Machine Learning for Networking

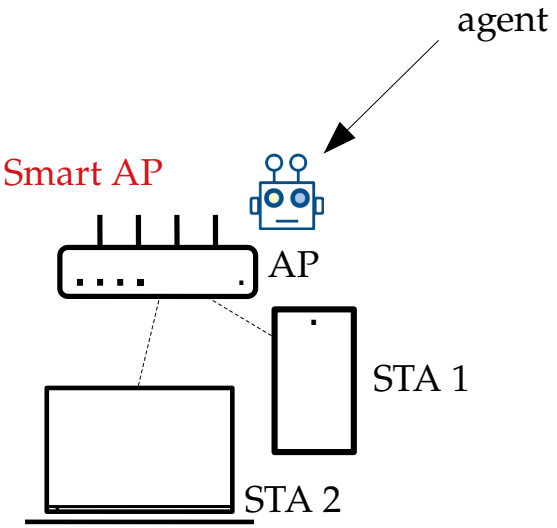
A dynamic Wi-Fi scenario

Session 5 – Searching (exploring) for a good solution

Boris Bellalta: boris.bellalta@upf.edu

Contents

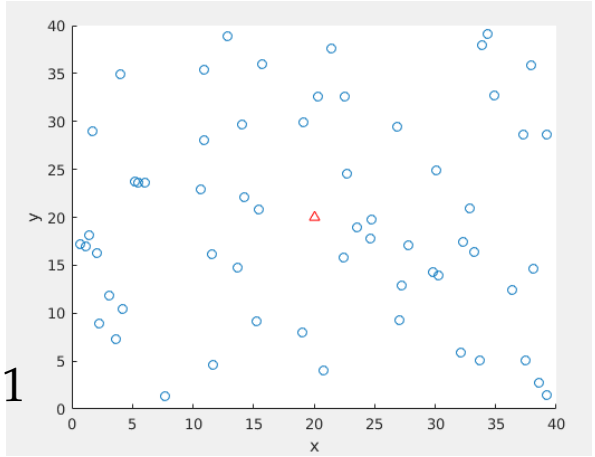
- A case study: A Wi-Fi network where the number and position of the stations change with time.
- Exploration & Exploitation trade-off
- A solver: ε-greedy



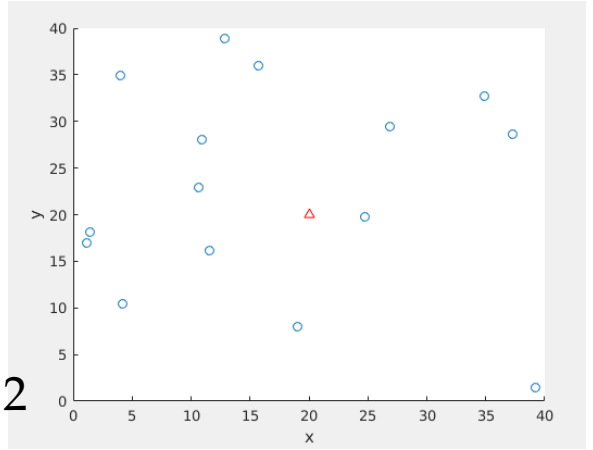
The case study

- We have a Wi-Fi network where the number of devices and their position change from time to time.

