Machine Learning for Networking

Reinforcement Learning

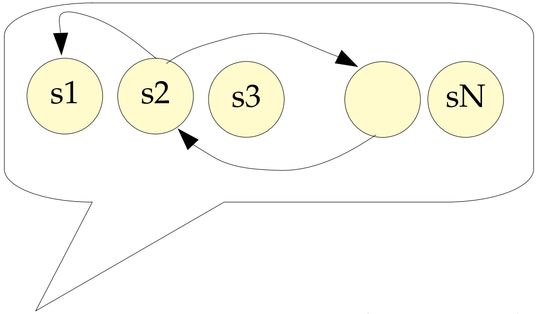
Session 9 – Q-learning

Boris Bellalta: boris.bellalta@upf.edu

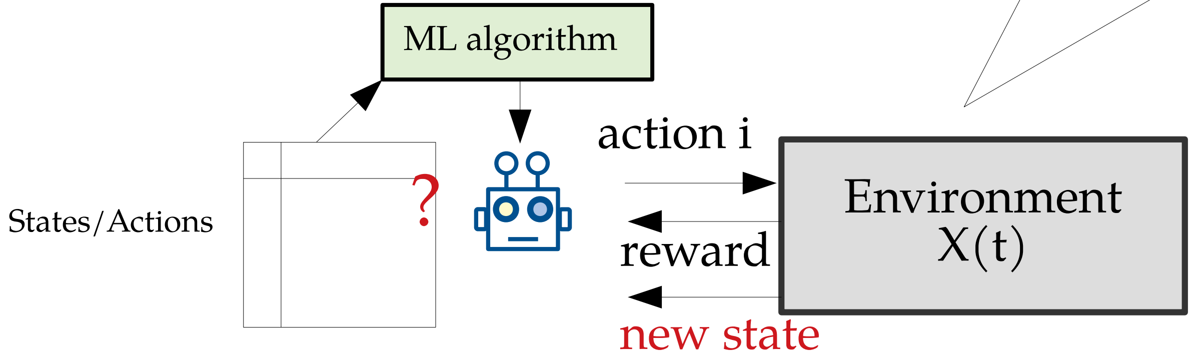
Contents

- The channel selection problem
- MDPs
- Q-learning

Reinforcement Learning



The environment can be represented by a set of states

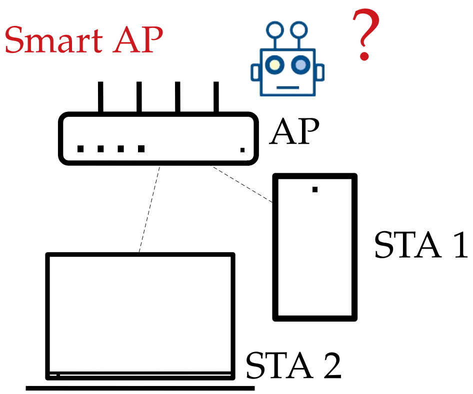


The agent learns by interacting with the environment.

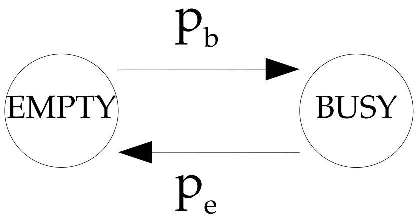
The ML algorithm tells the agent how to do that interaction to maximize the reward.

