A (not so) short introduction to Reinforcement Learning

Jost Tobias Springenberg
springj@informatik.uni-freiburg.de

Machine Learning Lab - University of Freiburg

December 23, 2016

Disclaimer

- Reinforcement Learning (RL) is a huge field, I will focus on the very basics that you need for your exercise

Overview

What is a MDP ?

- Before I detail anything complicated let us look at the framework of Markov Decision Processes (MDPs) which we will use to describe all RL algorithms
- The situation an MDP seeks to model:

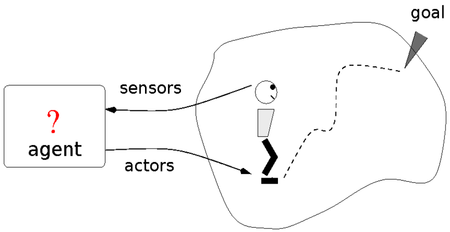
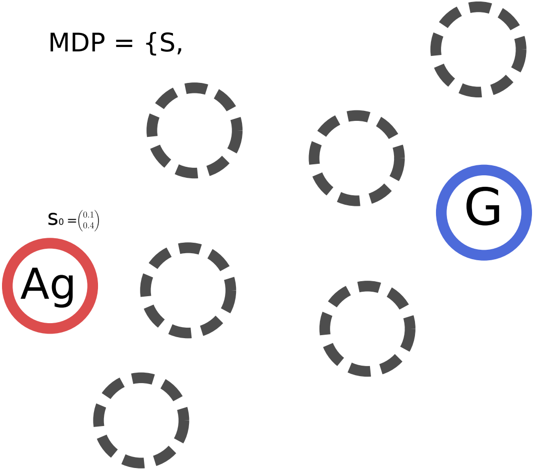


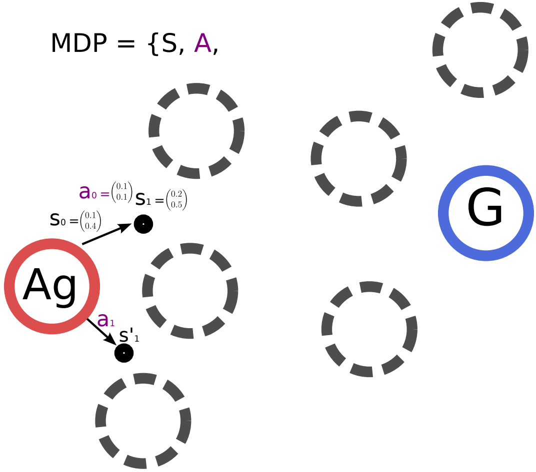
Figure: Taken from RL leacture by Martin Riedmiller

What is a MDP ?

MDP = {S,

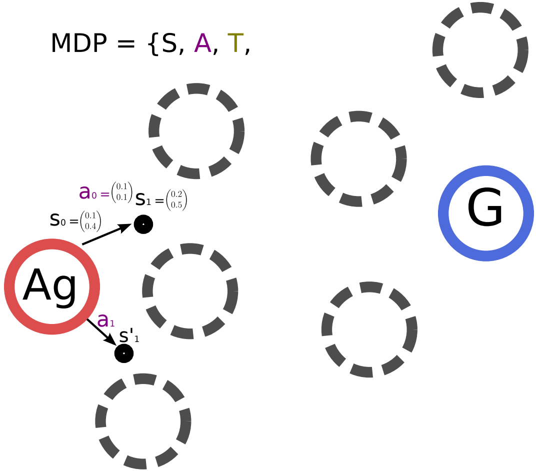


What is a MDP ?



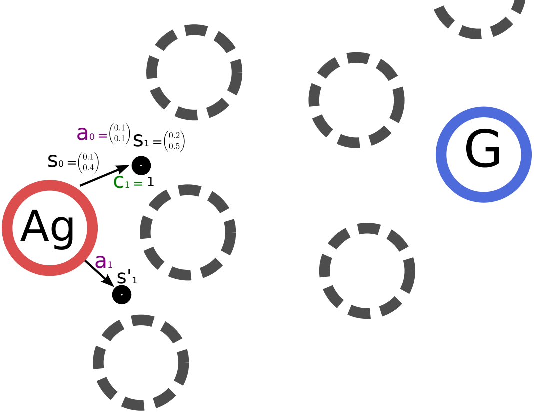
What is a MDP ?

