Pricing & Social Influence

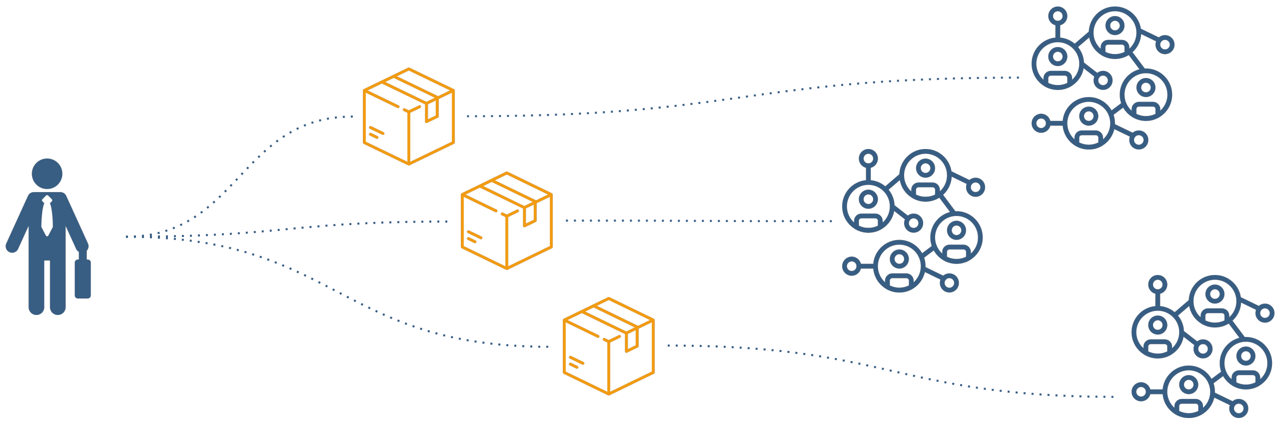
Data Intelligence Applications

Meta Samuele Metaj Stiven Salamino Manuel Serna Giuseppe

POLITECNICO MILANO 1863

1. The problem

The goal is modeling a scenario in which a seller is pricing some products and spends a given budget on social networks to persuade more and more nodes to buy the products.



POLITECNICO MILANO 1863

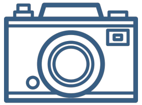
1. The products

Television

500, 690, 750, 850

Laptop

500, 690, 750, 850



Camera

500, 690, 750, 850

1. The features

Gender: Male/Female

Age: <18, 18-25, 26-34, 35-50, >50

Location: Africa, Asia, Europe, North America, South America, Australia

Interests: technology, sports, politics, science, economics, health

Authority: a node connected to many neighbors can be seen as an influencer

1. The social networks

Facebook

age

Twitter

interests

Instagram

authority

2. Greedy algorithm

`greedy_algorithm(graph, budget, k)`

Search the best set of seeds adding one at the time until we reach the budget

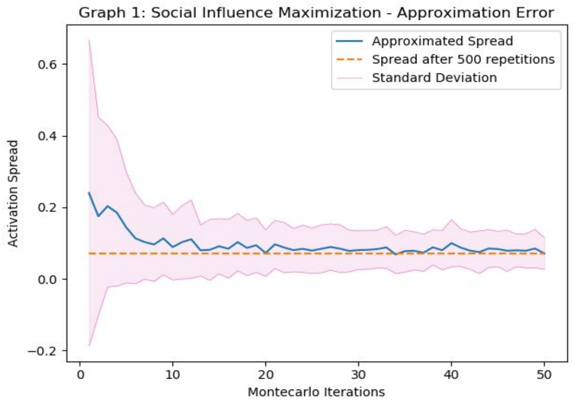
`montecarlo_sampling(seeds, max_repetitions)`

