Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Reinforcement Learning

Uwe Dick

Overview

- Problem Statements
- Examples
- Markov Decision Processes
- Planning – Fully defined MDPs
- Learning – Partly defined MDPs
- Monte-Carlo
- Temporal Difference

Problem Statements in Machine Learning

- Supervised Learning:
  Learn a decision function from examples of correct decisions.

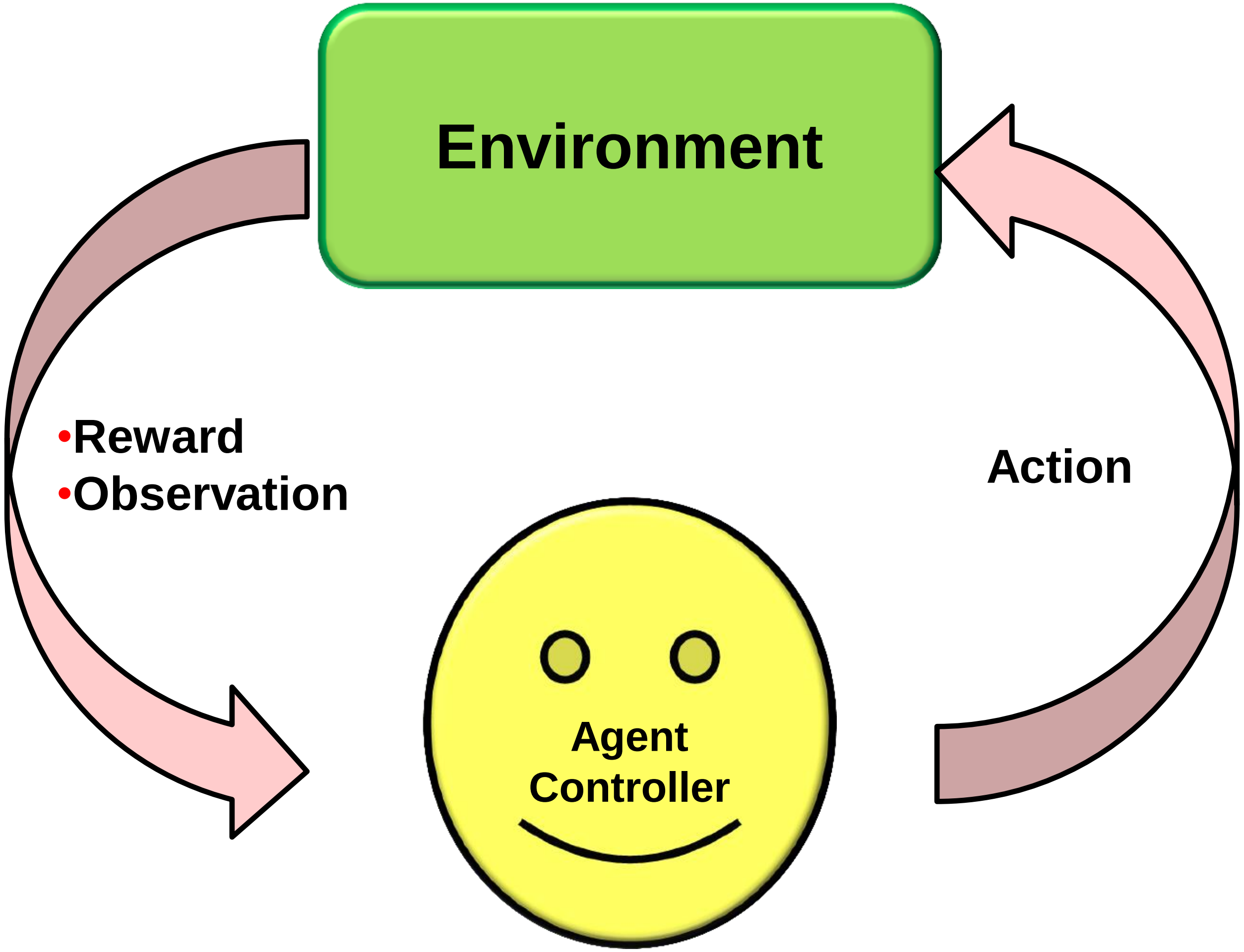
- Unsupervised Learning:
  Learn e.g. how to partition a data set (clustering) without knowledge of a correct partitioning.

- Reinforcement Learning:
  Learn how to make a sequence of decisions. The quality of each decision may depend on the complete decision sequence.
  $\rightarrow$ Temporal Credit Assignment Problem.

Examples

- Backgammon: How much does one move influence the outcome of a game?
- Robot Football: We want to score a goal. But which sequence of moves gives highest chance to do so?
- Helicopter Flight: What do we have to do to fly a maneuver without crashing in unknown environments?

Learning from Interactions



When Reinforcement Learning?

- Delayed reward for actions.
  (Temporal credit assignment problem)

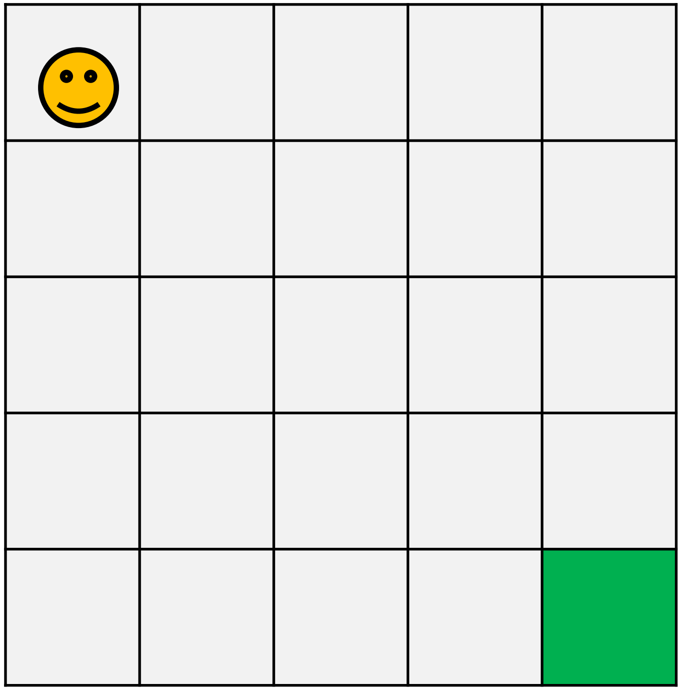
- Control problems.

- Agents – The full AI problem.

What is Reinforcement Learning?

- RL methods are „Sampling based methods to solve optimal control problems " (Richard Sutton)
- Search for an optimal policy (function from states to actions).
- Optimality: Policy with highest expected reward.
- Other definitions for optimal learning: Fast learning without making too many mistakes.

Example: Gridworld



Markov Decision Processes

- (Finite) Markov Decision Process: Tuple (S, A, R, P)
- S : Finite state space (set of states).
- A : Finite action space (set of actions).
- P : Transition probabilities.

$$P(s'|s,a) \;\; s, s' \in S, a \in A$$

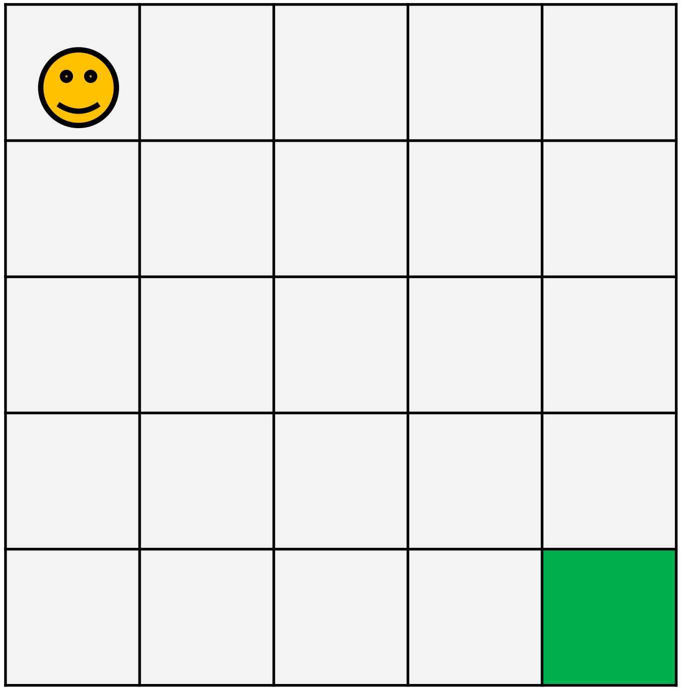
- R : Expected Immediate Reward.

$$R : (S \times A) \to \mathbb{R}$$

- Discount factor $0 \leq \gamma < 1$.

