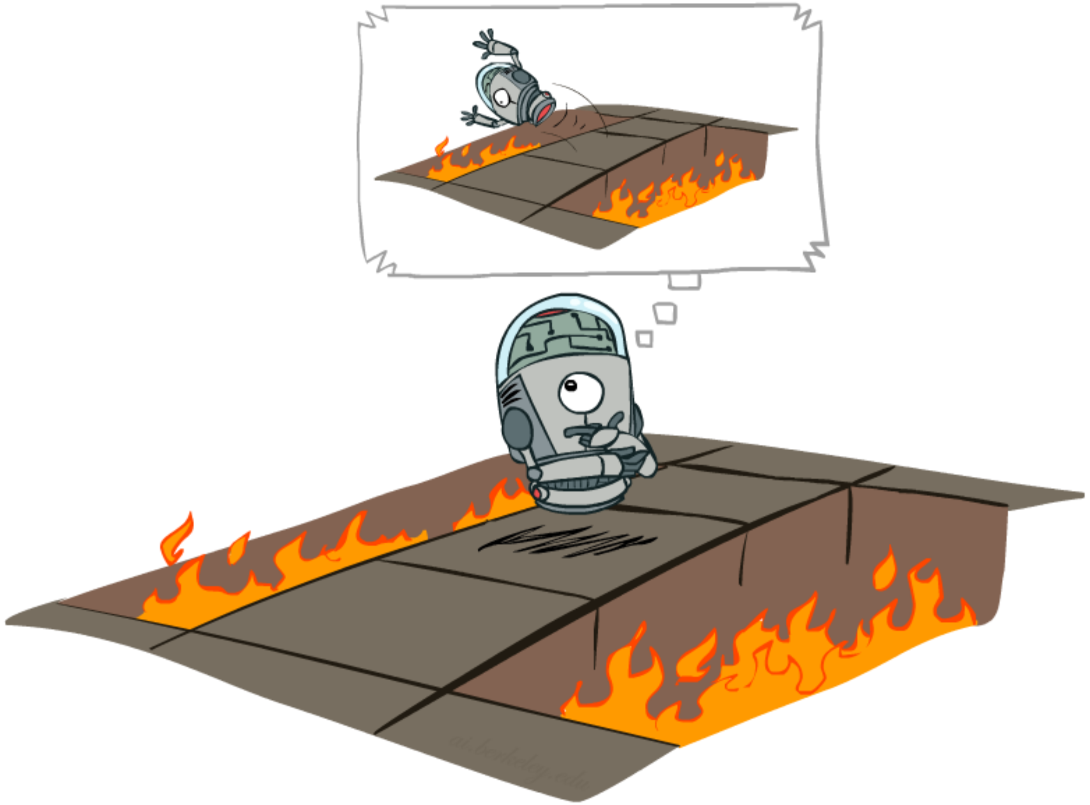
MACHINE LEARNING AND
APPLICATIONS GROUP

Reinforcement learning (solving unknown MDPs)

Predrag Vasilić

- We are still considering a Markov decision process (MDP)
- What if transitions $P_{sa}(s')$ and rewards $R(s)$ (or $R(s, a, s')$) are unknown?
- We still want to learn the optimal policy!!!
- **We need to learn online!**



Offline solution
(MDP)

Online learning
(RL)

Picture taken from [3]

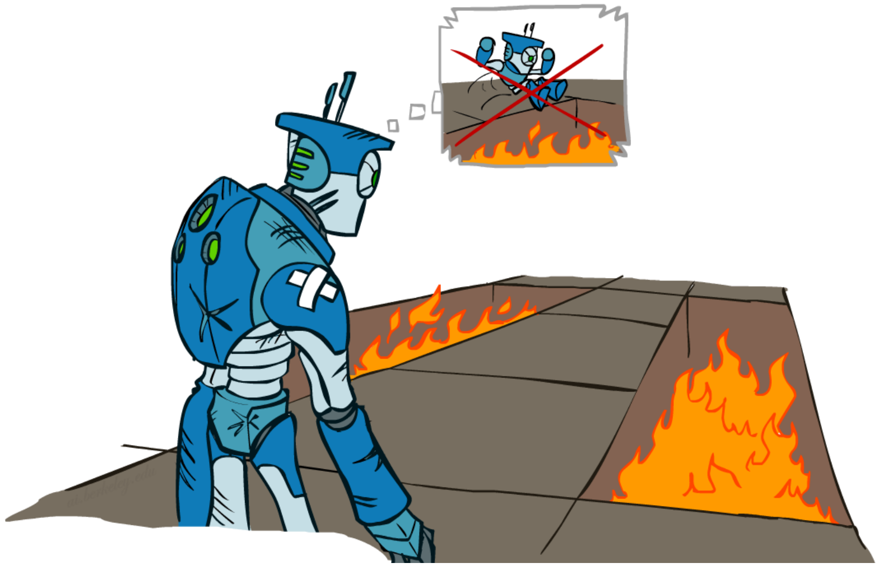
Online learning

- ***Agent learns from success and failure, from reward and punishment***
- $P_{sa}(s')$ unknown – we don't know the results of our actions
- R(s) unknown until we reach s

Picture taken from [3]

How to learn?

- We need to **explore** the problem
- With time, we start to **exploit** actions from our experience
- **Compromise**
- **Regret** is inevitable – we will certainly make errors



Picture taken from [3]

Model based learning

- We start by choosing actions randomly
- How can we estimate transitions?

$$\widehat{P_{sa}}(s') = \hat{P}(s'|s,a) = \frac{\#in\ s\ applied\ a\ and\ got\ to\ s'}{\#in\ s\ applied\ a}$$

- If we have never chosen a in s, we can use $\widehat{P_{sa}}(s') = \frac{1}{N_s}$

Policy iteration with learning

- Initialize π randomly
- Repeat {
  - (a) Execute π in the MDP for some number of trials
  - (b) Using the accumulated experience in the MDP, update our estimates for $\widehat{P_{sa}}(s')$
  - (c) Estimate V^{π}
  - (d) Update π to be the greedy policy with respect to V^{π}

  }

Model free learning

- Is it necessary to learn the transition model ($P_{sa}(s')$) ?
- Example: average age of MLA participant

Unknown P(A): "Model Based"	Unknown P(A): "Model Free"
$\hat{P}(a) = \frac{\text{num}(a)}{N}$ $E[A] \approx \sum_a \hat{P}(a) \cdot a$	$E[A] \approx \frac{1}{N} \sum_i a_i$

Example taken from [3]

Example taken from [3]