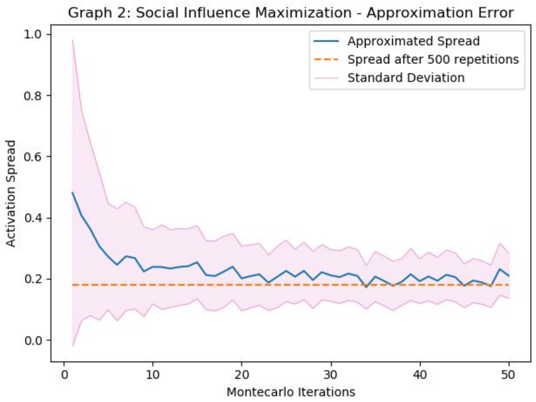
Given a set of seeds, evaluate the spread in the network through different simulations

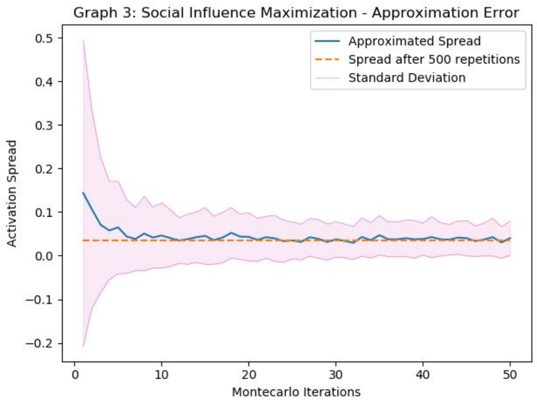
`approximation_error(graphs, budget, scale_factor, n_exp)`

Plot the result of running the greedy algorithm on the different networks

2. Greedy algorithm - Results







POLITECNICO
MILANO 1863

3. Greedy algorithm over all networks

Now we want to perform jointly social influence in the three social networks

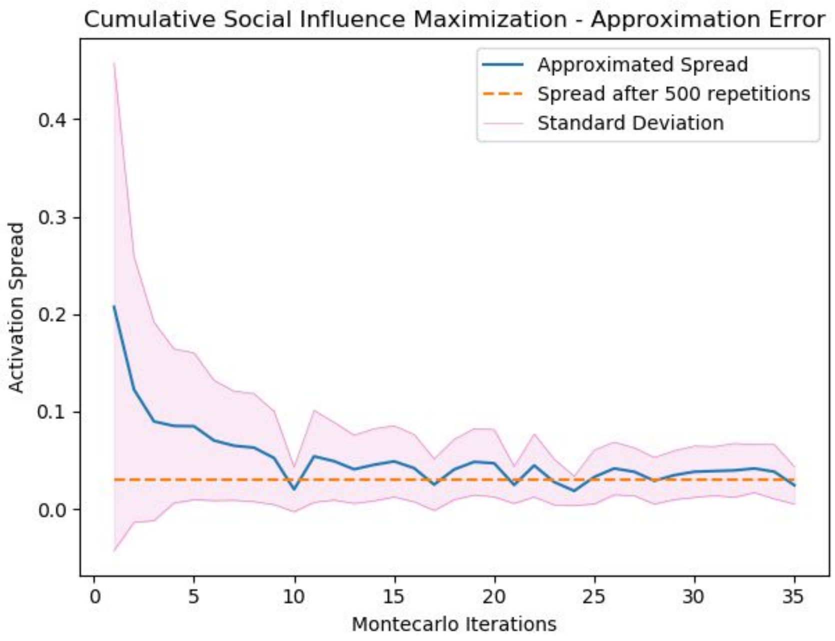
```
cumulative_greedy_algorithm(graphs, budget, k)
```

Until the budget is not finished, search in the different networks the seed that will provide the best marginal increase and add it to the set of seeds

```
cumulative_approximation_error(graphs, budget, scale_factor, n_exp)
```

Plot the results of running the cumulative greedy algorithm on the three social networks

3. Greedy algorithm over all networks - Results



POLITECNICO
MILANO 1863

4. Unknown activation probabilities

We consider the adjacency matrix as the true values to which we want to converge

We model each edge as a Bernoulli variable and we define a Beta distribution over them

