REINFORCEMENT LEARNING

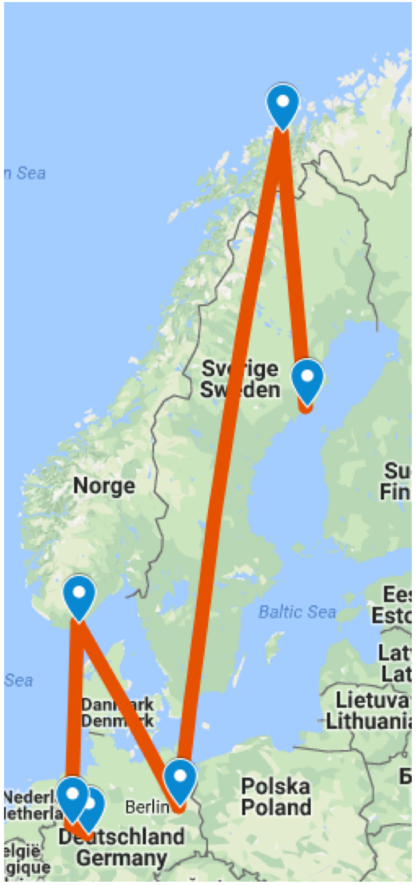
Chapter 21.1-21.3

Timotheus Kampik

UMEÅ UNIVERSITY

THAT’S ME

- Bachelor’s in Information Systems (Kristiansand)
- Master’s Computer Science (Tromsø)
- Currently pursuing PhD at Umeå University:
  - WASP Sweden Autonomous Systems
- Industry background in
  - enterprise systems, product management
  - open source software
- Research interests:
  - AI and decision making
  - Artificial intelligent agents of bounded rationality



EXPECTED LEARNING OUTCOMES

- Recap: Bellman equation and value iteration to solve Markov Decision Process (MDP) problems
- Understand active and passive reinforcement learning
- Be able to conceptualize the exploration vs. exploitation dilemma
- Understand Q-learning
- Be able to implement multi-armed bandits
- **Gain an intuition of how reinforcement learning can be applied**

AGENDA I

- Review: Bellman Equation & MDPs
- RL overview
- Why is RL important?
- Passive RL
  - Direct utility estimation
  - Temporal difference learning
- Active RL (continued)
  - ϵ-greedy
  - Exploration vs. exploitation
  - ϵ-greedy with decaying ϵ

AGENDA II

- Active RL (continued)
  - Q-learning
  - Multi-armed bandits
- Examples
  - Robotics (Boston Dynamics)
  - Music recommender system (Spotify)
  - Basic research (UmU)
- Assignment preview

REVIEW: MARKOV DECISION PROCESSES

$$U_{i+1}(S) = R(S) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_i(s')$$

REVIEW: MARKOV DECISION PROCESSES

Bellman equation:

$$U(s) = \max_{a \in A(s)} \left(R(s,a) + \gamma\, U(s')\right)$$

- s: Current state
- $A(s)$: all possible actions at state s
- s' : Future state
- $R(s,a)$: Immediate reward of S after action a
- γ: Discount factor

➔ Take the action that maximizes the immediate reward plus all time-discounted future rewards

REVIEW: VALUE ITERATION I

0	0
-1	10

REVIEW: VALUE ITERATION II

-0.1	7.2
-1	10

REVIEW: VALUE ITERATION III

5	7.84
-1	10

PROBLEMS WITH MDPS

?

PROBLEMS WITH MDPS

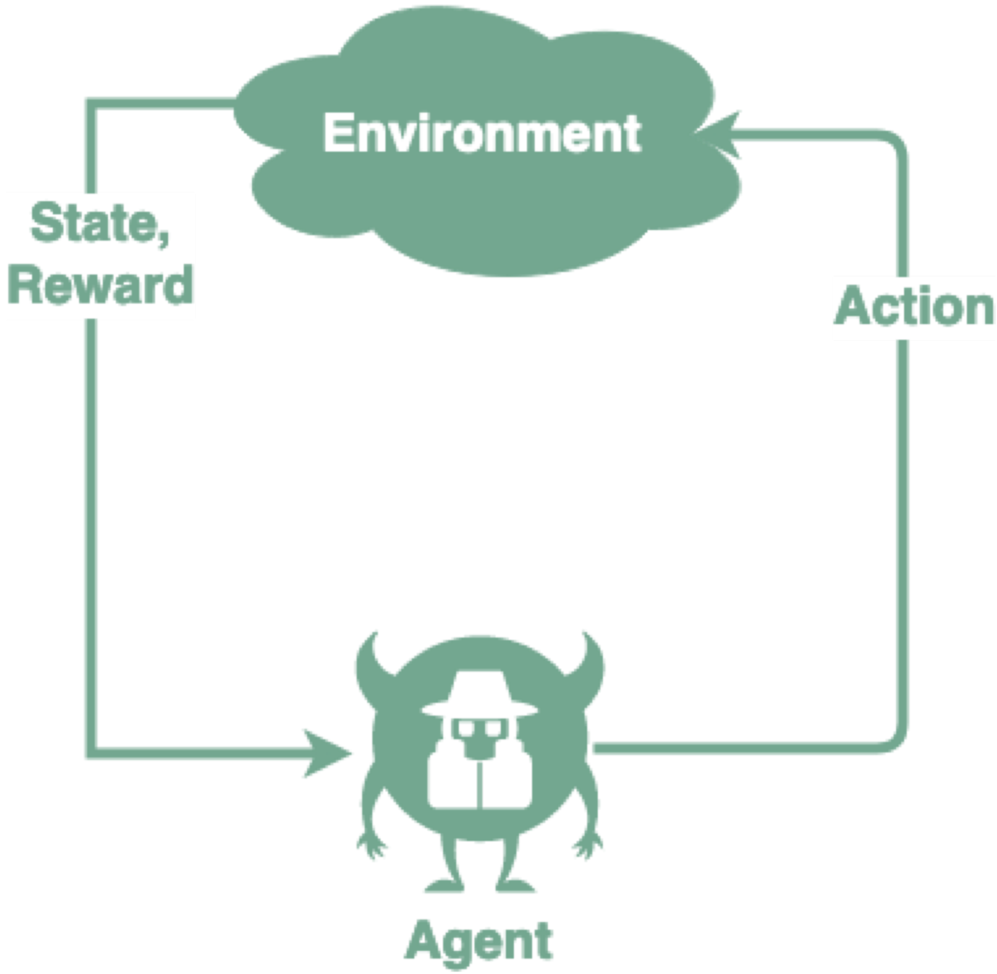
- Simplistic: states and rewards often not fully known

