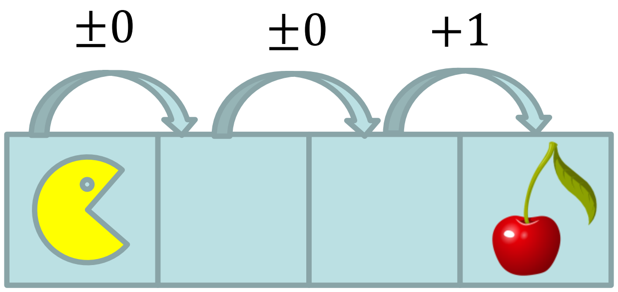
Temporal Difference Learning & Policy Iteration

Advanced Topics in Reinforcement Learning Seminar

WS 15/16



by Tobias Joppen

Overview

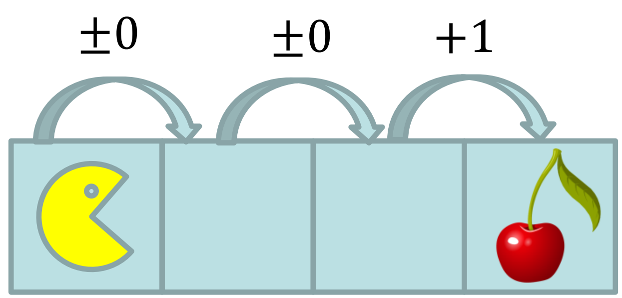
- Introduction
  - Reinforcement Learning Model
  - Learning procedure
  - Markov Property
  - Value Function
- Policy Iteration
- Temporal Difference Learning
  - Idea
  - Update Rule
- Application

Introduction (Reinforcement Learning)

- Reinforcement learning ⊆ machine learning
- Learn by reinforcements (good / bad moves)
- Want: a policy how to behave in different situations
- Mostly sequential problems

$$\text{state} \xrightarrow{\text{action}} \text{state} \xrightarrow{\text{action}} \text{state} \xrightarrow{\text{action}} \text{state} \overset{\dots}{\to} \dots \left(\xrightarrow{\text{action}} \text{final state}\right)$$

- Delayed Rewards



Reinforcement-Learning Model

„An agent is connected to its environment via perception and action“

- Leslie Pack Kaelbling

Formal model consists of:

- A discrete set of states S
- A discrete set of actions A
- A set of scalar reinforcement signals (Rewards)

Main Goal:

Find a policy $\pi : S \rightarrow A$, mapping states to actions, that maximizes some long-run measure of reinforcement. ($\pi(s)$ is the chosen action in state s)

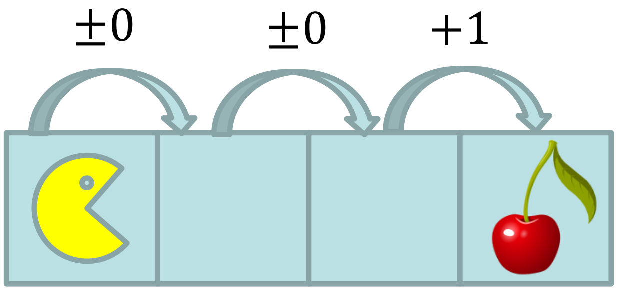
Main procedure

Agent is in state s_1 and can choose from actions $\{a_1, \ldots, a_n\}$.

Agent chooses action a_i.

Agent gets a reinforcement (reward) r_1 and is given its new state s_2 and a new set of possible actions.

This loops until a given final state is reached or the procedure gets canceled.



Markov Decision Process

In general: **Environment non-deterministic**

(Same action in same state can lead to different reinforcements and states)

But: The probability for a outcome is fixed (**Static environment**).

The **markov property**:

The outcome of an action in a given state does not depend on anything but the state/action pair. (Does not depend on earlier actions or states)

Markov Decision Process

State transition function $T: S \times A \rightarrow \Pi(S)$

members of $\Pi(S)$ are probability distribution over the set S

for each state there is a probability that a pair (s, a) leads into this state

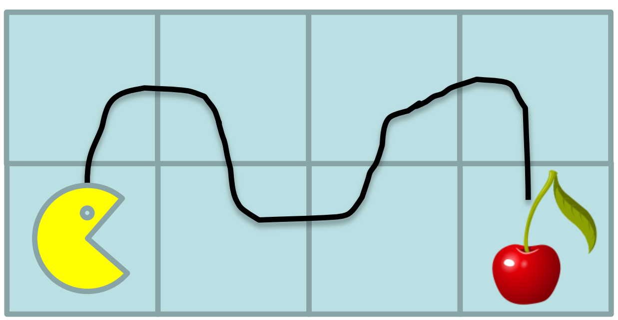
We write $T(s, a, s')$ for the probability of making a transition from state s to state s' using action a.