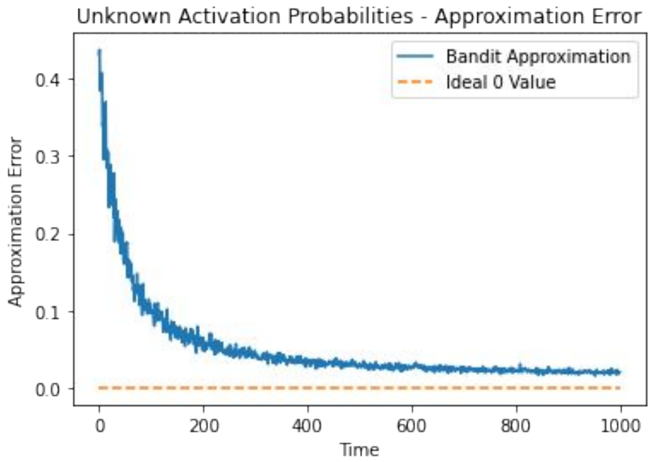
```
choose_seeds(graph, budget, epsilon, simulations)
```

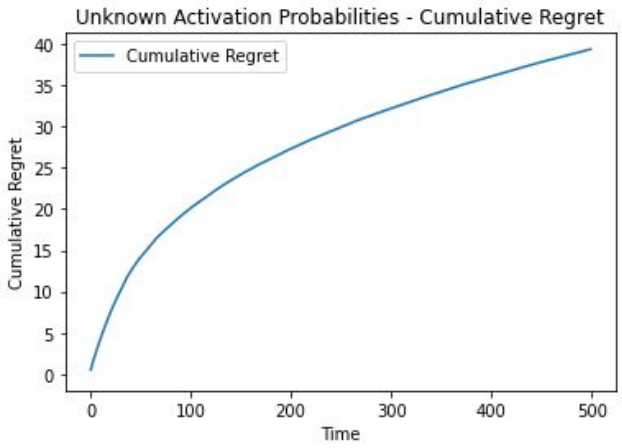
We use the greedy algorithm to select the set of seeds. The parameter epsilon allows to regulate the exploration/exploitation dilemma

```
influence_episode(seeds, true_graph.adj_matrix)
```

Collect a sample for each edge and observe which edges will activate, updating the Beta parameters

4. Unknown activation probabilities - Results





POLITECNICO
MILANO 1863

5. Cumulative revenue maximization

We assume to have an unique seasonal phase and that the activations of the graph are known

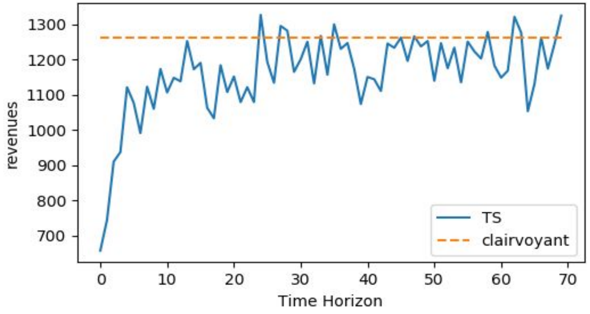
`TS_Learner(n_arms, arms)`

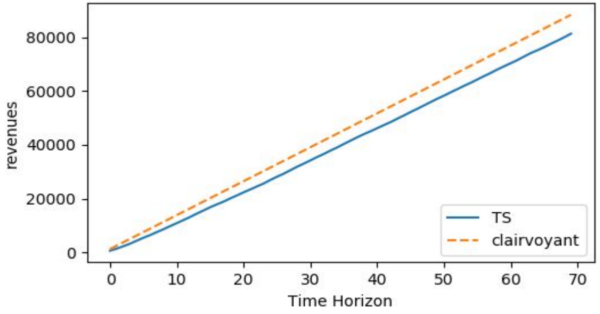
Builds a class to manage the Thompson Sampling operations: the choice of the arm to pull and, given the rewards, the update of the distributions.

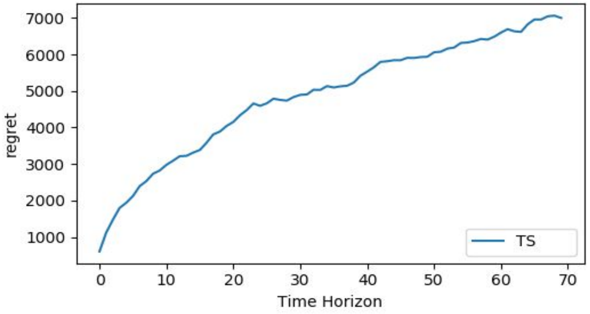
`Environment(n_arms, probabilities, horizon)`

Simulates the environment where the arms are: it gives the rewards based on the real probabilities for the event to occur.

