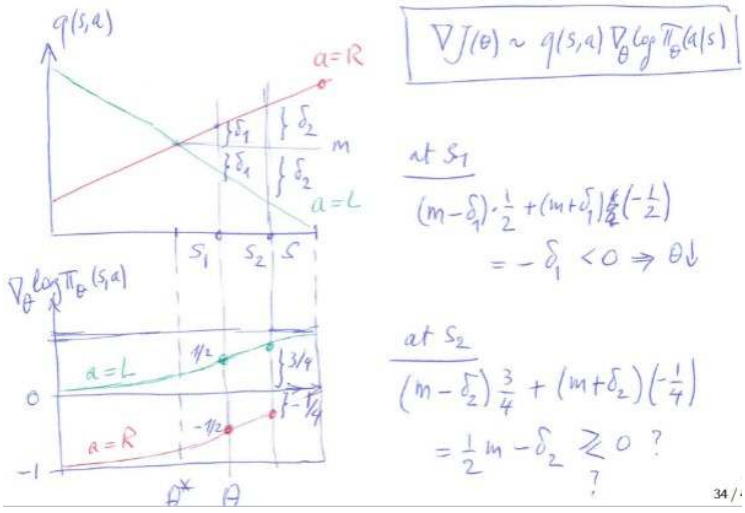
$$= -\frac{1}{1+e^{(s-\theta)}}$$

$$= -\pi_\theta(a=L|s) = \pi_\theta(a=R|s) - 1$$

- Worked example



- Advantage of baseline



- Introducing baselines: A2C
  - Policy gradient theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) q(s_t, a_t)\right]$$

  - **Problem:** value of ***q*(*s*, *a*) is not very informative**;

- We need a reference point or **baseline**: natural choice = $v(s)$.
- **Advantage:** Relative value of an action as compared to action in that state:

$$A(s,a) := q(s,a) - v(s)$$

- **Advantage actor-critic (A2C)**

$$\nabla_\theta J(\theta) \propto \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)(q_{\pi_\theta}(s_t, a_t) - v_{\pi_\theta}(s_t)) \right]$$

- Estimating Advantage (not in exam the below slides)
  - Typically **two neural networks** to estimate
    1. Policy → π_θ
    2. value functions and advantage: → v_w, q_w
    3. Network weights: θ and w
  - **Computational strategy:**
    - Estimate $v(s)$
    - Estimate $q(s, a)$ using Bellman equations:

$$q(s_t, a_t) = \mathbb{E}\left[r_{t+1} + \gamma v(s_{t+1})\right]$$

  - Along **sampled trajectory:**

$$\hat{q}(s_t, a_t) = r_{t+1} + \gamma \hat{v}(s_{t+1})$$

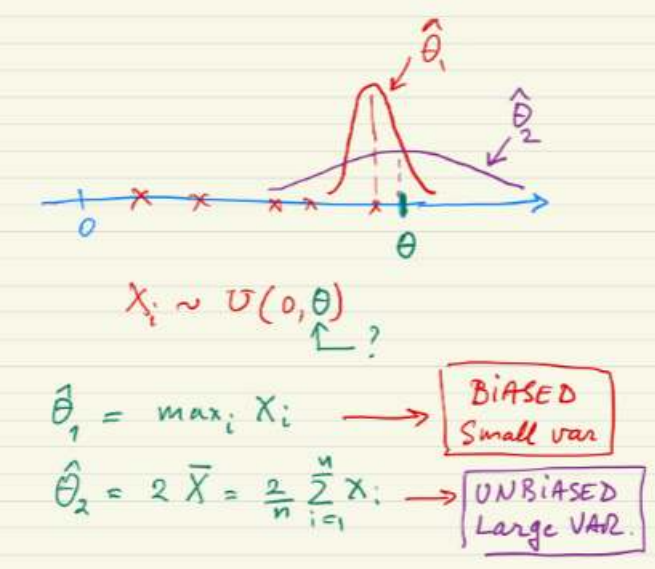
$$\hat{A}(s_t, a_t) = r_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t)$$

  - ***N*-steps returns** along **sampled trajectory**:

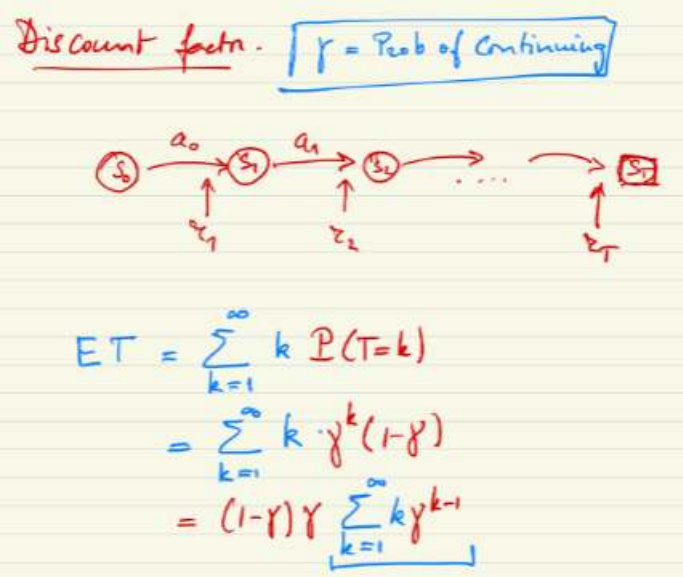
$$\hat{q}(s_t, a_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t_3} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n \hat{v}(s_{t+n})$$

  - Combination of biased and unbiased estimate:
    - Actual returns: **unbiased but very high variance** (sample paths can be very different!)
    - **Bias** due to inclusion of estimate , but lower variance (average over all actions);

- Mathematical aside (1): Bias vs. Variance



- Mathematical aside (2a): Discount factor



- Mathematical aside (2b): Discount factor

$$\sum_{k=1}^{\infty} k\gamma^{k-1} = \frac{d}{d\gamma}\left(\sum_{k=1}^{\infty}\gamma^{k}\right)$$

$$= \frac{d}{d\gamma}\left(\frac{\gamma}{1-\gamma}\right)$$

$$= \frac{(1-\gamma)-\gamma(-1)}{(1-\gamma)^2} = \frac{1}{(1-\gamma)^2}$$

$$ET = (1-\gamma)\gamma\frac{1}{(1-\gamma)^2} = \frac{\gamma}{1-\gamma}$$

Eg:
$\gamma = 0.9$
$ET \simeq 9$.

- Mathematical aside (2c): Discount factor