Reinforcement Learning

Lukáš Hruška, MSc.
Ján Magyar, MSc.

Slovak Artificial Intelligence Meetup

January 22nd, 2020

Why do we need machine learning?

- problems are often far too complex to express formally
    - try to describe a chair
- supervised vs unsupervised learning

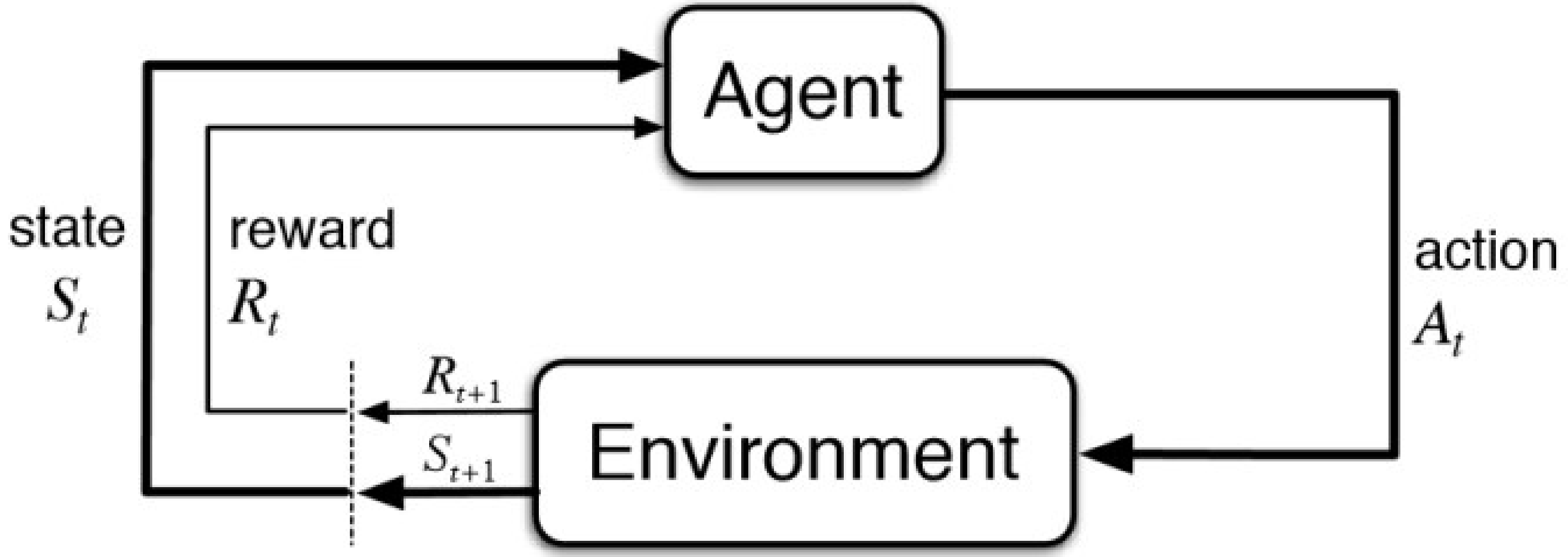
Definition of basic elements

- agent
- environment
- state
- action
- reward function
- value
- policy

Dynamic programming and Markov decision processes

- Markov chain - stochastic model, describes a sequence of possible events, where the next event depends only on the state resulting from the current one
- decision making approach for situations, where the outcome is not fully in the control of the decision maker.
- goal: obtain optimal policy → describes the best action for each state in MDP

Reinforcement learning



Model-free vs model-based

- model-free
  - learn value function and policy from the interaction (the agent knows how to act, but does not understand why)
  - appropriate when collection of data is not a problem
  - random sampling methods are included here (Monte Carlo)
  - decisions are only based on expected reward
- model-based
  - first approximates the environment’s behavior (model) from state transitions and outcomes
  - searches model to find appropriate actions
  - appropriate when collection of large quantities data is problematic / expensive
  - understands, what would (is likely to) happen if a given action is taken