5. Cumulative revenue maximization - Results(1)

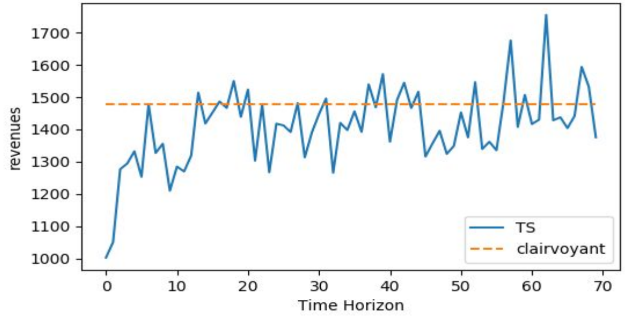


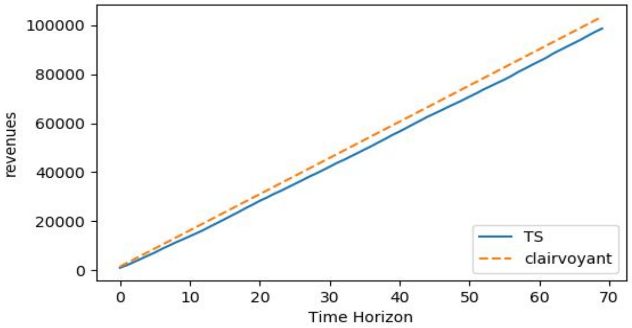


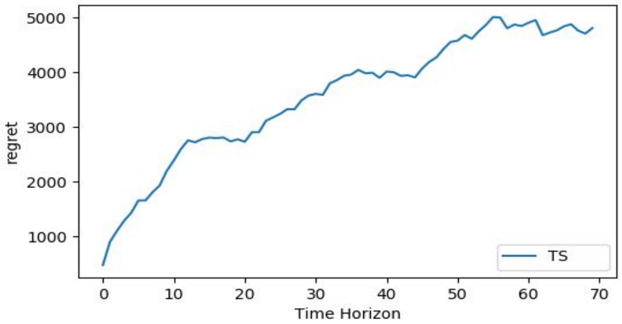


POLITECNICO
MILANO 1863

5. Cumulative revenue maximization - Results(2)

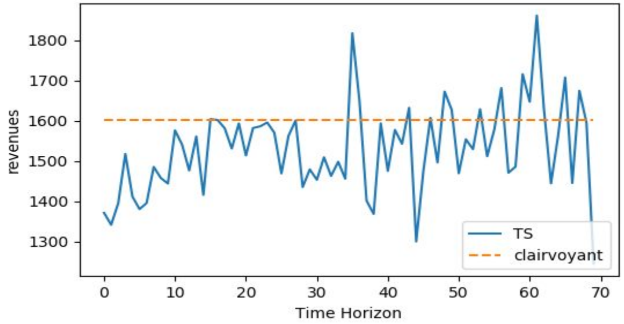


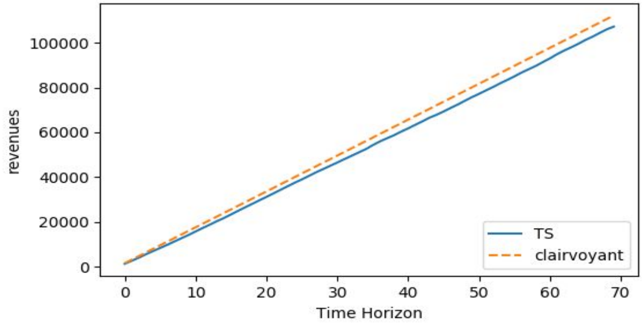


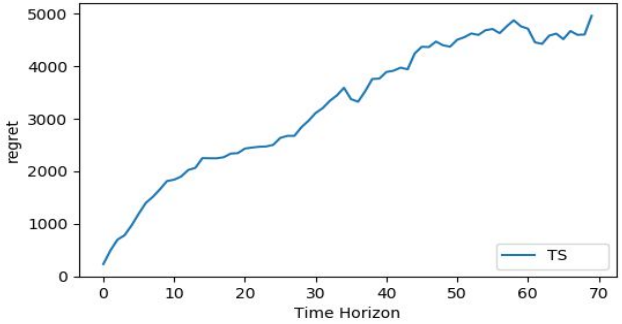


POLITECNICO
MILANO 1863

5. Cumulative revenue maximization - Results(3)







POLITECNICO
MILANO 1863

6. Pricing with seasonal phases

15
november

01
december

25
december

15
january

1st season

BEFORE CHRISTMAS

2nd season

CHRISTMAS

3rd season

AFTER CHRISTMAS

POLITECNICO
MILANO 1863

6. Pricing with seasonal phases

