Machine Learning for Networking

Reinforcement Learning I

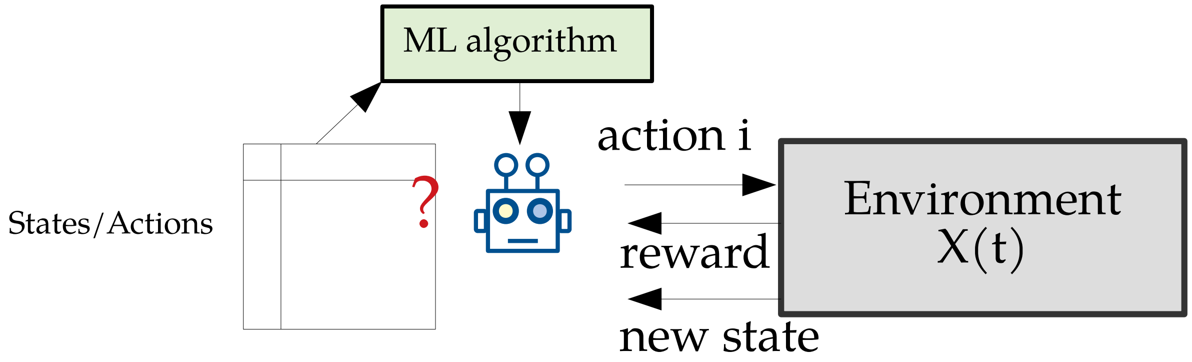
Session 6 – Multi-armed Bandits

Boris Bellalta: boris.bellalta@upf.edu

Contents

- Reinforcement Learning
- The channel selection problem
- Brute force: Check all channels for M episodes each
- Multi-armed bandits
  - ε-greedy
  - Thompson sampling
  - UCB

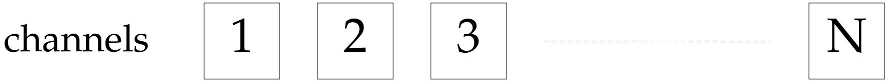
Reinforcement Learning

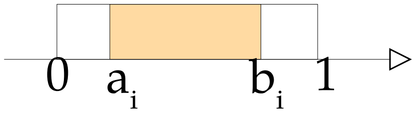


The agent learns by interacting with the environment.

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

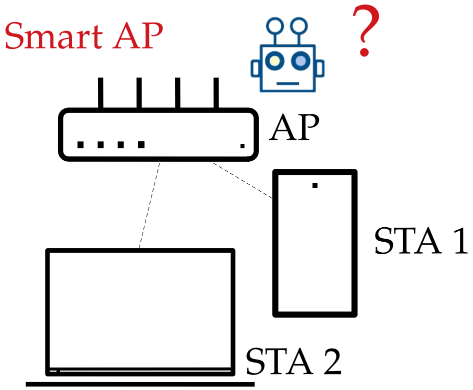
Channel selection problem





$a_i <= b_i$

$\rho(i)$ = occupancy channel i [0,1]