$\text{MDP} = \{S, A, T,$

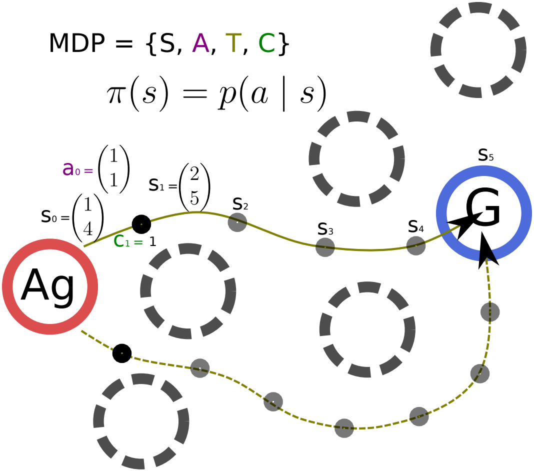


What is a MDP ?

MDP = {S, A, T, C}



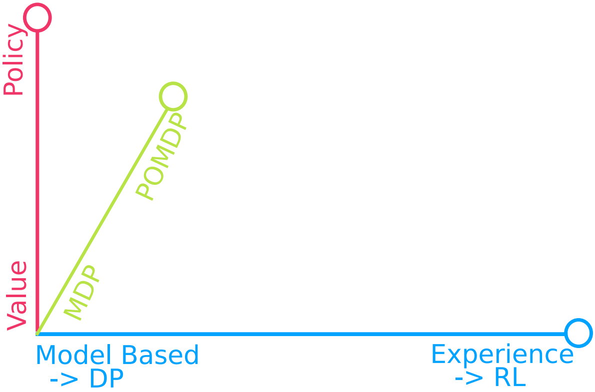
What is a MDP ?



MDP = {S, A, T, C}
$\pi(s) = p(a \mid s)$
$a_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
$s_1 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$
$s_0 = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$
$c_1 = 1$
s_2
s_3
s_4
s_5
Ag
G

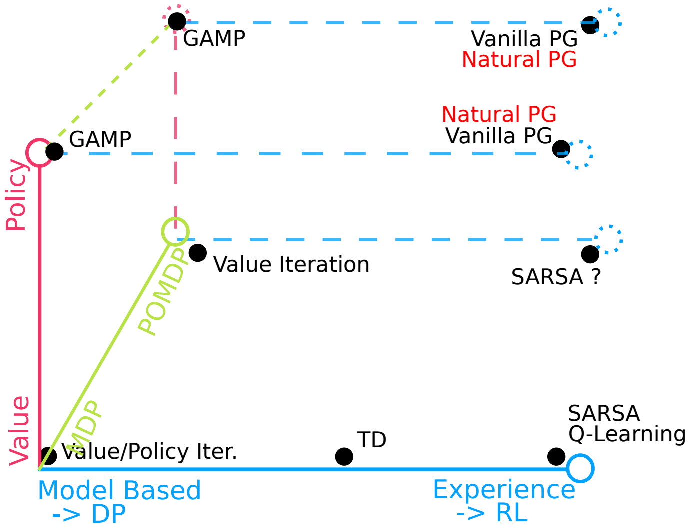
The Dimensions of RL

- We are going to look at RL methods by considering three axes:



The Dimensions of RL

- We are going to look at RL methods by considering three axes:



RL Definitions

Ok let us formalize

- A MDP is a 5-tuple

$$\text{MDP} = (S, A, P, R)$$

RL Definitions

Ok let us formalize

- A MDP is a 5-tuple

$$\text{MDP} = (S, A, P, R)$$

- Discrete timesteps t with states and actions $s_t \in S$, $a_t \in A$

RL Definitions

Ok let us formalize

- A MDP is a 5-tuple

$$\text{MDP} = (S, A, P, R)$$

- Discrete timesteps t with states and actions $s_t \in S$, $a_t \in A$
- Deterministic rewards: $r_t = r(s_t, a_t) \mapsto \mathbb{R}$

RL Definitions

Ok let us formalize

- A MDP is a 5-tuple

$$\text{MDP} = (S, A, P, R)$$

- Discrete timesteps t with states and actions $s_t \in S$, $a_t \in A$
- Deterministic rewards: $r_t = r(s_t, a_t) \mapsto \mathbb{R}$
- Probabilistic transition model P: $p(s_{t+1} \mid s_t, a_t)$
  - Assume the *markov property*: the future depends on the past only through the present state