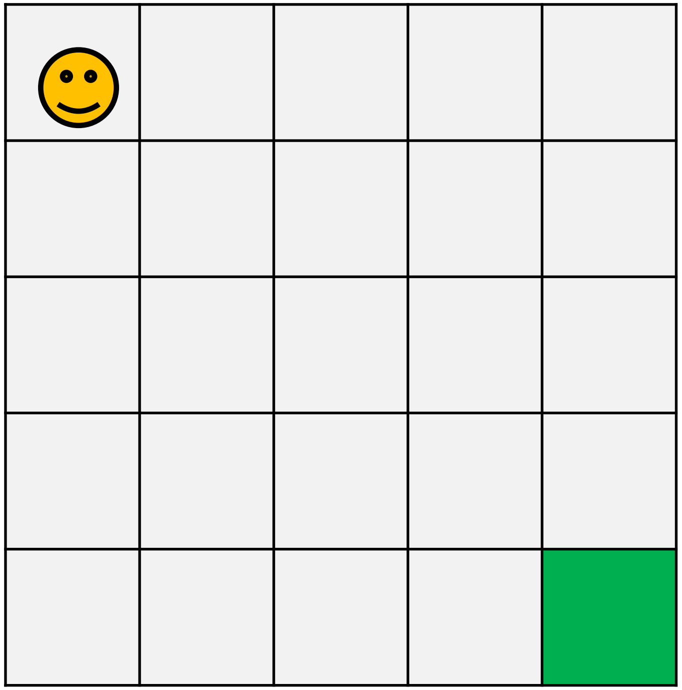
Example: Gridworld

- State space S
  - Start state $s_s \in S$
  - Target state $s_z \in S$



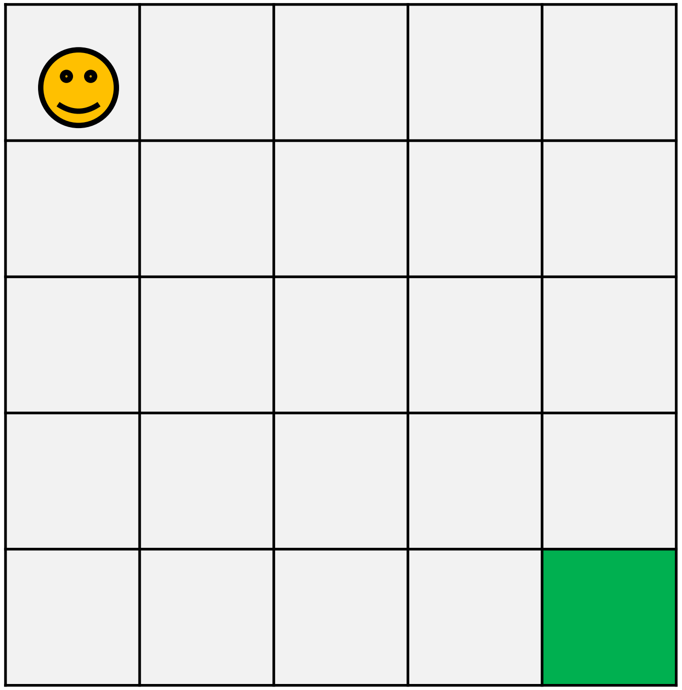
Example : Gridworld

- State space *S*
- Action space *A*
  - *A*=(left, right, up, down)



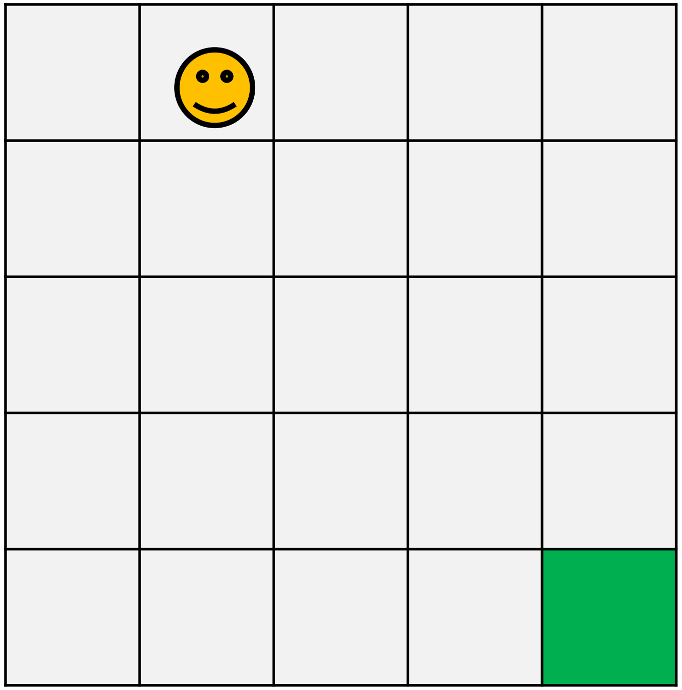
Example: Gridworld

- State space *S*
- Action space *A*
- Transition probabilities *P*
  - $P((1,2)|(1,1), \text{right}) = 1$



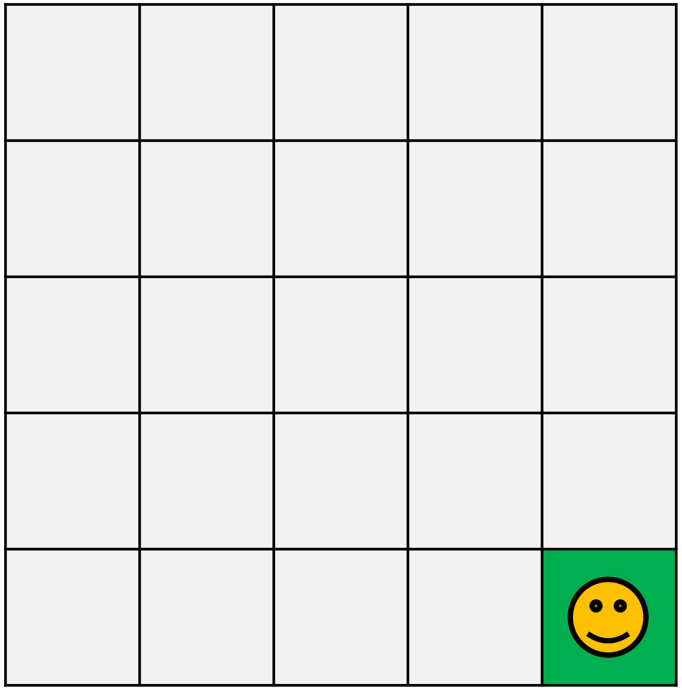
Example: Gridworld

- State space *S*
- Action space *A*
- Transition probabilities *P*
- Immediate Reward *R*
  - $R((1,1),\text{right}) = 0$



Example: Gridworld

- State space S
- Action space A
- Transition probabilities P
- Immediate Reward R
  - $R((4,5),\text{down}) = 1$



Markov Decision Processes

- (Finite) Markov Decision Process: Tuple (S, A, R, P)
- S : Finite state space (set of states).
- A : Finite action space (set of actions).
- P : Transition probabilities.

$$P(s'|s,a) \;\; s, s' \in S, a \in A$$

- R : Expected Immediate Reward.

$$R : (S \times A) \to \mathbb{R}$$

- Discount factor $0 \leq \gamma < 1$.

MDP

- A deterministic stationary policy maps states to actions.

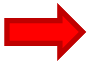
$$\pi : S \to A$$

- Stochastic Policy: Function from states to distribution of actions.