Direct utility estimation

- We have a fixed policy π !!!
- This is called Passive learning
- DUE:
  - Whenever you are in s, remember the final the total reward:
$$V^{(i)}(s) = R(s) + \gamma R\left(s_1^{(i)}\right) + \gamma^2 R\left(s_2^{(i)}\right) + \cdots$$
  - We get $\hat{V}^{\pi}(s)$ by averaging over $V^{(i)}(s)$:

$$\hat{V}^{\pi}(s) = \frac{1}{N}\sum_{i=1}^{N} V^{(i)}(s)$$

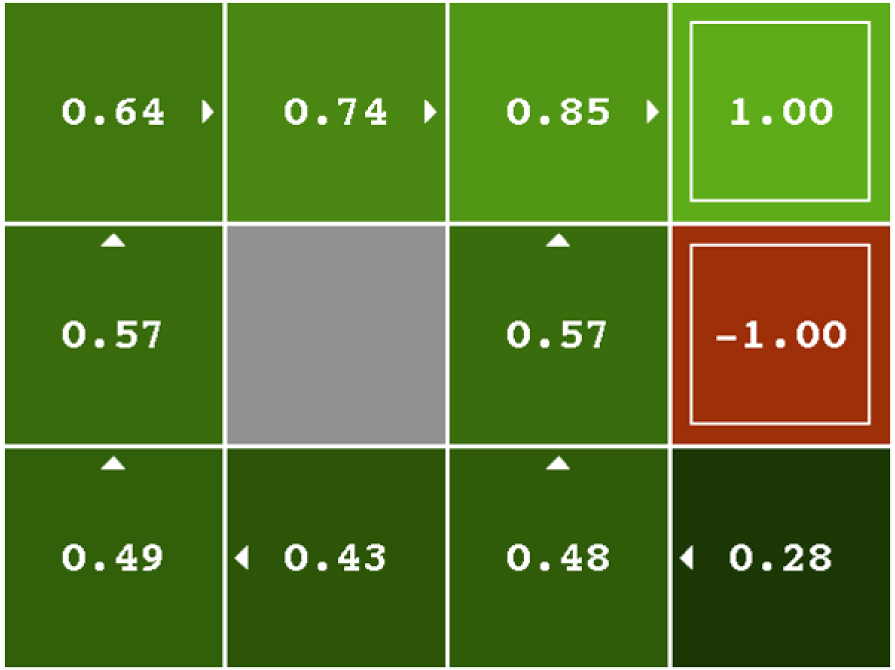
Direct utility estimation

- It misses important information: **utilities of states are not independent**
- *Some state is likely to have high utility, if it's neighbors have high utilities!*

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s')V^{\pi}(s')$$

Idea - average over samples:

$$v^{(i)}(s) = R(s) + \gamma \hat{V}^{\pi}\left(s'^{(i)}\right)$$



Picture taken from [3]

Temporal-difference learning(TD)

- We keep the current estimation of $\hat{V}^{\pi}(s)$
- Whenever we end up in s, we compute:

$$v(s) = R(s) + \gamma \hat{V}^{\pi}(s')$$

- We apply the update to $V^{\pi}(s)$:

$$\hat{V}^{\pi}(s) \leftarrow \hat{V}^{\pi}(s) + \alpha\left[v(s) - \hat{V}^{\pi}(s)\right], \qquad \alpha \in (0,1)$$

- Mean of $\hat{V}^{\pi}(s)$ converges to $V^{\pi}(s)$

Temporal-difference learning

- $\hat{V}^{\pi}(s) = (1-\alpha)\hat{V}^{\pi}(s) + \alpha v(s)$
- Exponential moving average
- The running interpolation update: $\bar{x}_n = (1-\alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$
- Makes recent samples more important:

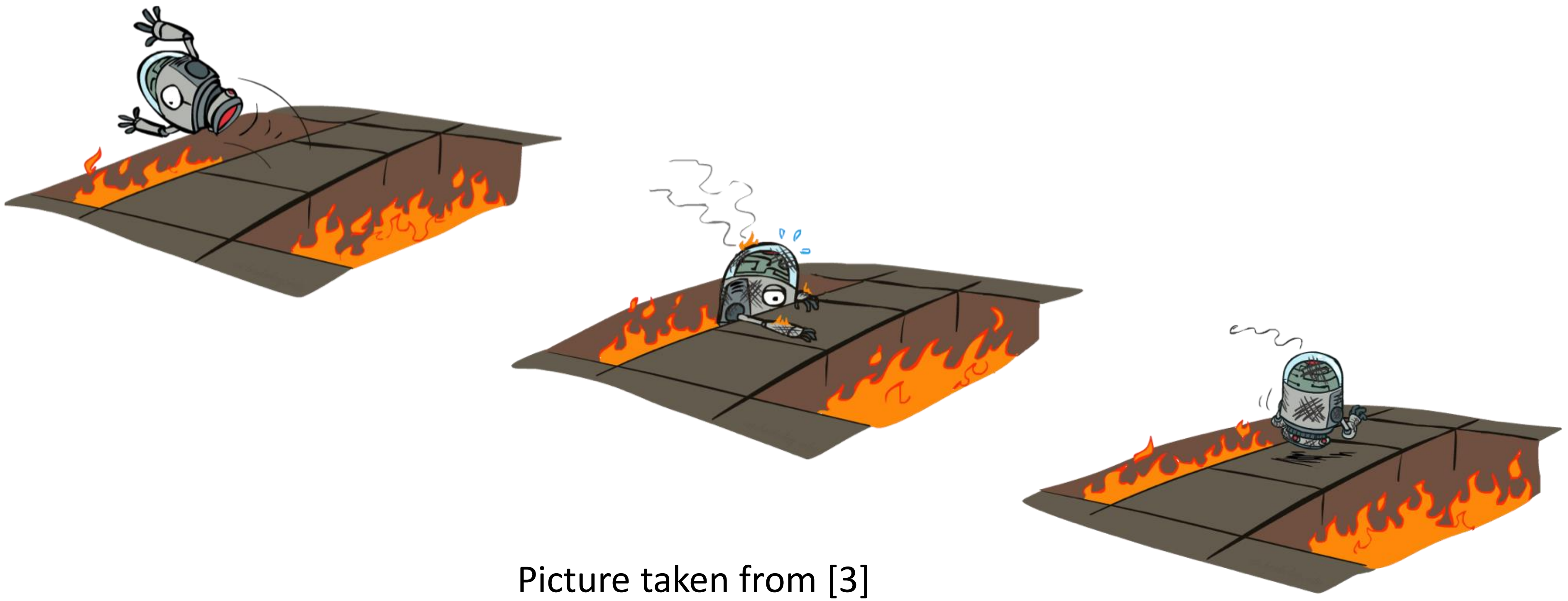
$$\bar{x}_n = \frac{x_n + (1-\alpha) \cdot x_{n-1} + (1-\alpha)^2 \cdot x_{n-2} + \ldots}{1 + (1-\alpha) + (1-\alpha)^2 + \ldots}$$