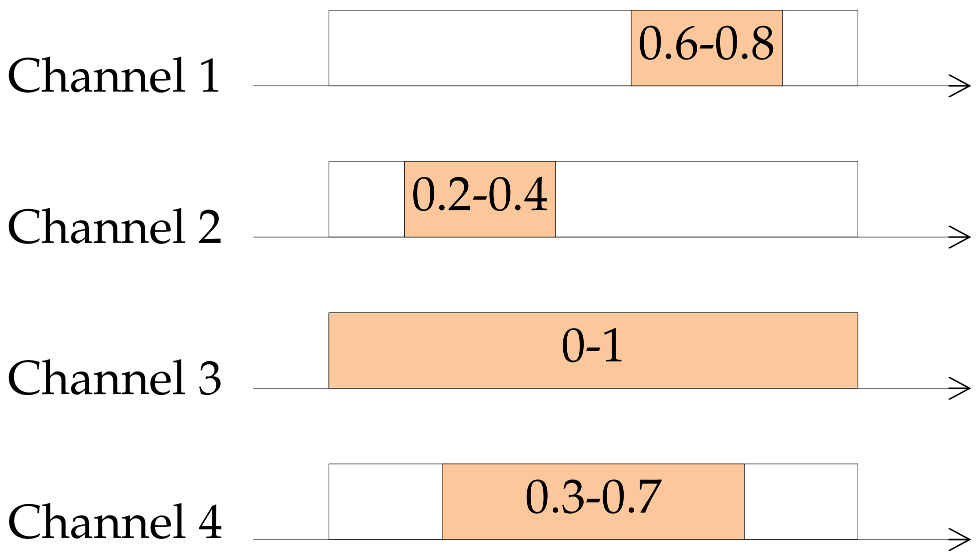


Reward using channel i: $V(i) = 1-\rho(i)$

Channel selection problem

- An example with 4 channels

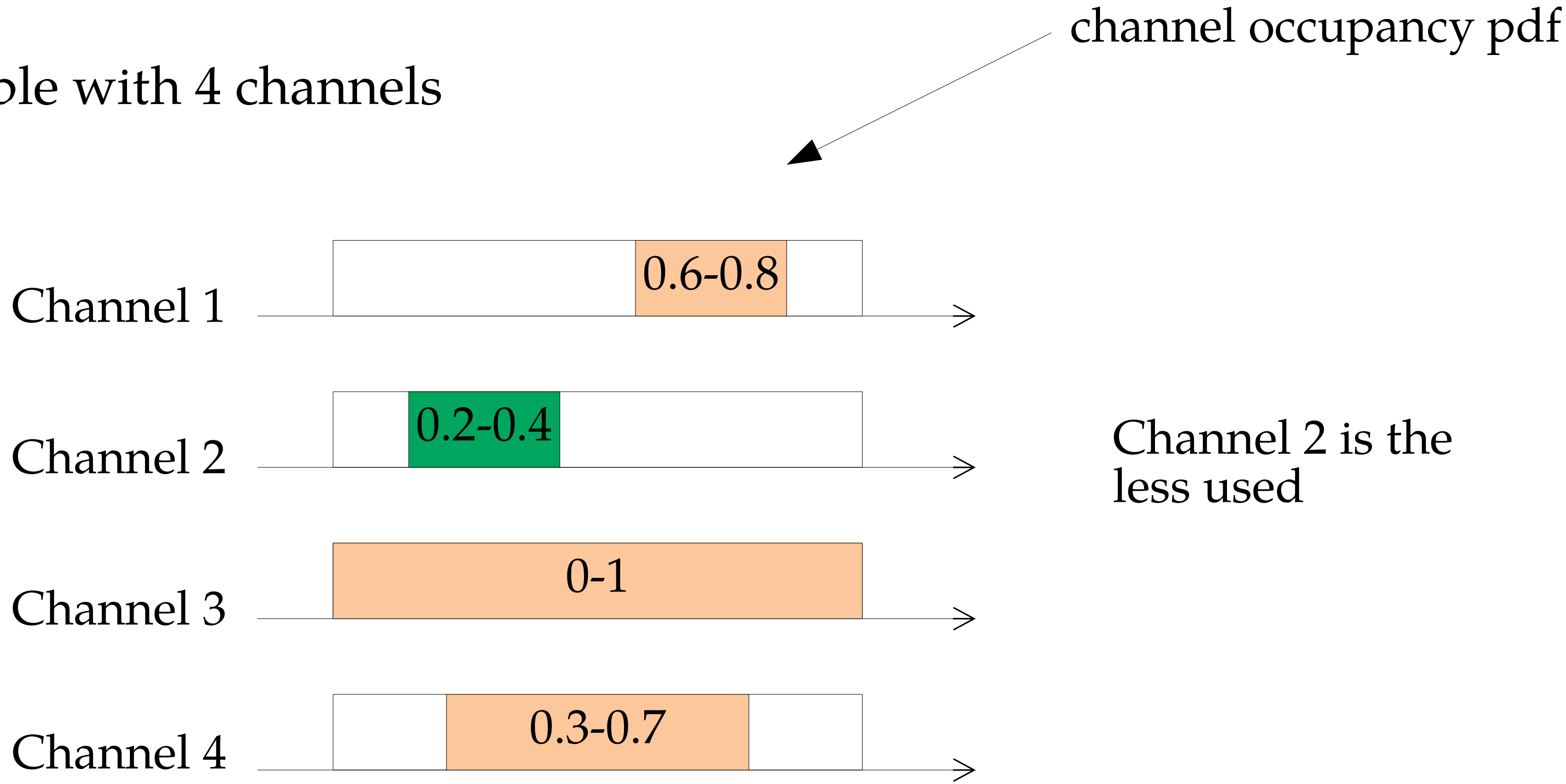
channel occupancy pdf



Which channel
would you pick?