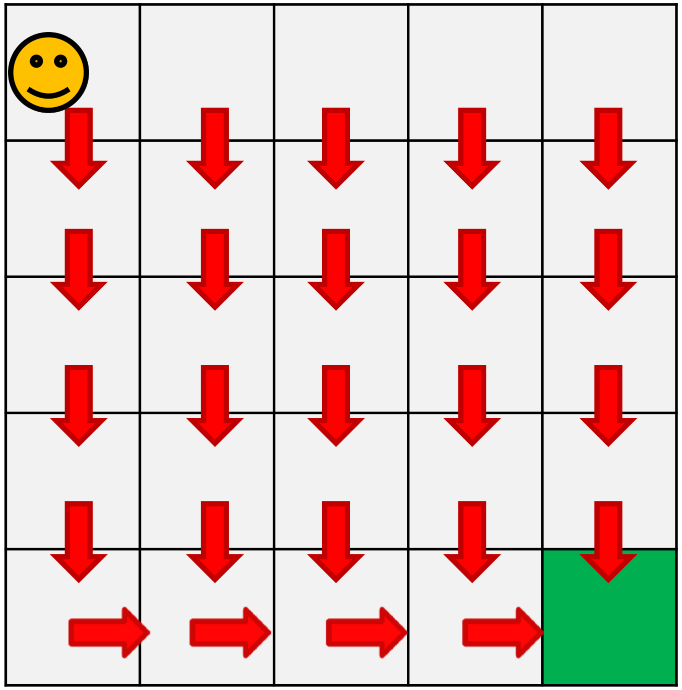
- Goal: Find policy π, that maximizes the expected cumulative (discounted) reward.

$$E_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right]$$

Example: Gridworld

- State space *S*
- Action space *A*
- Transition probabilities *P*
- Immediate Reward *R*
- Discount factor $\gamma = 0{,}9$
- Policy π
  - Good Policy π_2 
- Expected discounted Reward

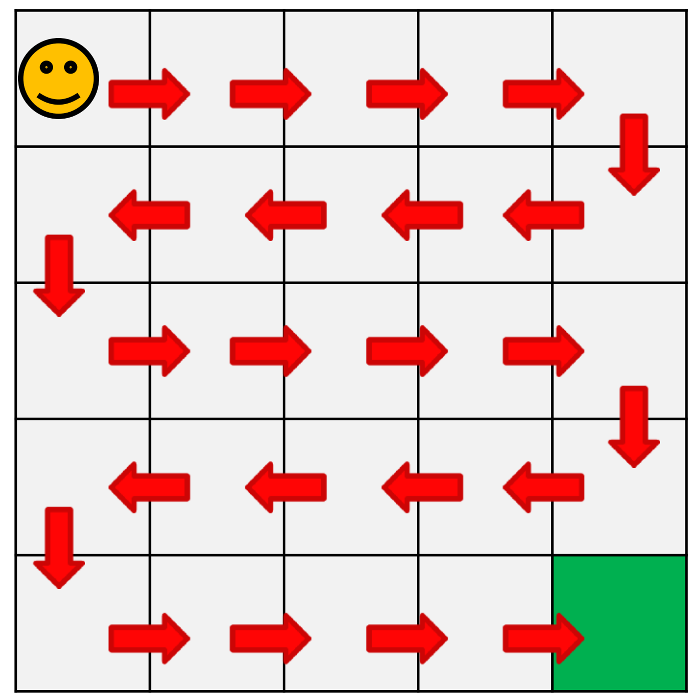
$$E_{\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,\pi(s_t)) \mid s_0 = s_s\right] = 0{,}9^7$$



Example : Gridworld

- State space S
- Action space A
- Transition probabilities P
- Immediate Reward R
- Discount factor $\gamma = 0{,}9$
- Policy π
  - Bad Policy π_1
- Expected discounted Reward

$$E_{\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,\pi(s_t)) \mid s_0 = s_s\right] = 0{,}9^{23}$$



Markov Property

- Markov Property:

$$\begin{aligned} P(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) &= P(s_{t+1} | s_t, a_t) \\ R(s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) &= R(s_t, a_t) \end{aligned}$$

- In order to model real world scenarios as MDPs, sequences of observations and actions have to be aggregated into states.

- Markov property rarely fulfilled in reality.

Value Functions

- Value function $V^{\pi}(s)$ for a state s and policy π describes the expected discounted cumulative reward that will be observed when starting in s and performing actions according to. π.

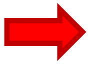
$$V^{\pi}(s_t) = E_{\pi,P}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k}))\right]$$

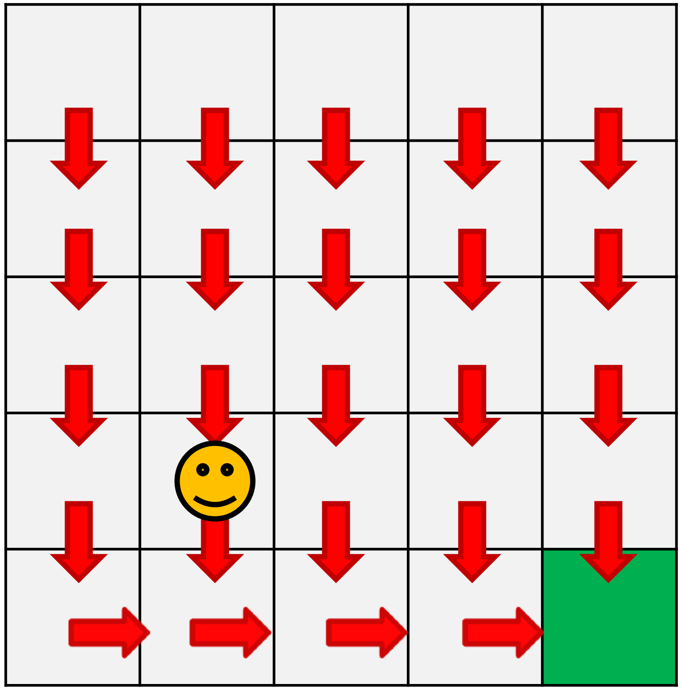
- There always exists an optimal deterministic stationary policy π^*, which maximizes the value function.

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

$$V^{\pi^*}(s) = V^*(s)$$

Example: Gridworld

- State space *S*
- Action space *A*
- Transition probabilities *P*
- Immediate Reward *R*
- Discount factor $\gamma = 0,9$
- Policy π
  - Good Policy π_2 
- Expected discounted Reward



$$V^{\pi_1}(s_t) = E_{\pi,P}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k}))\right] = 0,9^3$$

Value Functions

- Value function for state action pair:

$$Q^\pi(s_t, a_t) = R(s_t, a_t) + E_{\pi,P}\left[\sum_{k=1}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k}))\right]$$

- Optimal value function:

$$\begin{aligned} Q^*(s_t, a) &= R(s_t, a) + \gamma E_P[V^*(s_{t+1})] \\ \text{und } V^*(s_t) &= \max_a Q^*(s_t, a) \\ Q^*(s, a) &= \max_\pi Q^\pi(s, a) \\ \pi^*(s) &= \arg\max_a Q^*(s, a) \end{aligned}$$

- Assumption: Value function can be stored in (large) table (One entry for each state-action pair).

Continuous State Spaces

- In the real world state spaces are (often) continuous or very large.
- The same can hold for action spaces.
- Representation of value function and/or policy via function approximation methods.
    - E.g. representation of value function as parametric function with parameter vector θ and features (basis functions) $\phi_i : S \times A \to \mathbb{R}$:

$$\hat{Q}(s,a;\theta) = \sum_{i=1}^{N} \phi_i(s,a) \cdot \theta_i$$

Continuous State Spaces

- Alternatively, use representation of policy as parametric function of state-dependent features

$$\pi(s;\theta) = \sum_{i=1}^{N} \phi_i(s) \cdot \theta_i$$

- Such problems will be covered next week!

- Today: „Idealized“ problems with small and discrete state and action spaces.

Bellman Equations

- Bellman equations describe a recursive property of value functions. (Because of Markov property)

$$
\begin{aligned}
V^{\pi}(s_t) &= E_{\pi,P}\Big[\sum_{k=0}^{\infty}\gamma^k R(s_{t+k},\pi(s_{t+k}))\Big] \\
&= E_{\pi,P}\Big[R(s_t,\pi(s_t)) + \gamma\sum_{k=0}^{\infty}\gamma^k R(s_{t+k+1},\pi(s_{t+k+1}))\Big] \\
&= E_{\pi,P}\Big[R(s_t,\pi(s_t)) + \gamma V^{\pi}(s_{t+1})\Big] \\
&= R(s_t,\pi(s_t)) + \gamma \sum_{s_{t+1}\in S} P(s_{t+1}|s_t,\pi(s_t))V^{\pi}(s_{t+1})
\end{aligned}
$$

Bellman Equations

- State action value functions:

$$Q^\pi(s_t, a_t) \quad = \quad R(s_t, a_t) + \gamma E_{\pi,P}\Big[Q^\pi(s_{t+1}, \pi(s_{t+1}))\Big]$$

- The Bellman equations constitute a system of linear equations.

$$\begin{aligned} V^\pi &= R + \gamma P^\pi V^\pi \\ \rightarrow V^\pi &= (I - \gamma P^\pi)^{-1} R \end{aligned}$$

Bellman Operators

- Notation using (linear) operators:

$$V^\pi = T^\pi V^\pi$$

- with linear operator T^π:

$$(T^\pi V)(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V(s')$$

➔ V^π is a fixed point of the Bellman operator T^π.