- Forgets about the past (distant past values were wrong anyway)
- Decreasing learning rate (α) can give converging averages

Slide taken from [3]

Active Reinforcement Learning

- So far, we used TD to learn utilities for a fixed policy π
- We now want to learn the optimal policy π^*



Picture taken from [3]

Q-values

- $Q(s,a)$(Q-value) : The total expected reward if we apply action a in state s, and from there we act optimally

$$Q^*(s,a) = R(s) + \gamma \sum_{s'} P_{sa}(s')V^*(s')$$

$$V^*(s) = R(s) + \gamma \max_a \sum_{s'} P_{sa}(s')V^*(s')$$

- $V^*(s) = \max_a Q^*(s,a)$
- $\pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$
- $Q^*(s,a) = R(s) + \gamma \sum_{s'} P_{sa}(s') \max_{a'} Q^*(s',a')$

Q–value iteration

$$V_t(s) = R(s) + \gamma \max_a \sum_{s'} P_{sa}(s')\, V_{t-1}(s')$$

- Analog with Value iteration
- MDP model is known
- Algorithm:
  - Start with $Q_0(s, a) = 0$ for each s, a
  - For $t = 1, 2, \dots$ until convergence

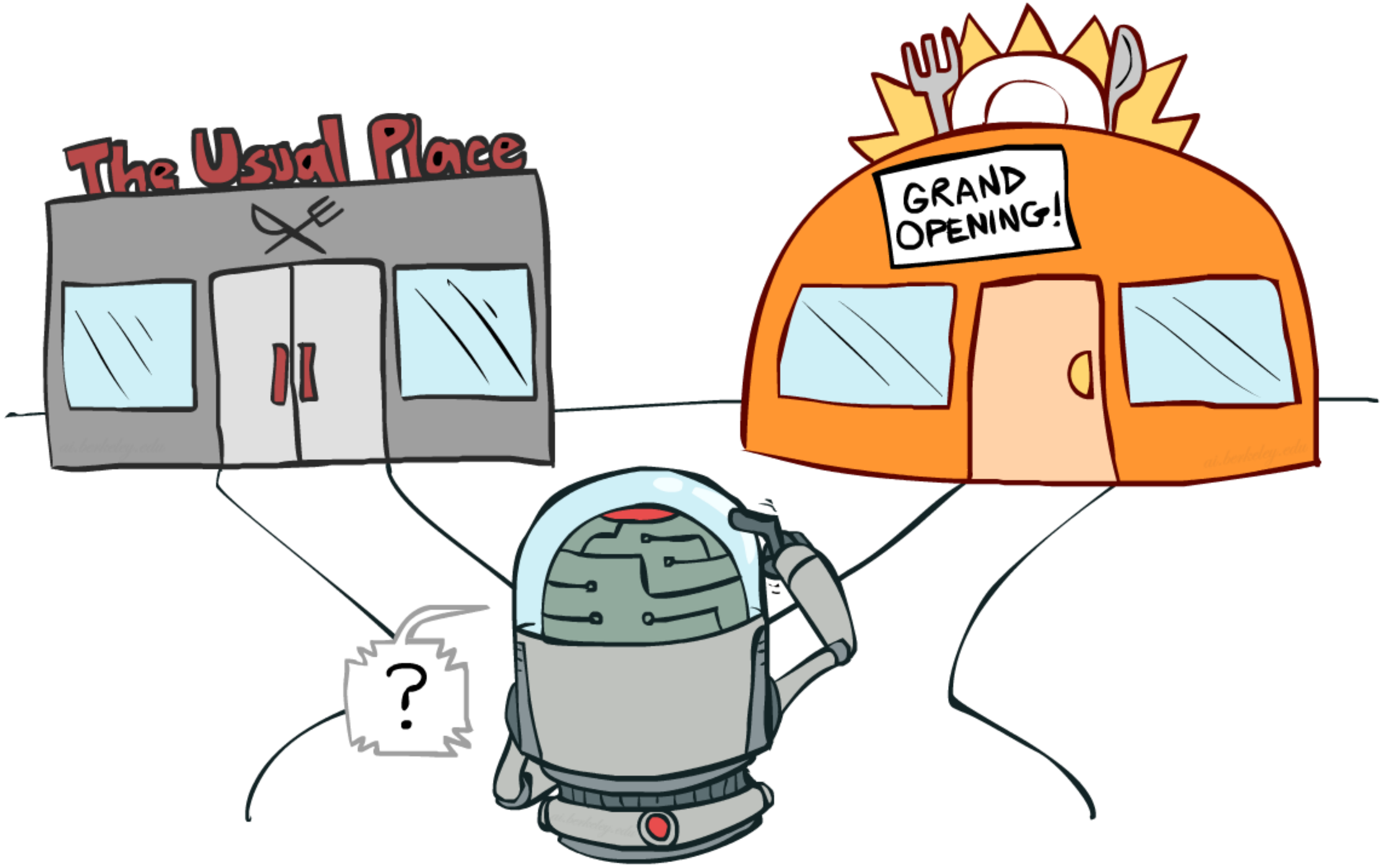
$$Q_t(s,a) = R(s) + \gamma \sum_{s'} P_{sa}(s') \max_{a'} Q_{t-1}(s', a'), \qquad \forall\, s, a$$

- Q_t converges to Q^*
- We can compute π^* and V^*

Q-learning

- For unknown models - TD with Q-values
- Q-learning algorithm:
  - Receive a new sample (s, a, s')
  - Consider your old estimate: $Q(s,a)$
  - Compute the new sample estimate:
$$q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$
  - Update the Q-value with running average:
$$Q(s,a) = Q(s,a) + \alpha\,[q(s,a) - Q(s,a)]$$
- Q-learning converges to Q^*, if we explore enough !!!

Exploration vs. Exploitation



Picture taken from [3]

How to explore?

- How to choose actions in states?
- ε-greedy exploration:
  - With probability ε, we choose actions randomly
  - Otherwise, we use our current policy estimation ($\underset{a}{\operatorname{argmax}}\, Q(s, a)$)
- Start with high value of ε and decrease it with time

How to explore?

