Aalto University
School of Electrical Engineering

ELEC-E8125 Reinforcement Learning
Exploration and exploitation

Joni Pajarinen

7.11.2023

Learning goals

- Understand how to select actions that allow us to learn the best action

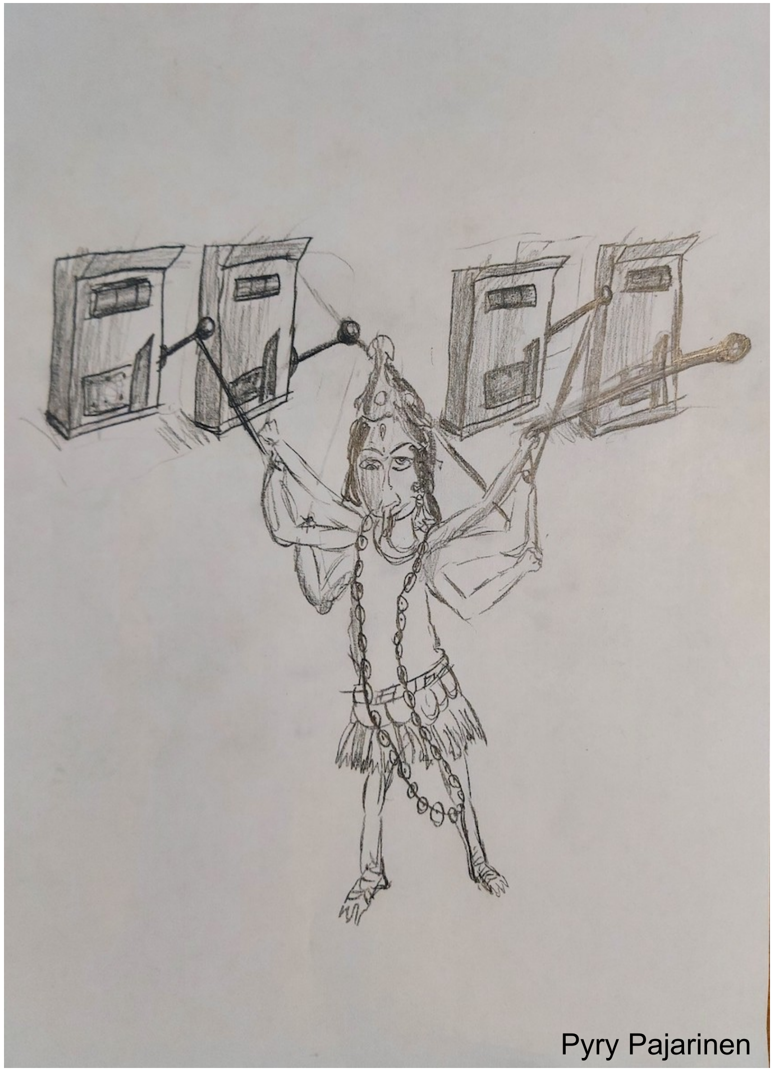
Exploration vs. exploitation

- Exploration: try out actions to learn which actions are the best
- Exploitation: select high performance actions (according to current knowledge)

Have we already done something like this?

Multi-armed bandit

- Multi-armed bandit has K arms
- Pulling bandit arm k corresponds to action $a=k$
- Pulling an arm yields a reward from an unknown probability distribution $P(r|a)$
- Special case of an MDP without states
- How to get maximum total reward?



Pyry Pajarinen

How to select arms so that we get maximum reward?

Greedy approach in the multi-armed bandit setting

- For each arm, we estimate mean action value

$$Q(a) = \frac{1}{N(a)} \sum_{n=1}^{N(a)} r_n(a)$$

- Greedy approach chooses action with highest action value estimate:

$$\hat{a} = argmax_a Q(a)$$

- Do we find the best action? Why / why not?