RL Definitions

Ok let us formalize

- A MDP is a 5-tuple

$$\text{MDP} = (S, A, P, R)$$

- Discrete timesteps t with states and actions $s_t \in S$, $a_t \in A$
- Deterministic rewards: $r_t = r(s_t, a_t) \mapsto \mathbb{R}$
- Probabilistic transition model P: $p(s_{t+1} \mid s_t, a_t)$
  - Assume the *markov property*: the future depends on the past only through the present state

  $\rightarrow$ $p(s_{t+1} \mid s_t, a_t)$ independent of previous states and actions

RL Definitions

Ok let us formalize

- A MDP is a 5-tuple

$$\text{MDP} = (S, A, P, R)$$

- Discrete timesteps t with states and actions $s_t \in S$, $a_t \in A$
- Deterministic rewards: $r_t = r(s_t, a_t) \mapsto \mathbb{R}$
- Probabilistic transition model P: $p(s_{t+1} \mid s_t, a_t)$
  - Assume the *markov property*: the future depends on the past only through the present state

  $\rightarrow$ $p(s_{t+1} \mid s_t, a_t)$ independent of previous states and actions
- The agents policy is described as: $\pi(s_t, a_t) = p(a_t \mid s_t)$

Model Based RL

$\rightarrow$ Question How can we figure out what action to take in a state ?

Model Based RL

→ Question How can we figure out what action to take in a state ?

- We can assign a Value V to each state as follows.
- Let us assume a finite horizon

$$V_T^{\pi}(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\}$$

Model Based RL

→ Question How can we figure out what action to take in a state ?

- We can assign a Value V to each state as follows.
- Let us assume a finite horizon

$$V_T^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\}$$

- Note: if we knew the V^π $\forall\pi$ behaving optimally: pick policy with maximum expected reward when starting in s_0

Model Based RL

→ Question How can we figure out what action to take in a state ?

- We can assign a Value V to each state as follows.
- Let us assume a finite horizon

$$V_T^{\pi}(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\}$$

- Note: if we knew the V^{π} $\forall \pi$ behaving optimally: pick policy with maximum expected reward when starting in s_0
- Problem: But we dont really want to enumerate all of these

Model Based RL

→ Question How can we figure out what action to take in a state ?

- We can assign a Value V to each state as follows.
- Let us assume a finite horizon

$$V_T^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\}$$

- Note: if we knew the V^π $\forall\pi$ behaving optimally: pick policy with maximum expected reward when starting in s_0
- Problem: But we dont really want to enumerate all of these
- Solution: Classic dynamic programming

Model Based RL - Dynamic Programming (DP)

- Solution: Classic dynamic programming
- Wanted: $\pi^\star = \arg\max_\pi V_T^\pi(s)$

Model Based RL - Dynamic Programming (DP)

- Solution: Classic dynamic programming
- Wanted: $\pi^\star = \arg\max_\pi V_T^\pi(s)$
- Bellmans optimality principle states that for optimal $\pi^\star$ we can reformulate

$$
\begin{aligned}
V_T^{\pi^\star}(s_t) &= \mathbb{E}_{\pi^\star, P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\} \\
&= \max_a r(s,a) + \sum_{s' \in S} p(s' \mid s, a) V_{T-1}^{\pi^\star}(s')
\end{aligned}
$$

Model Based RL - Dynamic Programming (DP)

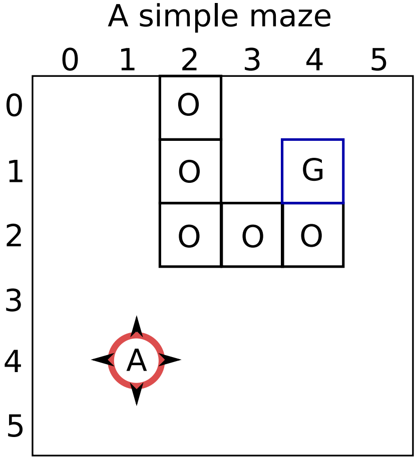
- Solution: Classic dynamic programming
- Wanted: $\pi^\star = \arg\max_\pi V_T^\pi(s)$
- Bellmans optimality principle states that for optimal $\pi^\star$ we can reformulate

$$
\begin{aligned}
V_T^{\pi^\star}(s_t) &= \mathbb{E}_{\pi^\star, P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\} \\
&= \max_a r(s,a) + \sum_{s' \in S} p(s' \mid s, a) V_{T-1}^{\pi^\star}(s')
\end{aligned}
$$