We consider now three different phases with abrupt transitions and we still assume to know the activation probabilities of the social networks

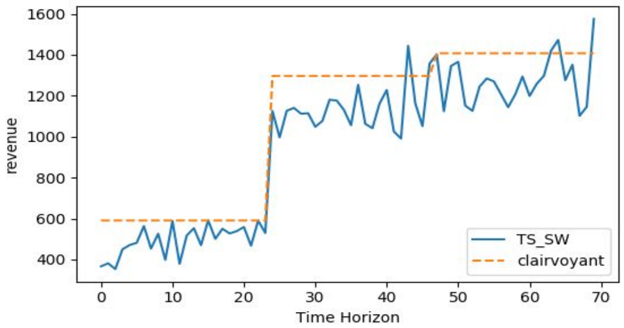
`SWTS_Learner(n_arms, arms, window_size, horizon)`

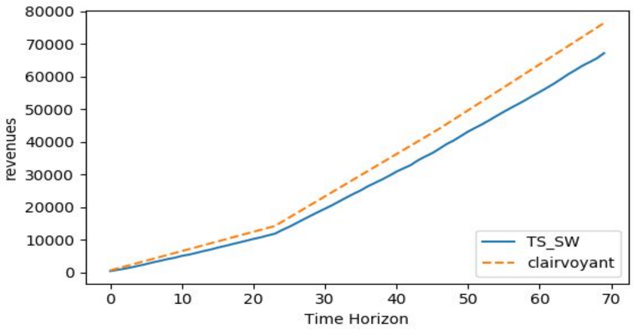
Builds the class to manage the operation of the Thompson Sampling with Sliding Window: the choice of the arm to pull and, given the rewards, the update of the distributions, taking into account only the rewards obtained inside the Sliding Window.

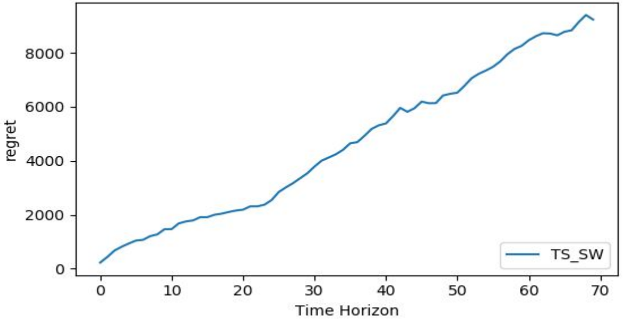
`Non_Stationary_Environment(n_arms, probabilities, horizon)`

Simulates the environment of the arms: it gives the rewards based on the real probabilities for the event to occur, taking into account the phase in that specific moment.