Channel selection problem

channels 1 2 3 ... N



Channel model: ON-OFF

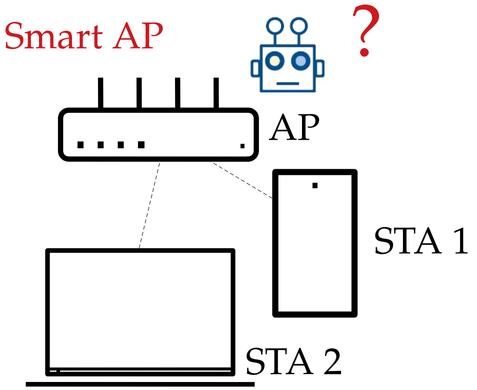


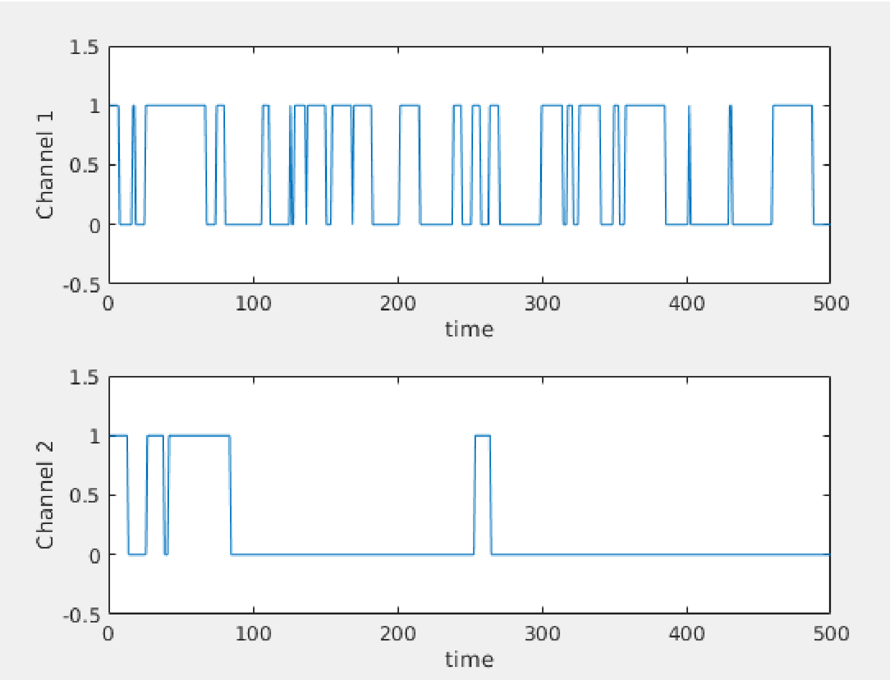
Reward empty: +1
Reward busy: -1

Goal: Maximize Accumulated reward.

ON/OFF channel model

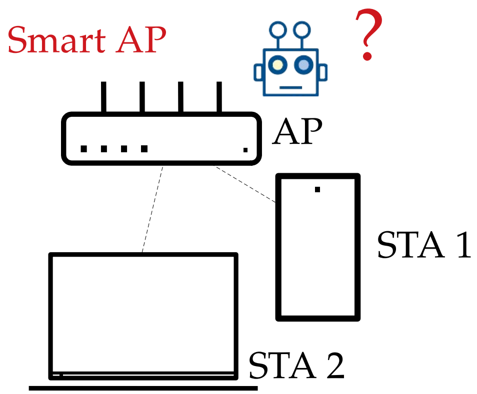
channels | 1 | 2





States & Actions

channels 1 2



States (4)