- State space grows quickly, more so with POMDPs

➔ Most real-world problems are too complex to be solved with MDPs

REINFORCEMENT LEARNING

REINFORCEMENT LEARNING (RL)



REINFORCEMENT LEARNING (RL)

- An **agent** learns through iterative inter**actions** with an **environment**
- "Trial and error" approach (very roughly)
- RL log entry: tuple (**State, Action, Time, Reward**)
- **How to select actions that maximize long-term rewards?**
- How to design rewards?

PASSIVE VS. ACTIVE VS. INVERSE RL

- **Passive**: policy is known/fixed: learn utilities of states
  - Direct utility estimation
  - Adaptive programming
  - Temporal difference learning

- **Active**: policy is learned as we go along/dynamic
  - Active temporal difference learning
  - Q-learning
  - State-action-reward-state-action (SARSA)
  - Multi-armed bandits

- **Inverse**: learn policy of an agent we observe

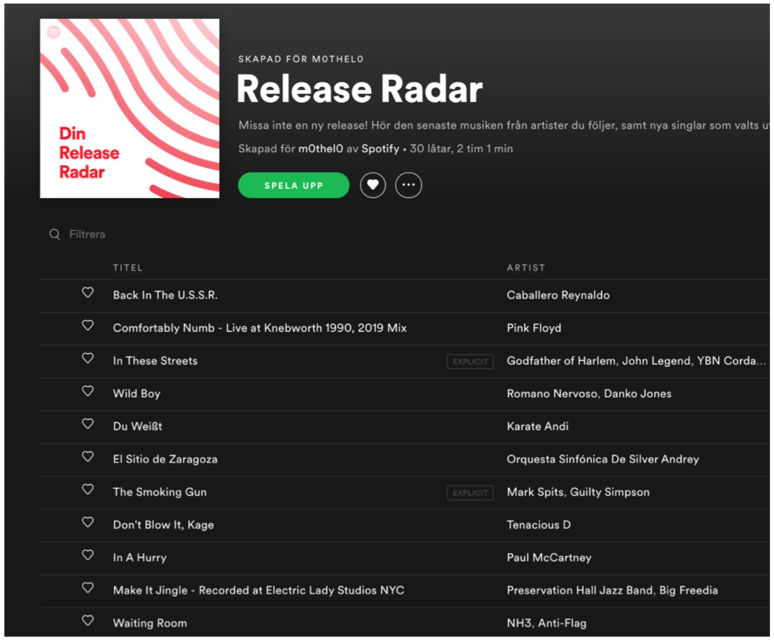
MOTIVATION: WHY RL?

- "Traditional" learning is just correlation and clustering
  - ➔Does not allow for great degree of autonomy
- Planning cannot solve many problems in dynamic real-world environments
  - ➔Computationally too complex

MOTIVATION: WHY RL?

Use Cases?

MOTIVATION: WHY RL?



McInerney, James, et al. "Explore, exploit, and explain: personalizing explainable recommendations with bandits." *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 2018.

https://robots.ieee.org/robots/spotmini/

Hwangbo, Jemin, et al. "Learning agile and dynamic motor skills for legged robots." *arXiv preprint arXiv:1901.08652* (2019).

PASSIVE REINFORCEMENT LEARNING

PASSIVE RL

- Agent interacts with environment using a fixed policy
- The agent use the fixed policy π
- Evaluate policy π
- Passive RL does not dynamically choose actions

Input
1. History of taken Actions
2. Resulting transitions
3. Rewards

→ Algorithm →

Output
Expected utility of every state

DIRECT UTILITY ESTIMATION

- Essentially supervised learning
- Run policy several times
  For each time:
  - Update expected utility of state with:
    ”experienced” reward +
    future rewards at the given state
- Utility of state s, given policy π:

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' | s, \pi(s)) U^{\pi}(s')$$

s: $state$, π: $policy$
γ: $discount\ factor$
s': $future\ state$

DIRECT UTILITY ESTIMATION

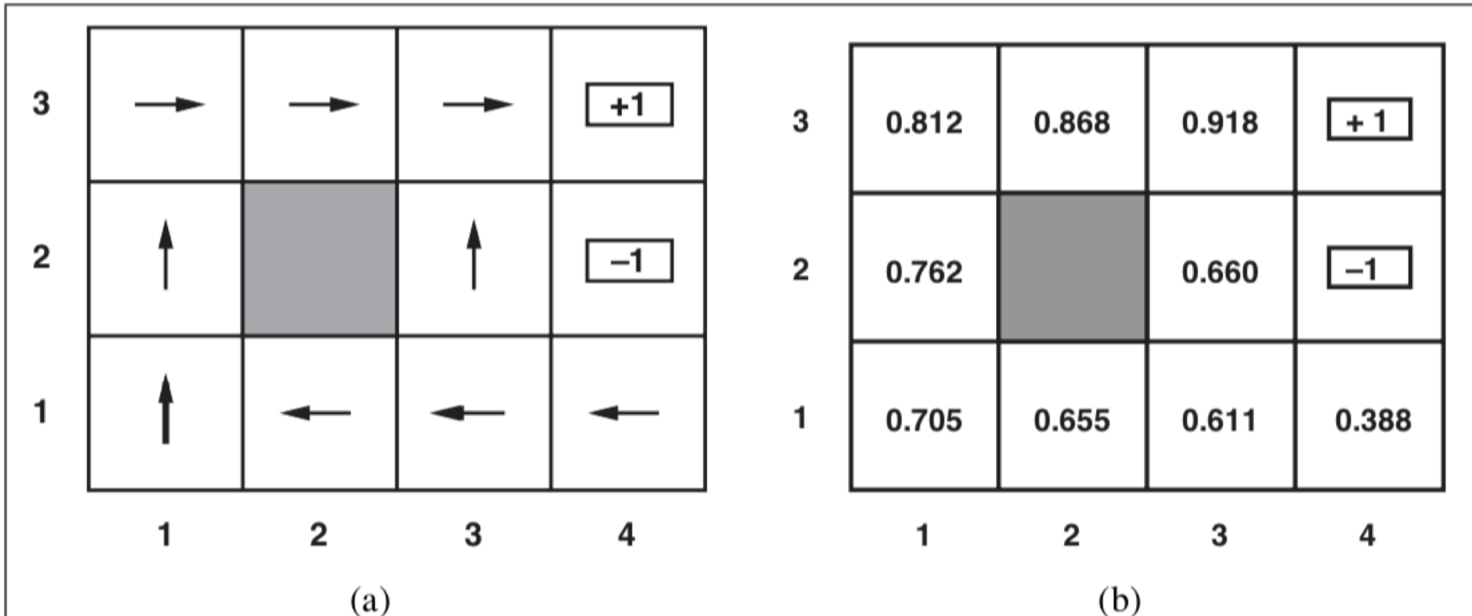


Figure 21.1 (a) A policy π for the 4×3 world; this policy happens to be optimal with rewards of $R(s) = -0.04$ in the nonterminal states and no discounting. (b) The utilities of the states in the 4×3 world, given policy π.

Russel, Norvig: Artificial Intelligence: A Modern Approach

DIRECT UTILITY ESTIMATION

- (1,1): -0.04 → (1,2): -0.04 → (1,3): -0.04 → (2,3): -0.04 → (3,3): -0.04 → (4,3): +1

- (1,1): -0.04 →(2,1): -0.04 →(3,1): -0.04 → (3,2): -0.04 →(4,2): -1

What is the estimated utility of state (1,1)?

DIRECT UTILITY ESTIMATION

- (1,1): -0.04 → (1,2): -0.04 → (1,3): -0.04 (2,3): -0.04 → (3,3): -0.04 → (4,3): +1
- (1,1): -0.04 →(2,1): -0.04 →(3,1): -0.04 → (3,2): -0.04 →(4,2): -1

What is the estimated utility of state (1,1)?

➔ (1 - 5 x 0.04 - 1 – 4 x 0.04) / 2

= - 0.18

DIRECT UTILITY ESTIMATION

Reduces the RL problem to an inductive learning problem

- Misses that utilities are not independent
- No learning until the end of trial → converges slowly

TEMPORAL-DIFFERENCE LEARNING

- Adjust (update) the current estimate of utility of each state
- By observing actions, transitions, and rewards
- It shows how much we under/over estimated the utility of the current state and then adjust it based on the observed successor s'
- Each time we move from s to s' we update the utility estimation
- Basis for Q-learning algorithm (active RL)

TEMPORAL-DIFFERENCE LEARNING

function PASSIVE-TD-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  persistent: π, a fixed policy
    U, a table of utilities, initially empty
    N_s, a table of frequencies for states, initially zero
    s, a, r, the previous state, action, and reward, initially null

  if s' is new **then** $U[s'] \leftarrow r'$
  if s is not null **then**
    increment $N_s[s]$
    $U[s] \leftarrow U[s] + \alpha(N_s[s])(r + \gamma U[s'] - U[s])$
  if s'.TERMINAL? **then** $s, a, r \leftarrow$ null **else** $s, a, r \leftarrow s', \pi[s'], r'$
  return a

Russel, Norvig: Artificial Intelligence: A Modern Approach

TEMPORAL-DIFFERENCE LEARNING

$$U[s] \leftarrow U[s] + \boldsymbol{\alpha}(\mathrm{N}_s[s])(r + \gamma U[s'] - U[s])$$