Bellman Optimality Equations

- Bellman Equations for control problem.
- Recursive property of optimal value functions.

$$V^*(s_t) \quad = \quad \max_a E_{\pi,T}\Big[R(s_t, a) + \gamma V^*(s_{t+1})\Big]$$

$$Q^*(s_t, a_t) \quad = \quad R(s_t, a_t) + \gamma E_{\pi,T}\Big[\max_a Q^*(s_{t+1}, a)\Big]$$

Model Knowledge

- Different problem formulations for differing knowledge about MDPs.

- MDP fully defined.
  $\rightarrow$ Planning.

- MDP only partly defined.
  We can gather experience by interacting with the environment.
  $\rightarrow$ Reinforcement Learning.

Types of Reinforcement Learning

- Reinforcement Learning methods can be distinguished w.r.t. their usage of those interactions.

- Indirect methods:
  - Model learning.

- Direct methods:
  - Direct policy search.
  - Value function estimation
    - Policy Iteration.
    - Value Iteration.

MDP Fully Defined – Planning with Policy Iteration

- Both reward function R and transition probabilities P are defined.

- **Policy Iteration** is a general algorithm for computing the optimal policy.

- Iterate the following 2 Steps for computation of optimal policy for k=0,1,… Initialize π_0 randomly.
  - Policy Evaluation: Compute Q^π for fixed $\pi_{k.}$
  - Policy Improvement: Determine next π_{k+1}.

Policy Evaluation

- First step in each iteration: Evaluate quality of current (approximation of optimal) policy.

- Policy Evaluation computes value function $V^{\pi'}$ or $Q^{\pi'}$ for fixed π'.

- Bellman Equations constitute system of linear equations.
- However, state space is usually too large to solve system of linear equations with standard solvers.

Policy Evaluation with Value Iteration

- Value Iteration for policy evaluation is an iterative algorithm that computes value function Q^{π_k} for current policy π_k as the limit of a sequence of approximations Q_i.

$$\forall s \in S, a \in A:$$

$$Q_{i+1}(s,a) = E_{\pi,P}\left[R(s,a) + \gamma Q_i\,(s', \pi_k(s')) \right]$$

$$= R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) Q_i\,(s', \pi_k(s'))$$

Policy Evaluation with Value Iteration

- Value Iteration for policy evaluation is an iterative algorithm that computes value function V^{π_k} for current policy π_k as the limit of a sequence of approximations:

$$\forall s \in S:$$

$$\begin{aligned} V_{i+1}(s) &= E_{\pi,P}\left[R(s,\pi_k(s)) + \gamma V_i\,(s')\right] \\ &= R(s,\pi_k(s)) + \gamma \sum_{s'} P(s' \mid s,\pi_k(s)) V_i\,(s') \end{aligned}$$

Policy Iteration

k=0. Repeat until $\forall s \in S : \pi_k(s) = \pi_{k+1}(s)$

- Evaluate current policy, e.g. using Value Iteration.

$$\text{for } i = 1 \ldots :$$

$$\forall s \in S, a \in A :$$

$$Q_{i+1}(s,a) = E_{\pi,P}\left[R(s,a) + \gamma Q_i(s', \pi_k(s')) \right]$$

$$= R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) Q_i(s', \pi_k(s'))$$

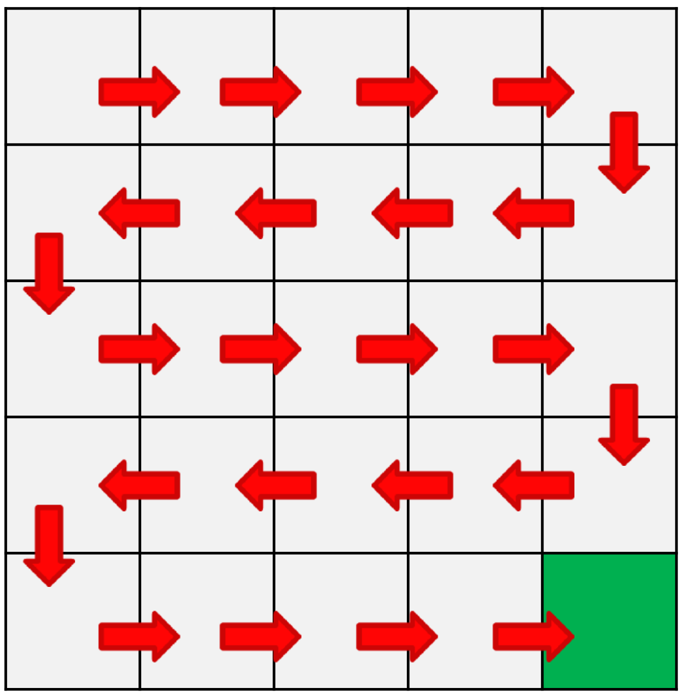
- Greedy Policy Improvement:

$$\forall s \in S : \; \pi_{k+1}(s) = \arg\max_a Q^{\pi_k}(s,a)$$

Example: Gridworld

- Discount factor $\gamma = 0,9$
- Start Policy π_1 ➡

- Policy Iteration:
  - Compute V^{π_1} with sequence of approximations V_i

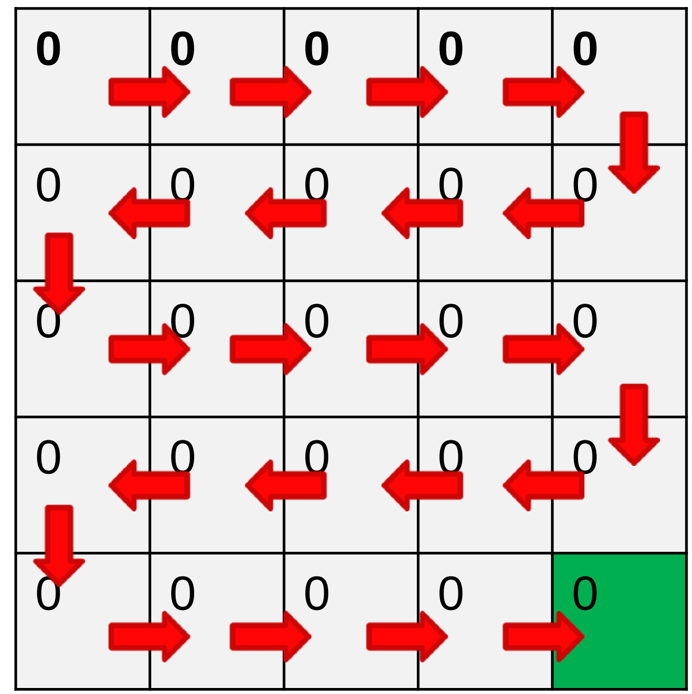


Example: Gridworld

- Discount factor $\gamma = 0{,}9$
- Start Policy π_1

- Policy Iteration:
  - Compute V^{π_1} with sequence of approximations V_i

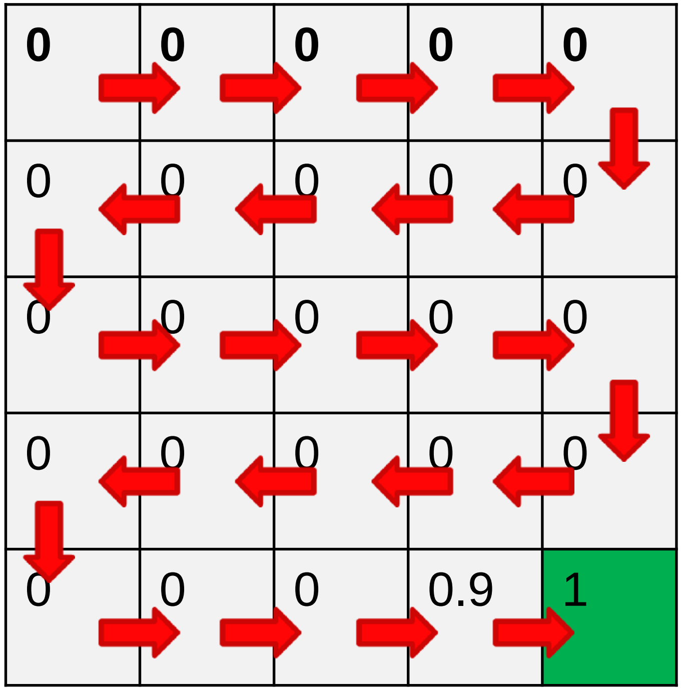
V_0



Example: Gridworld

- Discount factor $\gamma = 0,9$
- Start Policy π_1 ➡
- Policy Iteration:
  - Compute V^{π_1} with sequence of approximations V_i