6. Pricing with seasonal phases - Results(1)

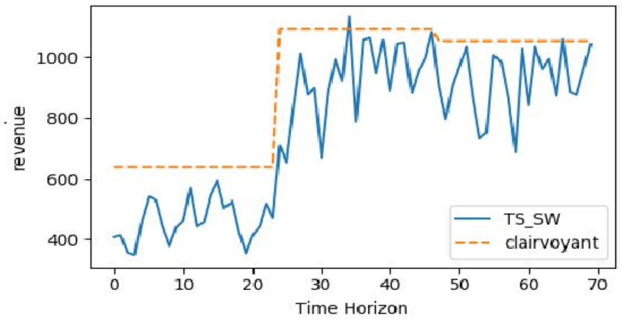


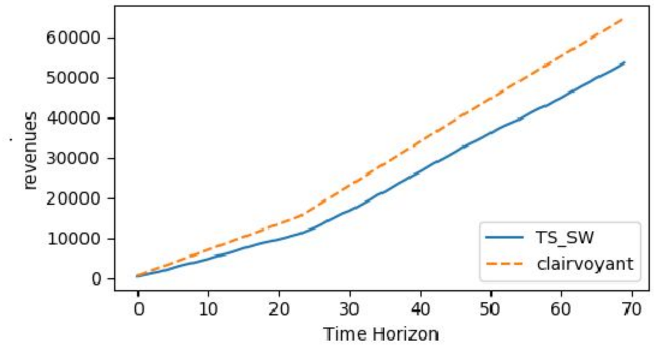


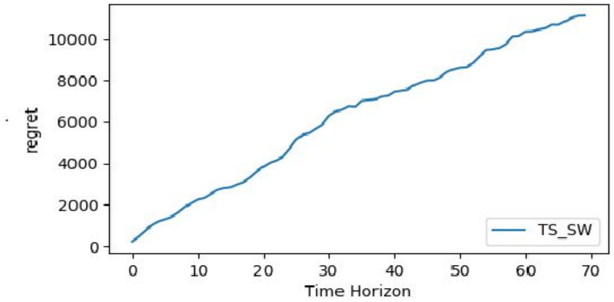


POLITECNICO
MILANO 1863

6. Pricing with seasonal phases - Results(2)

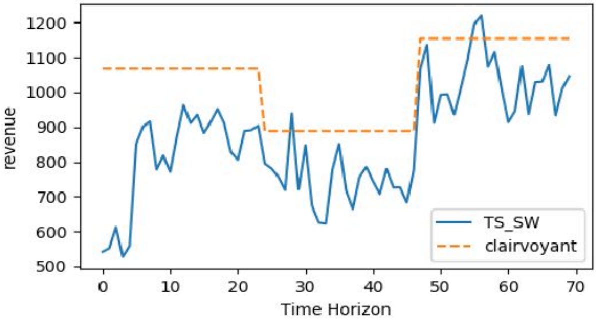


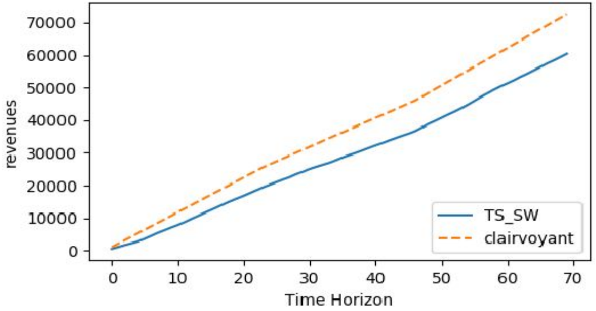


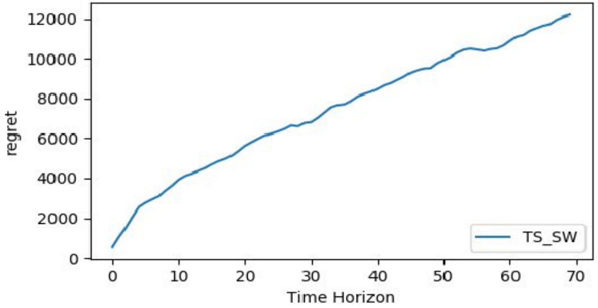


POLITECNICO
MILANO 1863