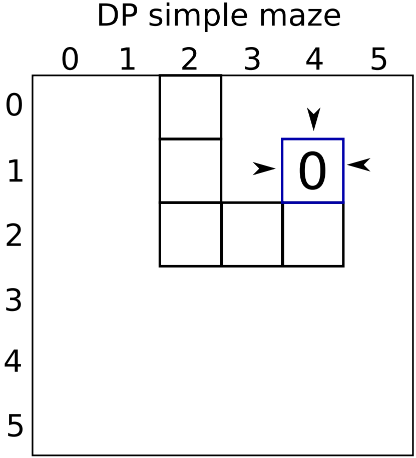
$\rightarrow$ Set $V_0^{\pi^\star}(s') = \max_a r(s,a)$ and iterate

Model Based RL - Dynamic Programming (DP)

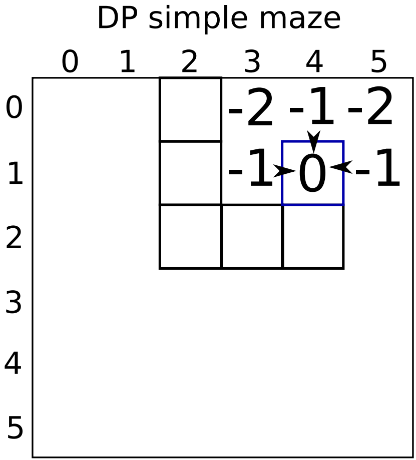
- But is this actually any use ? Show me a solvable problem!



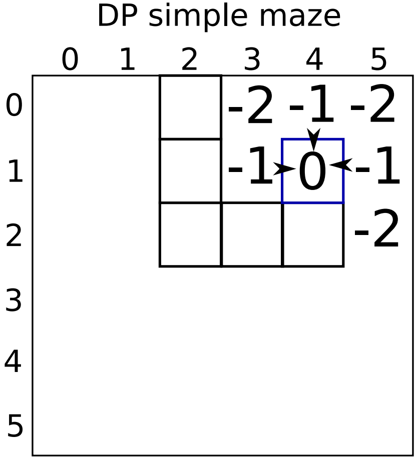
Model Based RL - Dynamic Programming (DP)



Model Based RL - Dynamic Programming (DP)



Model Based RL - Dynamic Programming (DP)



→ lather, rinse, repeat

Model Based RL - DP Problems

- DP works great but only allows Agent to “live” for N steps ...

Model Based RL - DP Problems

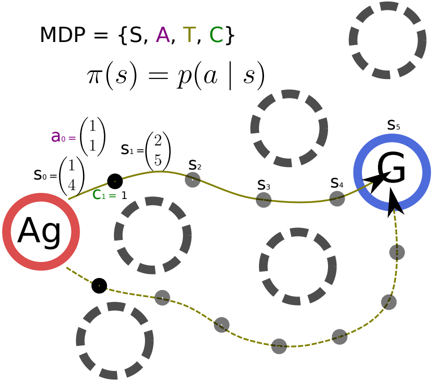
- DP works great but only allows Agent to "live" for N steps ...
- That is not the problem we really care for at all !

Model Based RL - DP Problems

- DP works great but only allows Agent to "live" for N steps ...
- That is not the problem we really care for at all !
- Remember (continuous states/actions, infinite horizon):

MDP = {S, A, T, C}

$$\pi(s) = p(a \mid s)$$



Model Based RL - DP Problems

- Recall the definition of the state value:

$$V_T^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\}$$

Model Based RL - DP Problems

- Recall the definition of the state value:

$$V_T^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\}$$

- Why don't we consider an infinite horizon ?

$$V_\infty^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{\infty} r_t \mid s_o = s\}$$

Model Based RL - DP Problems

- Recall the definition of the state value:

$$V_T^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\}$$

- Why don't we consider an infinite horizon ?

$$V_\infty^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{\infty} r_t \mid s_o = s\}$$