Finding best action: example on blackboard

Epsilon-greedy in the multi-armed bandit setting

- Epsilon greedy chooses action with highest value estimate $Q(a)$ with fixed probability $1-\epsilon$
- and uniformly randomly chosen action with probability ϵ

actions

Total number of samples

- Tries out every action approximately at least $\epsilon N/|A|$ times
- Do we find the best action? Is epsilon-greedy sample efficient?
- How to improve?

Sample efficiency: example on blackboard

Trading off exploration vs. exploitation in the multi-armed bandit setting

- Goal: find best action using only few tries / samples
- Try out actions if they can be optimal but not otherwise: how to quantify this?
- The more we try out an action a the more certain we are about our estimate $Q(a)$

- We will discuss two approaches:
  - Upper confidence bound (UCB) approach
  - Thompson sampling

Upper confidence bound

- Estimate additional upper confidence term $U(a)$ for each action based on N(a), number of tries of action a
- When N(a) is low, $U(a)$ should be high
- When N(a) is high, $U(a)$ should be low
- Select action that maximizes the sum $\hat{Q}(a) = \underbrace{Q(a)}_{\text{Exploitation}} + \underbrace{U(a)}_{\text{Exploration}}$
- $\rightarrow$ tries out actions where we are uncertain about the current value estimate
- How to compute $U(a)$?

Computing upper confidence bound

- For selecting $U(a)$, let's use **Hoeffding's Inequality:**

For i.i.d. random variables $X_1, \ldots, X_M$ in $[0,1]$ where the mean estimate after M samples is $\bar{X}_M = \frac{1}{M}\sum_{m=1}^{M} X_m$, it is true that

$$P(E[X] > \bar{X}_M + u) \leq e^{-2Mu^2}$$

- Let's apply the inequality to the bandit action a :

$$P(E[Q(a)] > Q(a) + U(a)) \leq e^{-2N(a)U(a)^2}$$

Estimate of action value Q(a) using N(a) samples

True expected action value Q(a)

Computing upper confidence bound

- Limit probability of true value to exceed upper bound:

$$P(E[Q(a)] > Q(a) + U(a)) \leq e^{-2N(a)U(a)^2} = p$$

$$\rightarrow U(a) = \sqrt{-1/2 \log p / N(a)}$$

- Choosing $p = N^{-4}$ yields

$$\hat{Q}(a) = Q(a) + U(a) = Q(a) + \sqrt{2 \log N / N(a)}$$

- This is the UCB1 formula. When N goes to infinity, maximum value error is $(\log N / N) const$

[Auer et al. *Finite-time analysis of the multiarmed bandit problem*, 2002]

Example algorithm using UCB1

1) For each action a, sample reward r and set Q(a) = r
2) Initialize N = |A| and N(a) = 1 for each action a
3) Sample r for action a that has the highest UCB1 value

$$\hat{Q}(a) = Q(a) + U(a) = Q(a) + \sqrt{2 \log N / N(a)}$$

4) Update mean Q(a): Q(a) = (r + N(a) Q(a)) / (N(a) + 1)
5) Update N = N + 1 and N(a) = N(a) + 1
6) Goto 3)

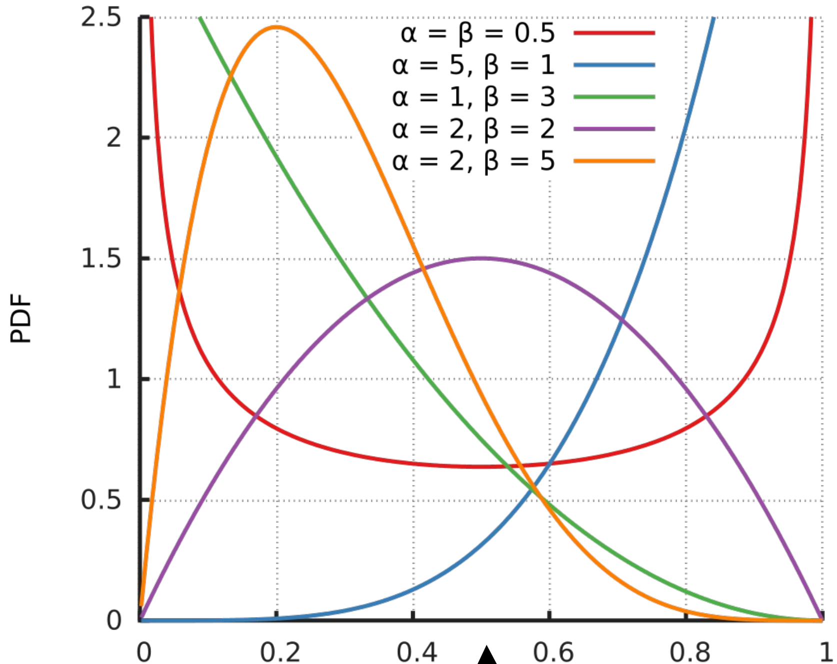
[Auer et al. *Finite-time analysis of the multiarmed bandit problem*, 2002]