t=1



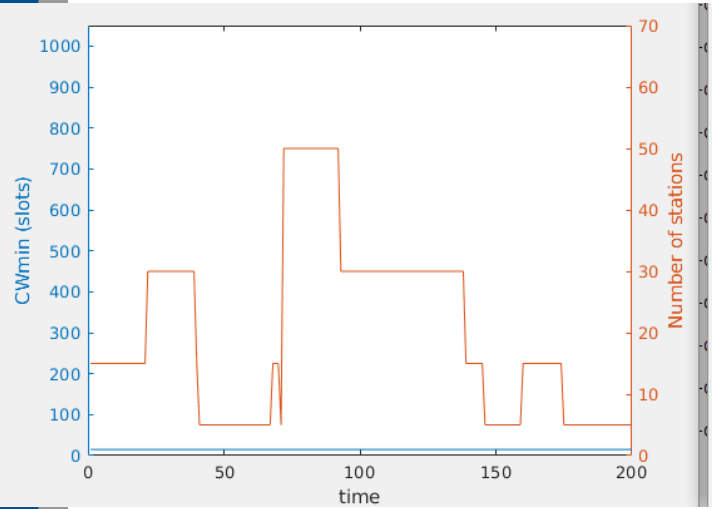
t=2

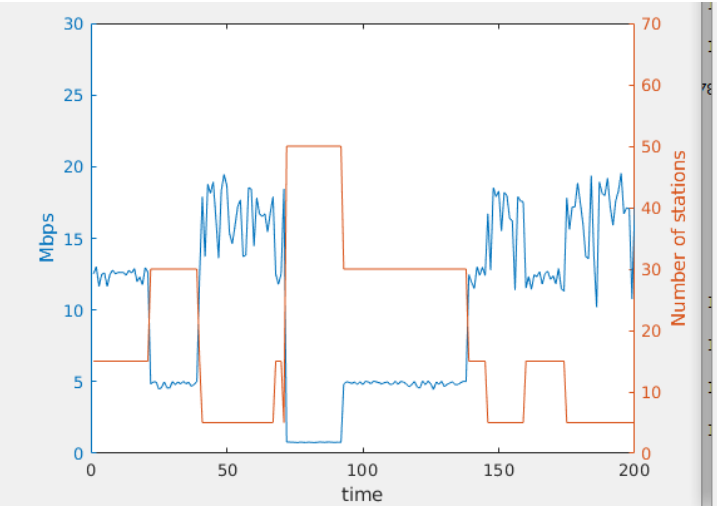


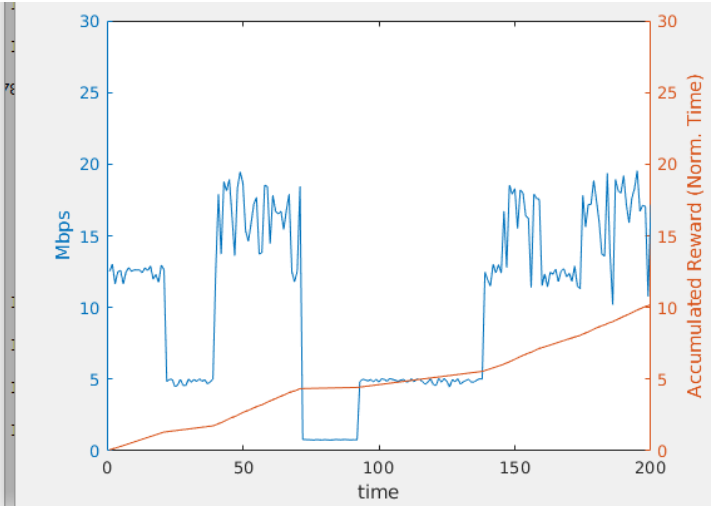
- What happens if we keep the same CWmin all the time?
  - Let's assume CWmin = CWmax

The case study

- What happens if we keep the same CWmin all the time?
- Pros: easy to implement | Cons: not adaptive to changing situations

