Reinforcement Learning (RL)

- RL differs from supervised learning in two crucial respects:
  1) SL is passive: model observes world and computes "predictions"
     RL is active: observes, makes predictions, and executes actions that influence the environment (in desired ways)
  2) SL is strongly supervised: for each training instance, the true answer is known
     RL is weakly supervised: a feedback signal is often only available after many actions

  example: learning chess: SL ≙ student has a coach who explains after every move whether it was good or bad and why
  RL ≙ feedback only game outcome "win", "loss", "draw" ≙ 1.5 bits after ≈ 40 actions

- feedback in RL is formally defined by "rewards": transition from timestep $t \to t+1$ produces reward R_{t+1}:
  $R_{t+1} > 0$ ≙ good, gain
  $R_{t+1} < 0$ ≙ bad, penalty
  $R_{t+1} = 0$ ≙ neutral, no feedback in this time step
  - extreme case: executes T time steps (chess $T \approx 40$) and then gets a single reward, all rewards for $t < T$ are neutral

$\Rightarrow$ makes learning difficult $\Rightarrow$ reward shaping ≙ manually introduces auxiliary rewards during training

<u>but</u>: reward shaping must be used with care, because auxilliary rewards are not present at test time and may cause training to converge to a bad local ~~mini~~ optimum ≙ algorithm "exploits" the shortcomings of the auxilliary rewards, e.g. infinite loops that do not help winning

· formal definitions: <u>agent</u> acts in an <u>environment</u> S

- environment is characterized by a state s_t at time t, states evolve in discrete timesteps $s_t \to s_{t+1}$
- ⇒ states form a chain $S_0 \to S_1 \to \dots \to S_t \to \dots \to S_T$
- some environments always reach a terminal state after finite time $T < \infty$, e.g. most games end with win or loss ≙ different "runs": "<u>episodes</u>"
- others can run indefinitely, e.g. typical in the real world
- transition probabilities of the environment: likelihood for the next state, when we are in a given state

$$p(\underbrace{s_{t+1} = s'}_{\text{next chess position}} \mid s_t = \underbrace{s}_{\text{current chess position}})$$

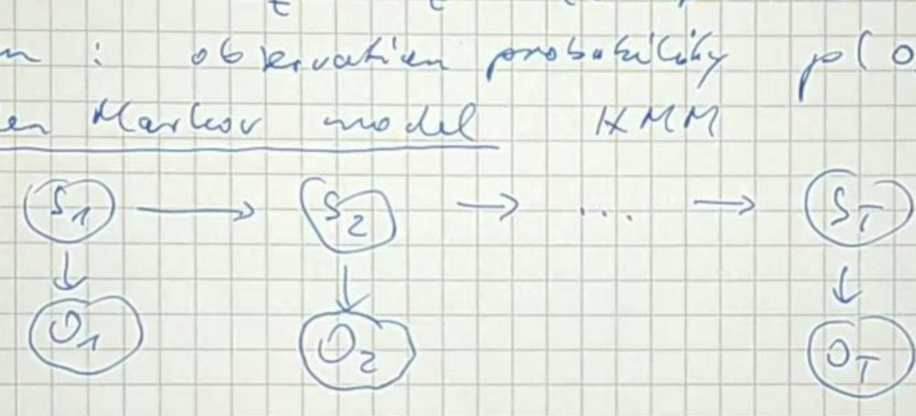
- crucial assumption: <u>Markov property</u>: the future only depends on the present, not on the past (chess: does not matter how one ends up in position s, the best next move is always <u>independent of history</u>)

$$s_{t-1} \perp s_{t+1} \mid s_t \Rightarrow \boxed{p(s_1, \dots, s_T) = p(s_1) \prod_{t=2}^{T} p(s_t \mid s_{t-1})}$$

states form <u>"Markov chain"</u>

- <u>observations</u>: the agent may not be able to know the present state exactly, but only get's indirect information via observations o_t
  - fully observable: $O_t = S_t$ (example: chess)
  - partially observable: $O_t \subset S_t$ (example: card game)
  - state is hidden: observation probability $p(O_t | S_t)$ — O_t only depends on the present S_t not the past $S_{t-1}, \dots$