Thompson sampling

- Idea: sample each action according to the probability of the action to be the best
- Requires computing for every action the probability of being the best action based on the history of all observed rewards
- Can utilize prior knowledge

Thompson sampling: Bernoulli bandits

- Each Bernoulli bandit produces a 1 with probability θ_k and a 0 with probability $1-\theta_k$
- Keep counts of 1s and 0s, α_k and β_k , for each arm k
- Algorithm main loop:
  - For each arm k sample θ_k from Beta(α_k, β_k)
  - $a = argmax_k \theta_k$
  - Sample r from $P(r|a)$
  - Update counts:
    - if r = 1: $\alpha_k = \alpha_k + 1$
    - If r = 0: $\beta_k = \beta_k + 1$



How to incorporate prior knowledge about the bandits?

Beta distribution

From multi-armed bandits to MDPs

- Can we utilize the insights in multi-armed bandits for exploration in MDPs?
- In an MDP, instead of Q(a) find Q(s,a)
  - Use multi-armed bandit to choose action
  - Evaluate Q(s,a) using Monte Carlo value estimation
  - How to generate a sequence of states and actions in Monte Carlo value estimation of Q(s,a)? What policy to use? How to simulate state transitions?

From multi-armed bandits to MDPs

- Can we utilize the insights in multi-armed bandits for exploration in MDPs?
- In an MDP, instead of Q(a) find Q(s,a)
  - Use multi-armed bandit to choose action
  - Evaluate Q(s,a) using Monte Carlo value estimation
  - In Monte Carlo value estimation, use a multi-armed bandit approach such as UCB1 as the policy!
  - Assume a known dynamics model such as $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t)$
  - Leads to **Monte Carlo tree search** (MCTS)

Reminder: spectrum of model-based RL

Time of planning

- **On-line**
  - **Discrete actions**
    - MCTS
    - ...
  - **Continuous actions**
    - iLQR
    - CEM
    - ...
- **Off-line**
  - **Simulate environment**
    - DYNA
    - Latent state models
    - ...
  - **Assist learning**
    - Policy backprop
    - ...

Monte Carlo tree search

- Search method for optimal decision making
- State-of-the-art for playing games (e.g. Alpha Go)

- Iteratively builds a search tree
  - Each search tree node is a multi-armed bandit
- Phases:
  - Selection: Choose a promising node to expand
  - Expansion: Add a new node
  - Simulation: Simulate value for new node
  - Backup: Back-up value to root (update values for parents)