Value-based vs policy-based methods

value-based

- learn a value function that maps each state action pair to a value
  - the action with the largest value will be taken for the next step
  - works well in finite action spaces, problematic in continuous ones

policy-based

- directly optimize a policy without using a value function
- works well with stochastic action-space

Q-learning

- model-free, value-based
- finds an optimal policy π^* for any finite Markov decision process
- exploration vs exploitation

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

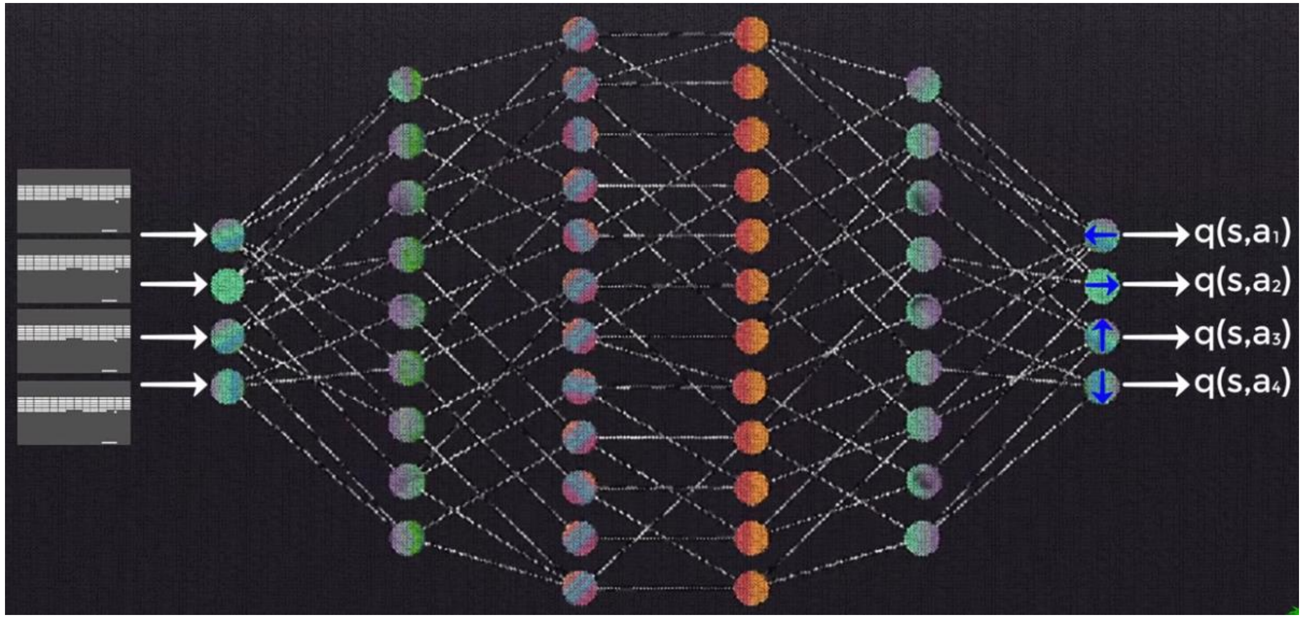
Variatons of Q-learning

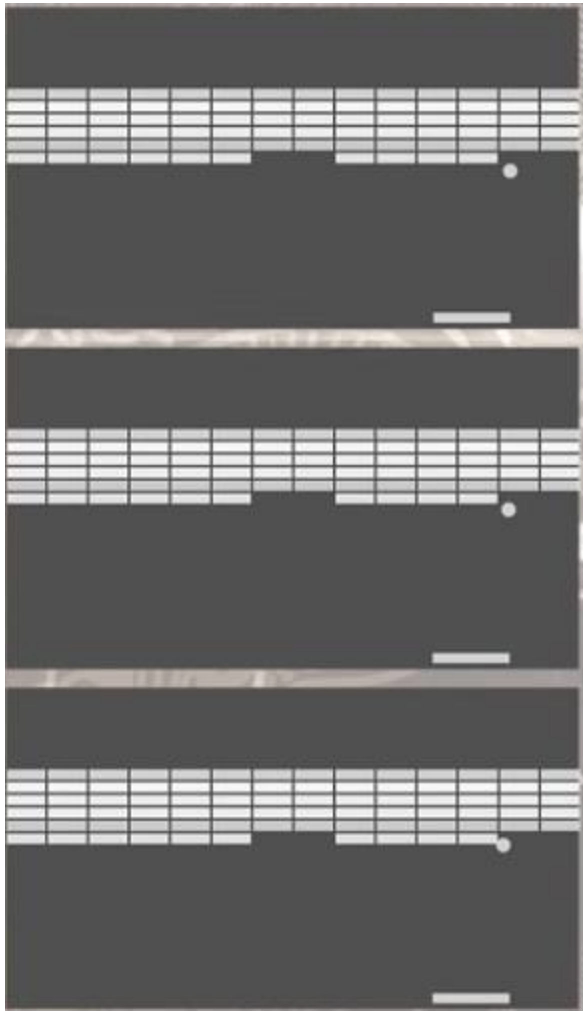
- Double Q-learning – off-policy, attempts to compensate for Q-learning's weakness, where it tends to overestimate action values in some stochastic environments caused by using the maximum action value as an approximation of the expected one