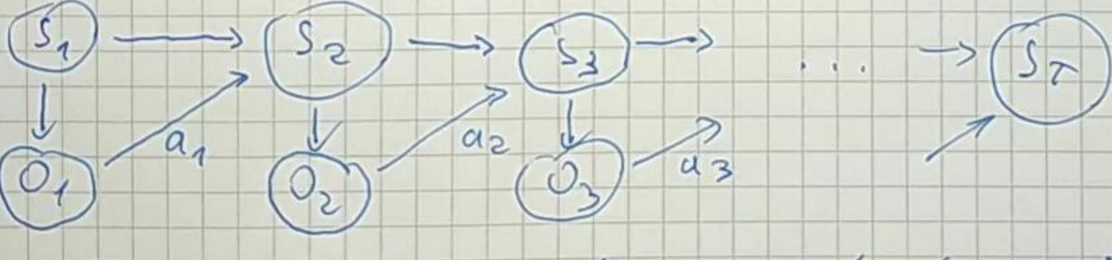
    → <u>hidden Markov model</u> HMM



examples: speech recognition

Poker: reactions of opponents

- so far, we only considered the environment passively, but actually the agent's <u>actions</u> can <u>influence the state transitions</u>



redefine state transitions:

$$p(S_{t+1} = s' | S_t = s, a_t = a)$$

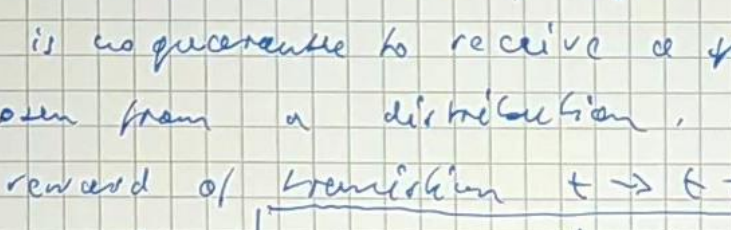
actions are chosen according to the <u>current observations</u> (history plays no role) $\hat{=}$ <u>policy</u> of the agent $\pi(O_t)$

- deterministic policy: $a_t = \pi(O_t)$ is always the same in state $O_t = o$
- probabilistic policy: $a_t \sim \pi(a | O_t)$ ⇒ can choose different actions in same situation (good ones with high probability)

- introduce rewards into the model, again ~~determin~~ probabilistically, e.g. in general there is no guarantee to receive a specific reward in a given situation, rewards are chosen from a distribution.

  convention: reward of transition $t \to t+1$ is called R_{t+1}

  <u>full transition model</u>: $\boxed{p(S_{t+1} = s', R_{t+1} = r \mid S_t = s, a_t = a)}$



transition function fully describes the behavior of the "world"

the agent is fully described by the observations that receives and the policy to choose actions: $p(O_t \mid S_t)$, $\pi(a_t \mid O_t)$

- simplification in the lecture: consider only fully observable worlds: $O_t \equiv S_t$

  $\Rightarrow p(O_t \mid S_t) = \delta(O_t - S_t)$, $\pi(a_t \mid S_t)$ conditioned on current state

  $\Rightarrow$ <u>Markov Decision Process</u> MDP (generalization: POMDP "partially observable MDP: $O_t \neq S_t$)

- Goal of RL: given training data, find the policy $\hat{\pi}(a_t \mid s_t)$ that maximizes the total reward

$\pi^*(a_t \mid s_t)$ unknown optimal policy, $\hat{\pi}(a_t \mid s_t) \approx \pi^*(a_t \mid s_t)$

- when the state transitions and/or policy are non-deterministic, the same initial state may result in many different games with different rewards
  $\Rightarrow$ take the <u>expected reward</u> in the optimization $\hat{=}$ <u>value function</u>

$$V_\pi(s) = \mathbb{E}_{p,\pi}\left[\sum_{t'=t+1}^{T} R_{t'} \mid S_t = s\right]$$

start state at time t

when T is very big or $T \to \infty$, $\sum R_{t'}$ may diverge $\Rightarrow$ discounted value function with discount factor γ (hyperparameter)

$$V_\pi(s) = \mathbb{E}_{p,\pi}\left[\sum_{t'=t+1}^{\infty} \gamma^{t'-t-1} R_{t'} \mid S_t = s\right]$$

$\gamma^0, \gamma^1 < 1, \ldots \gamma^k \lll 1 \quad k \to \infty$

$0 < \gamma \leq 1$: $\cancel{\gamma^0}$ $t' = t+1$.
$\Rightarrow$ future rewards are downweighted

