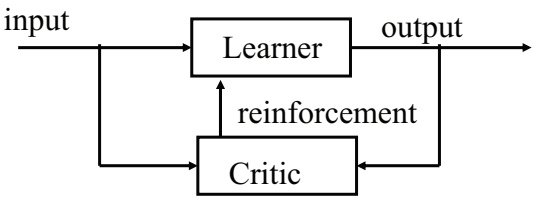
Reinforcement learning

- **We want to learn the control policy:** $\pi : X \rightarrow A$
- We see examples of **x** (but outputs *a* are not given)
- Instead of *a* we get a feedback (reinforcement, reward) from a **critic** quantifying how good the selected output was



- The reinforcements may not be deterministic
- **Goal:** find $\pi : X \rightarrow A$ with the best expected reinforcements

Gambling example.

- **Game:** 3 different biased coins are tossed
  - The coin to be tossed is selected randomly from the three options and I always see which coin I am going to play next
  - I make bets on head or tail and I always wage $1
  - If I win I get $1, otherwise I lose my bet
- **RL model:**
  - **Input:** X – a coin chosen for the next toss,
  - **Action:** A – choice of head or tail,
  - **Reinforcements:** {1, -1}
- **A policy** $\pi : X \rightarrow A$

  Example: π: | **Coin1** → ***head***
 Coin2 → ***tail***
 Coin3 → ***head*** |

Gambling example.

- **RL model:**
  - **Input:** X – a coin chosen for the next toss,
  - **Action:** A – choice of head or tail,
  - **Reinforcements:** {1, -1}
  - **A policy** π: | **Coin1** → ***head***
 Coin2 → ***tail***
 Coin3 → ***head*** |

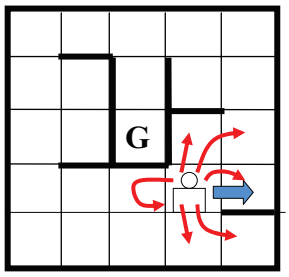
- **Learning goal: find** $\pi : X \rightarrow A$ π: | **Coin1** → ***?***
 Coin2 → ***?***
 Coin3 → ***?*** |

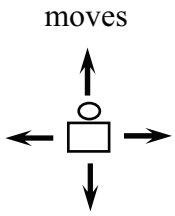
  maximizing future expected profits

  $E(\sum_{t=0}^{\infty} \gamma^t r_t)$ γ a discount factor = present value of money

Agent navigation example.

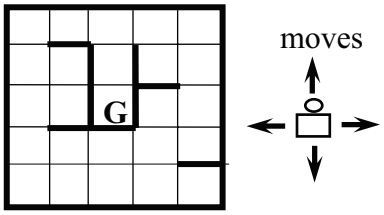
- **Agent navigation in the Maze**:
  - 4 moves in compass directions
  - Effects of moves are stochastic – we may wind up in other than intended location with non-zero probability
  - **Objective:** reach the goal state in the shortest expected time





Agent navigation example

- **The RL model:**
  - **Input:** X – position of an agent
  - **Output:** A –a move
  - **Reinforcements:** R
    - -1 for each move
    - +100 for reaching the goal
  - **A policy:** $\pi : X \rightarrow A$



π: | **Position 1** → ***right*** | **Position 2** → ***right*** | … | **Position 20** → ***left*** |

- **Goal: find the policy maximizing future expected rewards**

$$E(\sum_{t=0}^{\infty} \gamma^t r_t)$$

Objectives of RL learning

- **Objective:**
  Find a mapping $\pi^* : X \rightarrow A$
  That maximizes some combination of future reinforcements (rewards) received over time
- **Valuation models (quantify how good the mapping is):**
  - **Finite horizon model**

$$E(\sum_{t=0}^{T} r_t)$$ Time horizon: $T > 0$

  - **Infinite horizon discounted model**

$$E(\sum_{t=0}^{\infty} \gamma^t r_t)$$ Discount factor: $0 < \gamma < 1$

  - **Average reward**

$$\lim_{T \rightarrow \infty} \frac{1}{T} E(\sum_{t=0}^{T} r_t)$$

Exploration vs. Exploitation

- The (learner) actively interacts with the environment:
  - At the beginning the learner does not know anything about the environment
  - It gradually gains the experience and learns how to react to the environment
- **Dilemma (exploration-exploitation):**
  - After some number of steps, should I select the best current choice (**exploitation**) or try to learn more about the environment (**exploration**)?
  - **Exploitation** may involve the selection of a sub-optimal action and prevent the learning of the optimal choice
  - **Exploration** may spend to much time on trying bad currently suboptimal actions

Effects of actions on the environment

Effect of actions on the environment (next input **x** to be seen)

- No effect, the distribution over possible **x** is fixed; action consequences (rewards) are seen immediately,
- Otherwise, distribution of **x** can change; the rewards related to the action can be seen with some delay.

Leads to two forms of **reinforcement learning:**

- **Learning with immediate rewards**
  - **Gambling example**
- **Learning with delayed rewards**
  - **Agent navigation example**; move choices affect the state of the environment (position changes), a big reward at the goal state is delayed

RL with immediate rewards.