- Exploration function
- $N(s', a')$ is the number of times we have selected action a' in state s', and we define:

$$f(s', a') = Q(s', a') + \frac{k}{N(s', a')}$$

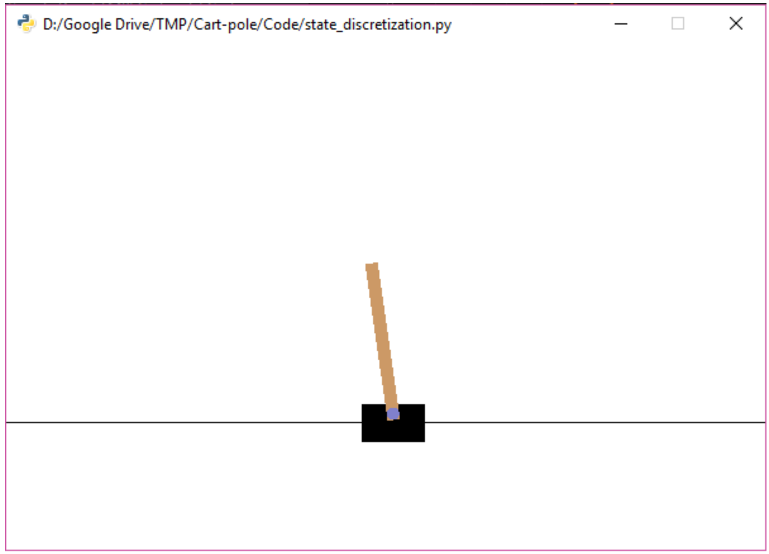
- Sample for Q-learning is now:

$$q(s, a) = R(s) + \gamma \max_{a'} f(s', a')$$

Discretization

- Methods defined so far supposed discrete states
- If states are continuous-we can discretize them
- Problem – curse of dimensionality
- Works for states with small dimensionality(not grater than 5)

Example: Cart-Pole (Inverted pendulum)



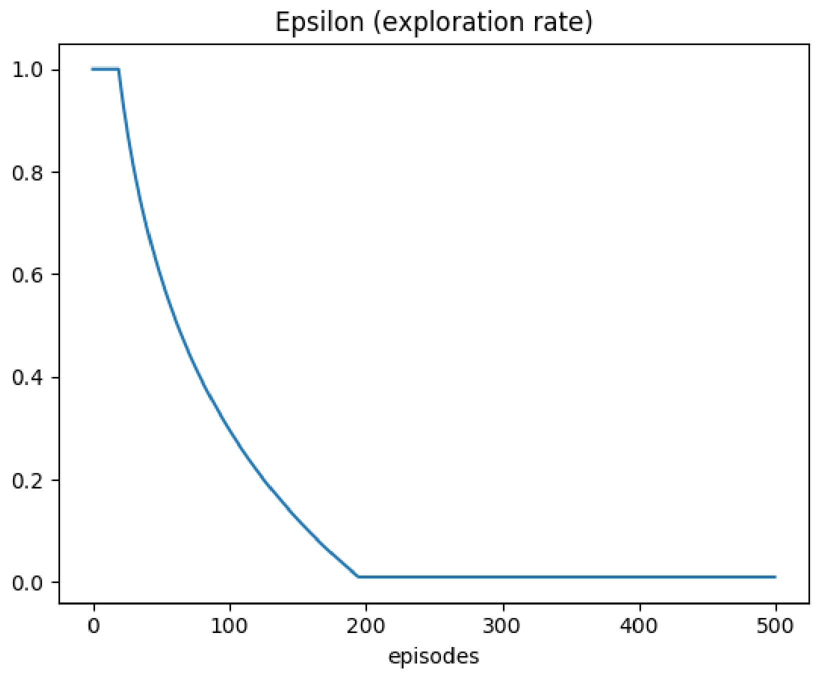
- Library: https://github.com/openai/gym (OpenAI Gym)
- Simulator: “CartPole-v1”
- State variables:
  - $x \in [-2.4, 2.4]$ – position
  - $\dot{x} \in [-\infty, \infty]$ – velocity
  - $\theta \in [-12°, 12°]$ –angular distance from the vertical position
  - $\dot{\theta} \in [-\infty, \infty]$ – angular velocity
- If any variable goes out of range, the episode ends
- If the variables are in their range for 500 steps (T=0.02s), the episode ends
- Two available actions on each step: apply force to the left or right
- Each step has a living reward of +1, so do the terminal states

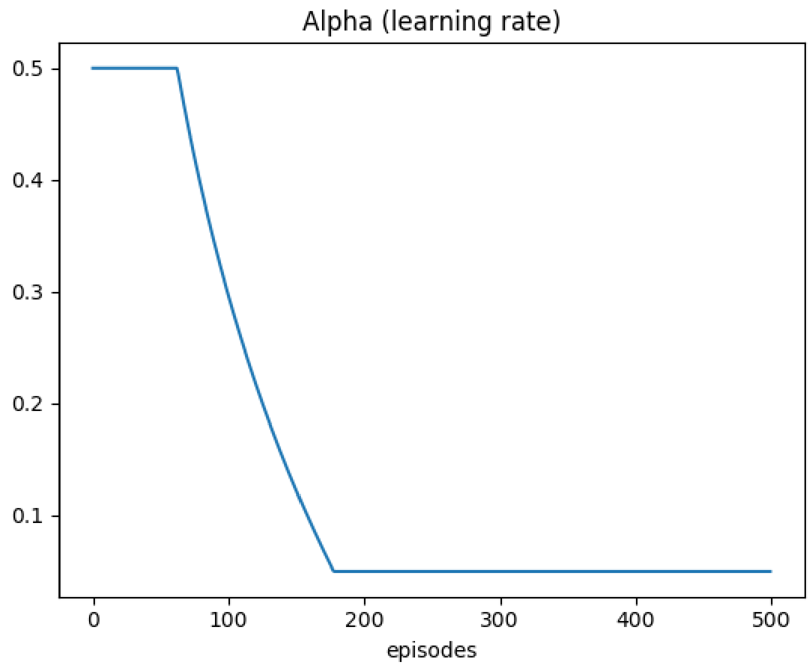
Example: Cart-Pole (Inverted pendulum)

- Can we apply temporal difference Q-learning?
- We could discretize the state space:
  - x: $[-\infty, 0.8),\ [-0.8, 0.8), [0.8, \infty)$
  - $\dot{x}$: $[-\infty, 0.5),\ [-0.5, 0.5), [0.5, \infty)$
  - θ: $[-\infty, -8)°,\ [-8, -4)°, [-4, 0)°, [0, 4)°, [4, 8)°, [8,\ \infty)°$
  - $\dot{\theta}$: $[-\infty, -30)°/s,\ [-30, -15)°, [-15, 0)°, [0, 15)°, [15, 30)°, [30,\ \infty)°$
- Problem – curse of dimensionality
- Usually works for states with small dimensionality(some authors say to use it with ≤ 5 dimensions)

