# KDD Challenge

Team 9
Bao Long Vu
BELAID Mohamed Karim
Van Bach Nguyen

Universität Passau 2019-07-16

## Content

- Introduction to KDDCup
- Genetic Algorithm
- Bayesian Optimisation
- Sequence Breaking + Q-Learning

## Introduction to the KDD Cup

- Motivation: Malaria is dangerous disease in Africa. Human are taking 2 interventions to prevent it:
  - Insecticide spraying (IRS)
  - Distributing bed nets (ITN)
- But people don't know how much percent of the region's area they should spray the medicines.
- The problem: given an simulated environment, find a policy, which is 5 tuples of (IRS, ITN) so that they yield the best reward from the environment:
  - Find: $(IRS_1, ITN_1)$, $(IRS_2, ITN_2)$, $(IRS_3, ITN_3)$, $(IRS_4, ITN_4)$, $(IRS_5, ITN_5)$ -> max rewards
- The domain of IRS and ITN is [0.0, 1.0]
- The reward's domain is not publicly announced but we believe it reaches its peak at 600,00.
- In the first month of the challenge, the environment allows us to do 1000 testing times, after that it is updated to only 100 times, so it is very limited to find the best policy.

## Introduction to the KDD Cup

- The problem is framed as a Reinforcement Learning problem.
- Proposed solutions:
  - Genetic Algorithm
  - Bayesian Optimization
  - Sequence breaking + Q-Learning

## Genetic Algorithm

## Genetic Algorithm

Genetic algorithm is a evolution algorithm and is inspired by natural selection. The following procedure has been applied:

- Create a population of 5 policies randomly and give them to the environment to get the rewards.
- Initialize a noise value
- Continually create new policy until the number of testing times is exhausted
  - New policy is created by chosen the 2 best previous policies and mix them together using “Crossover” operator. (Exploitation)
  - New policy will then be changed by “Mutation” operator with the noise value. (Exploration)
  - Noise value will be gradually decreased as we focus more on exploitation on later childs

# Genetic Algorithm

Example: Supposed we create the following policies randomly and theirs rewards

| | |
|---|---|
| [0.13, 0.97], [0.91, 0.82], [0.21, 0.13], [0.00, 0.30], [0.20, 1.00] | 12.9 |
| [0.34, 0.25], [0.01, 0.99], [0.00, 0.27], [1.00, 0.30], [0.02, 0.34] | -3.7 |
| [0.03, 0.88], [0.78, 0.25], [0.64, 0.02], [0.19, 0.69], [0.09, 0.02] | 47.2 |
| [1.00, 1.00], [0.53, 0.72], [0.28, 0.73], [0.01, 1.00], [0.40, 0.83] | 103.2 |
| [0.18, 0.27], [0.64, 0.12], [0.00, 0.00], [0.21, 0.62], [0.00, 1.00] | 5.4 |

Mixing the 2 best policies with cross over at position 2 and 4, we have 2 new policies:

[0.03, 0.88], [0.53, 0.72], [0.64, 0.02], [0.01, 1.00], [0.09, 0.02]
[1.00, 1.00], [0.78, 0.25], [0.28, 0.73], [0.19, 0.69], [0.40, 0.83]

## Genetic Algorithm

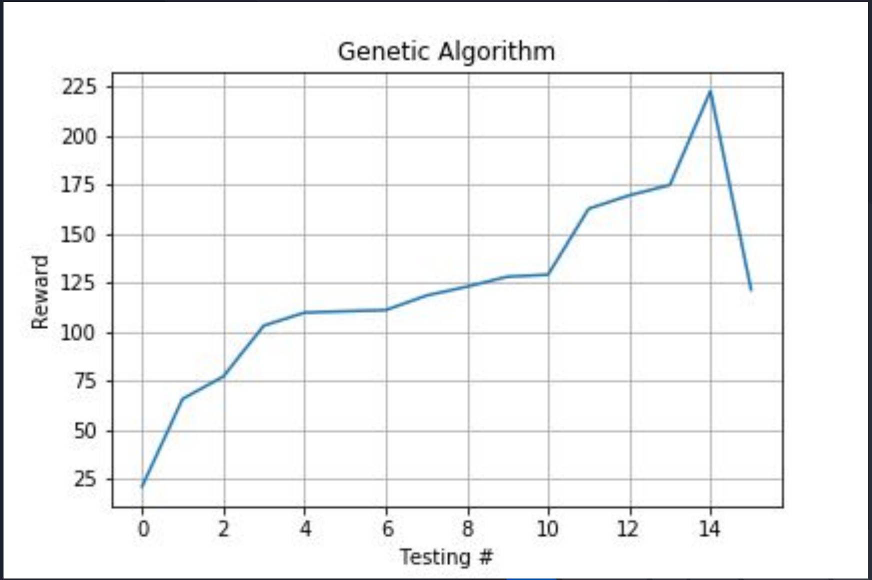
These new policies will be mutated at position 4 with noise value of 0.1:

Before mutation:
[0.03, 0.88], [0.53, 0.72], [0.64, 0.02], [0.01, 1.00], [0.09, 0.02]
[1.00, 1.00], [0.78, 0.25], [0.28, 0.73], [0.19, 0.69], [0.40, 0.83]

After mutation:
[0.03, 0.88], [0.53, 0.72], [0.64, 0.02], [0.02, 1.00], [0.09, 0.02]
[1.00, 1.00], [0.78, 0.25], [0.28, 0.73], [0.20, 0.70], [0.40, 0.83]

They will then be evaluated by the environment and added to the population. Noise value will be decreased to 0.095 and the process continues.

On average, Genetic Algorithm can get around 200.00 points worth of reward after 100 evolution times.



## Bayesian Optimisation

## BO: How does it work ?

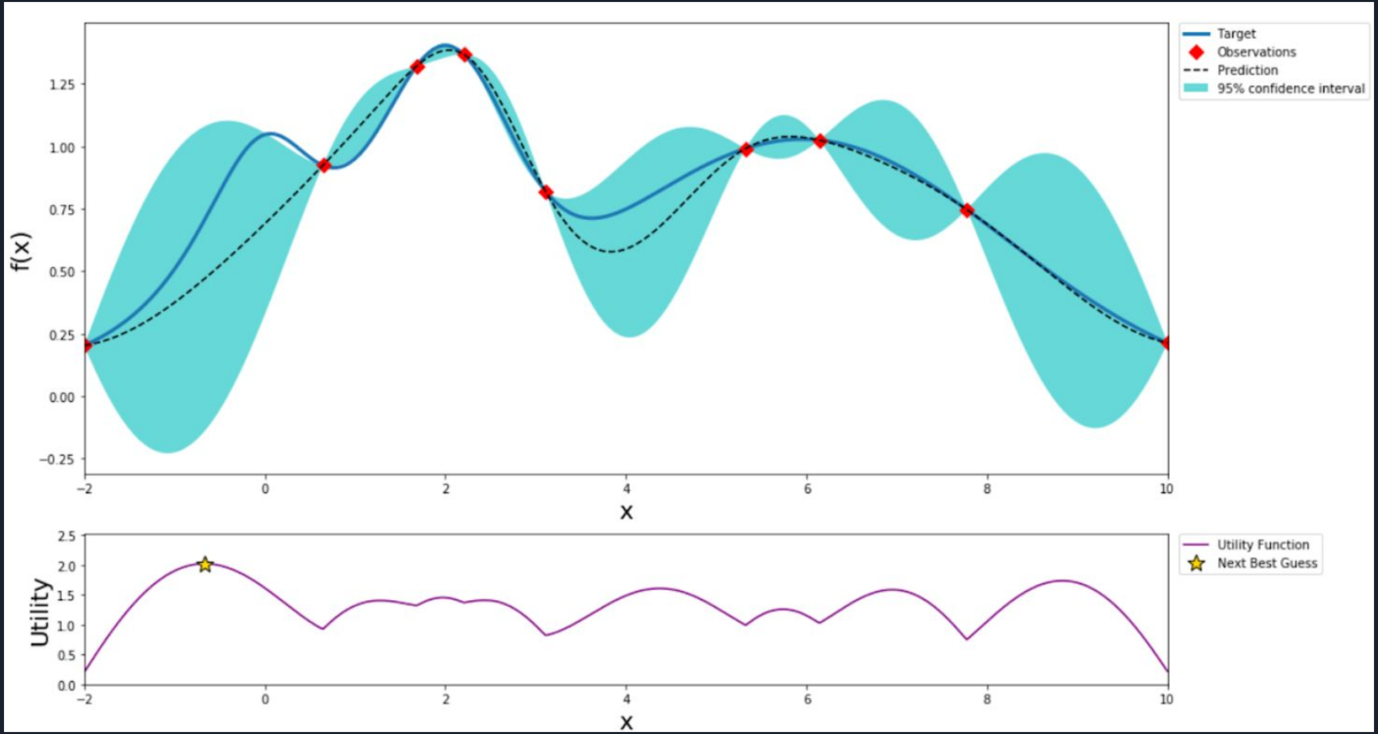


Figure: Example of 1-Dim function approximation using Bayesian Optimisation (UBC) [1]