- SARSA – on-policy, the update function does not consider the optimal action value, but the one that the algorithm would actually take

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Deep Q-network

- attempts to estimate Q-values using a neural network
- impractical with large or continuous action spaces
- limited to finite number of actions



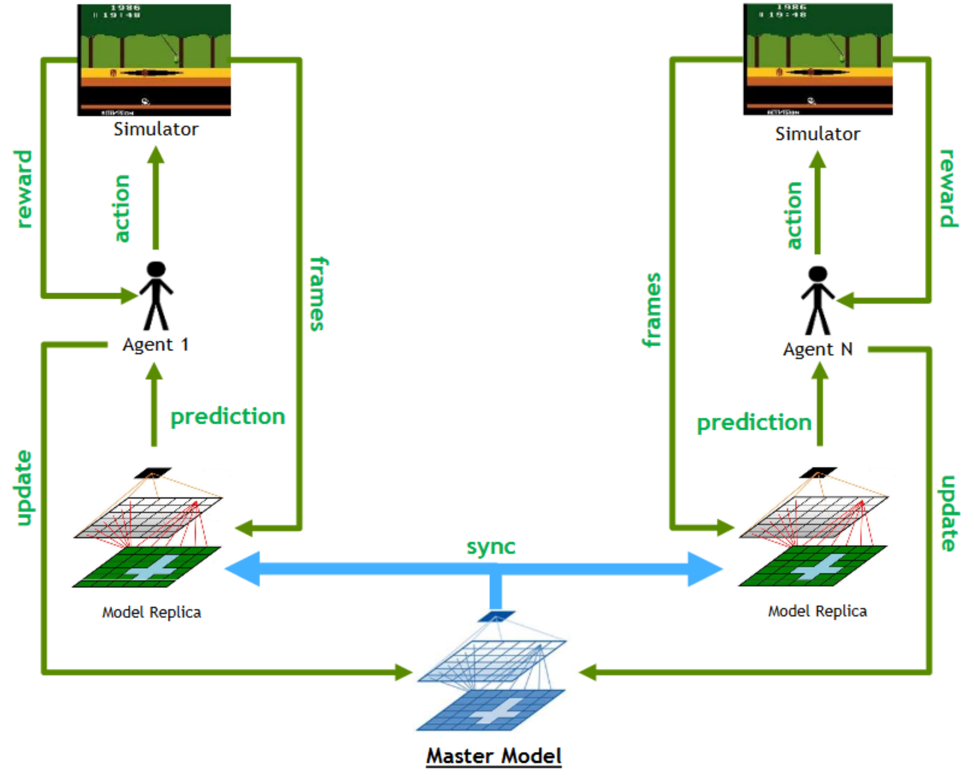


Actor-critic RL

- what would happen if an actor using the previous algorithms was put in a situation with unfavorable conditions?
- what would happen, if the state and action spaces were infinite?
  - infinite Q-table
- actor
  - takes the current environment state and determines the best action to take from there
  - policy-based
- critic
  - takes the current environment state and the action and returns the score representing, how good the action is for the given state
  - value-based