6. Pricing with seasonal phases - Results(3)





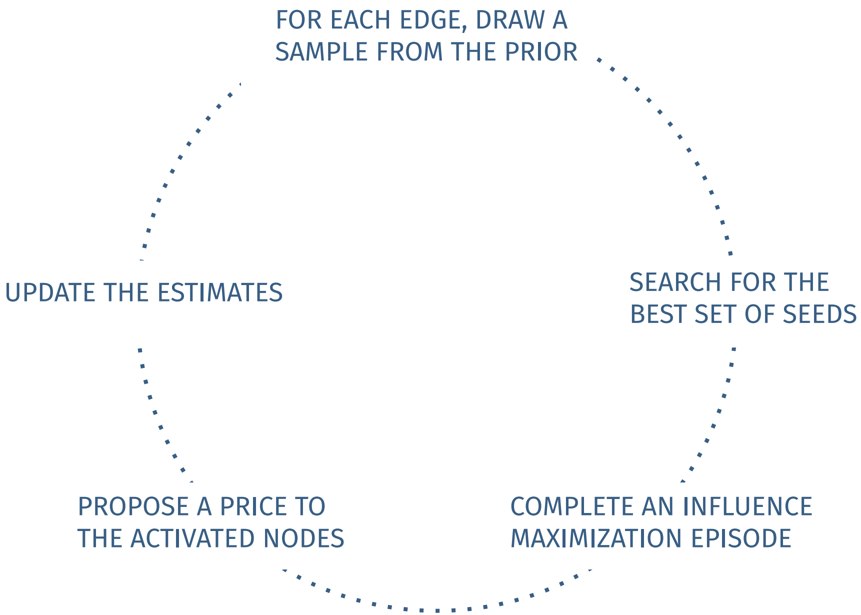


POLITECNICO MILANO 1863

7. Pricing and social influence learning

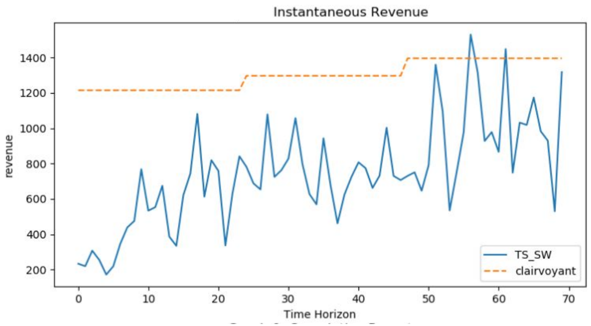
Now we cannot consider the activation probabilities as known a priori

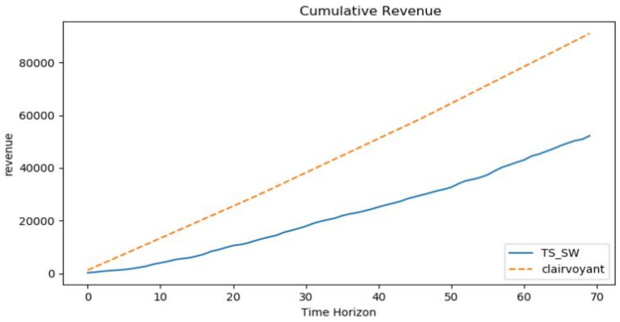
The seller needs to learn both the activation probabilities and the conversion rate curves simultaneously