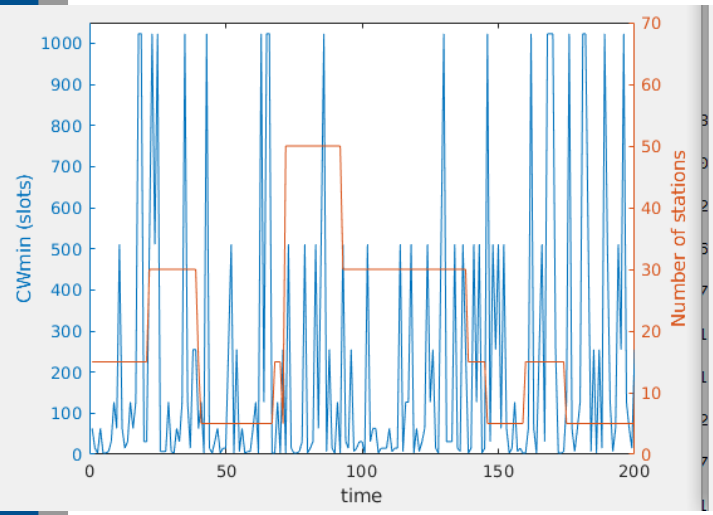


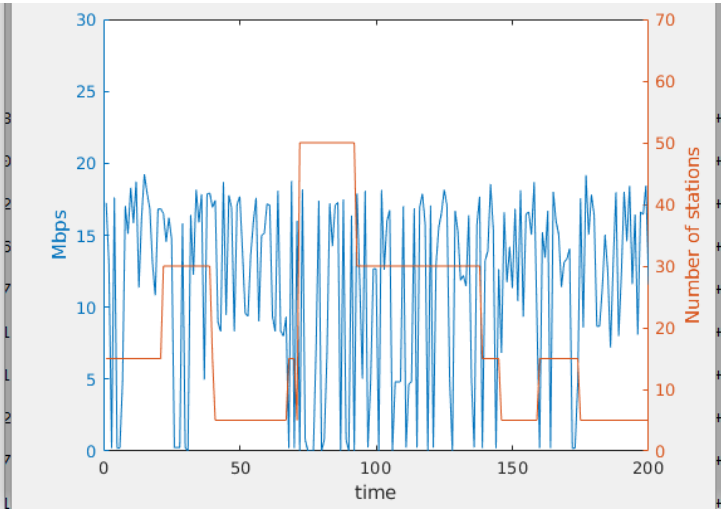


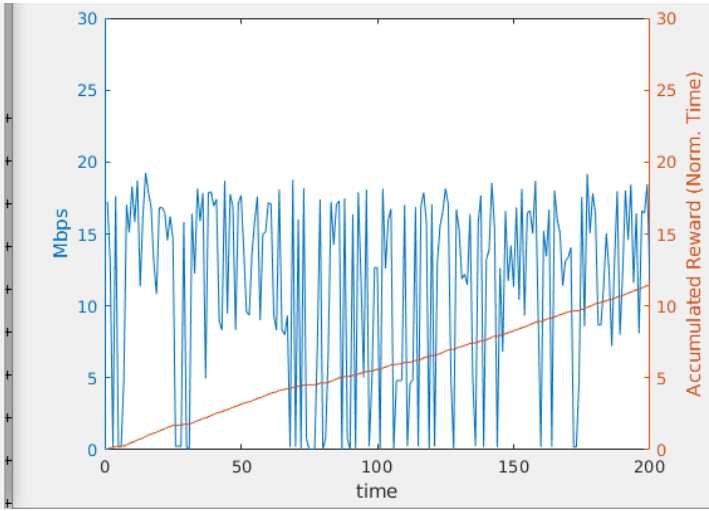


How to improve?

- We can use a ML agent placed to the AP to learn what is the best CWmin value. However, to start, let us allow the agent just tries random CWmin… we may be lucky!







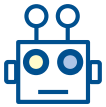
Pros: sometimes works | Cons: sometimes not → A kind of average

How to improve?

- We will assume the agent starts with ‘0’ knowledge about the relationship between the number of stations and the CWmin.
  - After the previous lectures, we know that adding more stations, increases the collision probability, and so reduces the network performance.
  - The agent does not know that. It has to find by itself a way to map:
    - CWmin values
    - Number of stations
    - Performance obtained

Picking Random values

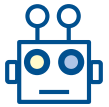
- Important: we are not learning
- However, picking random CWmin, allows us to gather useful information we can use:
  - Relationship between number of stations and CWmin value in terms of throughput
  - This is: we can learn the function f:
    - [S] = f(N,CWmin)



Exploration / Exploitation

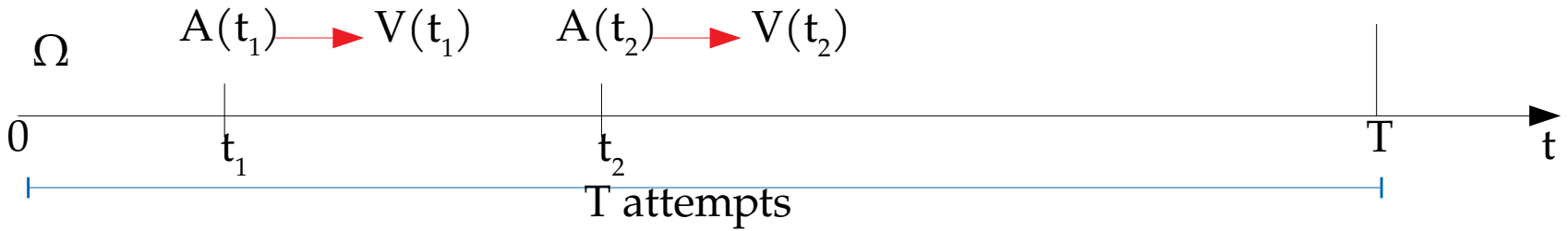
- Picking random values: required to explore (=learn)
- We need to consolidate what we have learned → exploitation

- **Example**: An AP has to select a channel to operate. It knows channel 1 will offer it a performance P_1, which is good but not excellent. AP ignores if another channel may offer it a higher performance. Should it explore other channels or use channel 1?
- The exploration-exploitation tradeoff: Why is it a tradeoff? Well, we have to choose between what we know, and the risk that we the new things we discover do not satisfy us. Also, there could be an extra cost to explore.