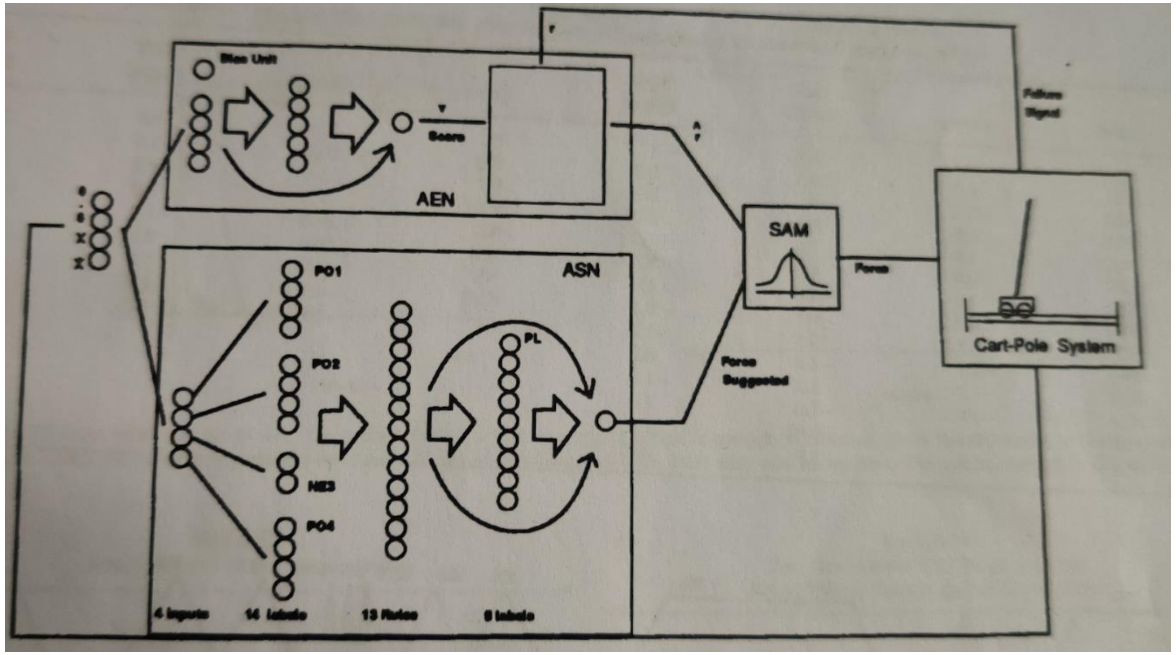
Asynchronous Advantage Actor-Critic Algorithm (A3C)

- policy gradients are an online method
- only data obtained using the current policy can be used