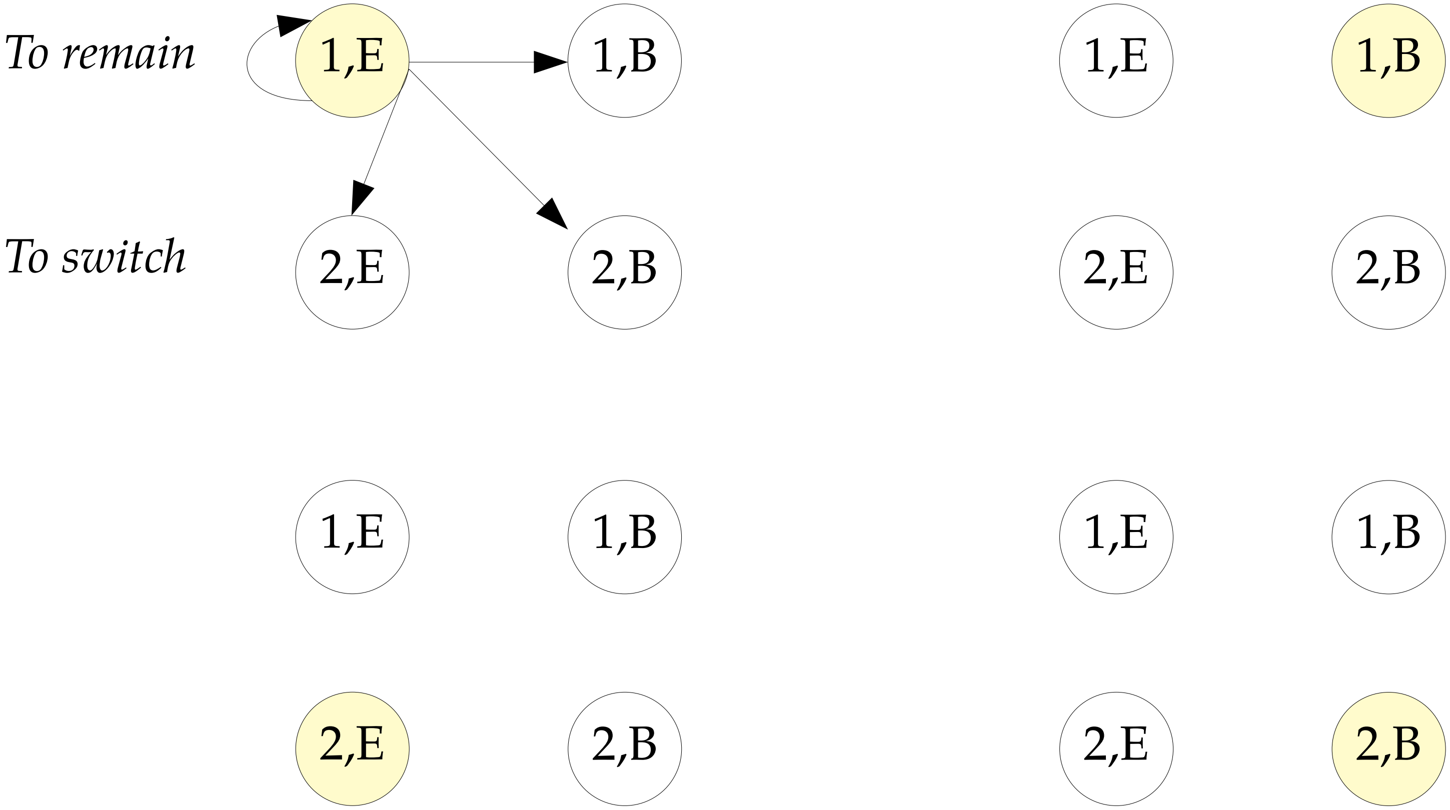
- 1,E
- 1,B
- 2,E
- 2,B

actions

- a_1=Remain in the same channel
- a_2=Switch to the other channel

(transitions not included)

Representation



Markov Decision Process (MDP)

- Our example is a MDP
  - Markov: next state of the process depends only on current state
- MDP
  - Set of states: the state space
  - Set of actions (they could depend on the current state)
  - Transition probabilities (which depend on the actions we take)
  - Rewards (which depend if our actions were good or bad)
- Solving the MDP → "learning"
  - Finding the best action to take when we are in a given state
- Q-learning is an algorithm to solve a MDP

Q-learning

- Table to store the pair (state,action), i.e. the Q-table

State	Remain	Switch
1,E		
1,B		
2,E		
2,B		

- A mechanism to explore the state space
  - Epsilon-greedy (for example)

- A way to update the Q-table (from wikipedia)

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

Q-learning

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

- Learning rate: how we replace old with new information (reward)
  - 0: we don't learn at all, 1: we use only instantaneous information
- Discount factor: determines the importance of future rewards
  - *"If I do this, I will get that in the future"*
  - 0: only rewards from the current state are considered
  - 1: only considers "expectations", may create instability (better to set to lower values)

Q-learning, the code

```
for t=1:TimeHorizon-1

    % I take a new action

    if rand < Epsilon
       p = randi(K); % I explore an action randomly
       explore(t)=1;
    else
       [~,p]=max(Q_table(state(t),:)); % I take the best action from the state in which I'm
    end

    action(t)=p;
    action_ntimes(state(t),action(t))=action_ntimes(state(t),action(t)) + 1;

    % Next State (from the action I have taken)

    if(state(t)==1)
       if(action(t)==1) % stay
           state(t+1) = 1 + ChS1(t+1);
           channel(t+1)=1;
       else
           state(t+1) = 3 + ChS2(t+1);
           channel(t+1)=2;
       end
    end
```

Q-learning, the code

```
% Next State (from the action I have taken)

if(state(t)==1)
   if(action(t)==1) % stay
       state(t+1) = 1 + ChS1(t+1);
       channel(t+1)=1;
   else
       state(t+1) = 3 + ChS2(t+1);
       channel(t+1)=2;
   end
end

if(state(t)==2)
   if(action(t)==1) % stay
       state(t+1) = 1 + ChS1(t+1);
       channel(t+1)=1;
   else
       state(t+1) = 3 + ChS2(t+1);
       channel(t+1)=2;
   end
end
```

```
if(state(t)==3)
   if(action(t)==1) % stay
       state(t+1) = 3 + ChS2(t+1);
       channel(t+1)=2;
   else
       state(t+1) = 1 + ChS1(t+1);
       channel(t+1)=1;
   end
end

if(state(t)==4)
   if(action(t)==1) % stay
       state(t+1) = 3 + ChS2(t+1);
       channel(t+1)=2;
   else
       state(t+1) = 1 + ChS1(t+1);
       channel(t+1)=1;
   end
end
```

Q-learning, the code

```
% Reward (consequence of the action)

switch state(t+1)
    case 1
        reward(t)=Reward_Good;
    case 2
        reward(t)=Reward_Bad;
    case 3
        reward(t)=Reward_Good;
    case 4
        reward(t)=Reward_Bad;
end


AcReward = AcReward + reward(t);

% Update Q-Table

Q_table(state(t),action(t))=(1-alfa)*Q_table(state(t),action(t)) + alfa*(reward(t) + gamma*max(Q_table(state(t+1),:)));
```

Activity

- Test case I and case II, and follow the Q-learning evolution
- Remove the ‘pause’ command from the two files, and increase the TimeHorizon to 500.
- For Case 2
  - Test different values of α given γ=0.9, and compare which gives you the best accumulated. reward.
  - For the best value of α, test different γ values. Which is the one that gives the best accumulated. Reward?
- Using Case 1 or Case 2, create Case 3, and play with different channels. Try:
  - the case where the two channels have the same parameters
  - the case where the two channels have the same prob o stay in the busy and idle state, but different transition probabilities (one that changes fast, and one that changes slowly)

  → Are the results, and Q-table values consistent?