- Not bounded, we don't want infinite costs

→ Solve this via discounting with $\gamma \leq 1$

$$V^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{\infty} \gamma^t r_t \mid s_o = s\}$$

Model Based RL - DP Problems

- Recall the definition of the state value:

$$V_T^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{T-1} r_t \mid s_o = s\}$$

- Why don't we consider an infinite horizon ?

$$V_\infty^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{\infty} r_t \mid s_o = s\}$$

- Not bounded, we don't want infinite costs

→ Solve this via discounting with $\gamma \leq 1$

$$V^\pi(s) = \mathbb{E}_{\pi,P}\{\sum_{t=0}^{\infty} \gamma^t r_t \mid s_o = s\}$$

- Discounting is like assuming a non-zero interest rate — money arriving in the future is worth less than money arriving now.

Model Based RL - Value iteration

- Can we solve for the optimal value function (and hence policy) ?

Model Based RL - Value iteration

- Can we solve for the optimal value function (and hence policy) ?
- Solution is to iteratively do DP like updates with discounting

Algorithm 2: Value Iteration

```
repeat
    for s ∈ S do
        update V values using current estimate V_i
        V*_{i+1}(s) ← max_a Σ_{s'∈S} p(s' | s, a)(r(s, a) + γ V*_i(s'))
until converged
return V*
```

$$V^\star_{i+1}(s) \leftarrow \max_a \sum_{s' \in S} p(s' \mid s, a)(r(s, a) + \gamma V^\star_i(s'))$$

Model Based RL - Value iteration

- Can we solve for the optimal value function (and hence policy) ?
- Solution is to iteratively do DP like updates with discounting

Algorithm 3: Value Iteration

repeat
  for $s \in S$ **do**
    update V values using current estimate V_i
    $V^\star_{i+1}(s) \leftarrow \max_a \sum_{s' \in S} p(s' \mid s, a)(r(s,a) + \gamma V^\star_i(s'))$
until *converged*
return $V^\star$

→ Converges under mild assumptions

Model Based RL - Value iteration

- Can we solve for the optimal value function (and hence policy) ?
- Solution is to iteratively do DP like updates with discounting

Algorithm 4: Value Iteration

repeat
  for $s \in S$ **do**
    update V values using current estimate V_i
    $V^\star_{i+1}(s) \leftarrow \max_a \sum_{s' \in S} p(s' \mid s, a)(r(s,a) + \gamma V^\star_i(s'))$
until *converged*
return $V^\star$

→ Converges under mild assumptions

→ However only in the limit of infinite iterations ...

Model Based RL - Further Problems

- But what if we don't have a model ?
  - $\rightarrow$ Learn a model (see e.g. PILCO, or our ADPRL paper)

    movie1
  - $\rightarrow$ Use model free methods (e.g. Q-learning, SARSA)

Model Based RL - Further Problems

- But what if we don't have a model ?
  - → Learn a model (see e.g. PILCO, or our ADPRL paper)

    movie1
  - → Use model free methods (e.g. Q-learning, SARSA)
- And what if we have continuous states ?
  - → Use function approximation (e.g. a Neural Network)

Model Free RL

- But what what if we don't have a model ?

Algorithm 5: Value Iteration

```
repeat
    for s ∈ S do
        update V values using current estimate V_i
        V*_{i+1}(s) ← max_a Σ_{s'∈S} p(s' | s, a) (r(s, a) + γ V*_i(s'))
until converged
return V*
```

$$V^\star_{i+1}(s) \leftarrow \max_a \sum_{s' \in S} p(s' \mid s, a) \left(r(s,a) + \gamma V^\star_i(s')\right)$$

(Somewhat) Model Free RL - TD-learning

- Somewhat Model Free RL $\rightarrow$ use sampling

Algorithm 6: TD learning

repeat

choose $a = \arg\max_a \; r(s,a) + \sum_{s'} p(s' \mid s,a) V(s')$

record transition (s, a, r, s')

update V using sample (td) error $\delta_t = r(s,a) + \gamma V_i(s') - V_i(s)$

$V_{i+1}(s) \leftarrow V_i(s) + \alpha \delta_t$

until *converged*

return V

(Somewhat) Model Free RL - TD-learning

- Almost there!