- $U[s]$: estimate of reward in previous state s
- $\boldsymbol{\alpha}$: learning rate
- $\mathrm{N}_s[s]$: frequency of state s
- r: reward, as just received in state s
- $\gamma U[s'] - U[s]$: discounted reward of current state s' – reward of previous state
  ➔ How good is current state compared to previous state?
  ➔ If reward in previous state was higher, we discount utility, else we add utility to estimation

PROBLEM WITH PASSIVE REINFORCEMENT LEARNING

- In passive learning we can estimate utilities and transition probabilities for a **fixed policy**
  - using passive recordings of an agent interacting with the environment.
- But not for any action that is not in the policy
- The agent cannot discover the environment to find better policies, it can only use the action which is defined by the fixed policy.

ACTIVE REINFORCEMENT LEARNING

ACTIVE RL

- The agent attempts to find the optimal policy
- Or at least a "good" policy
- By exploring the world taking different actions

➔ The agent learns as it goes along and adjusts its policy step-by-step

ACTIVE LEARNING

- Methods are similar to the passive learning but with ability of using the new freedom (choosing actions)
- Instead of using the expected utility for a fixed policy, the agent use expected utility for the best policy
- Agent can select any action to take (not only those that are defined by the fixed policy)
- Therefore, can explore the environment and improve the policy
- Its all about Exploration vs Exploitation

$\varepsilon - greedy$

- With probability of ε ($0 < \varepsilon < 1$):
  - Execute random action

- With probability of 1- ε:
  - Execute action with highest expected utility, given current knowledge

- Update expected utility, given (state, action)

EXPLORATION-EXPLOITATION DILEMMA

?

EXPLORATION-EXPLOITATION DILEMMA

- **Explore**:
  try to find better actions
- **Exploit**:
  execute action with highest expected utility, given the knowledge we have

- **Explore too much**
  - ➔ ***regret* caused by lack of commitment**
- **Exploit too much**
  - ➔ ***regret* caused by lack of knowledge**
  - ➔ **get stuck in local maximum**

Decaying ε – greedy

- With probability of ε ($0 < \varepsilon < 1$):
  - Execute random action
- With probability of $1 - \varepsilon$:
  - Execute action with highest expected utility, given current knowledge
- Update expected utility, given (state, action)
- Decrease ε (multiply by factor x, $0 < x < 1$)

Q-LEARNING

- Q-learning learns **action-utility** instead of learning utilities
  - $U(s) = \max_{a} Q(s, a)$
- Does not need a model of $P\left(s^{'} \middle| s, a\right)$:
  - Probability of being in state s', given prior state s and action a

  → **model-free**
- Q-learning is off-policy,
- Action-utility assignment analogous to temporal difference learning:

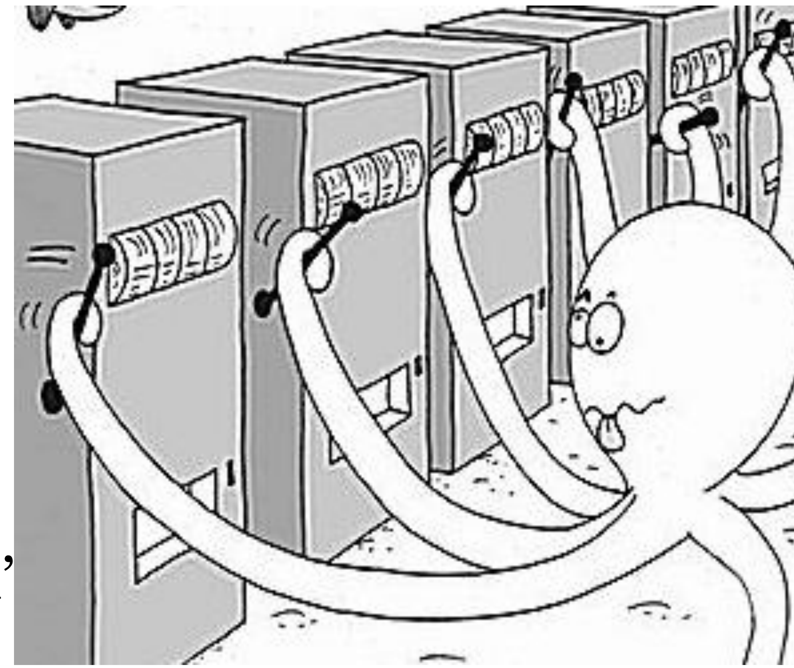
$$Q(s, a) \leftarrow Q(s, a) + \alpha(R(s) + \gamma \max_{a\prime} Q(s', a') - Q(s, a))$$

Q-LEARNING ALGORITHM

1. Start in state *s*
2. Take action a based on exploration/exploitation strategy (epsilon-greedy or similar)
3. Based on the utility of the new state *s'*: update the utility of previous state *s*
4. Execute the policy
5. Update the current state *s'*
6. Repeat steps

MULTI-ARMED BANDITS (MAB)

- *N* possible actions
- Each action has unknown expected reward (random variable)
- Goal:
  - find best (or at least “good” action)



http://www.primarydigit.com/blog/multi-arm-bandits-explorationexploitation-trade-off