Measurement of performing good

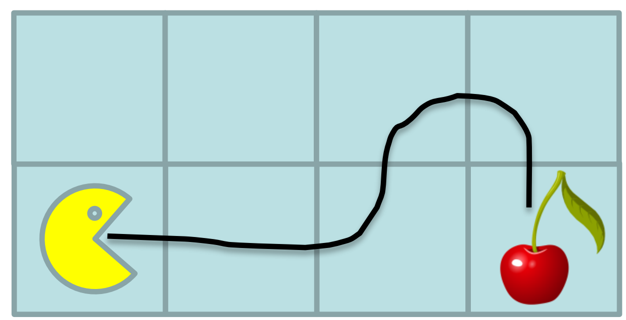
- the learner needs information about the quality of a state
- want to find optimal policy π^*

A value function V represents the expected future reward for a current state, if the agent follows policy π:

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



$<$



Optimality

TECHNISCHE UNIVERSITÄT DARMSTADT

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

The optimal policy is the policy with the highest value for all states s:

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

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1

Optimal value function $V^*(s)$ → optimal policy:

$$\pi^*(s) = \arg\max_{a}[R(s,a) + \gamma \sum_{s' \in S} T(s,\pi(s),s')V^*(s')]$$

Policy Improvement

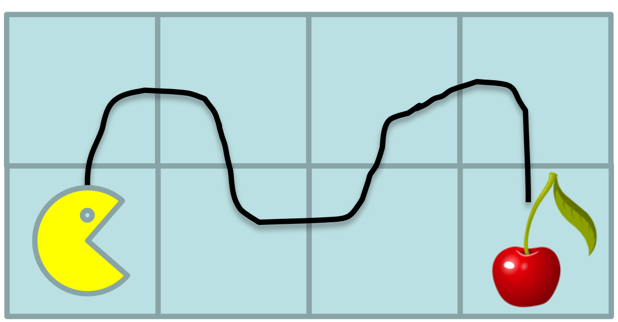
The task: improve a policy π, such that it performs better.

That means, it has a bigger cummulated expected reward afterwards (for each state s).

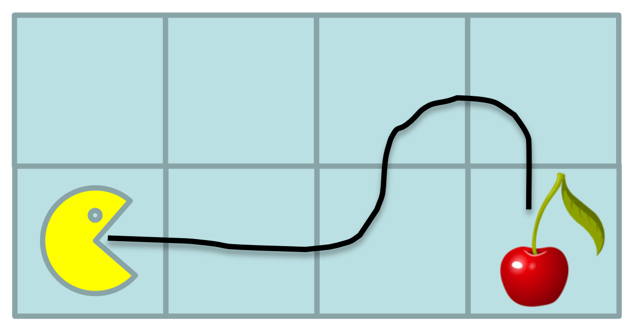
If it is $V^{\pi}(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} T(s,\pi(s),s') \; V^{\pi}(s')$

$$< R(s,a) \quad + \gamma \sum_{s' \in S} T(s,a,s') \quad V^{\pi}(s')$$

than it is better to choose action a instead of $\pi(s)$ in the current state s_t. So one can improve the policy by choosing acion a‘ instead of a_t in state s_t.



$<$



Policy Iteration

If this improvement is done multiple times with all possible states, this is called policy iteration.

One iteration consists of:

1. Calculating the value function of the current policy π for each state s

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

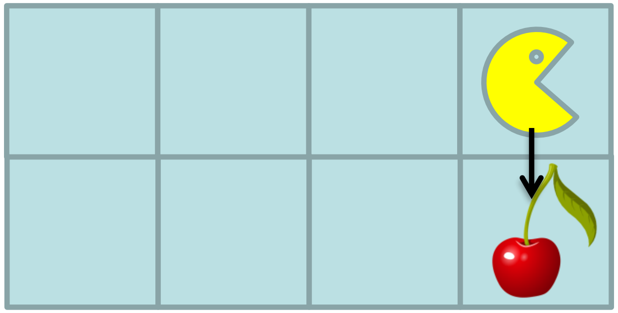
2. Improve the policy at each state

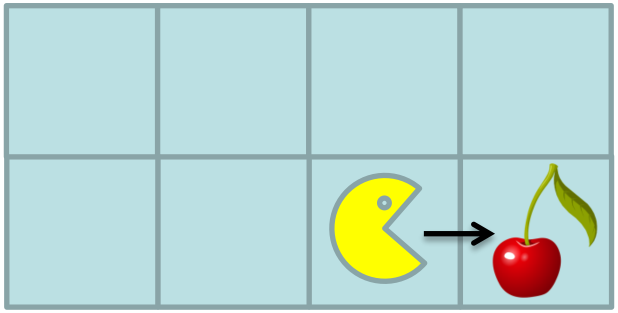
$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_2} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} V^*$$

Policy Iteration

Why does it work (why does it take multiple iterations)?