(Somewhat) Model Free RL - TD-learning

- Almost there!
- Need a way to query the value of actions without using the model

(Somewhat) Model Free RL - TD-learning

- Almost there!
- Need a way to query the value of actions without using the model
- Introducing the Q-function

$$Q^{\pi}(s,a) = r(s,a) + \sum_{s'} p(s' \mid s,a) V^{\pi}(s')$$

Model Free RL - Q-learning

Algorithm 7: Q-learning

repeat

  choose $a = \arg\max_a Q(s, a)$

  record transition (s, a, r, s')

  update Q using sample Q-error

  $\delta_t = r(s, a) + \gamma \max_{a'} Q_i(s', a') - Q_i(s, a)$

  $Q_{i+1}(s, a) \leftarrow Q_i(s, a) + \alpha \delta_t$

until *converged*

return Q

Model Free RL - Q-learning

Algorithm 8: Q-learning

repeat

  choose $a = \begin{cases} \arg\max_a Q(s,a) & \text{with probability } \epsilon \\ a \in A & \text{else choose randomly} \end{cases}$

  record transition (s, a, r, s')

  update Q using sample Q-error

  $\delta_t = r(s,a) + \gamma \max_{a'} Q_i(s', a') - Q_i(s,a)$

  $Q_{i+1}(s,a) \leftarrow Q_i(s,a) + \alpha\delta_t$

until *converged*

return Q

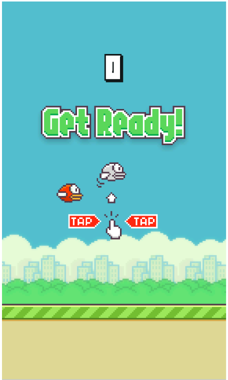
Model Free RL - Q-learning

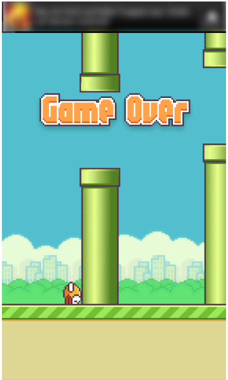
$\rightarrow$ Remarkably Q-learning converges

- Okay, we don't need a model anymore, let's see some more applications!

Model Free RL - Q-learning

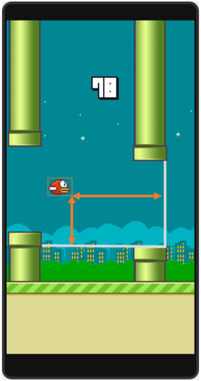
- Finally solving some important problems!





Model Free RL - Q-learning

- Solving FlappyBird with a simple table based Q-learning approach (see http://sarvagyavaish.github.io/FlappyBirdRL/)
- 2-dimensional State-space S: (x, y) pixel distance to tube discretized in 4 pixel steps



Model Free RL - Q-learning

- Solving FlappyBird with a simple table based Q-learning approach (see http://sarvagyavaish.github.io/FlappyBirdRL/)

Model Free RL - Q-learning

- Solving FlappyBird with a simple table based Q-learning approach (see http://sarvagyavaish.github.io/FlappyBirdRL/)
- 2-dimensional State-space S: (x, y) pixel distance to tube discretized in 4 pixel steps

Model Free RL - Q-learning

- Solving FlappyBird with a simple table based Q-learning approach (see http://sarvagyavaish.github.io/FlappyBirdRL/)
- 2-dimensional State-space S: (x, y) pixel distance to tube discretized in 4 pixel steps
- 2-actions $A = \{\text{tap}, \text{donothing}\}$

Model Free RL - Q-learning

- Solving FlappyBird with a simple table based Q-learning approach (see http://sarvagyavaish.github.io/FlappyBirdRL/)
- 2-dimensional State-space S: (x, y) pixel distance to tube discretized in 4 pixel steps
- 2-actions $A = \{\text{tap}, \text{donothing}\}$
- Reward

$$r(s, a) = \begin{cases} 1 & \text{if alive} \\ -1000 & \text{otherwise} \end{cases} \tag{1}$$

Model Free RL - Q-learning

- Solving FlappyBird with a simple table based Q-learning approach (see http://sarvagyavaish.github.io/FlappyBirdRL/)
- 2-dimensional State-space S: (x, y) pixel distance to tube discretized in 4 pixel steps
- 2-actions $A = \{\text{tap}, \text{donothing}\}$
- Reward

$$r(s, a) = \begin{cases} 1 & \text{if alive} \\ -1000 & \text{otherwise} \end{cases} \tag{1}$$

- Solved using on-line Q-learning (training took about 6-7 hours)
- Show VIDEO

Model Free RL - Q-learning

So are we there yet ?