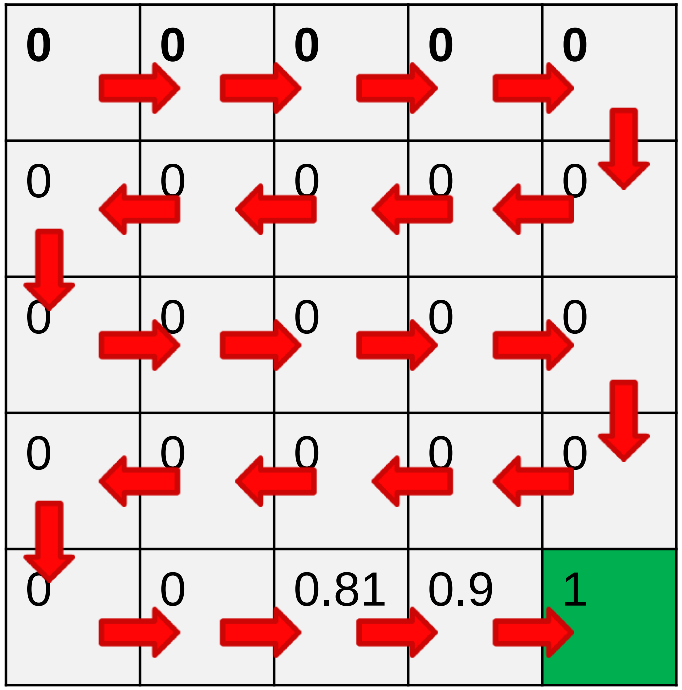
V_1



Example: Gridworld

- Discount factor $\gamma = 0,9$
- Start Policy π_1 ➡
- Policy Iteration:
  - Compute V^{π_1} with sequence of approximations V_i

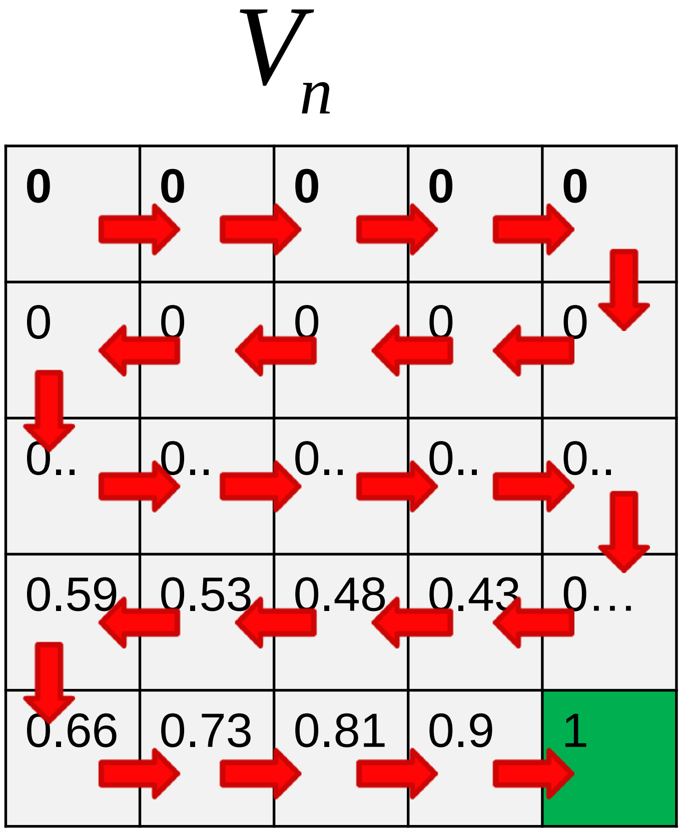
V_2



Example: Gridworld

- Discount factor $\gamma = 0,9$
- Start Policy π_1

- Policy Iteration:
  - Compute V^{π_1} with sequence of approximations V_i

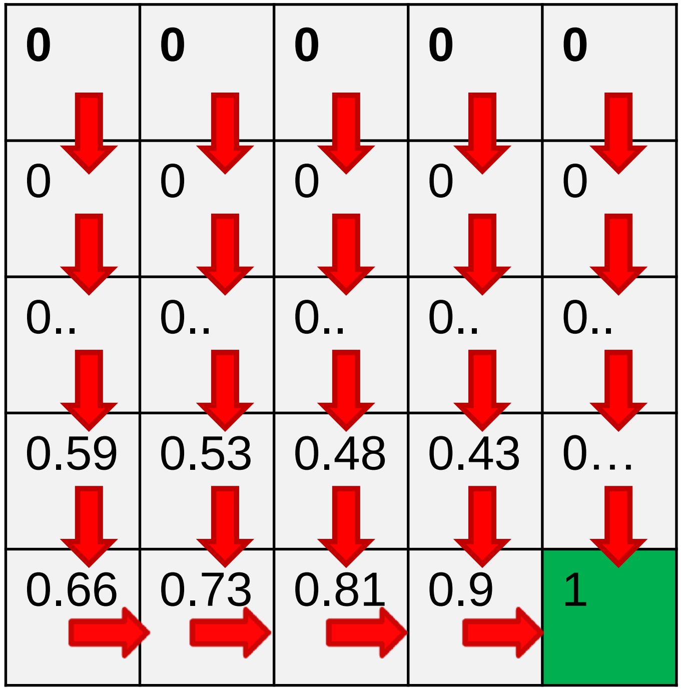


Example: Gridworld

- Discount factor $\gamma = 0,9$
- Start Policy π_1

- Policy Iteration:
  - Compute V^{π_1} with sequence of approximations V_i
  - Policy Improvement: Compute greedy Policy π_2 $(Q^{\pi_k}(s,a) = R(s,a) + \gamma E[V^{\pi_k}(s')])$

$$V^{\pi_1}$$



Policy Evaluation

- In the limit $k \to \infty$, V_k converges to V^π.
  Rate of convergence O(γ^k): $||V_k - V^\pi||$ = O(γ^k)

- Proof e.g. using Banach fixed-point theorem.
- Let B=(B,||.||) be a Banach space.
- Let T be an operator T:$B \to B$, such that
  $||TU - TV|| \leq \gamma\, ||U - V||$ with γ<1.
  T is a γ-contraction mapping.
- Then T admits a unique fixed-point V. Furthermore, for all $V_0 \in B$, the sequence $V_{k+1}=T\,V_k$, $k \to \infty$ converges to V. Also, $||V_k - V|| = O(\gamma^k)$

Policy Evaluation

- The Bellman operator T^π

$$(T^\pi V)(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V(s')$$

is a contraction mapping with contraction constant γ. It follows that the sequence that results from iteratively applying the operator

$$V_{k+1}(s) = (T^\pi V_k)(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V_k(s')$$

converges to V^π.

Policy Evaluation: Contraction Mapping

- What is a contraction mapping?

$$\|T^\pi V_{k+1} - T^\pi V_k\| \quad \leq \quad \gamma \|V_{k+1} - V_k\|$$

- According to the algorithm:

$$\|V_{k+2} - V_{k+1}\| \quad \leq \quad \gamma \|V_{k+1} - V_k\|$$

- Distance to the real value function reduces per iteration by a factor γ (using sup norm).

$$\|V^\pi - V_{k+1}\| = \|T^\pi V^\pi - T^\pi V_k\| \leq \gamma \|V^\pi - V_k\|$$

Policy Improvement

- Greedy Policy Improvement

$$\pi_{k+1}(s) \quad = \quad \arg\max_a Q^{\pi_k}(s,a)$$

- Policy Improvement Theorem:

  Let π‘ and π be deterministic policies with: For all $s \in S$: $Q^{\pi}(s, \pi‘(s)) \geq V^{\pi}(s)$.

  Then $V^{\pi‘}(s) \geq V^{\pi}(s)$

Value Iteration

- Value Iteration for control problem:

$$V_{k+1}(s_t) = \max_a \left[R(s_t, a) + \gamma \sum_{s_{t+1}} T(s_{t+1}|s_t, a) V_k(s_{t+1}) \right]$$

$$Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s'} T(s'|s, a) \max_{a'} Q_k(s', a')$$

- Converges to Q^* for $k \to \infty$
- Similar proof.

Value Iteration

- Algorithm:
  - Initialize Q, e.g. Q = 0
  - for k=0,1,2,…:
    - foreach (s,a) in (S x A):

$$Q_{k+1}(s,a) \quad = \quad R(s,a) + \gamma \sum_{s'} T(s'|s,a) \max_{a'} Q_k(s',a')$$

  - until $Q_{k+1} = Q_k$

- Alternative convergence criterion:
  - Max distance to optimal Q^* smaller than ς.
  - Conservative Choice: $K = \log_\gamma \frac{\varsigma(1-\gamma)^2}{2\|R\|_\infty}$

MDP Partially Undefined—Reinforcement Learning

- Indirect Reinforcement Learning: Model based.