- **Strategy**: sequence of actions towards achieving a certain goal (i.e., maximize the network performance).
- **Environment**: well, everything...
- **Something obvious**: If we completely know the environment, to select the best strategy is easy, and we will achieve the goal. How to completely know an environment? If it is small, we can simply use brute force. Otherwise, no way, and those are the interesting cases.
- **Warning**: We can only completely learn stationary environments (which does not mean deterministic), or that remain stationary for a large enough period. If they change, all previously collected information may be useless.
- **Dynamism**: An environment may seem small, and so one may consider to first learn everything, and then just pick the best 'strategy'. However, what happens if the environment changes after you have learn it, or during the learning phase? We need to be able to adapt.

Exploration & Exploitation tradeoff



- Environment: Ω
- Set of actions: $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$
- Action selected at time t: $A(t) = a,\ a \in \mathcal{A}$
- Reward at time time t: $V(t) = \Omega(A(t))$

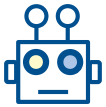
Goal: max accumulated reward

$$W = \sum_{t=0}^{T} V(t), \rightarrow \max_{A(t)}(W)$$

Find the sequence $A(t)$, t=1..T, that maximizes W.

How to find the best A(t), t=1...T? A solver: ε-greedy

- At $t = 0$, we set-up a Q-table of size K (one position per action), all set to 0. We also set $n = 0$, and $W = 0$.
  - The agent selects an action from $\mathcal{A}$ uniformly at random: $P(A(t) = a) = \frac{1}{K}$
  - We obtain the corresponding reward $V(0) = \Omega(A(0))$
  - We increase the number of attempts we take action a: $n_a = n_a + 1 = 1$
  - We update the Q-table: $Q(A(0)) = (Q(A(0)(n_a - 1) + V(0))/n_a = V(0)$
  - We increase the accumulated reward: $W = W + V(0) = V(0)$
- At $t > 0$, repeat, until $t = T$:
  - At time t the agent decides:
    - To explore with prob.: ϵ. Next action is selected uniformly at random from $\mathcal{A}$
    - To explot with prob.: $1 - \epsilon$. Next action $A(t) = \text{argmax}_{a \in \mathcal{A}} Q(a)$
  - We obtain the corresponding reward $V(t) = \Omega(A(t))$
  - We increase the number of attempts we take action a: $n_a = n_a + 1$
  - We update the Q-table: $Q(A(t)) = (Q(A(t)(n_a - 1) + V(t))/n_a$ → We can simply use the last reward, or a different approach
  - We increase the accumulated reward: $W = W + V(t)$



Exercise

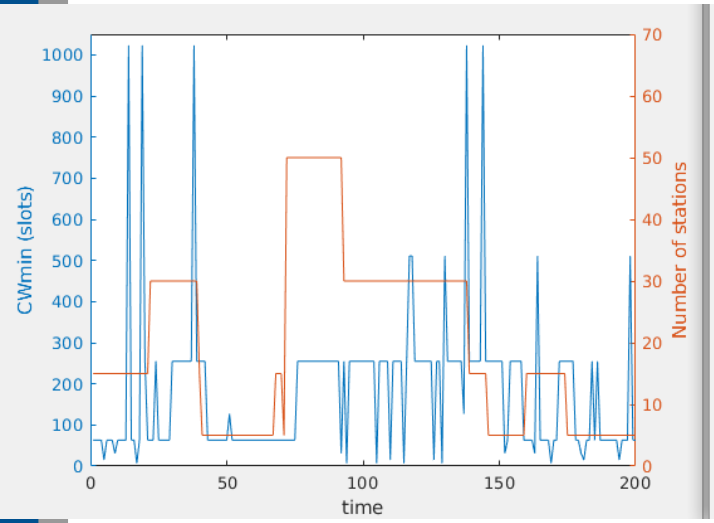
- Implement ε-greedy in the Example5 code
- Does it work?
- To execute the code: example5(policy,seed)
  - Policy = 1 (fixed); 2 (random); 3 (e-greedy)
  - seed: use the same in all cases, to compare