## BO & KDD Cup: Advantages

01 (+) Interpolation:
between points saves a lot of queries

02 (+) Active Learning:
Find the best next point to query

03 (+) Reinforcement Learning:
Kappa parameter represent the exploration vs exploitation dilemma

## BO: How does it work ?

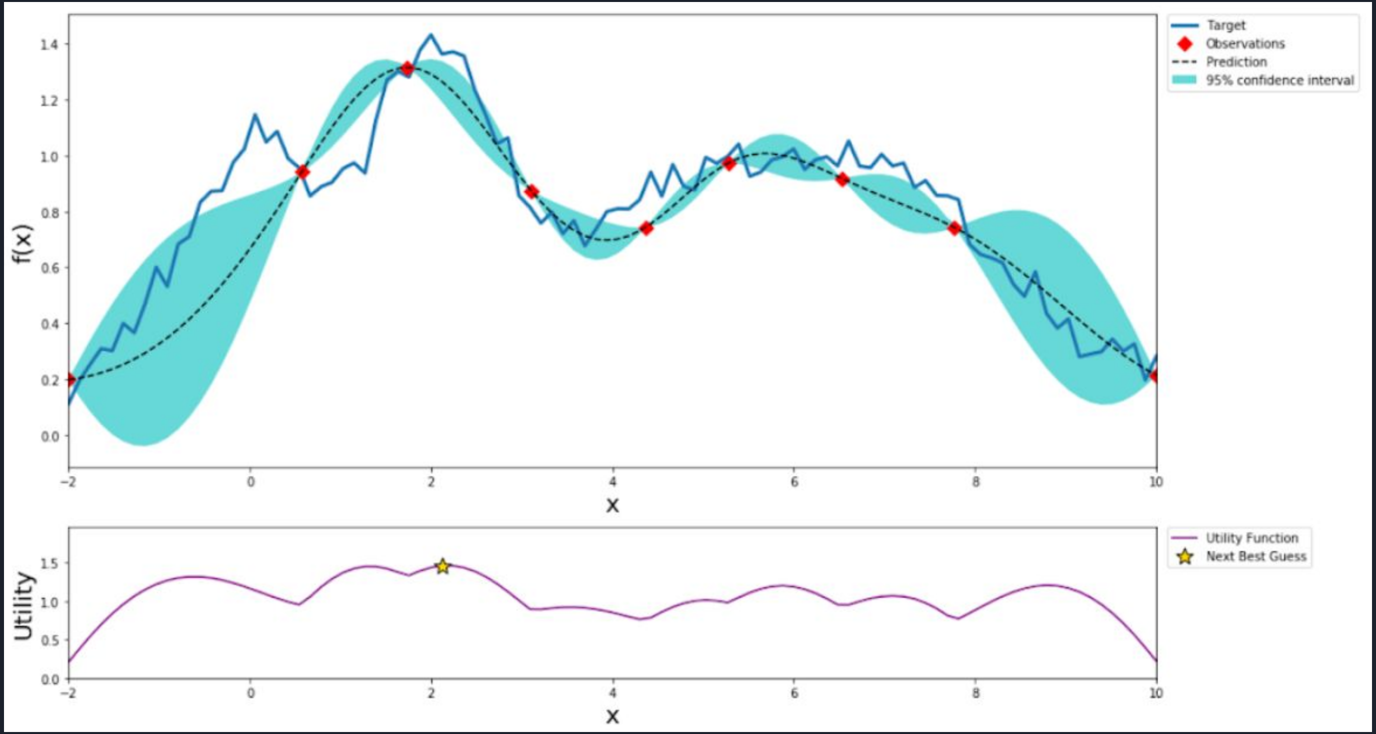
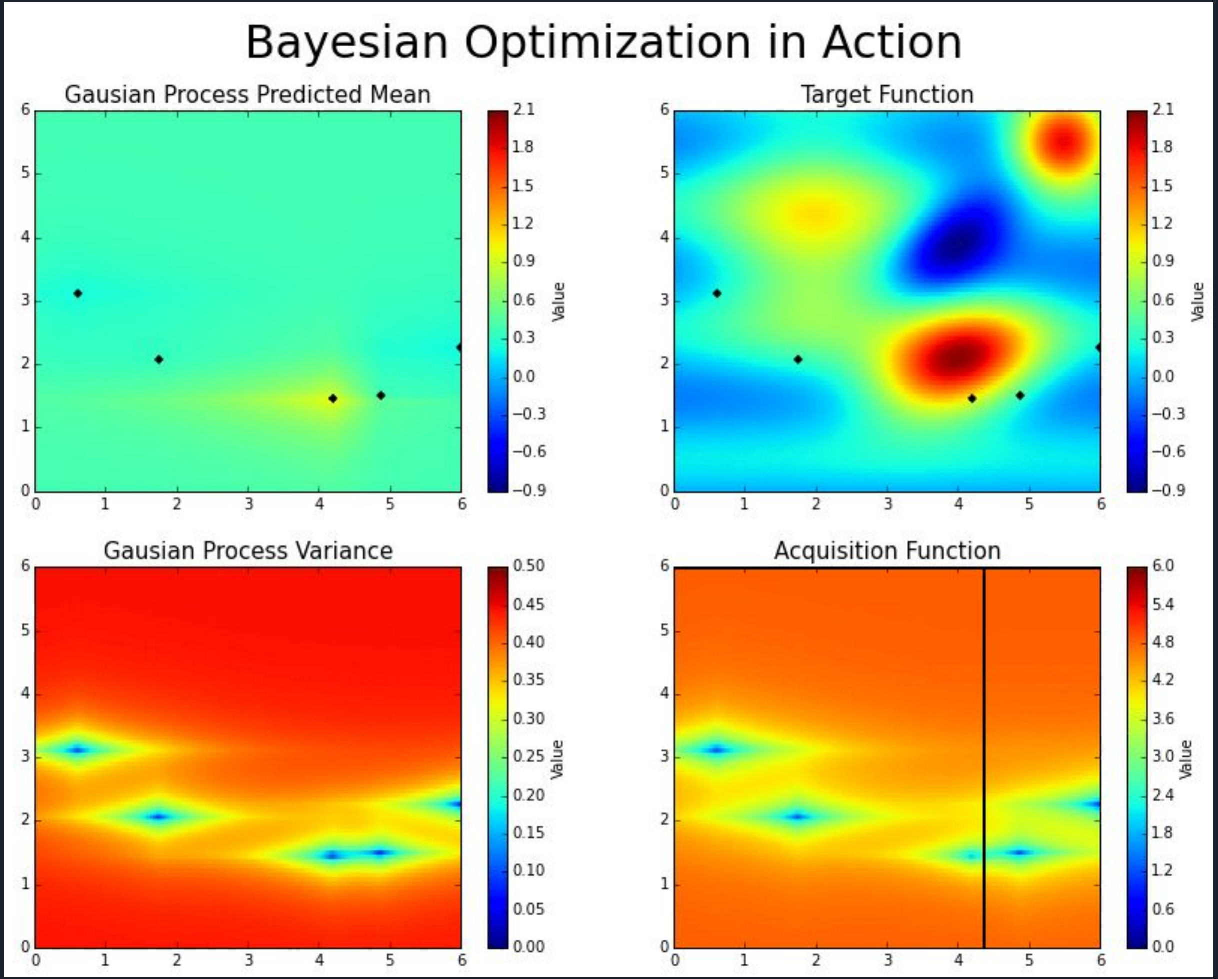