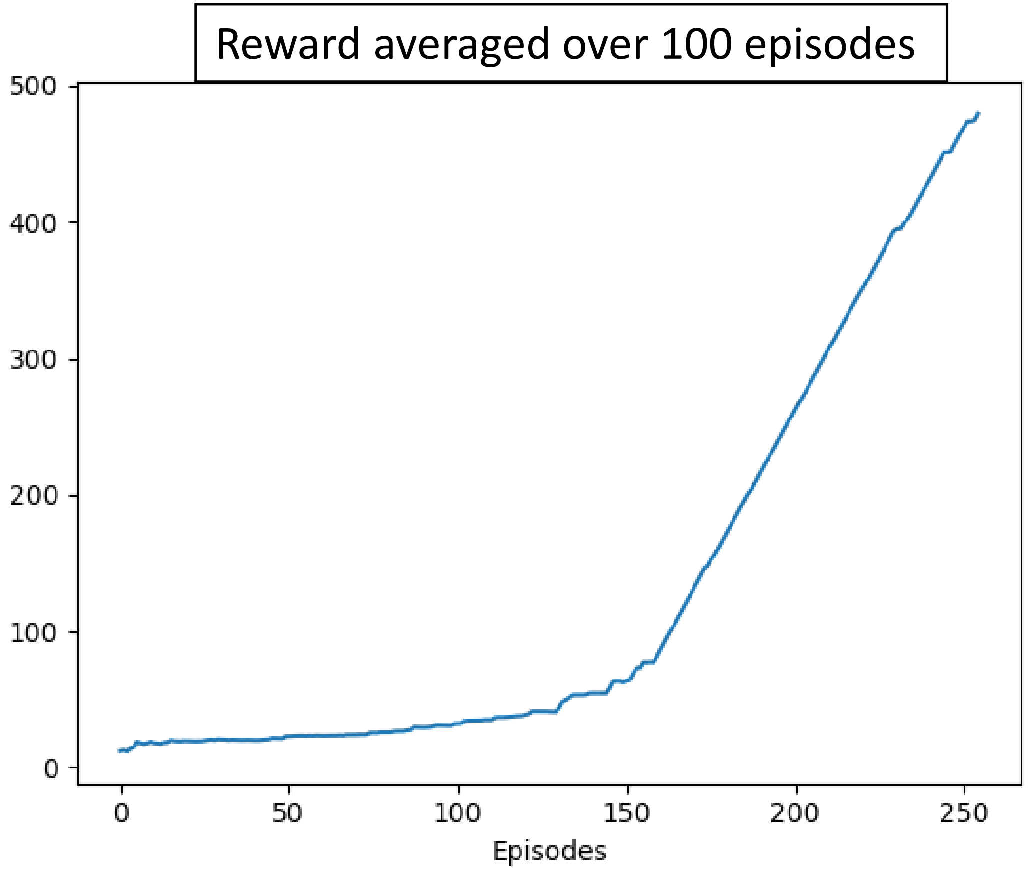
Example: Cart-Pole (Inverted pendulum)

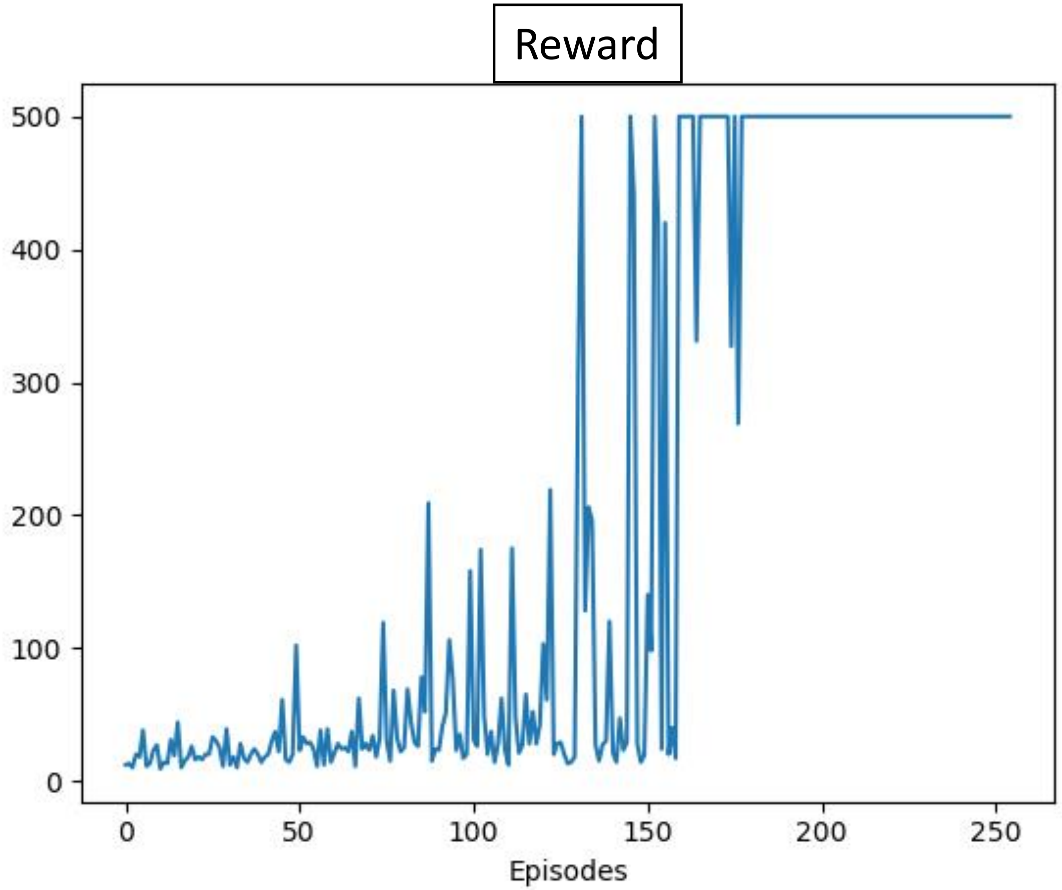
- Adaptive exploration rate ε
- Adaptive learning rate α