Channel selection problem

- An example with 4 channels



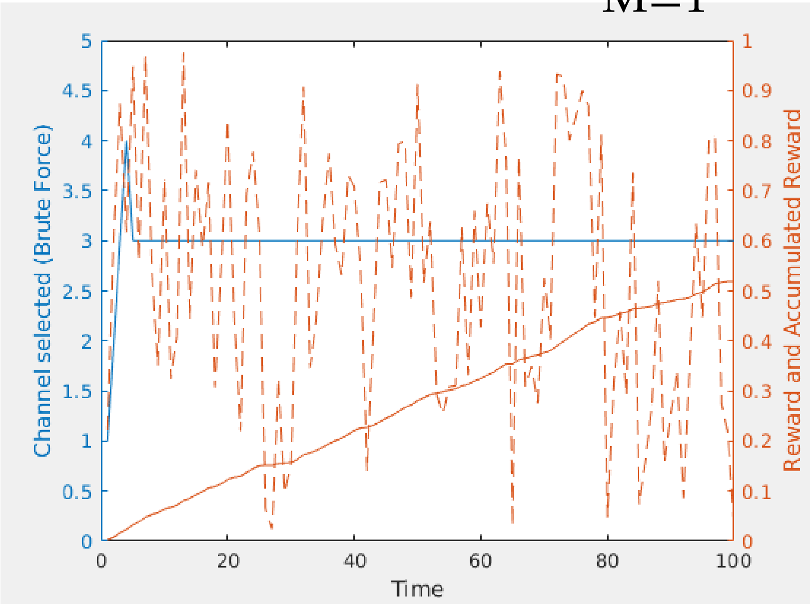
Channel 2 is the less used

Brute force: I check all channels

```
M=1;
Q_BF = zeros(1,N);
R=zeros(1,TimeHorizon);
ch_selected = zeros(1,TimeHorizon);

% I check all the channels

time_to_check_all = 1;

for n=1:N
    for m=1:M
        ch_selected(time_to_check_all) = n;
        rho = ChannelOccupancy(a(n),b(n));
        R(time_to_check_all) = 1-rho;
        Q_BF(n)=Q_BF(n) + (1-rho);
        time_to_check_all = time_to_check_all + 1;
    end
    Q_BF(n)=Q_BF(n)/M;
end

% I pick the one with highest reward

disp(Q_BF);

[~,n_best]=max(Q_BF);

for t=time_to_check_all:TimeHorizon
    ch_selected(t) = n_best;
    rho = ChannelOccupancy(a(n_best),b(n_best));
    R(t) = 1-rho;
end
```