$\rightarrow$ Problems with Q-learning

Model Free RL - Q-learning

So are we there yet ?

$\rightarrow$ Problems with Q-learning

- $\rightarrow$ Curse of dimensionality I: table is not going to work for high dimensions

Model Free RL - Q-learning

So are we there yet ?

→ Problems with Q-learning
  → Curse of dimensionality I: table is not going to work for high dimensions
    → We have only considered discrete states, next: function approximation

Model Free RL - Q-learning

So are we there yet ?

- → Problems with Q-learning
  - → Curse of dimensionality I: table is not going to work for high dimensions
    - → We have only considered discrete states, next: function approximation
  - → Curse of dimensionality II: Efficient Exploration is key! Even with only 2 state-dimensions + 2 actions FlappyBird took hours ..

Model Free RL - Q-learning

Modern approaches / Function approximation

- To (partially) solve the curse of dimensionalty one can try to use function approximation

Model Free RL - Q-learning

Modern approaches / Function approximation

- To (partially) solve the curse of dimensionalty one can try to use function approximation

→ Idea: replace Q-table by a function approximator

  - use Q-error $\delta_t = r(s, a) + \gamma \max_a Q_i(s', a) - Q_i(s, a)$ as the gradient

Model Free RL - From Q-learning to DQN

Algorithm 9: Approximate Q-learning

repeat

choose $a = \begin{cases} \arg\max_a Q(s,a) & \text{with probability } \epsilon \\ a \in A & \text{else choose randomly} \end{cases}$

record transition (s, a, r, s')

update Q using sample Q-error

$$\delta_t = r(s,a) + \gamma \max_{a'} Q_i(s', a') - Q_i(s,a)$$

$$Q_{i+1}(s,a) \leftarrow Q_i(s,a) + \alpha\delta_t$$

until *converged*

return Q

Model Free RL - From Q-learning to DQN

Algorithm 10: Approximate Q-learning

repeat

choose $a = \begin{cases} \arg\max_a Q_W(s,a) & \text{with probability } \epsilon \\ a \in A & \text{else choose randomly} \end{cases}$

record transition (s, a, r, s')

update Q_W using sample Q-error

$$\delta_t = r(s,a) + \gamma \max_{a'} Q_W(s',a') - Q_W(s,a)$$

$$Q_{i+1}(s,a) \leftarrow Q_i(s,a) + \alpha\delta_t$$

until *converged*

return Q

Model Free RL - From Q-learning to DQN

Algorithm 11: Approximate Q-learning

repeat

chose $a^\star = \begin{cases} \max_a Q_W(s,a) & \text{with probability } \epsilon \\ a \in A & \text{else chose randomly} \end{cases}$

record transition (s, a, r, s')

update Q_W using sample Q-error as derivative

$$\delta_t = r(s,a) + \gamma \max_{a'} Q_W(s', a') - Q_W(s,a)$$

$$W \leftarrow W + \alpha \left(\delta_t \frac{\partial Q_W(s,a)}{\partial W} \right)$$

until *converged*

return Q_W

Model Free RL - Q-learning

Modern approaches / Function approximation

- To (partially) solve the curse of dimensionalty one can try to use function approximation

→ Idea: replace Q-table by a function approximator

  - use Q-error $\delta_t = r(s,a) + \gamma \max_a Q_i(s', a) - Q_i(s,a)$ as the gradient

→ Problem: What is the function we are minimizing anyways ? Convergence ?

Model Free RL - Q-learning

Modern approaches / Function approximation

- To (partially) solve the curse of dimensionalty one can try to use function approximation

→ Idea: replace Q-table by a function approximator

  - use Q-error $\delta_t = r(s, a) + \gamma \max_a Q_i(s', a) - Q_i(s, a)$ as the gradient

→ Problem: What is the function we are minimizing anyways ? Convergence ?

  - Unfortunately approximate Q-learning may not converge

Model Free RL - Q-learning

Modern approaches / Function approximation

- To (partially) solve the curse of dimensionalty one can try to use function approximation

$\rightarrow$ Idea: replace Q-table by a function approximator

  - use Q-error $\delta_t = r(s,a) + \gamma \max_a Q_i(s', a) - Q_i(s,a)$ as the gradient

$\rightarrow$ Problem: What is the function we are minimizing anyways ? Convergence ?