MAB – EPSILON-GREEDY

- N arms, $0 < \varepsilon < 1$
- At iteration i, $0 < i < N$:
  - Pull arm i.
  - Log reward returned by arm i.
- At iteration i, $i > N$:
  - If $\varepsilon > random(0,1)$: Pull random arm
  - Else: Pull arm with highest expected reward
  - Updated expected reward of pulled arm

MAB – EPSILON-DECAY

- N arms, $0 < \varepsilon < 1$, $0 < x < 1$
- At iteration i, $0 < i < N$:
  - Pull arm i.
  - Log reward returned by arm i.
- At iteration i, $i > N$:
  - If $\varepsilon > random(0,1)$: Pull random arm
  - Else: Pull arm with highest expected reward
  - Updated expected reward of pulled arm
  - $\varepsilon \leftarrow \varepsilon * x$

MAB – OTHER ALGORITHMS

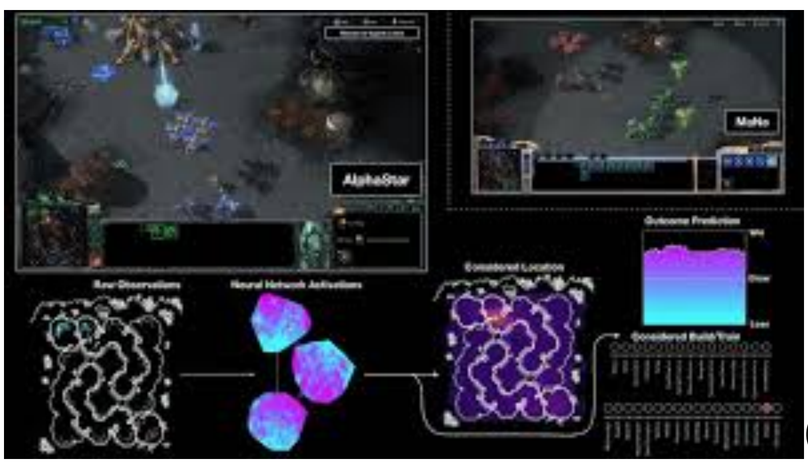
- Decay function for epsilon
- ”Discard” arms that are clearly bad
- Thompson sampling:
  - Assumes known initial distribution over action values
  - Allows (theoretically) to compute optimal exploration vs. exploitation balance

EXAMPLES

THE OBVIOUS ONES

https://www.netflix.com/se/title/80190844

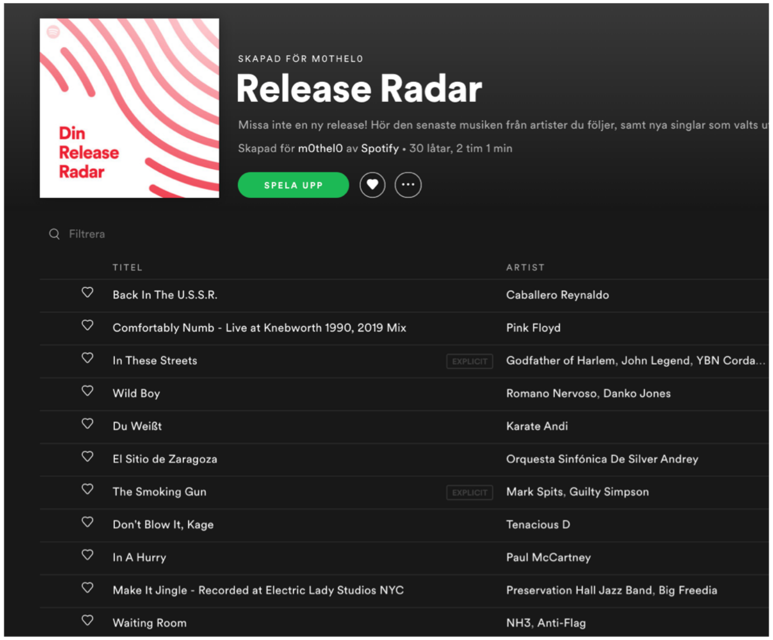
https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go



https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

https://arxiv.org/pdf/1312.5602.pdf

SPOTIFY



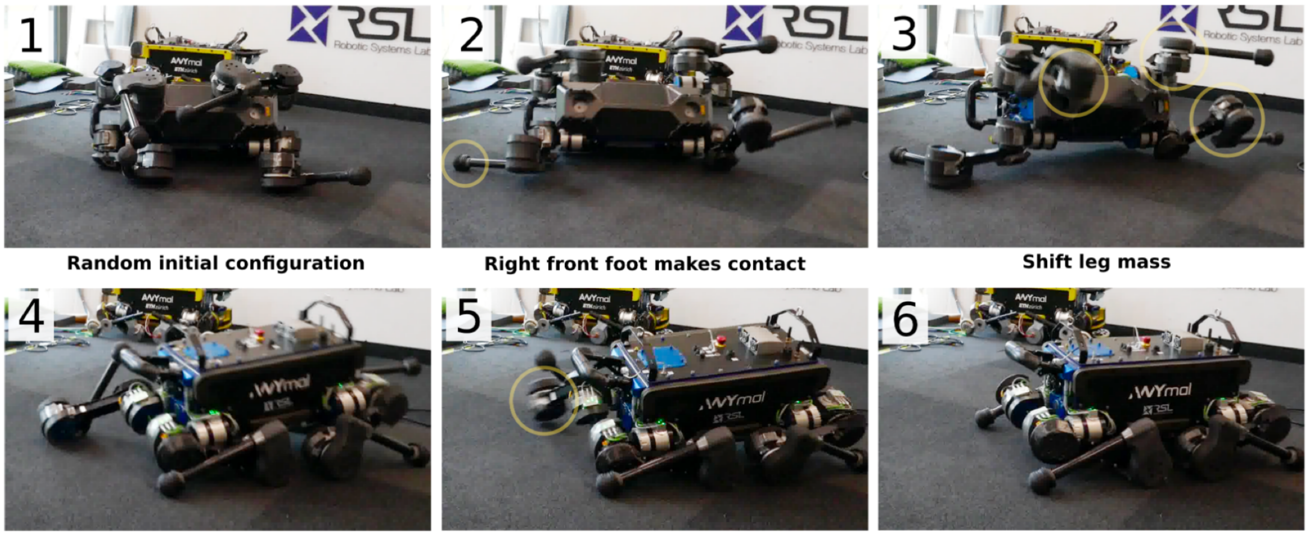
McInerney, James, et al. "Explore, exploit, and explain: personalizing explainable recommendations with bandits." *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 2018.