- After 1 Iteration, at least this is optimal:

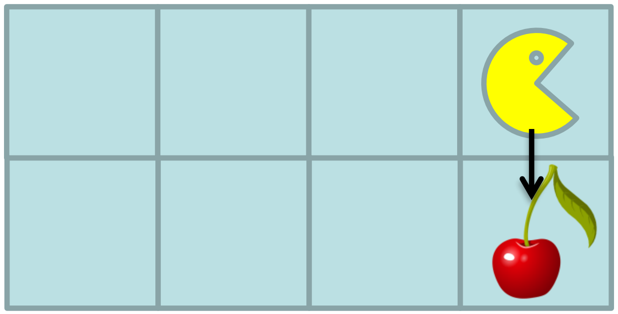


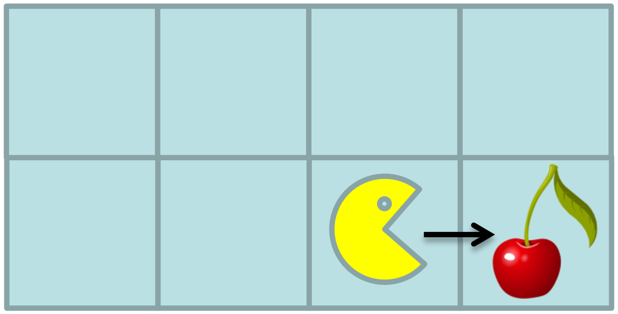


Policy Iteration

Why does it work (why does it take multiple iterations)?

- After 1 Iteration, at least this is optimal:





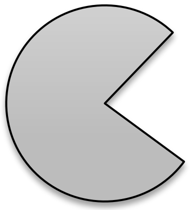
This works, but

- Needs many policy evaluations (What is $V^{\pi}(s)$?)
- Need a model of the environment (need to know R and T to get $V^{\pi}(s)$)
- May have large trajectories (expensive computation, memory usage)

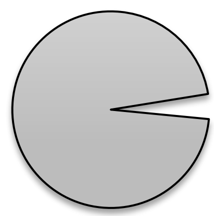
Can we learn $V^{\pi}(s)$ without knowing R and T ?

?

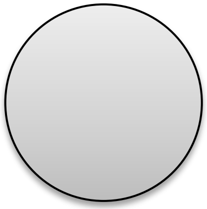
Can we learn $V^{\pi}(s)$ without knowing R and T ?



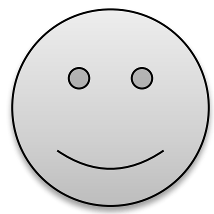
Can we learn $V^{\pi}(s)$ without knowing R and T ?



Can we learn $V^{\pi}(s)$ without knowing R and T ?



Can we learn $V^{\pi}(s)$ without knowing R and T ?



Can we learn $V^{\pi}(s)$ without knowing R and T ?

Yes!

How to learn $V^{\pi}(s)$ without knowing R and T ?

- Introduction
  - Reinforcement Learning Model
  - Learning procedure
  - Markov Property
  - Value Function
- Policy Iteration
- Temporal Difference Learning
  - Idea
  - Update Rule

How to learn $V^{\pi}(s)$ without knowing R and T ?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Introduction
  - Reinforcement Learning Model
  - Learning procedure
  - Markov Property
  - Value Function
- Policy Iteration
- Temporal Difference Learning
  - Idea
  - Update Rule

Temporal difference learning

- Following a policy π gives experience
- Use this to update estimate V of V^π
- Begin with *incorrect* estimate V

  the true values V^π are unknown

  (otherwise there is nothing you need to learn)
- Learn correct value function

0	0	0	0
0	0	0	0

(using a good policy)

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1

Temporal difference learning

Recall the value function:

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

For deterministic environments, the value function is easier!

- Lets take a look at this …

Since $T(s, a, s')$ is 0 for $a \neq \pi(s)$ and 1 otherwise,

in deterministic environment it is:

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma\ V^{\pi}(\delta(s, \pi(s)))$$

where $\delta : S \times A \rightarrow S$ is the transition function

Temporal difference learning

Example (non-deterministic environment):

- The agent is in state s at time step t (say s_t)
- $V(s_t)$ is the expected future reward
- Agent performs action $\mathrm{a} := \pi(s_t)$
- Leads to state s_{t+1} with reinforcement $R(s,a)$
- Now we have a new expected future reward: $V(s_{t+1})$
- Is $V(s_t) = R(s,a) + \gamma V(s_{t+1})$?
  - For V^{π} it‘s true: $V^{\pi}(s) = R(s,a) + \gamma\ V^{\pi}(s_{t+1})$

Temporal difference learning

$V(s_t) = R(s,a) + \gamma V(s_{t+1})$ in common doesn‘t hold, because the expectations are not correct.