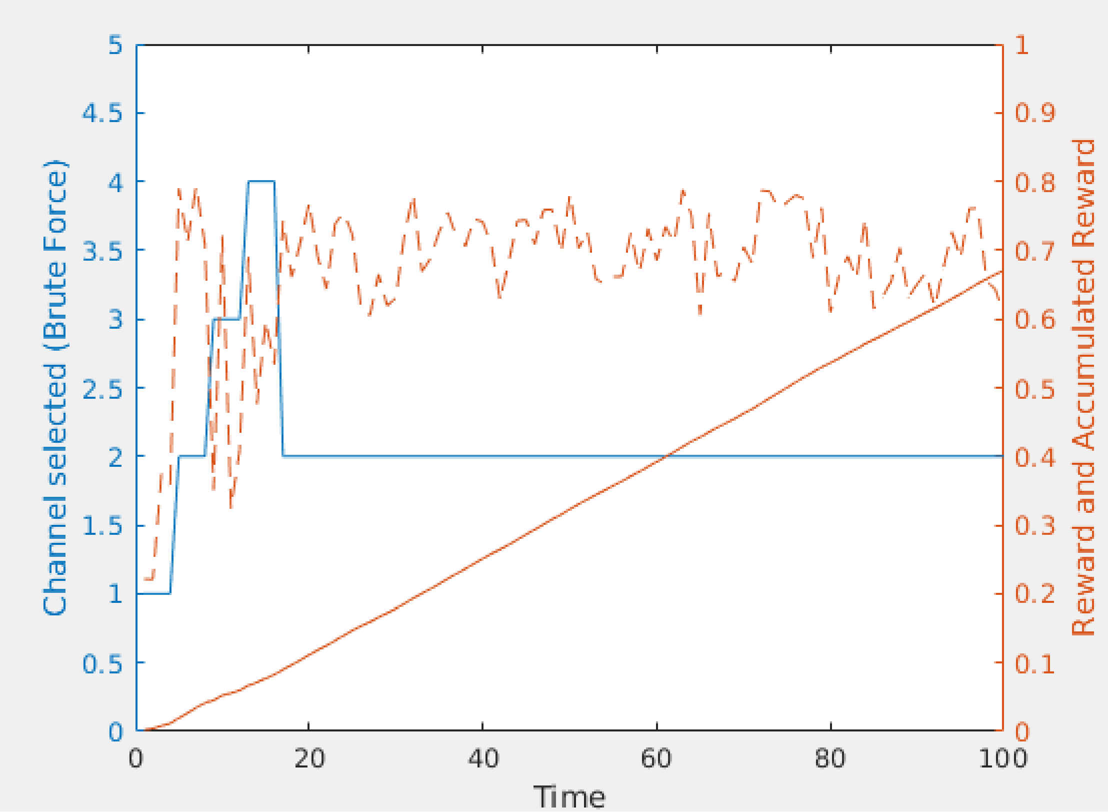
How many 'samples' I take per channel.

M=1



The agent keeps using the 'best' option found before.

Brute force: I check all channels



M=4

3.1. The Multi-Armed Bandits Framework

In the online learning literature, several MAB settings have been considered such as stochastic bandits [25, 26, 27], adversarial bandits [28, 29], restless bandits [30], contextual bandits [31] and linear bandits [32, 33], and numerous exploration-exploitation strategies have been proposed such as *ε-greedy* [34, 27], *upper confidence bound* (UCB) [26, 35, 36, 27], *exponential weight algorithm for exploration and exploitation* (EXP3) [28, 27] and *Thompson sampling* [25]. The classical multi-armed bandit problem models a sequential interaction scheme between a learner and an environment. The learner sequentially selects one out of K actions (often called *arms* in this context) and earns some rewards determined by the chosen action and also influenced by the environment. Formally, the problem is defined as a repeated game where the following steps are repeated in each round $t = 1, 2, \ldots, T$:

1. The environment fixes an assignment of rewards $r_{a,t}$ for each action $a \in [K] \stackrel{\text{def}}{=} \{1, 2, \ldots, K\}$,
2. the learner chooses action $a_t \in [K]$,
3. the learner obtains and observes reward $r_{a_t,t}$.