EXPLAINABLE BANDITS

Use novel **explainable** personalized recommendations generated by multi-armed bandits

- Make exploration explainable
- Bandit dynamically changes explanation type
- Recommendations on two dimension:
    - Recommended item
    - Explanation of recommended item

BOSTON DYNAMICS (ETH ZÜRICH, INTEL)



Random initial configuration | Right front foot makes contact | Shift leg mass

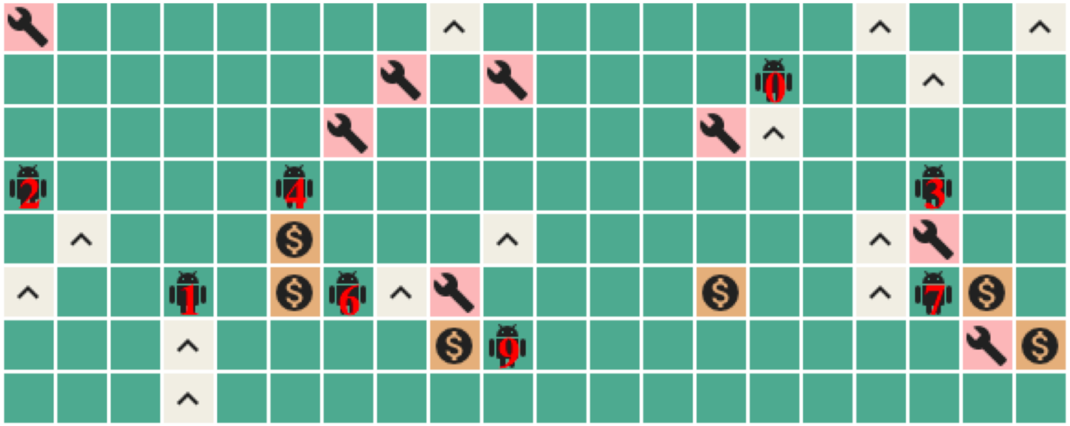
Initial impact | Roll using momentum and retract knees | Final configuration

Hwangbo, Jemin, et al. "Learning agile and dynamic motor skills for legged robots." *arXiv preprint arXiv:1901.08652* (2019).

RL FOR ROBOTS

- "Learn" controller that manages robot's locomotion skills "best"
- Train in simulation
- Eventually out-perform hand-crafted controllers
- Still needs control theory, though!

UMU: RL-REWARDS AND FAIR EQUILIBRIA



Kampik and Spieker. "Learning Agents of Bounded Rationality: Rewards Based on Fair Equilibria."