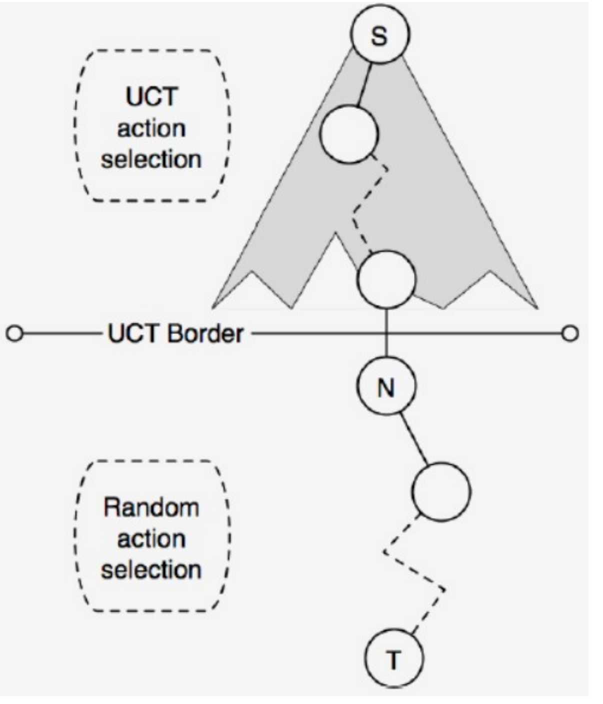
Using e.g. UCB1

Monte Carlo value estimation

Blackboard: example tree. Each node corresponds to a state.

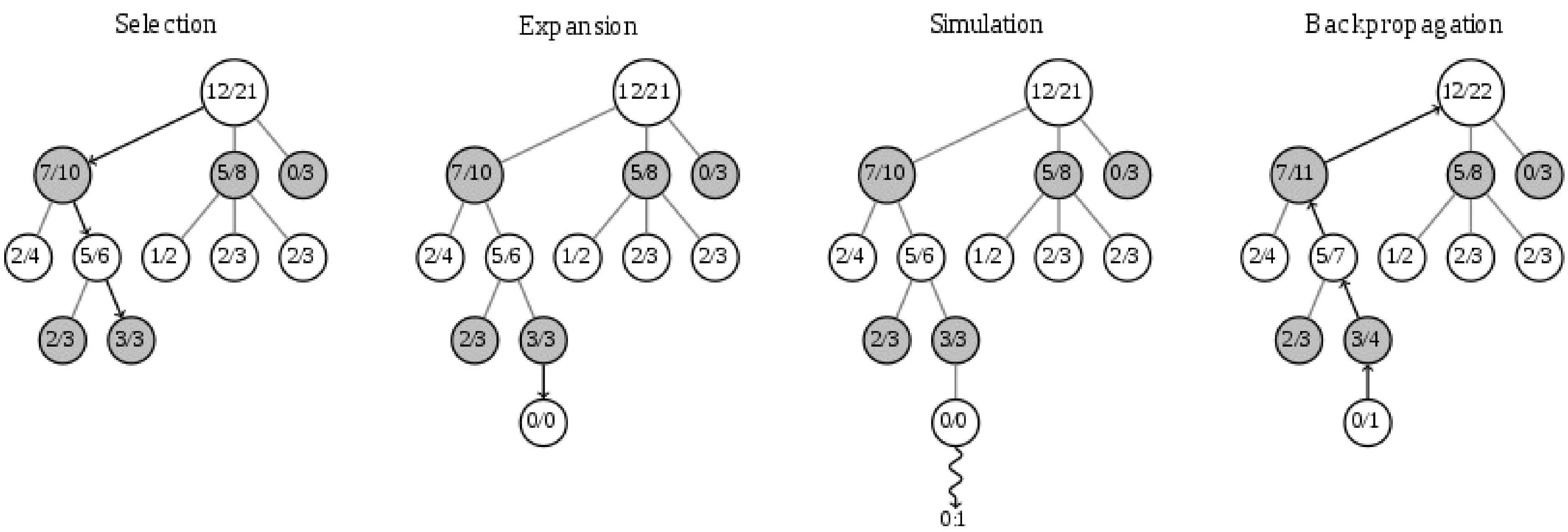
MCTS operation

- From start node *S* choose actions to walk down tree until reaching a leaf node.
- Choose an action and create a child node for that action.
- Perform a **random** roll-out (take random actions) until end of episode (or for a fixed horizon).
- Record returns as value for child node and back up value to root.