GARIC

- actor-critic based
- action-selection network



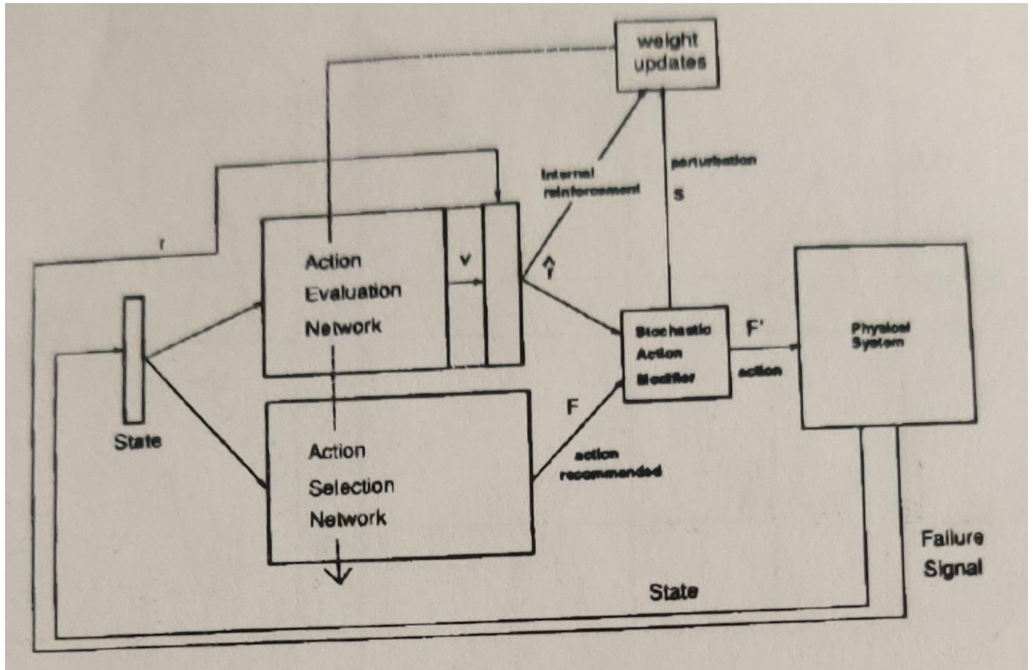


Fig. 3. The architecture of GARIC.

Inverse reinforcement learning

- how could we teach AI to drive a car?
- no reward function is given
- it is inferred from the observed behavior of an expert
  - premise: the observed behavior is (close to) optimal

Problems inherent to reinforcement learning

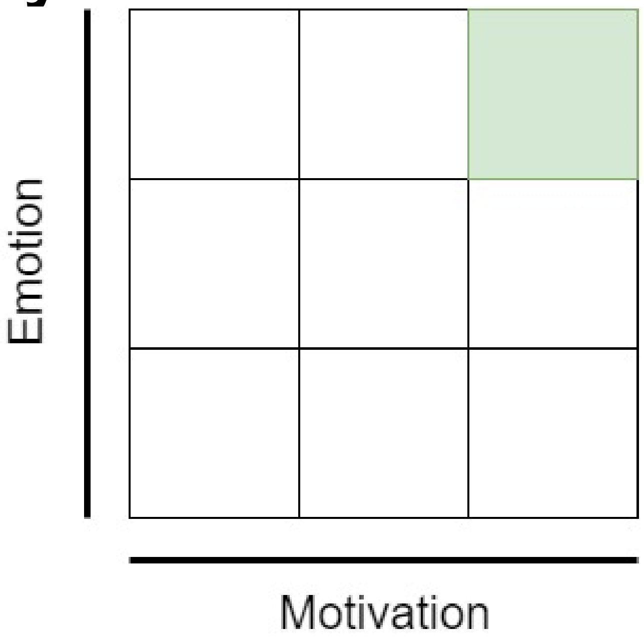
- faulty reward functions
- wireheading
- the agent adapting the environment
- balancing exploration and exploitation

Faulty reward functions

- the agent adapts its behavior based on the observed rewards
- what if the reward function does not reflect the desired goal?