<u>horizon</u> for reward calculation: narrow ~~wide~~ when γ small

- <u>optimal policy maximizes the expected reward</u>

$$\boxed{\begin{aligned} \pi^*(s) &= \cancel{\max_\pi} \arg\max_\pi V_\pi(s) \\ v^*(s) &= V_{\pi^*}(s) \end{aligned}}$$

(deterministic policy case)

- simplification: introduce "return": sum of rewards:

$$G_t = \sum_{t'=t+1}^{T} R_{t'} \quad \text{or} \quad G_t = \sum_{t'=t+1}^{\infty} \gamma^{t'-t-1} R_{t'}$$

$\gamma = 1$ equal

$$\boxed{G_t = R_{t+1} + \underline{\gamma\, G_{t+1}}}$$

- rewrite value function with "return":

$$v_\pi(s) = \mathbb{E}_{p,\pi}\left[G_t \mid S_t = s \right] = \mathbb{E}_{p,\pi}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right]$$

- assume that state and action spaces are discrete and finite

$\Rightarrow$ replace expectation with a sum

$$v_\pi(s) = \underbrace{\sum_a}_{\text{all permissible actions in state } s} \pi(a \mid s) \underbrace{\sum_{s', r}}_{\substack{r: \text{ all possible rewards from transition } s \to s' \\ s': \text{ all possible states that can follow state } s}} p(s', r \mid s, a) \Big[\overbrace{r}^{\text{immediate reward}} + \underbrace{\mathbb{E}_{p,\pi}\left[\gamma G_{t+1} \mid S_{t+1} = s' \right]}_{\text{expected return of the rest of the game, starting in state } s'} \Big]$$

now $\mathbb{E}_{p,\pi}\left[\gamma G_{t+1} \mid S_{t+1} = s' \right) = v_\pi(s') \cdot \gamma$

$\Rightarrow$ define $v_\pi(s)$ recursively in terms of $v_\pi(s')$

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

<u>Bellmann equation</u> for value function $\hat{=}$ enforces self-consistency of the value function

since states are discrete, we have finitely many numbers $v_\pi(s)$

for a given policy π $\Rightarrow$ Bellmann eq. is just a linear system

<u>example</u> simple game with 4 squares, allowed actions $A \to B$, $A \to C$, $A \leftarrow B$, $B \to D$

A = start	B
C = monster	D = treasure

game terminates when in state C or D

deterministic rewards: $r_{A \to C} = -100$, $r_{B \to D} = 100$

$r_{A \to B} = r_{B \to A} = -1$ (avoid infinite loop $A \leftrightarrow B$)

- define the "world" $p(s', r \mid s, a)$:

s'	r	s	a	$p(..)$
B	-1	A	$\rightarrow$	0.8
C	-100 (action ignored)	A	$\rightarrow$	0.2
C	-100	A	$\downarrow$	0.8
B	-1	A	$\downarrow$	0.2
D	100	B	$\downarrow$	0.8
A	-1	B	$\downarrow$	0.2
A	-1	B	$\leftarrow$	0.8
D	100	B	$\leftarrow$	0.2

- easy to see optimal policy:

$$\pi^*(s=A) = \rightarrow$$
$$\pi^*(s=B) = \downarrow$$

[$\pi^*(s=C)$, $\pi^*(s=D)$ undefined, because game over]

- write out Bellmann equations for π^* ($\gamma = 1$) no discount)

$$v^*(s=A) = 0.8\,[-1 + v^*(s=B)] + 0.2\,[-100 + \underbrace{v^*(s=C)}_{=0\text{, because game over}}]$$

$$v^*(s=B) = 0.8\,[100 + \underbrace{v^*(s=D)}_{=0}] + 0.2\,[-1 + v^*(s=A)]$$

$$20.8 = -v^*(s=A) + 0.8\,v^*(s=B)$$
$$-79.8 = 0.2\,v^*(s=A) - v^*(s=B)$$

$$v^*(s=A) \approx 51 \qquad v^*(s=B) \approx 90$$

- easily calculate that any other policy has lower value, for example
$\pi^\circ(s=A) = \downarrow$, $\pi^\circ(s=B) = \leftarrow$ $\quad V^\circ(s=A) = -91 \quad V^\circ(s=B) = -54$