- Learn Modell of MDP:
  - Reward function R
  - Transition probabilities P

- Apply planning algorithm as before, e.g. Policy Iteration.

Policy Iteration (Again)

- Even without learning a model of the MDP, we can apply the same principles.

- As before: Iterate the following 2 Steps for computation of optimal policy:
  - Policy Evaluation: Compute Q^{π} for fixed $\pi_{k.}$
  - Policy Improvement: Determine next π_{k+1}.

- Policy Evaluation step changes for partly defined MDPs.

Policy Evaluation: Monte-Carlo Methods

- Learn from episodic interactions with the environment.
- Goal: Learn $Q^{\pi}(s,a)$.

- Monte-Carlo Estimation of $Q^{\pi}(s,a)$: Compute mean of sampled cumulative rewards.

- Unbiased Estimation of real rewards. Variance reduces with $1/n$.

Policy Evaluation: Monte-Carlo Methods

- Computation time of estimation is independent from size of state space.

- Problem: If π is deterministic, many state action pairs $Q(s,a)$ will never be observed.
- Problems in Policy Improvement step.

- Solution: stochastic policies, e.g. ϵ-greedy policies.

Greedy and ϵ-Greedy Policies

- Greedy: $$\pi(s) = \arg\max_a Q(s,a)$$

- ϵ-greedy:
$$\pi(s) = \begin{cases} \arg\max_a Q(s,a) & \text{with probability } \varepsilon \\ \text{random action} & \text{with probability } 1-\varepsilon \end{cases}$$
  - Notation as distribution:
$$\pi(s,a) = \begin{cases} \varepsilon & \text{if } \; a = \arg\max_{a'} Q(s,a') \\ \dfrac{1-\varepsilon}{|A|-1} & \text{otherwise} \end{cases}$$

- ϵ-greedy allows for exploration.

Stochastic Policy: Softmax

- Current estimation of value function should have influence on probabilities.
  $\rightarrow$ soft max

- Example: Gibbs distribution:

$$\pi(a) = \frac{e^{Q_t(a)/\tau_t}}{\sum_{i=1}^{|A|} e^{Q_t(a_i)/\tau_t}}$$

- τ_t is also called the temperature parameter.

Temporal Difference Learning

- Idea: Update states based on *estimates* of other states. Natural formulation as online learning method.

- Also applicable to incomplete episodes.

- Disadvantage compared to Monte-Carlo:
  - Stronger influence (more damage) if Markov property violated.

Policy Evaluation: Value Iteration

- Idea: Update states based on *estimates* of other states. Natural formulation as online learning method.
- Same idea as before in fully defined case.
- Value Iteration for Policy Evaluation.
  Iteratively sample action a_t and observe next state s_{t+1}. Update Q according to:

$$Q_{t+1}^{\pi_k}(s_t, a_t) = E_{a' \sim \pi}\left[R(s_t, a_t) + \gamma Q_t^{\pi_k}(s_{t+1}, a') \right]$$
$$= R(s_t, a_t) + \gamma \sum_{a'} \pi(s, a') Q_t^{\pi_k}(s_{t+1}, a')$$

Policy Evaluation: Value Iteration

- Exploration / exploitation problem. Same as for Monte-Carlo methods.

- If policy stochastic: Sample state-action sequence $s_1 a_1 s_2 a_2 s_3 a_3 s_4 a_4 \ldots$ on-policy according to $a_t \sim \pi(s_t)$.

- Or sample $s_1 a_1 s_2 a_2 s_3 a_3 s_4 a_4 \ldots$ off-policy according to stochastic off-policy $a_t \sim \pi_b(s_t)$.

Policy Iteration (Again)

- As before: Iterate the following 2 Steps for computation of optimal policy:
  - Policy Evaluation: Compute Q^{π} for fixed $\pi_{k.}$
  - Policy Improvement: Determine next π_{k+1}.
- Policy Evaluation either with Monte-Carlo sampling or value iteration.

N-step Returns

- General update rule:

$$V(s_t) \leftarrow (1-\alpha_t)V(s_t) + \alpha_t \Delta V(s_t)$$

- Temporal difference methods perform 1-step updates:

$$\Delta_1 V(s_t) = R_t + \gamma V(s_{t+1})$$

- Monte-Carlo methods make updates, that are based on complete episodes:

$$\Delta_\infty V(s_t) = R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \ldots$$

- N-Step Updates:

$$\Delta_n V(s_t) = R_0 + \gamma R_1 + \gamma^2 R_2 + \ldots + \gamma^{n-1} R_{n-1} + \gamma^n V(s_n)$$

TD(λ)

$$
\begin{array}{llllllll}
 & R_0 & R_1 & R_2 & R_3 & \ldots & R_k & R_{k+1} \quad \ldots \\
\Delta_1 : & R_0 + \gamma V(s_1) & & & & & & \\
\Delta_2 : & R_0 + \gamma R_1 + & \gamma^2 V(s_2) & & & & & \\
\Delta_3 : & R_0 + \gamma R_1 + & \gamma^2 R_2 + & \gamma^3 V(s_3) & & & & \\
 & & \vdots & & & & & \\
\Delta_k : & R_0 + \gamma R_1 + & \gamma^2 R_2 + & \gamma^3 R_3 & + \ldots & + \gamma^k V(s_k) & & \\
 & & \vdots & & & & & \\
\Delta_\infty : & R_0 + \gamma R_1 + & \gamma^2 R_2 + & \gamma^3 R_3 & + \ldots & + \gamma^k R_k + \gamma^{k+1} R_{k+1} + \ldots & &
\end{array}
$$

TD(λ)

$$
\begin{array}{lllllllll}
 & & R_0 & R_1 & R_2 & R_3 & \dots & R_k & R_{k+1} \quad \dots \\
w_1 & \Delta_1 : & R_0 + & \gamma V(s_1) & & & & & \\
w_2 & \Delta_2 : & R_0 + & \gamma R_1 + & \gamma^2 V(s_2) & & & & \\
w_3 & \Delta_3 : & R_0 + & \gamma R_1 + & \gamma^2 R_2 + & \gamma^3 V(s_3) & & & \\
 & & & & \vdots & & & & \\
w_k & \Delta_k : & R_0 + & \gamma R_1 + & \gamma^2 R_2 + & \gamma^3 R_3 & + \dots & + \gamma^k V(s_k) & \\
 & & & & \vdots & & & & \\
w_\infty & \Delta_\infty : & R_0 + & \gamma R_1 + & \gamma^2 R_2 + & \gamma^3 R_3 & + \dots & + \gamma^k R_k + & \gamma^{k+1} R_{k+1} + \dots
\end{array}
$$

- Idea: Weighted sum over all n-step returns.

$$
\Delta = \sum_{k=1}^{\infty} w_k \Delta_k V(s_0)
$$

TD(λ)

$$
\begin{array}{lll}
 & & R_0 \quad\quad R_1 \quad\quad\quad R_2 \quad\quad\quad R_3 \quad\quad\quad \ldots \quad R_k \\
(1-\lambda) & \Delta_1 : & R_0 + \gamma V(s_1) \\
(1-\lambda)\lambda & \Delta_2 : & R_0 + \gamma R_1 + \gamma^2 V(s_2) \\
(1-\lambda)\lambda^2 & \Delta_3 : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 V(s_3) \\
 & & \vdots \\
(1-\lambda)\lambda^{k-1} & \Delta_k : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \ldots + \gamma^k V(s_k) \\
 & & \vdots
\end{array}
$$

- TD(λ) Update: $\Delta(\lambda) = \sum_{k=1}^{\infty} (1-\lambda)\lambda^{k-1} \Delta_k V(s_0)$
- $0 \leq \lambda \leq 1$ interpolates between 1-step and MC.

Bias-Variance-Tradeoff

$$
\begin{array}{lllllllll}
 & R_0 & R_1 & R_2 & R_3 & \dots & R_k & R_{k+1} & \dots \\
\Delta_1 : & R_0 + \gamma V(s_1) \\
\Delta_2 : & R_0 + \gamma R_1 + \gamma^2 V(s_2) \\
\Delta_3 : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 V(s_3) \\
 & \vdots \\
\Delta_k : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k V(s_k) \\
 & \vdots \\
\Delta_\infty : & R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \dots + \gamma^k R_k + \gamma^{k+1} R_{k+1} + \dots
\end{array}
$$

More variance

Less Bias

Eligibility Traces

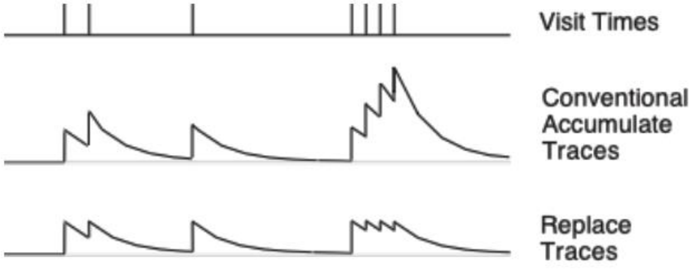
- Algorithmic view on TD(λ)
- Use additional variable $e(s)$ for every state $s \in S$.