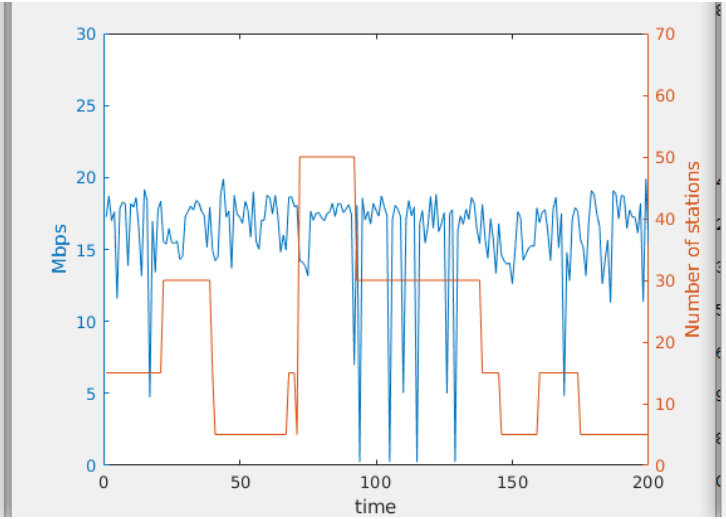
```
function ExampleLecture5(Policy,seed)

rng(seed);

%Policy = 1;

SimHorizon=200;

ProbChangeNumContenders = 0.1;

N_values = [5 15 30 50];

N=zeros(1,SimHorizon);
N(1) = N_values(randi(length(N_values),1));
%N(1)=15;


for i=1:SimHorizon
    if(i> 20 && rand < ProbChangeNumContenders)
        N(i+1) = N_values(randi(length(N_values),1));
    else
        N(i+1)=N(i);
    end
end

CWmin_values = [3 7 15 31 63 127 255 511 1023];
```

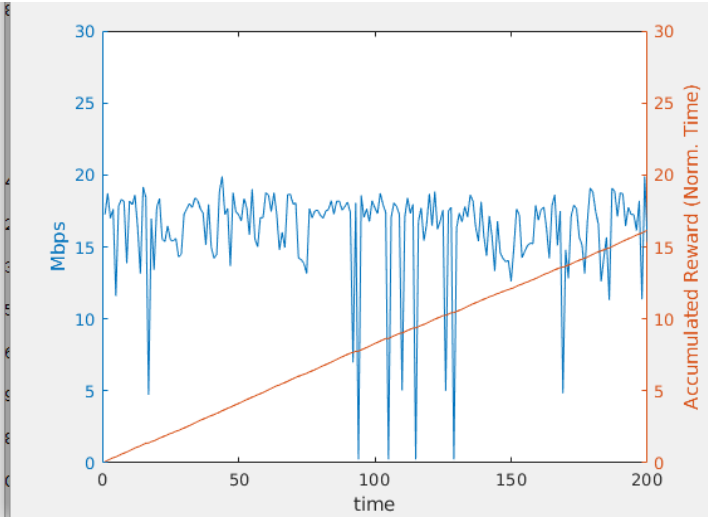
```
% ------ To implement epsilon greedy
epsilon=0.1;
Q_EG=zeros(1,length(CWmin_values)); % Q-table
n_a = zeros(1,length(CWmin_values)); % number of times action a is picked
a=1; % action picked in EG
% ------

for i=1:SimHorizon


    switch  Policy
        case 1  % Policy 1 - Fixed value
            CWmin(i) =15;

        case 2 % Policy 2 - I pick a random CWmin value

            CWmin(i) = CWmin_values(randi(length(CWmin_values),1));

        case 3 % Policy 3 - E-greedy
            if(i==1) % Exploration
                a = randi(length(CWmin_values),1);
                CWmin(1) = CWmin_values(a);
            else
                if(rand() < epsilon) % Exploration

                    % To be filled

                else % Exploitation

                    % To be filled
                end
            end
    end

    S(i)=WLANPerformance(N(i),CWmin(i));

    % Policy E-greedy
    if (Policy == 3)
          n_a(a)=n_a(a)+1; % Update times action a is picked
          Q_EG(a)=(S(i)+Q_EG(a)*(n_a(a)-1))/n_a(a); % Update Q-table
    end
```

ε-greedy

- We can observe how from time to time, the agent explores
  - Sometimes it finds a better ‘action’, and sometimes not







Practical Ways to improve ε-greedy

- Explore only when there are changes on the number of stations
- Reduce the exploration prob. (ε) as the time goes on.
  - Set ε at a high value (i.e., 1) when a change is detected.
  - Reduce it a every attempt: $\varepsilon_{t+1}=\varepsilon_t-\Delta$; with $\Delta=1/Ne$
  - Ne = number of iterations where we can explore

It's up to you to try!!