MABs → Decide what action to take under uncertainty

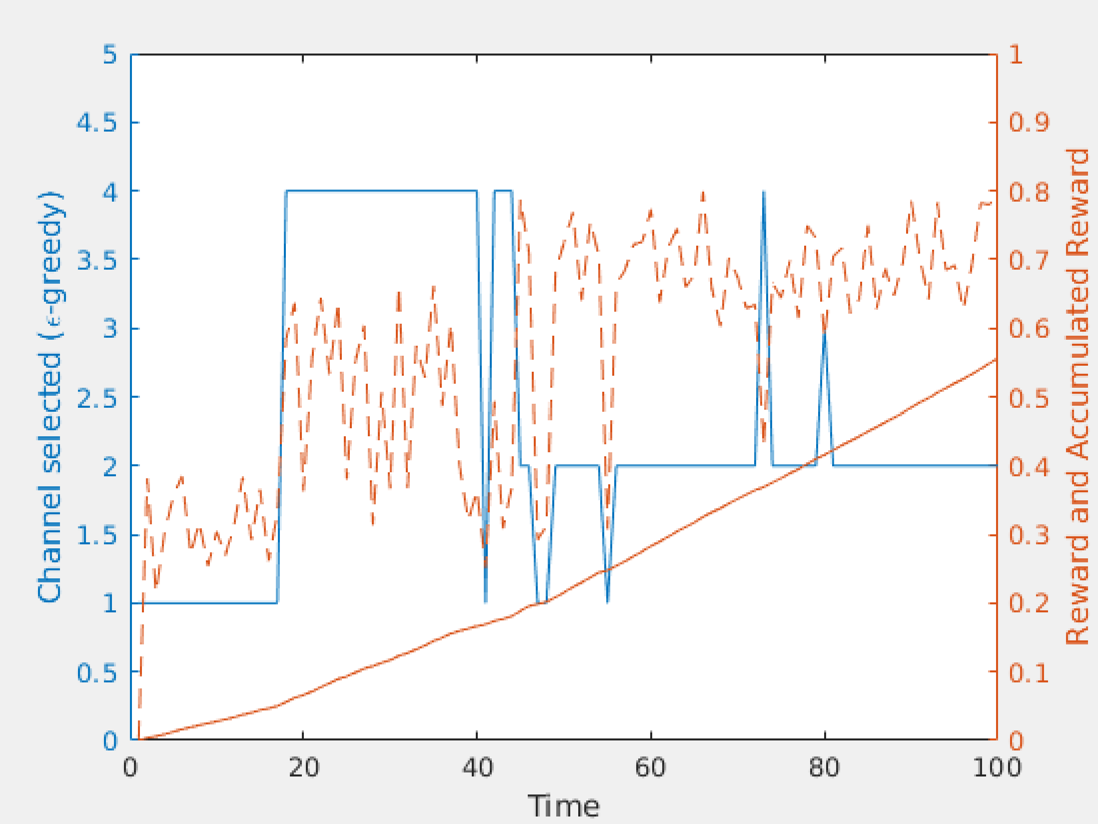
Epsilon-greedy

```
for t=2:TimeHorizon

    if(rand < epsilon) % explore
        n_sel=randi(N,1);
    else % exploit
        [~,n_sel]=max(Q_EG);
    end

    ch_selected_EG(t) = n_sel;
    n_times_ch_selected_EG(n_sel)=n_times_ch_selected_EG(n_sel)+1;
    rho = ChannelOccupancy(a(n_sel),b(n_sel));
    R_EG(t) = 1-rho;
    Q_EG(n_sel)=(Q_EG(n_sel)*(n_times_ch_selected_EG(n_sel)-1) + (1-rho))/n_times_ch_selected_EG(n_sel);

    epsilon = epsilon - Delta;
end
```

The exploration-exploitation tradeoff

To reduce the exploration as the time goes on
(assuming we have already learn, and so it's time to exploit)

Epsilon Greedy



Thompson Sampling

- We try to improve how we explore.
- We assume the reward of each action follows a Gaussian distribution
- For each action, we estimate its mean, and adjust the variance proportionally to the number of times we played that action in the past
  - Actions more played have less variance, and the opposite
  - **Idea**: the variance 'captures' how much confident I'm on the reward I will obtain when playing a certain arm.
  - If I have played many times a certain arm, I will have a better knowledge.
    - *True if the system is stationary!*