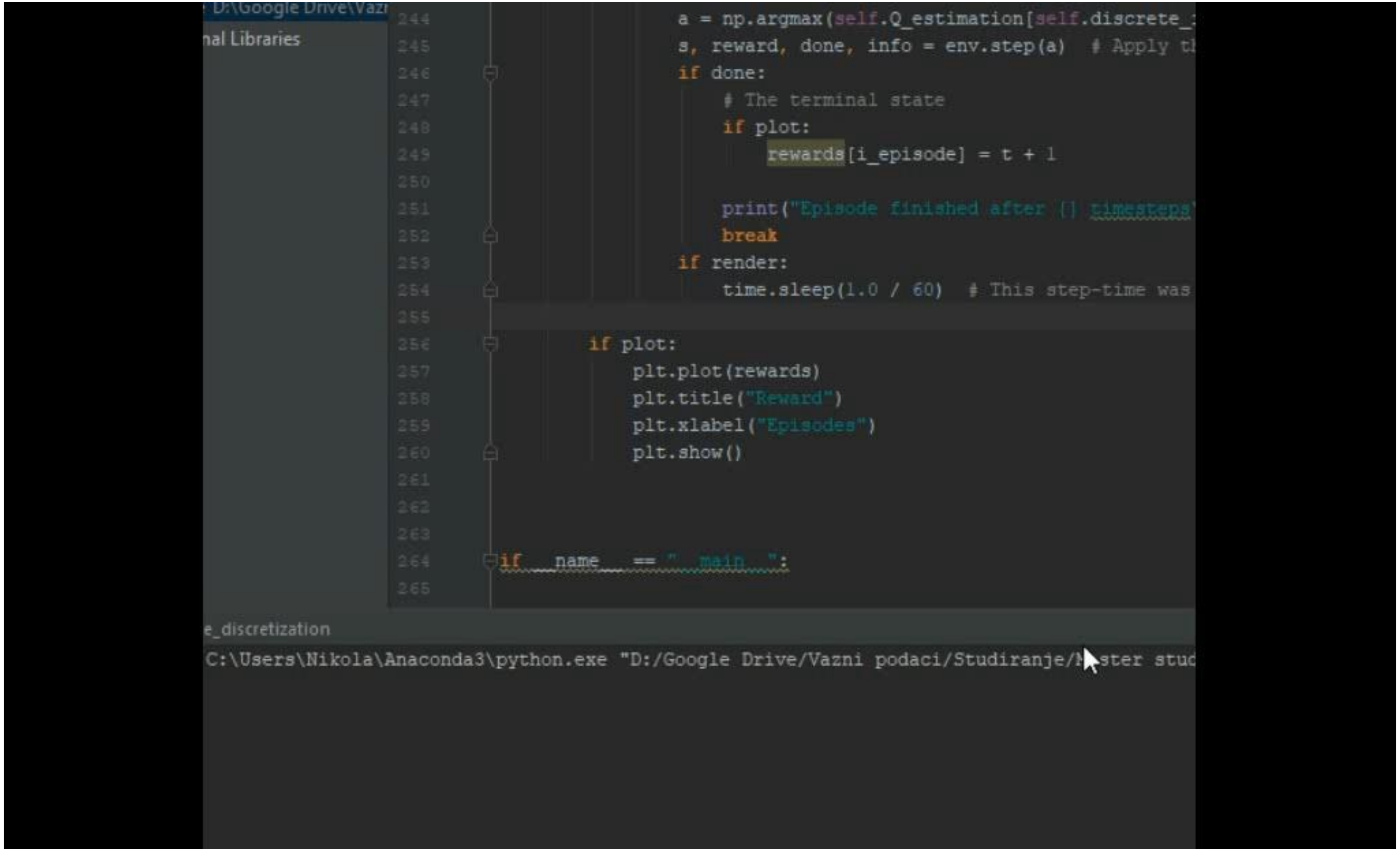


Example: Cart-Pole (Inverted pendulum)



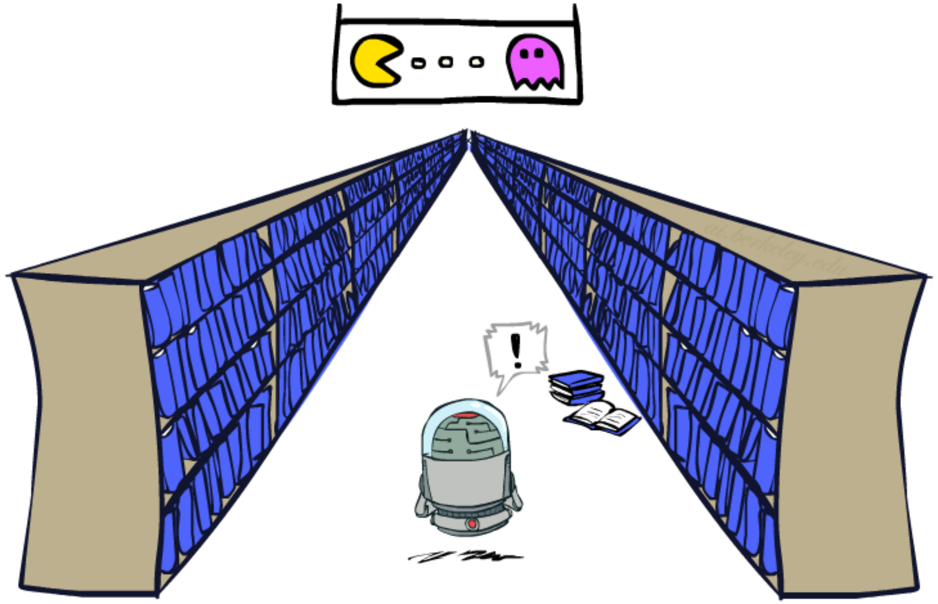


Example: Cart-Pole (Inverted pendulum)



Approximate Q-learning

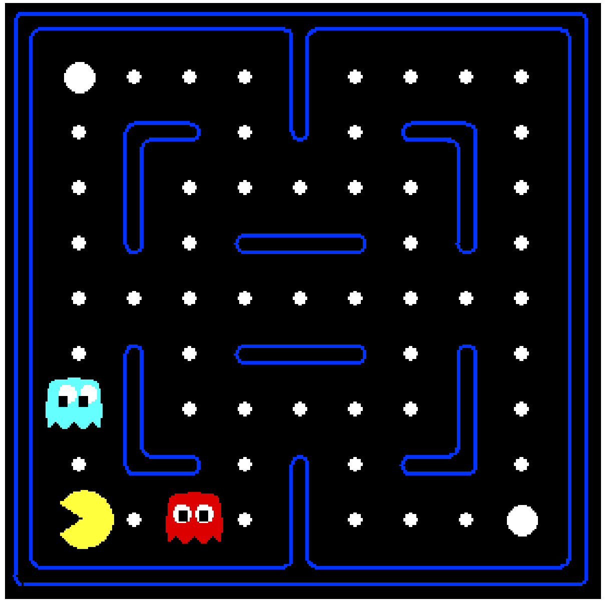
- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
- Instead, we want to generalize