MULTI-AGENT GRID WORLD

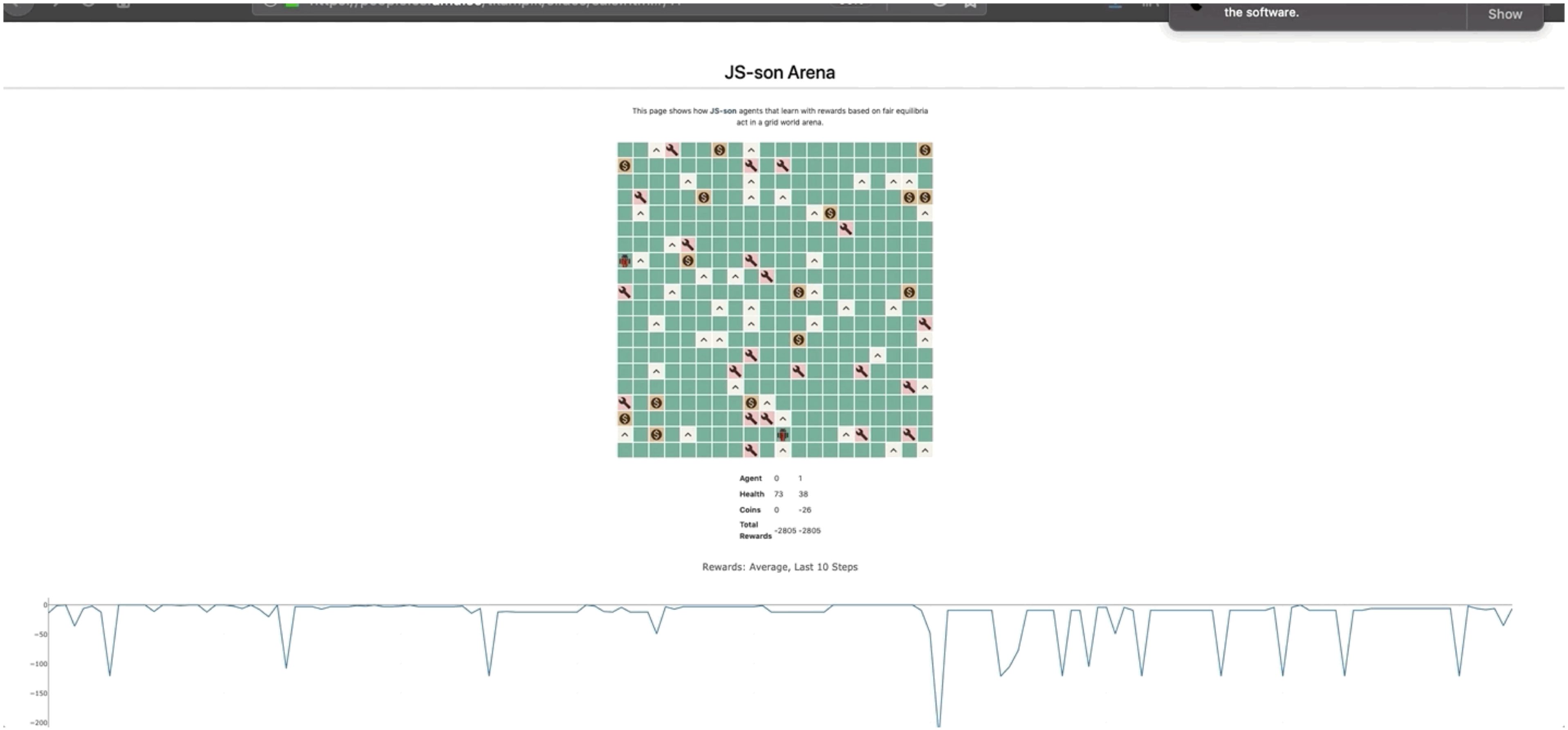
- Agents act in a grid world
- Should collect coins
- Loose health over time ➔ need to repair
- Collecting coins and reparations negatively affect other coins/health of others

➔ How to act sustainably as a society/community?

REWARD DESIGN

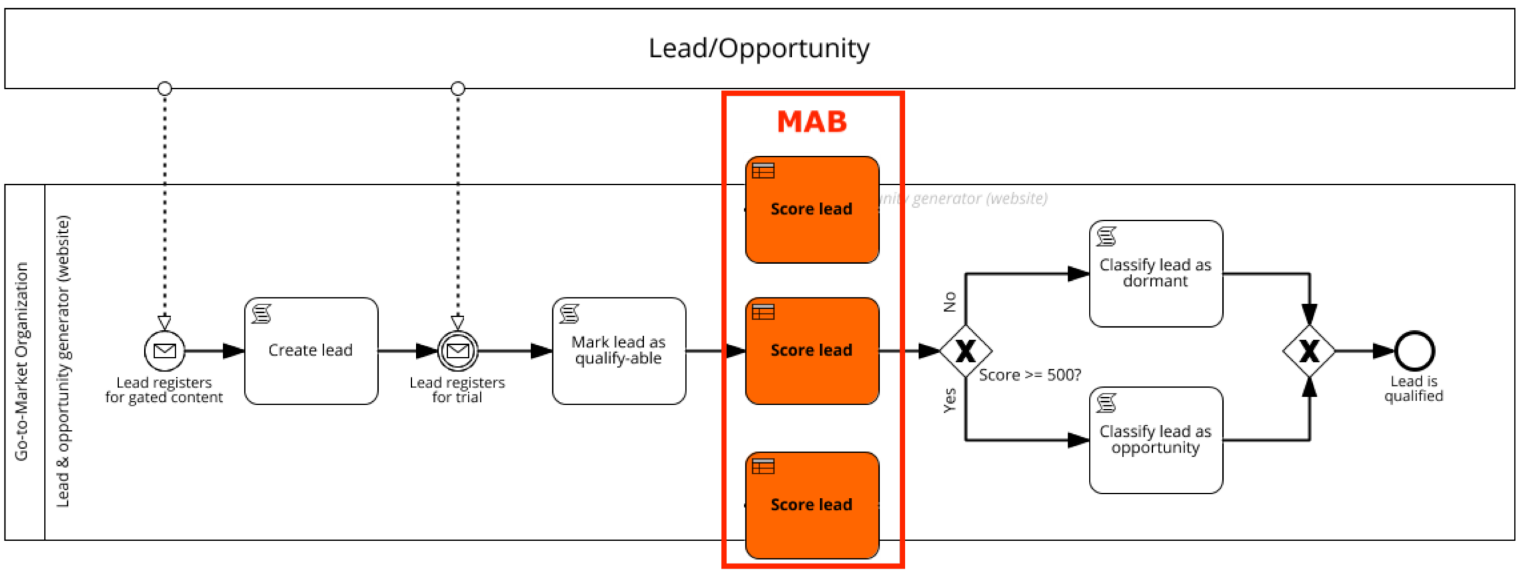
- All (both) agents are rewarded for *fairness*
- Rewards are based on:
  - How far are the *actual* actions from the closest *fair equilibrium*?
  - Smaller distance leads to higher reward