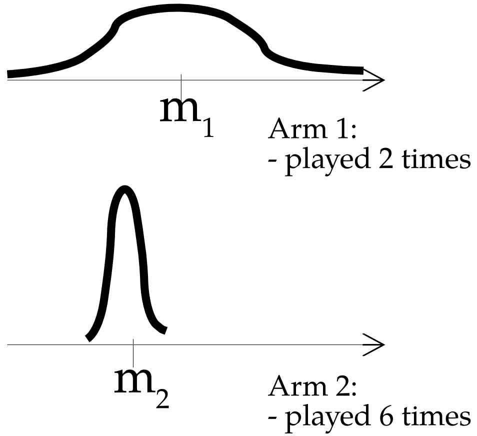
Thompson sampling

```
for t=2:TimeHorizon

    Sampling_arms=zeros(1,N);
    for i=1:N
        m = Q_TS(i);
        sigm = (1/(n_times_ch_selected_TS(i)+1));
        Sampling_arms(i)=m + sigm.*randn();
    end

    [~,n_sel]=max(Sampling_arms);

    ch_selected_TS(t) = n_sel;
    n_times_ch_selected_TS(n_sel)=n_times_ch_selected_TS(n_sel)+1;
    rho = ChannelOccupancy(a(n_sel),b(n_sel));
    R_TS(t) = 1-rho;
    Q_TS(n_sel)=(Q_TS(n_sel)*(n_times_ch_selected_TS(n_sel)-1) + (1-rho))/n_times_ch_selected_TS(n_sel);

end
```

mean

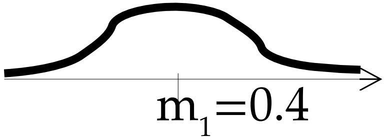
Std. dev

Gaussian RV



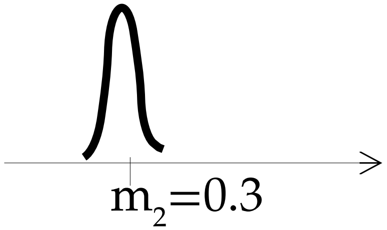
TS includes the ‘consolidated’ knowledge of a certain action when deciding to take it or not: playing arm 1 may give me higher reward, but I have more uncertainty