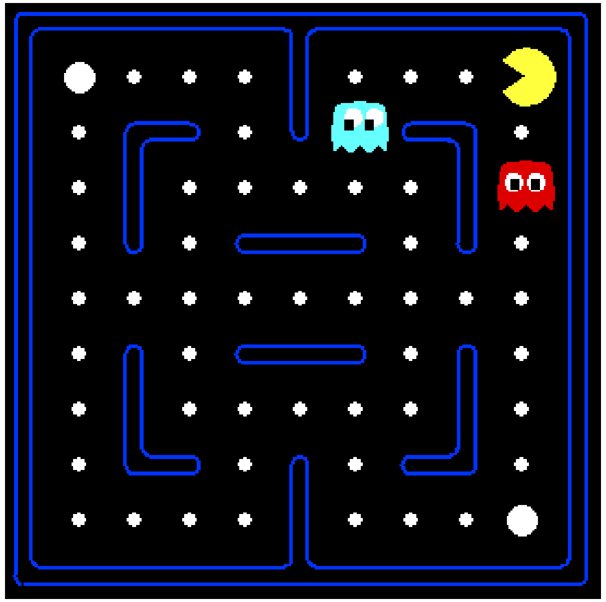
Parts taken from [3]

Example - Pacman

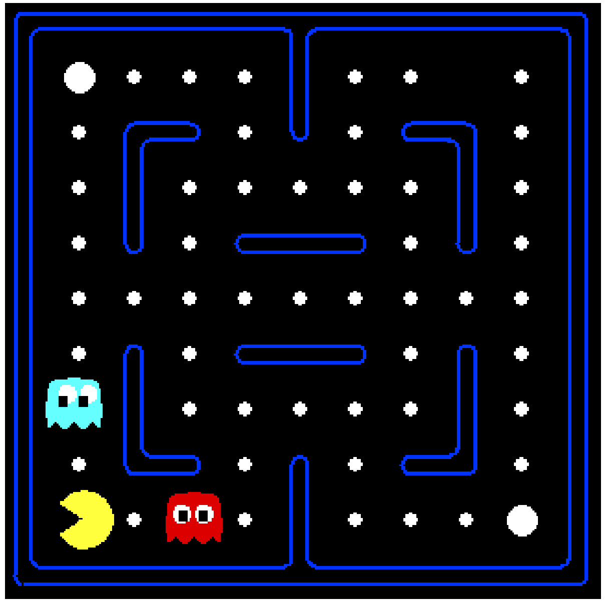
Let's say we discover through experience that this state is bad:



In naive q-learning, we know nothing about this state:



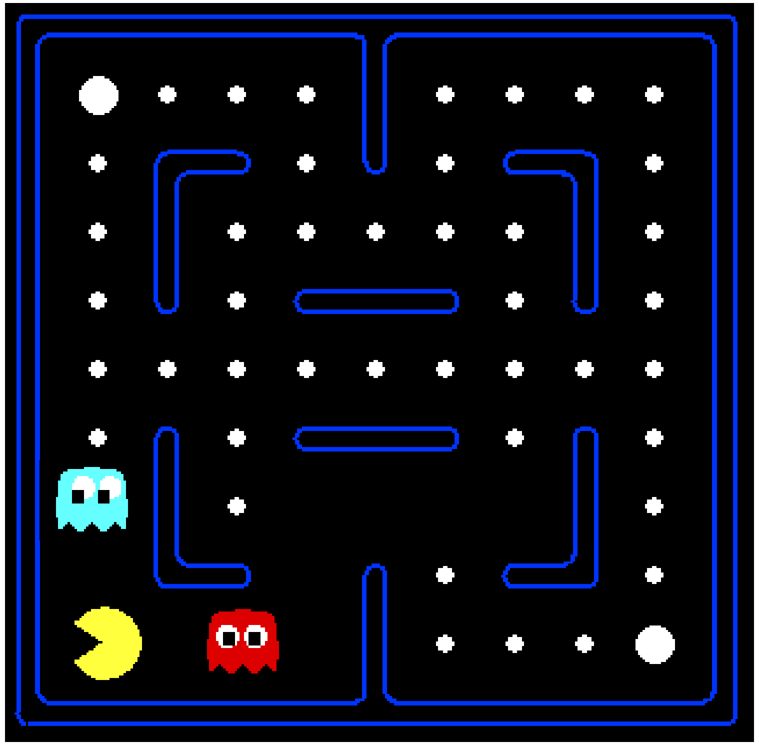
Or even this one!



Slide taken from [3]

Feature-Based Representations

- Describe a state using a vector of features (properties)
- Features are functions from states to real numbers (often 0/1) that capture important properties of the state
- Example features:
  - 1/Distance to closest ghost
  - 1/Distance to closest dot
  - Number of ghosts
  - 1 / (dist to dot)^2
  - Is Pacman in a tunnel? (0/1)
  - …… etc.
  - Is it the exact state on this slide?



Slide taken from [3]

Aproximate Q-learning

- We approximate Q-value with a linear function of features:
$$Q(s,a) = \theta_1\phi_1(s,a) + \theta_2\phi_2(s,a) + \cdots + \theta_n\,\phi_n(s,a) = \boldsymbol{\theta}^T\boldsymbol{\phi}(s,a)$$
- Aproximate Q-learning algorithm:
  - Initialize $\boldsymbol{\theta}$=**0**

  Repeat{
    - Receive a new sample (s, a, s')
    - Compute the new sample estimate:
    $$q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$
    - Update the weights:
    $$\theta_j = \theta_j + \alpha[q(s,a) - Q(s,a)]\phi_j(s,a), \qquad j = 1,2,\ldots,n$$

  }
  - $\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a) = \underset{a}{\operatorname{argmax}}\, \boldsymbol{\theta}^T\boldsymbol{\phi}(s,a)$