- After observation <s_t,a_t,R_t,s_{t+1}>, compute

$$\begin{aligned} \delta_t &\leftarrow R_t + \gamma V(s_{t+1}) - V(s_t) \\ e(s_t) &\leftarrow e(s_t) + 1 \end{aligned}$$

- Update for all states

$$\begin{aligned} V(s) &\leftarrow V(s) + \alpha_t \delta_t e(s) \\ e(s) &\leftarrow \lambda \gamma e(s) \end{aligned}$$



Q-Learning

- Q-Learning Update:

$$\begin{aligned} Q(s_t, a_t) \quad &\leftarrow \quad (1 - \alpha_t) Q(s_t, a_t) \\ &\quad + \alpha_t \left(R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \right) \end{aligned}$$

- Converges to Q^* if
  - Every state will be observed infinitely often.
  - Step size parameters follow:

$$\sum_{t=0}^{\infty} \alpha_t = \infty \qquad \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

Q-Learning

- Off-Policy method. No exploration / exploitation problem.
- Learn optimal policy π^* while following another behavior policy π‘.
- Policy π‘ could e.g. be a stochastic policy with $\pi(s,a)>0$ for all s and a to guarantee convergence of Q.

SARSA

- SARSA: On-Policy Temporal Difference Method.

$$\begin{aligned} Q(s_t, a_t) \quad \leftarrow \quad & (1-\alpha_t)Q(s_t, a_t) \\ & +\alpha_t \left(R(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})\right) \end{aligned}$$

- Exploration / Exploitation Problem.
  - Use stochastic policy.

- SARSA performs 1-step temporal difference updates.

Problem Formulations

- Learn optimal policy.
  - Or best possible approximation.

- Optimal Learning: Make as few as possible mistakes during learning.
  - Exploration / exploitation problem.

Exploration / Exploitation Problem

- Tradeoff between

  - using the current best policy to maximize (greedy) reward.
    (Exploitation)

  - and exploring currently suboptimal actions whose values are still uncertain in order to find a potentially better policy.
    (Exploration)

Bandit Problem

- n-armed bandit problem:
  - n actions (arms / slot machines) .
  - Each action has different expected reward.
  - Expected reward unknown.
  - Problem: find best action without losing too much on the way.

- Expected reward for action a is $Q^*(a)$.
- Estimated expected reward after t trials:

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{n_a(t)}}{n_a(t)}$$

Greedy and ϵ-Greedy Policies

- Greedy:

$$\pi = \arg\max_a Q_t(a)$$

- ϵ-greedy

$$\pi = \begin{cases} \arg\max_a Q_t(a) & \text{mit Wahrscheinlichkeit } 1-\epsilon \\ \text{zufällige Aktion} & \text{mit Wahrscheinlichkeit } \epsilon \end{cases}$$

- ϵ-greedy allows for random exploration.

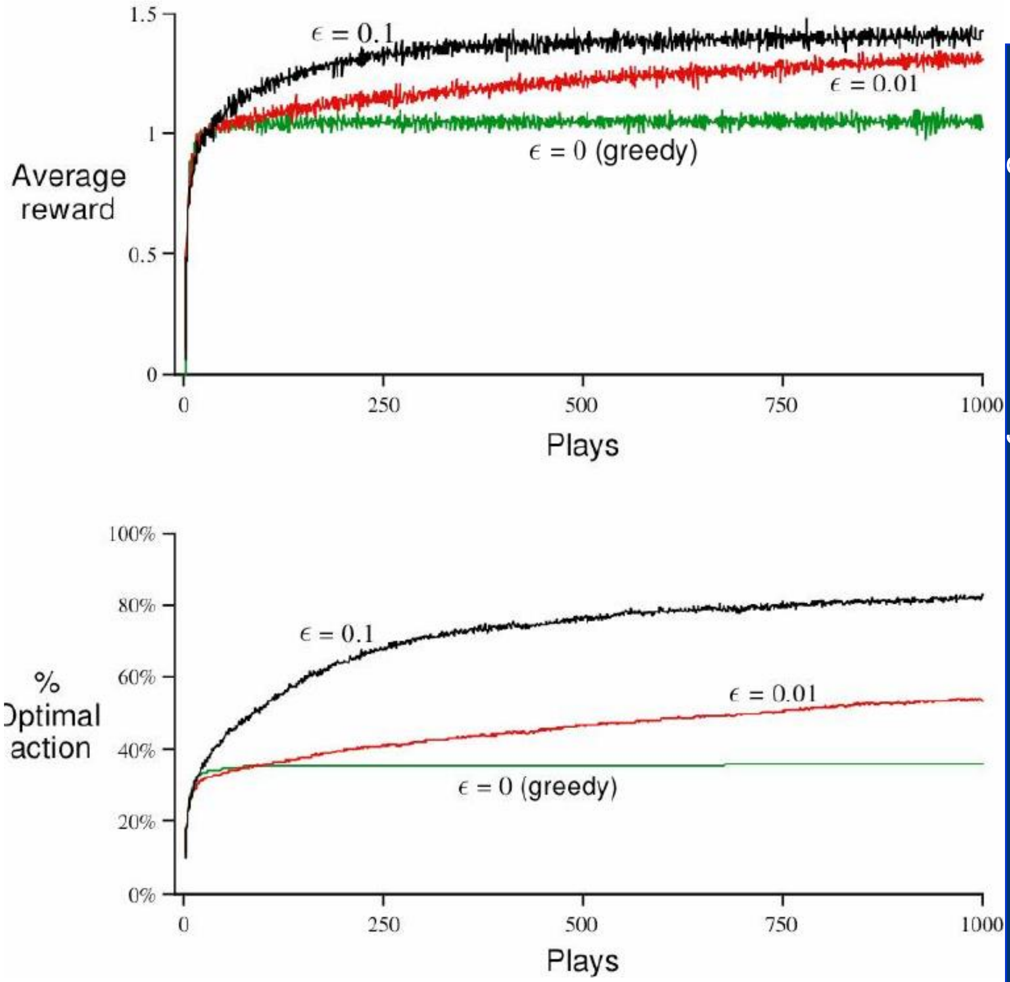
ϵ-Greedy Policies

- 10-armed bandit
- 2000 experiments
- For each experiment draw $Q^*(a)$ for all a:

$$Q^*(a) \sim \mathcal{N}(0, 1)$$

- Rewards are drawn from

$$R_t(a) \sim \mathcal{N}(Q^*(a), 1)$$



Optimism under Uncertainty

- One possible principle for solving the exploration/exploitation dilemma is „optimism under uncertainty“.

- Doesn‘t work in all environments.

- Could for example be implemented by using large initial values for Q.

Optimism under Uncertainty

- Upper Confidence Bound (UCB): [Auer et al. 02]
  - Assume that rewards are bounded by [0,1].

$$\pi = \arg\max_a \left[Q_t(a) + \sqrt{\frac{c_e \log(t)}{2 n_a(t)}} \right] \;, \; c_e \geq 2$$

- Good results for stationary environments and i.i.d. rewards.

Problem Formulations

- P,R known. $P(s`|s,a)$ can be queried.
- P,R not explicitly known. But we can sample from the distributions $P(s`|s,a)$. Assumption: Generative model of P and R.
- P,R not or only partly known. We can gain experience by interacting with the environment. $\rightarrow$ Reinforcement Learning.
- Batch Reinforcement Learning: We have to learn from a fixed set of episodes.

Large and Infinite State Spaces

- In realistic applications state spaces are usually very large or continuous.
- So far: Assumption that value function could be stored as a table.

- Different approaches:
  - Planning:
    - Monte-Carlo Sampling
    - Discretization with subsequent Value Iteration (or PI)
  - Approximation of value function with function approximation methods.
  - Direct learning of policy.