START
START
SCORE
0
LAPS
-/3
TIME
0:01
TURBO
MORE GAMES

Faulty reward functions – a case study



- the goal is to get the human into a state of positive happiness and high motivation
- reward function
  1. if the state does not change between two timesteps, give a reward of 0
  2. if the state changes, the reward is calculated as:

     new_motivation – old_motivation + new_emotion – old_emotion
  3. if the new state is positive happiness and high motivation, give a reward of 5

Solving faulty reward functions

- give sparse rewards
- define a single (or a handful) desired goal state that has a reward of 1 attached to it
- give a reward of 0 in any other case
- how does it affect training time and learning?
- introduce auxiliary tasks
- curriculum learning

Wireheading

- first observed in rats, later in humans
- not present with reward signals coming from outside the environment
- during wireheading a conscious agent influences the reward signal
- especially dangerous when the reward comes from humans

Wireheading in practice

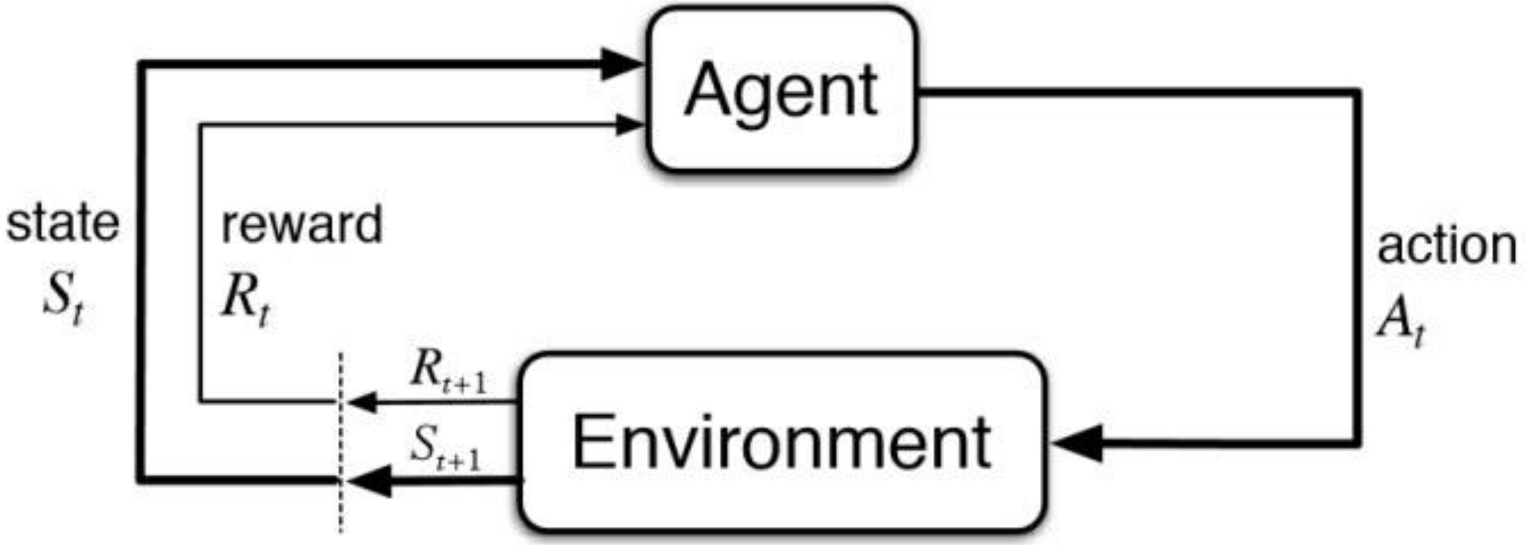
- adversarial training – AlphaGo
- the agent could potentially change its opponent’s behavior
- the agent could potentially take control of the reward signal
- the agent could potentially take control of the environment

Solving the wireheading problem

- distinguishing between rewards and reward signals
- reward signals should only describe the reward, not constitute it
- even if the agent takes control over the reward signal, it only loses information – it doesn’t pay off to overwrite the reward function

The agent adapting the environment

- a special case of wireheading
- the agent might be able to change the environment's behavior



- case study – recommendation systems

Balancing exploration and exploitation

- in the initial phase of learning, the agent should explore and look for more optimal policies
- later on during learning, it should rely on previously acquired knowledge about the environment
- critical especially in action-sensitive use cases (e.g. human—agent interaction)
- how to balance the exploration/exploitation rate?