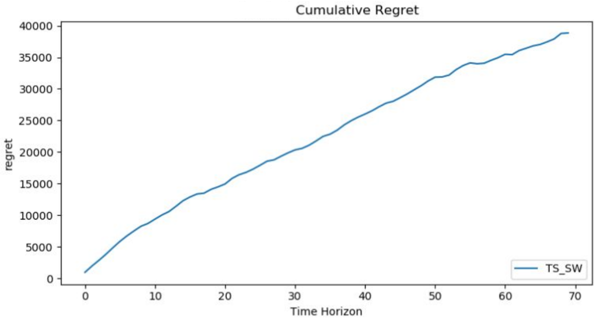


POLITECNICO MILANO 1863

7. Pricing and social influence learning - Results(1)

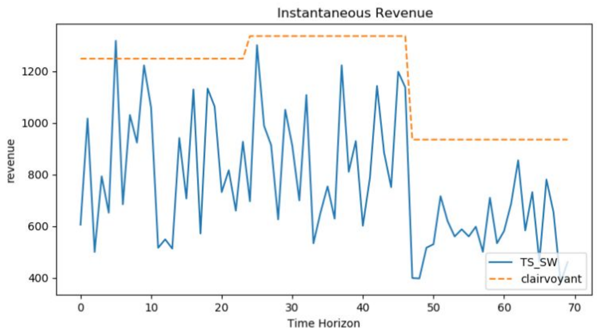


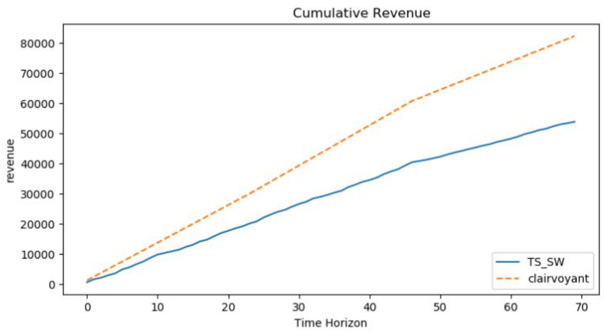


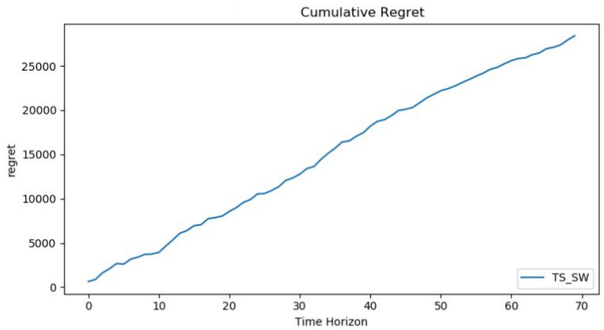


POLITECNICO
MILANO 1863

7. Pricing and social influence learning - Results(2)

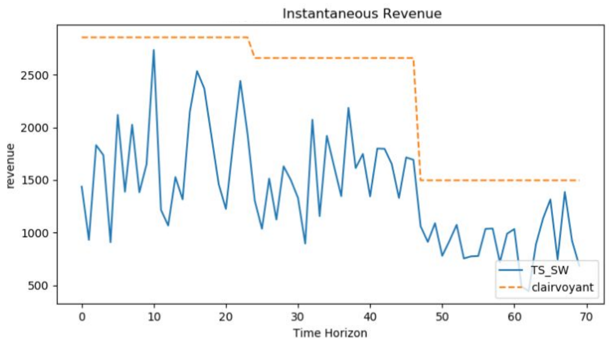


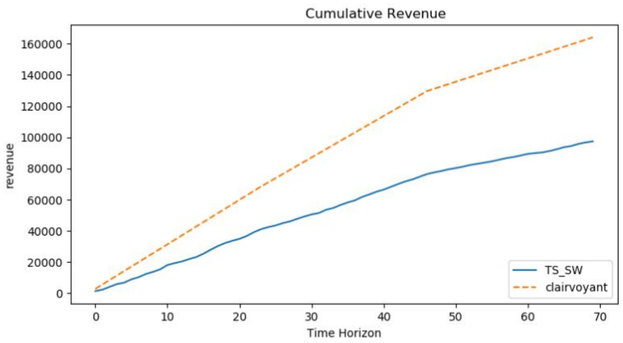


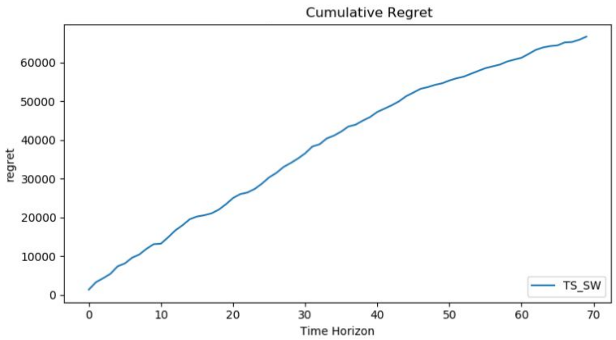


POLITECNICO
MILANO 1863

7. Pricing and social influence learning - Results(3)







POLITECNICO
MILANO 1863