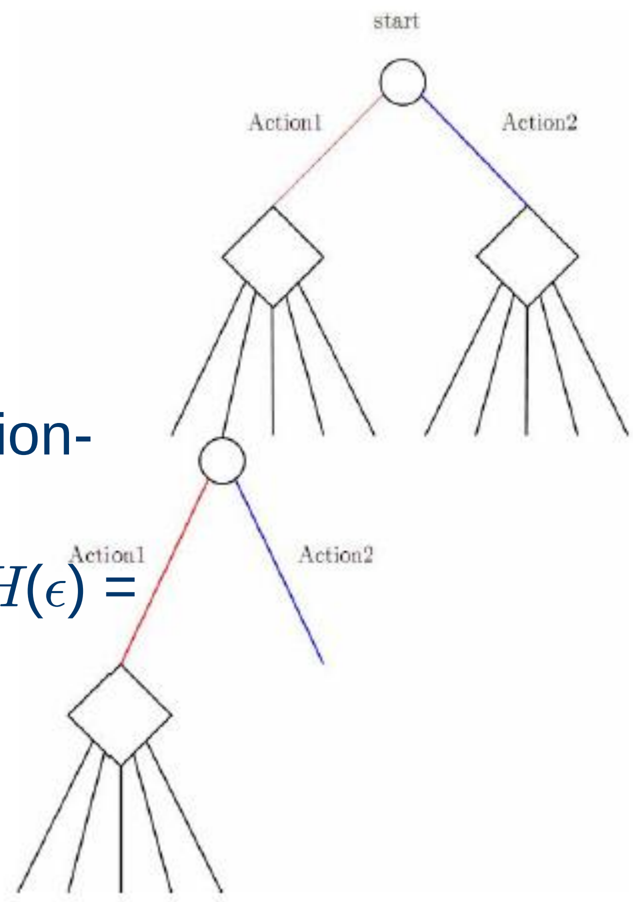
Approximation

- Types of approximations
  - Representation, e.g.
    - Value function $\hat{Q}(s,a;\theta) = \phi^T(s,a)\theta$
    - Policy $\pi(s,a;\theta) = h(\phi^T(s,a) \cdot \theta)$
  - Sampling
    - Online learning via interactions.
    - Sample from generative model of environment.
  - Maximization
    - Find the good action instead of best action for current state.

Monte-Carlo Sampling

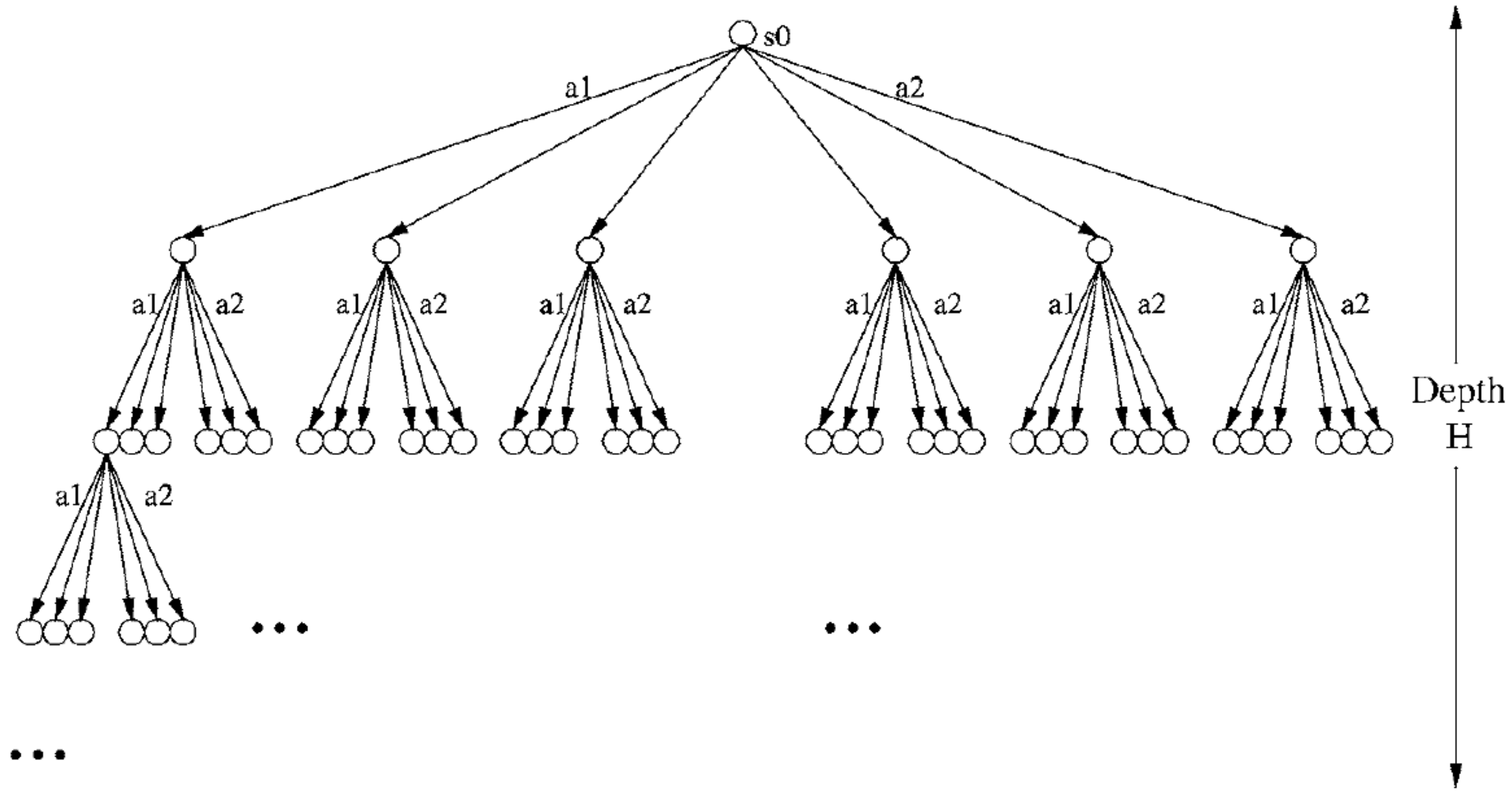
- Assume that S is very large
- Goal: Find Q, s.t. $||Q\text{-}Q^*||_\infty < \epsilon$.

- Sparse Lookahead Trees: [Kearns et al. 02]
  - Monte-Carlo: Sample sparse action-state tree.
  - Depth of tree: Effective horizon $H(\epsilon)$ = O($1/(1\text{-}\gamma)$ log($1/\epsilon(1\text{-}\gamma)$))
  - MC independent of |S|
  - But exponential in $H(\epsilon)$: minimal size of tree.



$$c|A|^{H(\epsilon)}$$

Sparse Lookahead Trees



Upper Confidence Bounds for Trees

- Improvement: Only inspect those parts of the tree that look promising.
- Optimism under uncertainty!
  - Same principle as for the bandit problem.
  - UCT: UCB for Trees.
    [Kocsis & Szepesvári 06]

$$\pi(s,a) = \arg\max_a \left[Q_t(s,a) + \sqrt{\frac{c_e \log(t)}{2n_{s,a}(t)}} \right] \;,\; c_e \geq 2$$

UCT Performance: Go

- Very good results in Go.
- 9x9 & 19x19

- Computer Olympics 2007 - 2009:
  - 2007 & 2008: 1st to 3rd places employed variants of UCT.
  - More general: Monte-Carlo Search Trees (MCST).
  - 2009: At least 2nd and 3rd employed variants of UCT.

Discretization

- Continuous state space S.
- Random Discretization Method: [Rust 97]
  - Sample states S‘ according to uniform distribution over state space.
  - Value iteration.
- Continuous value iteration:

$$V_{t+1}(s) = \max_a \left[R(s,a) + \gamma \int_{s'} p(s'|s,a) V_t(s') ds' \right]$$

- Discretization: Weighted Importance Sampling

$$\frac{\sum_{i=1}^{N} p(s_i|s,a) V(s_i)}{\sum_{j=0}^{N} p(s_j|s,a)} \rightarrow \int_{s'} p(s'|s,a) V(s') \text{ , für N} \rightarrow \infty$$

Discretization

- Compute value function $V(s)$ for states that are not in sample set S‘:
- Bellman update step:

$$V^*(s) = \max_a \left[R(s,a) + \gamma \sum_{i=1}^{N} \frac{p(s_i|s,a)}{\sum_{j=0}^{N} p(s_j|s,a)} V^*(s_i) \right]$$

- Guaranteed performance: [Rust97]
  Assumption: S=[0,1]^d

$$E[\|V_N(s) - V^*(s)\|_\infty^2] \leq \frac{Cd|A|^{5/4}}{(1-\gamma)^2 N^{1/4}}$$