  - Unfortunately approximate Q-learning may not converge

$\rightarrow$ Problem: Non-stationarity/Floating targets ? If Q changes the function we minimize changes

Model Free RL - Q-learning

Modern approaches / Function approximation

- To (partially) solve the curse of dimensionalty one can try to use function approximation

→ Idea: replace Q-table by a function approximator
  - use Q-error $\delta_t = r(s,a) + \gamma \max_a Q_i(s', a) - Q_i(s,a)$ as the gradient

→ Problem: What is the function we are minimizing anyways ? Convergence ?
  - Unfortunately approximate Q-learning may not converge

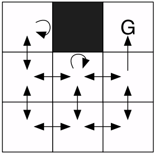
→ Problem: Non-stationarity/Floating targets ? If Q changes the function we minimize changes

→ Lots of exciting ongoing work: LSTD / NFQ / DQN etc.

Next exercise

NOTE: An exercise sheet is on the course webpage and in the git repository!

1) By hand, on a piece of paper solve the following discretized maze via Q-learning



2) Implement a simple neural Q-learning agent using a simple map in the same environment you already know

- API slightly changed so that you get a reward in each step
- hand in report showing the solution to 1) and explain what you did for 2)
- Exercise sheet and update code available **Monday 19.12.16** deadline for exercise **23.01.17**
- Bonus points for trying some more advanced ideas (e.g. target network, experience replay as in DQN)

Model Free RL - From Q-learning to DQN

Algorithm 12: Smoothed Approximate Q-learning (DQN)

---

repeat

chose $a^\star = \begin{cases} \max_a Q_W(s,a) & \text{with probability } \epsilon \\ a \in A & \text{else chose randomly} \end{cases}$

record transition (s, a, r, s')

update Q_W using sample Q-error as derivative

$\delta_t = r(s,a) + \gamma \max_{a'} Q_W(s', a') - Q_W(s,a)$

$W \leftarrow W - \alpha \left(\delta_t \frac{\partial Q_W(s,a)}{\partial W} \right)$

until *converged*

return Q_W

Model Free RL - From Q-learning to DQN

Algorithm 13: Smoothed Approximate Q-learning (DQN)

$T = \{\}$
repeat
  chose $a^\star = \begin{cases} \max\limits_a Q_W(s,a) & \text{with probability } \epsilon \\ a \in A & \text{else chose randomly} \end{cases}$
  record transition $T = T \cup \{(s,a,r,s')\}$
  update Q_W using sample Q-error as derivative
  for $s \in [1, K]$ **do**
    $(s,a,r,s') \sim T$
    $\delta = \delta + r(s,a) + \gamma \max\limits_{a'} Q_W(s',a') - Q_W(s,a)$
  $W \leftarrow W + \alpha \left(\delta \frac{\partial Q_W(s,a)}{\partial W} \right)$
until *converged*
return Q_W

Model Free RL - Q-learning

So are we there yet ?

$\rightarrow$ Further Problems with Q-learning

Model Free RL - Q-learning

So are we there yet ?

- $\rightarrow$ Further Problems with Q-learning
  - $\rightarrow$ Although Q-update looks like gradient we have "moving targets" (only partially addressed)

Model Free RL - Q-learning

So are we there yet ?

→ Further Problems with Q-learning
  - → Although Q-update looks like gradient we have “moving targets” (only partially addressed)
  - → Convergence can be slow

Model Free RL - Q-learning

So are we there yet ?

→ Further Problems with Q-learning
  → Although Q-update looks like gradient we have "moving targets" (only partially addressed)
  → Convergence can be slow
  → Reiterate: Efficient Exploration is key!

Model Free RL - Q-learning

So are we there yet ?

- → Further Problems with Q-learning
  - → Although Q-update looks like gradient we have “moving targets” (only partially addressed)
  - → Convergence can be slow
  - → Reiterate: Efficient Exploration is key!
  - → What about continuous actions ?

Model Free RL - Q-learning

So are we there yet ?

- → Further Problems with Q-learning
  - → Although Q-update looks like gradient we have "moving targets" (only partially addressed)
  - → Convergence can be slow
  - → Reiterate: Efficient Exploration is key!
  - → What about continuous actions ?
  - → When are we really going to have a fully observed state ?