Thompson sampling



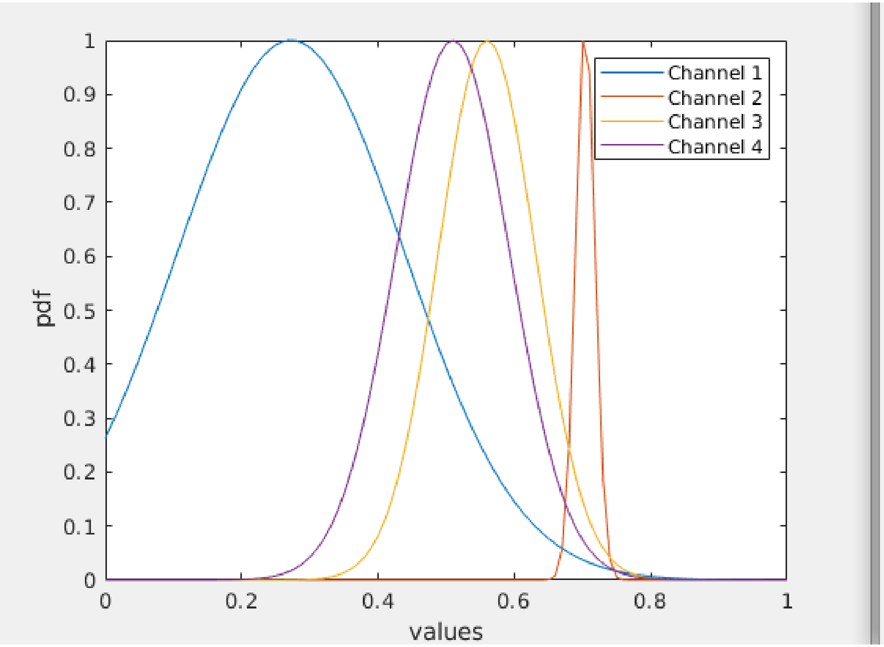
→ 0.6; 0.1; 0.05; 0.4; 0.8; …

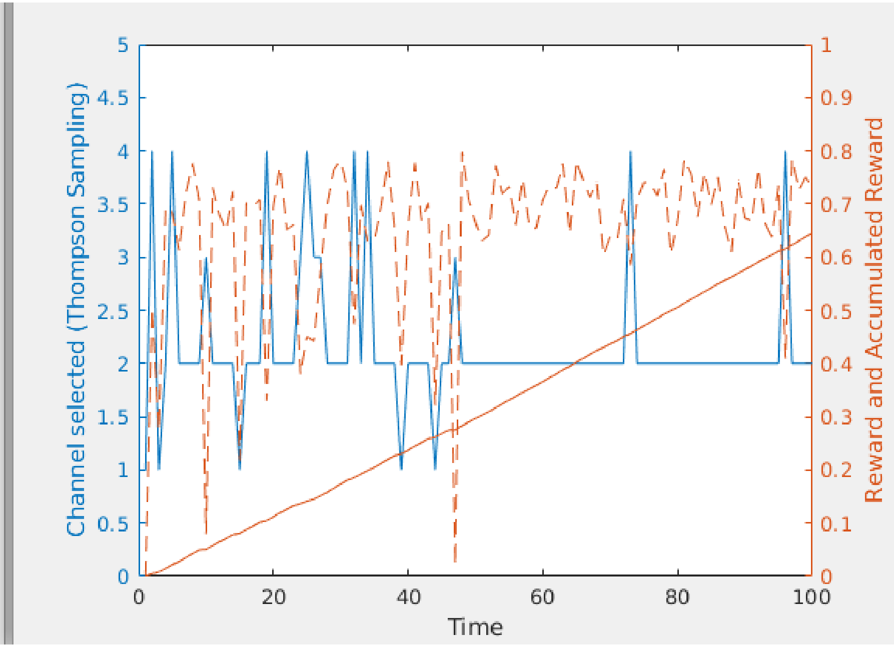
I may get more,
but I risk to get less



→ 0.2; 0.4; 0.3; 0.3; 0.4; …

Thompson Sampling





The pdfs used by TS to select the channel

Upper Confidence Bound

- The upper confidence bound (UCB) action-selection strategy is based on the principle of optimism in face of uncertainty.
- In each round, UCB selects the arm that is expected to give us the highest reward.
- Intuitively, UCB trades off exploration and exploitation as follows:
  - upon every time a suboptimal arm is chosen, the corresponding confidence bound will shrink significantly,
  - thus quickly decreasing the probability of drawing this arm in the future.
- However, after some time, it may try again arms that were before discarded.

UCB

```
% First Iteration

for i=1:N
    n_sel = i;
    ch_selected_UCB(n_sel) = n_sel;
    rho = ChannelOccupancy(a(n_sel),b(n_sel));
    R(i) = 1-rho;
    Q_UCB(n_sel)=Q_UCB(n_sel) + (1-rho);
    n_times_ch_selected_UCB(n_sel) = 1;
end

for t=N+1:TimeHorizon

    Sampling_arms=zeros(1,N);
    for i=1:N
        m = Q_UCB(i);
        Sampling_arms(i)= UCBx*m + sqrt(2*log(t)/n_times_ch_selected_UCB(i));
    end

    [~,n_sel]=max(Sampling_arms);

    ch_selected_UCB(t) = n_sel;
    n_times_ch_selected_UCB(n_sel)=n_times_ch_selected_UCB(n_sel)+1;
    rho = ChannelOccupancy(a(n_sel),b(n_sel));
    R_UCB(t) = 1-rho;
    Q_UCB(n_sel)=(Q_UCB(n_sel)*(n_times_ch_selected_UCB(n_sel)-1) + (1-rho))/n_times_ch_selected_UCB(n_sel);

end
```

A weight added by me, not part of the original UCB

Optimistic Function

Decreases with the number of times I play an arm, and increases with time.
→ after a while, I may re-play actions that I discarded in the past as they were not good (just in case)
→ Good option for non-stationary environments.