MCTS: Example search tree

- Value: number of won/simulated games



Node selection in MCTS

- Node selection in search has to balance between exploration and exploitation (note difference to RL, here exploration & exploitation only using simulation)
- Idea: Explore when uncertain of outcome
- Upper confidence bound 1 (UCB1) on trees (UCT)
  - A bound for value of a node (Kocsis & Szepesvari, 2006)

$$\hat{Q}(s,a) = Q(s,a) + c\sqrt{\frac{2\log N(s)}{N(s,a)}}$$

Exploration constant. Depends on the range of values. For guaranteed convergence, largest possible value minus smallest possible value.

MCTS simulation phase

- Perform one or several roll-outs from leaf node using random action selection
- Stop at terminal state or until a discount horizon is reached
- Estimate value of state as mean return of the *N(s)* simulations: $$V(s) = \frac{1}{N(s)} \sum_i G_i(s)$$

MCTS backpropagation

- After simulation phase backpropagate values to the root node
- Estimate value of state as mean return of the *N(s)* simulations:

$$V(s) = \sum_a \frac{N(s,a)}{N(s)} Q(s,a)$$

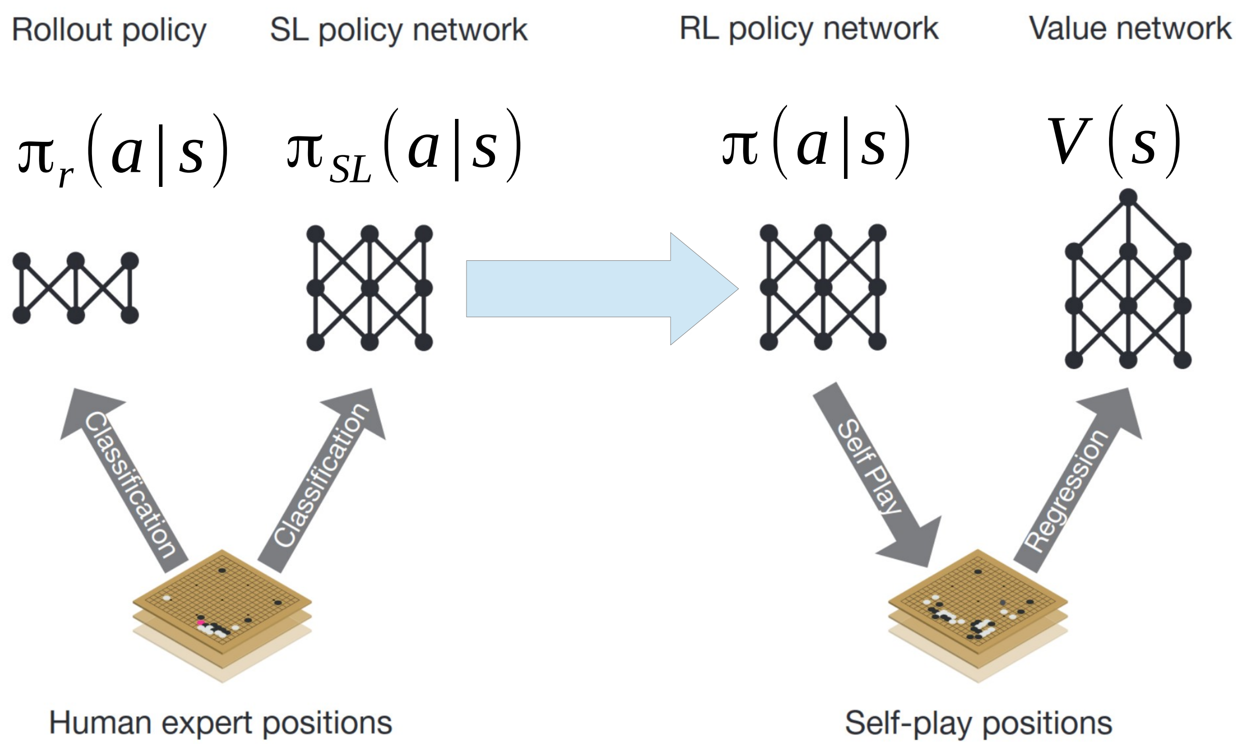
$$Q(s,a) = E_{s' \sim p(.|s,a)}[R(s,a) + V(s')]$$