FAIR EQUILIBRIUM - EXAMPLE



https://people.cs.umu.se/tkampik/slides/sais.html#/11

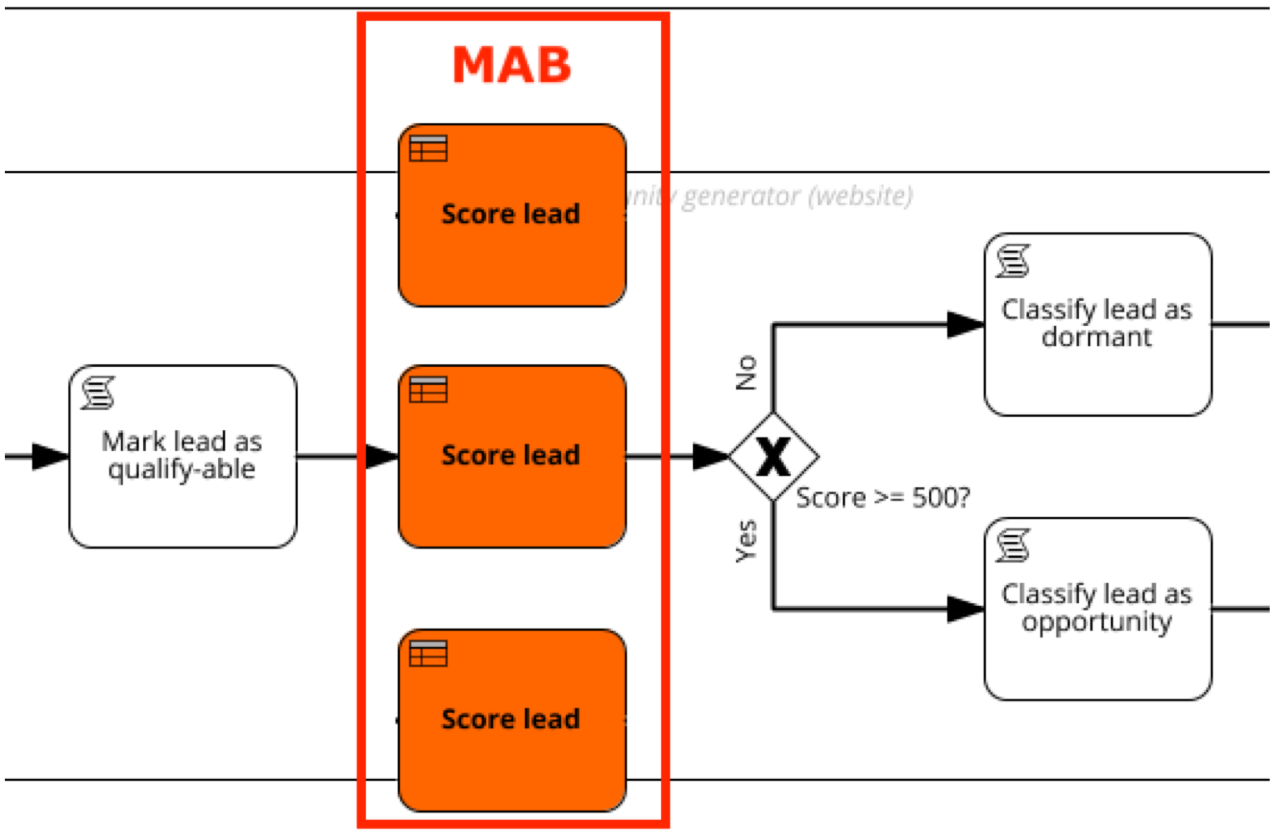
BANDITS FOR BUSINESS PROCESS MANAGEMENT



Mohan and Kampik. Work in progress.

UMEÅ UNIVERSITY

BANDITS FOR BUSINESS PROCESS MANAGEMENT



UMEÅ UNIVERSITY

BANDITS FOR BUSINESS PROCESS MANAGEMENT

- Use multi-armed bandits to test/simulate different task configurations before deploying at scale
- "Dynamic A/B testing"
- Especially useful in scenarios, where full-scale deployments are hard to change (e.g., smart contract)

LAB III – REINFORCEMENT LEARNING WITH MULTI-ARMED BANDITS

- Multi-armed bandits:

  Practical: Towards Data Science
  Academic: Paper
- In the lab, you will implement a multi-armed bandit to solve an example problem.
- Your bandit will need to beat a "naïve" benchmark.
- The best bandit will be determined.
- You will need to use git for version control: https://github.com/TimKam/multi-armed-bandit-lab
-

FURTHER READING

- *Russel, Norvig: Artificial Intelligence: A Modern Approach, chapters 21.1 – 21.3*

  plus:

  Literature about multi-armed bandits:
  - "Towards Data Science" introduction to multi-armed bandits
  - *Kuleshow, Precup: Algorithms for the multi-armed bandit problem*

FURTHER CODING

- *OpenAI Gym:* https://gym.openai.com/
- *Reinforcement learning in JavaScript:* https://metacar-project.com
- *Multi-armed bandits in Python:* https://github.com/bgalbraith/bandits