Temporal difference learning

$V(s_t) = R(s,a) + \gamma V(s_{t+1})$ in common doesn‘t hold, because the expectations are not correct.

Is $R(s,a) + \gamma V(s_{t+1})$ a better value for $V(s_t)$?

Should the old value be replaced by the new one?

Temporal difference learning

TECHNISCHE UNIVERSITÄT DARMSTADT

$V(s_t) = R(s,a) + \gamma V(s_{t+1})$ in common doesn‘t hold, because the expectations are not correct.

Is $R(s,a) + \gamma V(s_{t+1})$ a better value for $V(s_t)$?

Should the old value be replaced by the new one?

- No!
  - Because the term $V(s_{t+1})$ is just as wrong as $V(s_t)$ was
  - Because the learner would forget everything he learned before about $V(s_t)$

Temporal difference learning

$V(s_t) = R(s,a) + \gamma V(s_{t+1})$ in common doesn‘t hold, because the expectations are not correct.

Is $R(s,a) + \gamma V(s_{t+1})$ a better value for $V(s_t)$?

Should the old value be replaced by the new one?

- No!
  - Because the term $V(s_{t+1})$ is just as wrong as $V(s_t)$ was
  - Because the learner would forget everything he learned before about $V(s_t)$
- But $R(s,a) + \gamma V(s_{t+1})$ has a bit of truth in it:
  - $R(s,a)$ is the correct reinforcement value
  - $V(s_{t+1})$ hopefully converges to $V^{\pi}(s_{t+1})$

TD - Update Rule

So both $V(s_t)$ and $R(s,a) + \gamma V(s_{t+1})$ have some value:

$V(s_t)$ contains old knowledge

$R(s,a) + \gamma V(s_{t+1})$ contains new experience

TD - Update Rule

So both $V(s_t)$ and $R(s,a) + \gamma V(s_{t+1})$ have some value:

$V(s_t)$ contains old knowledge

$R(s,a) + \gamma V(s_{t+1})$ contains new experience

A proper update for a new value for $V(s_t)$ is a mixture of both!

TD - Update Rule

So both $V(s_t)$ and $R(s,a) + \gamma V(s_{t+1})$ have some value:

$V(s_t)$ contains old knowledge

$R(s,a) + \gamma V(s_{t+1})$ contains new experience

A proper update for a new value for $V(s_t)$ is a mixture of both!

A often used update-rule, known as $TD(0)$, is:

$V(s_t) \leftarrow V(s_t) + \alpha[R(s,a) + \gamma V(s_{t+1}) - V(s_t)]$

TD - Update Rule

So both $V(s_t)$ and $R(s,a) + \gamma V(s_{t+1})$ have some value:

$V(s_t)$ contains old knowledge

$R(s,a) + \gamma V(s_{t+1})$ contains new experience

A proper update for a new value for $V(s_t)$ is a mixture of both!

A often used update-rule, known as $TD(0)$, is:

$$V(s_t) \leftarrow V(s_t) + \alpha[R(s,a) + \gamma V(s_{t+1}) - V(s_t)]$$

Using $TD(0)$ the real V^π will be learned eventually!

(using good values for γ, α and running long enough)

Application

This only learnes V^{π}, not the best policy π^*!

Main Goal: Find a policy $\pi : S \rightarrow A$, mapping states to actions, that maximizes some long-run measure of reinforcement.

To achieve this, one has to:

- Learn the value of $S \times A$, not only S (V vs. Q-Function)
- Don‘t follow a single policy, but get actions using an *explorator*
- At the beginning: explore the state-space
- To the end: exploit the gathered knowledge

The end

Thanks for your attention!

Any questions?

References

Sutton, Richard S., and Andrew G. Barto. *Introduction to reinforcement learning*. Vol. 135. Cambridge: MIT Press, 1998.

DIETTERICH, ANDREW G. BARTO THOMAS G. "2 Reinforcement Learning and Its Relationship to Supervised Learning." *Handbook of learning and approximate dynamic programming* 2 (2004): 47.

Kunz, Florian. "An Introduction to Temporal Difference Learning.“

Sutton, Richard S. "Learning to predict by the methods of temporal differences." *Machine learning* 3.1 (1988): 9-44.

Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." *Journal of artificial intelligence research*(1996): 237-285.