MCTS extensions

- AlphaGo (2016)
  - Learn initial policy from expert demonstrations
  - Update policy using self-play and MCTS
- AlphaZero (2017, 2018)
  - No expert demonstrations needed
- MuZero (2020)
  - Similar to AlphaZero but interleaves model learning and MCTS
  - Does not require a known model

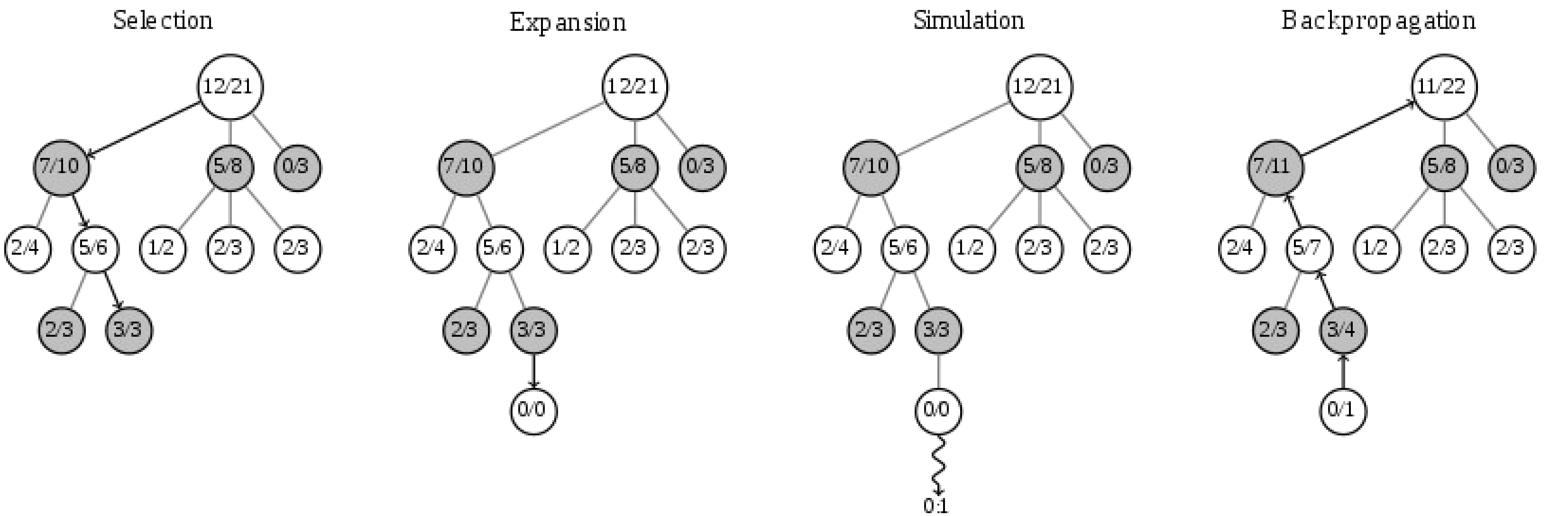
Example: Alpha Go (2016)

- Policy learned initially to imitate human players
- Updated through policy gradient and self-play



Example: Alpha Go (2016)

- Action chosen by bandit using Q(s,a) and policy
- Leaf-node value: estimated value V(s) plus roll-out value



Summary

- Balancing exploration and exploitation important for sample efficient reinforcement learning
- There are efficient approaches such as UCB and Thompson sampling for multi-armed bandit problems
- Monte Carlo tree search (MCTS) extends multi-armed bandits to model-based reinforcement learning
- Allows trading off between exploration and exploitation with proofs of convergence to an optimal solution

Next: Offline reinforcement learning

- Next week: Guest lecture on offline reinforcement learning by Mohammadreza Nakhaei
- No quiz for next week
- There will be a quiz for the lecture in two weeks!