Methods of exploration

- undirected exploration
  - random exploration
  - utility-driven probability distributions
- directed exploration
  - counter-based exploration
  - counter-based exploration with decay
  - error-based exploration
  - recency-based exploration

Random exploration

1. define **ε** that describes the exploration rate
2. generate random number **`num`** between 0 and 1
3. `if` **`num < ε`**:

    select random action with equal probability

   `else`:

    select action with the highest Q-value

Utility-driven probability distributions

1. define **ε** that describes the exploration rate
2. generate random number **`num`** between 0 and 1
3. `if` **`num < ε`**:

   select random action with probability based on utility

   `else`:

   select action with the highest Q-value

Counter-based exploration

- prefer states that have already been explored
- select actions deterministically – the action with the highest expected yield

$$eval_c(a) = \alpha \cdot f(a) + \frac{c(s)}{E[c|s,a]}$$

- α – constant (≥ 0) balancing exploitation and exploration
- $f(a)$ – Q-value (expected value of carrying out action a)
- $c(s)$ – counter of current state
- $E[c|s,a]$ – expected counter value of state after carrying out action a

Counter-based exploration with decay

- simple counter-based methods do not contain information regarding to *when* a state was last visited
- prefer states that were visited earlier during the learning
- at each time tick update the counter for each state

$$c(s) \leftarrow \lambda \cdot c(s)$$

- λ – constant, ≈ 1

Error-based exploration

- estimate the change of Q-value
- remember the last change of each state-action pair
- the higher the change, the more likely that neighboring states will be updated

$$eval_c(a) = \alpha \cdot f(a) + \frac{c(s)}{E[c|s,a]} + \beta \cdot E\left[\Delta\hat{V}_{last}\middle|s,a\right]$$

- β – constant (>*0*) determining the error-heuristic

Recency-based exploration

- prefer adjacent states that recurred less recently
- for each state remember a recency value $\rho(s)$ that describes the number of actions carried out since the last occurrence of the state

$$eval_c(a) = \alpha \cdot f(a) + \sqrt{E[\rho|s,a]}$$

- in the beginning, this sort of action selection leads to a quicker exploration of the entire state space

Adaptive methods comparison

exploration-technique	deterministic version average steps	stochastic version average steps
Uniform distribution Boltzmann distribution Semi-uniform distribution	43 000	43 000
Counter-based exploration Counter/error-based exploration Counter-based exploration with selective attention	4 700	4 800
Counter-based exploration with decay ($\lambda = 0.99999$)	5 800	7 300
Recency-based exploration	7 400	8 900