- **Game:** 3 different biased coins are tossed
  - The coin to be tossed is selected randomly from the three options and I always see which coin I am going to play next
  - I make bets on head or tail and I always wage $1
  - If I win I get $1, otherwise I lose my bet
- **RL model:**
  - **Input:** X – a coin chosen for the next toss,
  - **Action:** A – choice of head or tail,
  - **Reinforcements:** {1, -1}
- **Learning goal:** **find** $\pi : X \rightarrow A$

  maximizing the future expected profits over time

  $E(\sum_{t=0}^{\infty} \gamma^t r_t)$ γ a discount factor = present value of money

RL with immediate rewards

- **Expected reward**

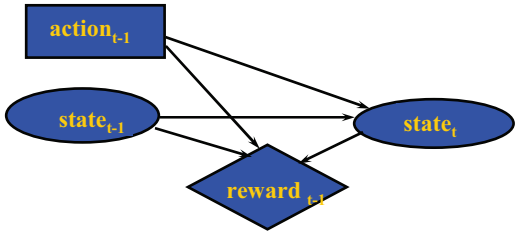
  $E(\sum_{t=0}^{\infty} \gamma^t r_t)$ γ - a discount factor = present value of money
- **Immediate reward case:**
  - Reward for the choice becomes available immediately
  - Our choice does not affect environment and thus future rewards

    $E(\sum_{t=0}^{\infty} \gamma^t r_t) = E(r_0) + E(\gamma r_1) + E(\gamma^2 r_2) + \ldots$

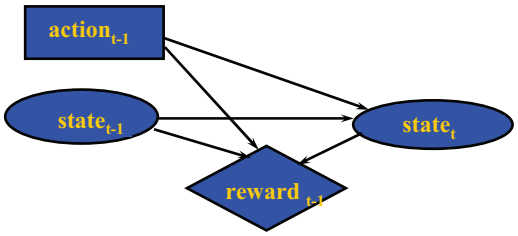
    $r_0, r_1, r_2 \ldots$ Rewards for every step
  - Expected one step reward for input **x** and the choice a: $R(\mathbf{x}, a)$

Learning with delayed rewards

- Actions, in addition to immediate rewards affect the next state of the environment and thus indirectly also future rewards
- We need a model to represent environment changes
- The model we use is called **Markov decision process (MDP)**
  - Frequently used in AI, OR, control theory
  - **Markov assumption**: next state depends on the previous state and action, and not states (actions) in the past



Markov decision process



Formal definition: 4-tuple (S, A, T, R)

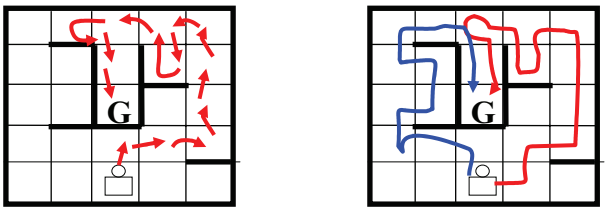
• **A set of states** S (X)	locations of a robot
• **A set of actions** A	move actions
• **Transition model** $S \times A \times S \rightarrow [0,1]$	where can I get with different moves
• **Reward model** $S \times A \times S \rightarrow \Re$	reward/cost for a transition

MDP problem

- We want to find the best policy $\pi^* : S \rightarrow A$
- **Value function** (V) for a policy, quantifies the goodness of a policy through, e.g. infinite horizon, discounted model

$$E(\sum_{t=0}^{\infty} \gamma^t r_t)$$

It: 1. combines future rewards over a trajectory
2. combines rewards for multiple trajectories (through expectation-based measures)



Value of a policy for MDP

- Assume a fixed policy $\pi : S \rightarrow A$
- How to compute the value of a policy under infinite horizon discounted model?

Fixed point equation:

$$V^{\pi}(s) = \underbrace{R(s, \pi(s))}_{\text{expected one step reward for the first action}} + \underbrace{\gamma \sum_{s' \in S} P(s' | s, \pi(s)) V^{\pi}(s')}_{\text{expected discounted reward for following the policy for the rest of the steps}}$$

$$\mathbf{v} = \mathbf{r} + \mathbf{U}\mathbf{v} \quad \Rightarrow \quad \mathbf{v} = (\mathbf{I} - \mathbf{U})^{-1}\mathbf{r}$$

– For a finite state space– we get a set of linear equations

Reinforcement learning of optimal policies

- **In the RL framework we do not know the MDP model !!!**
- **Goal:** learn the optimal policy

$$\pi^* : S \rightarrow A$$

- **Two basic approaches:**
  - **Model based learning**
    - Learn the MDP model (probabilities, rewards) first
    - Solve the MDP afterwards
  - **Model-free learning**
    - Learn how to act directly
    - No need to learn the parameters of the MDP
  - A number of clones of the two in the literature

Model-based learning

- We need to learn **transition probabilities** and **rewards**
- **Learning of probabilities**
  - ML or Bayesian parameter estimates
  - Use counts

$$\widetilde{P}(s' \mid s,a) = \frac{N_{s,a,s'}}{N_{s,a}} \qquad N_{s,a} = \sum_{s' \in S} N_{s,a,s'}$$

- **Learning rewards**
  - Similar to learning with immediate rewards

$$\widetilde{R}(s,a) = \frac{1}{N_{s,a}} \sum_{i=1}^{N_{s,a}} r_i^{s,a}$$

- **Problem: on-line update of the policy**
  - **would require us to solve an MDP after every update !!**