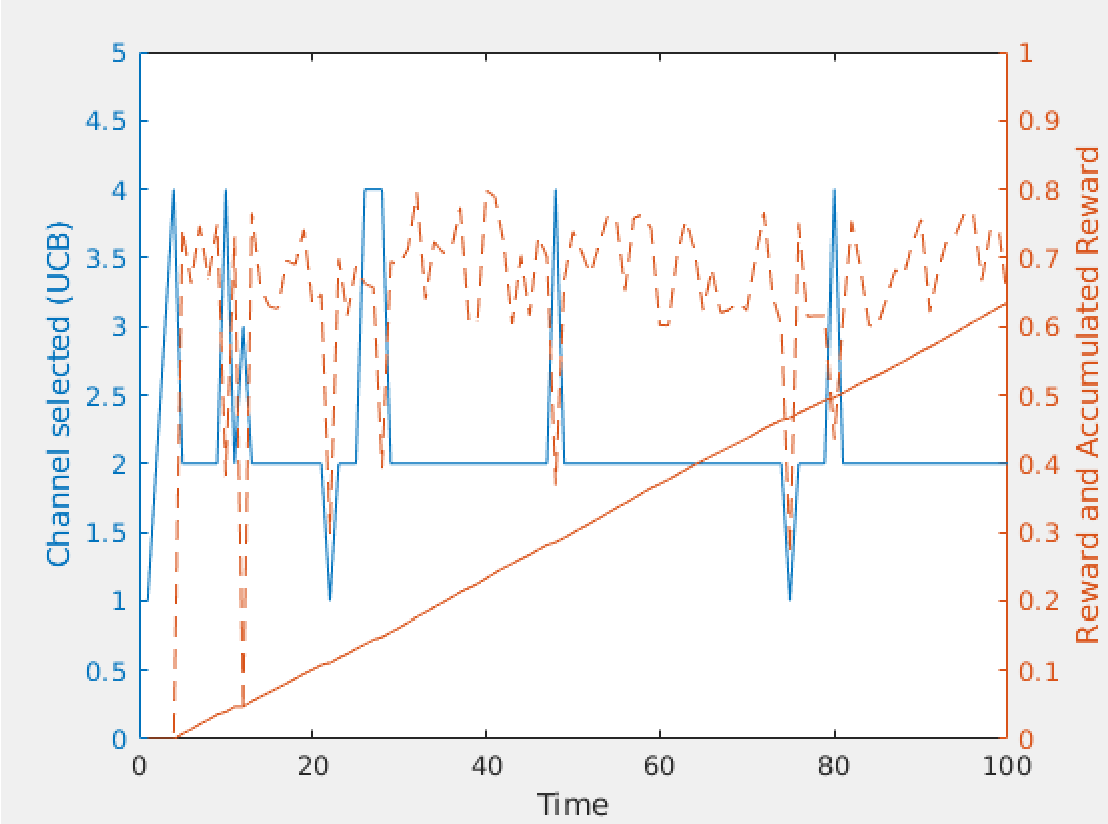
UCB

4 channels
(number of times
each channel has been selected)

Optimistic function

2	63	2	6
2.0746e+00	3.6964e-01	2.0746e+00	1.1978e+00
2	64	2	6
2.0779e+00	3.6732e-01	2.0779e+00	1.1997e+00
3	64	2	6
1.6992e+00	3.6788e-01	2.0810e+00	1.2015e+00
3	65	2	6
1.7017e+00	3.6559e-01	2.0842e+00	1.2033e+00
3	66	2	6
1.7043e+00	3.6335e-01	2.0873e+00	1.2051e+00
3	67	2	6
1.7067e+00	3.6115e-01	2.0903e+00	1.2068e+00
3	68	2	6
1.7092e+00	3.5900e-01	2.0933e+00	1.2086e+00
3	68	2	7
1.7116e+00	3.5951e-01	2.0963e+00	1.1205e+00

UCB



Channel selected (UCB)
Reward and Accumulated Reward
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
20
40
60
80
100

And the winner is?

```
Which is the best one? Brute Force (M=1) | EG | TS | UCB
    51.9072e+000    55.6482e+000    64.5326e+000    63.4469e+000
>>
```

Accumulated reward

Activity

- Download Example6.zip code
- Execute: example6(Nchannels, seed)
- Play with the number of channels, and seeds, and think about what you get.
- Go back to the case of 4 channels:
  - Test different 'occupancy' distributions of the channels, and see how the different algorithms respond.
- Homework:
  - Implement TS and UCB in the code of session 5 to select the CWmin.

Reading

- Wilhelmi, Francesc, Cristina Cano, Gergely Neu, Boris Bellalta, Anders Jonsson, and Sergio Barrachina-Muñoz. "**Collaborative spatial reuse in wireless networks via selfish multi-armed bandits.**" Ad Hoc Networks 88 (2019): 129-141. https://arxiv.org/pdf/1710.11403.pdf