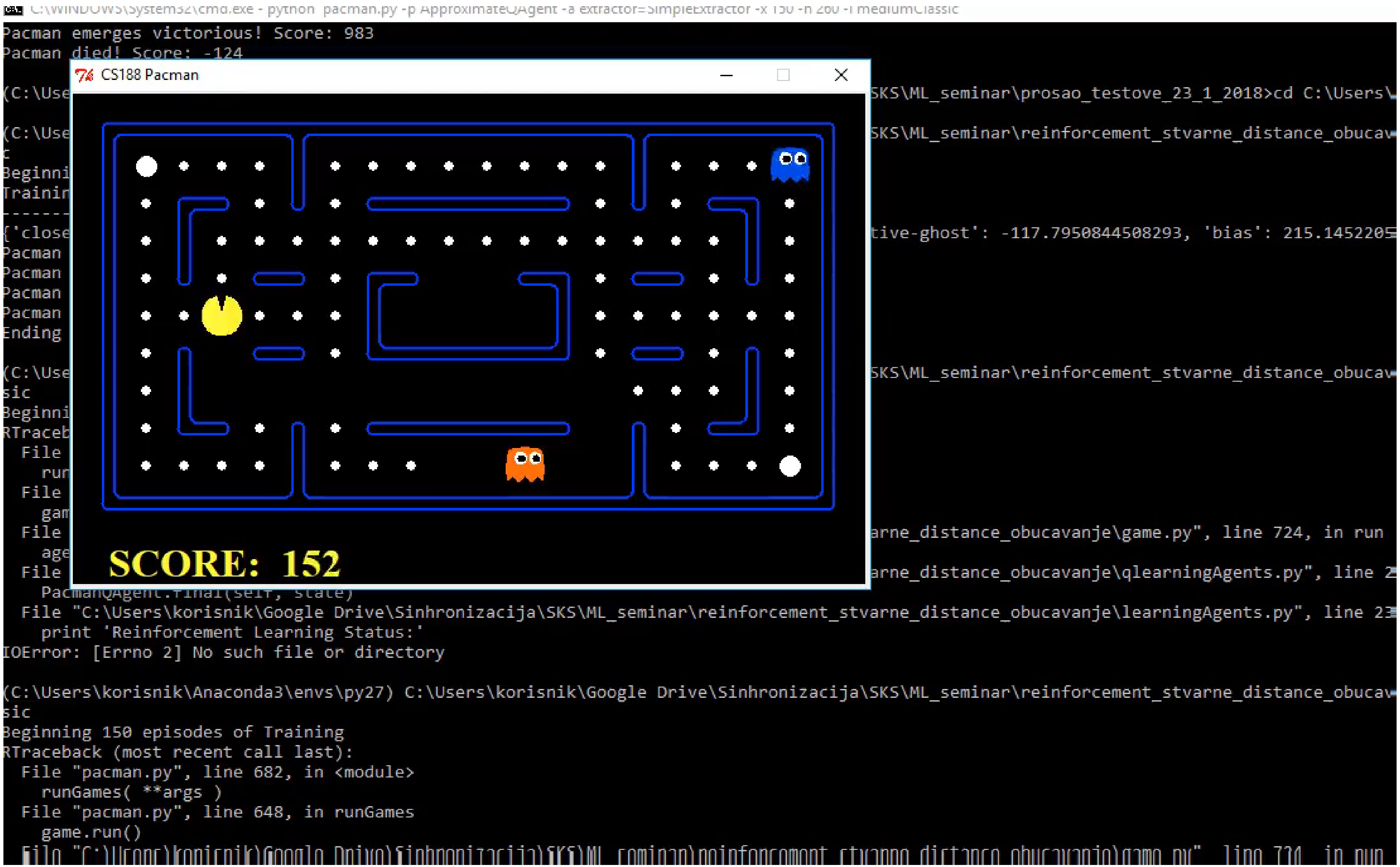
Example - Pacman

- Fetures for example:
  - $1/'closest - food'$
  - $'\# - of - ghosts - 1 - step - away'$
  - $'eats - food'$
  - $'bias'$
  - $1/\,'closest - scared - ghost'$
  - $1/'closest - active - ghost'$

Example-Pacman – first four features training

```
(C:\Users\korisnik\Anaconda3\envs\py27) C:\Users\korisnik\Documents>cd C:\Users\korisnik\Google Drive\Sinhronizacija\SKS\ML_seminar\prosao_testove_23_1_2018

(C:\Users\korisnik\Anaconda3\envs\py27) C:\Users\korisnik\Google Drive\Sinhronizacija\SKS\ML_seminar\prosao_testove_23_1_2018>python pacman.py -p ApproximateQAgent -a ext
Beginning 5 episodes of Training
Pacman died! Score: -172
Pacman died! Score: -387

(C:\Users\korisnik\Anaconda3\envs\py27) C:\Users\korisnik\Google Drive\Sinhronizacija\SKS\ML_seminar\prosao_testove_23_1_2018>python pacman.py -p ApproximateQAgent -a ext
Beginning 5 episodes of Training
Pacman died! Score: -156
Pacman died! Score: -189
Pacman emerges victorious! Score: 975
Pacman emerges victorious! Score: 970
Pacman emerges victorious! Score: 983
Training Done (turning off epsilon and alpha)
---------------------------------------------
{'closest-food': -0.08456937804251642, 'bias': 47.78197565415671, '#-of-ghosts-1-step-away': -19.839258573293158, 'eats-food': 68.06789429816786}

(C:\Users\korisnik\Anaconda3\envs\py27) C:\Users\korisnik\Google Drive\Sinhronizacija\SKS\ML_seminar\prosao_testove_23_1_2018>python pacman.py -p ApproximateQAgent -a ext
Beginning 5 episodes of Training
Pacman emerges victorious! Score: 981

(C:\Users\korisnik\Anaconda3\envs\py27) C:\Users\korisnik\Google Drive\Sinhronizacija\SKS\ML_seminar\prosao_testove_23_1_2018>python pacman.py -p ApproximateQAgent -a ext
Beginning 5 episodes of Training

(C:\Users\korisnik\Anaconda3\envs\py27) C:\Users\korisnik\Google Drive\Sinhronizacija\SKS\ML_seminar\prosao_testove_23_1_2018>python pacman.py -p ApproximateQAgent -a ext
Beginning 5 episodes of Training
Pacman died! Score: -303
Pacman emerges victorious! Score: 982
Pacman emerges victorious! Score: 946
Pacman died! Score: -256
Pacman emerges victorious! Score: 981
Training Done (turning off epsilon and alpha)
---------------------------------------------
{'closest-food': -0.46887783994894944, 'bias': 42.48683700913223, '#-of-ghosts-1-step-away': -20.057244137859165, 'eats-food': 63.73668004160507}

(C:\Users\korisnik\Anaconda3\envs\py27) C:\Users\korisnik\Google Drive\Sinhronizacija\SKS\ML_seminar\prosao_testove_23_1_2018>python pacman.py -p ApproximateQAgent -a ext
```

Example-Pacman – all six features after training



Questions?

References

- [1] Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach 3rd edition, Prentice Hall, 2009.
- [2] Andrew Ng, John Duchi, "Machine Learning - Lecture notes“
- [3] UC Berkeley: CS188 Intro to AI, lecture slides http://ai.berkeley.edu (last visited: 11.03.2018)
- [4]Faculty of Electrical Engineering, University of Belgrade: Statistička klasifikacija signala, materials from class, http://automatika.etf.bg.ac.rs/sr/13m051sks (last visited: 11.03.2018)

THANK YOU