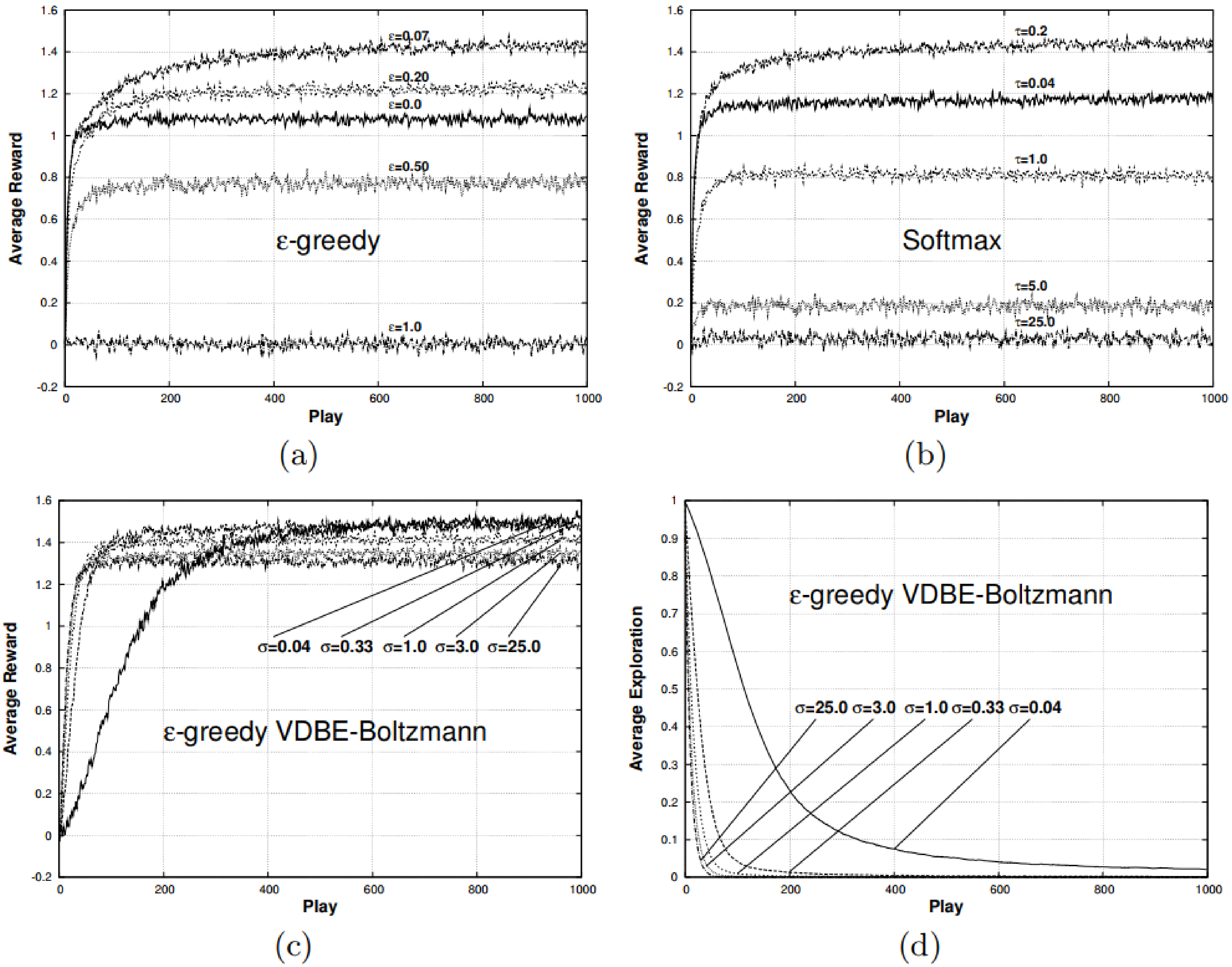
Value-difference based exploration

- uses softmax Boltzmann distribution
- exploration probability defined and updated for each state
- the agent should explore more states where the knowledge is uncertain

$$f(s,a,\sigma) = \frac{1 - e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}}{1 + e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}}$$

$$\varepsilon_{t+1}(s) = \delta \cdot f(s_t, a_t, \sigma) + (1 - \delta) \cdot \varepsilon_t$$

- σ – inverse sensitivity
- δ – influence of the action on exploration rate; $\delta = \frac{1}{|A(s)|}$


ε=0.07
ε=0.20
ε=0.0
ε=0.50
ε=1.0
ε-greedy
Average Reward
Play
(a)
τ=0.2
τ=0.04
τ=1.0
τ=5.0
τ=25.0
Softmax
Average Reward
Play
(b)
σ=0.04 σ=0.33 σ=1.0 σ=3.0 σ=25.0
ε-greedy VDBE-Boltzmann
Average Reward
Play
(c)
ε-greedy VDBE-Boltzmann
σ=25.0 σ=3.0 σ=1.0 σ=0.33 σ=0.04
Average Exploration
Play
(d)

Further reading

1. Tokic, Michel. "Adaptive ε-greedy exploration in reinforcement learning based on value differences." In *Annual Conference on Artificial Intelligence*, pp. 203-210. Springer, Berlin, Heidelberg, 2010.
2. Thrun, Sebastian B. "Efficient exploration in reinforcement learning." (1992).

questions?