Figure: Example of 1-Dim function approximation with noise


Bayesian Optimization in Action
Gausian Process Predicted Mean
Target Function
Gausian Process Variance
Acquisition Function
Value

## BO: Year 1 approximation

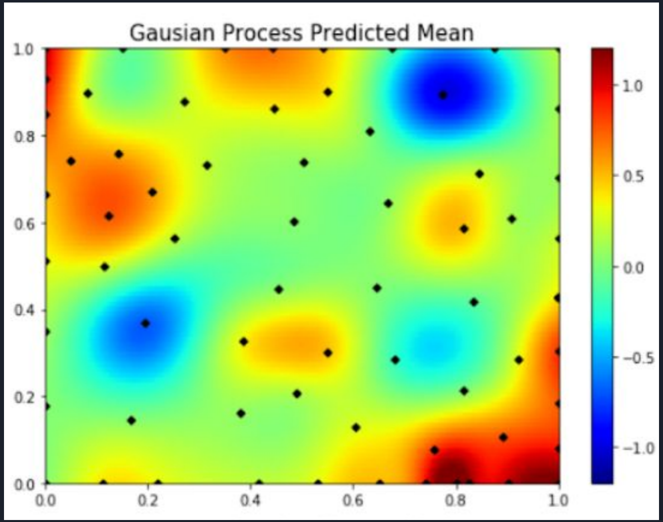


Figure: Approximation of Year 1's reward distribution using 64 points

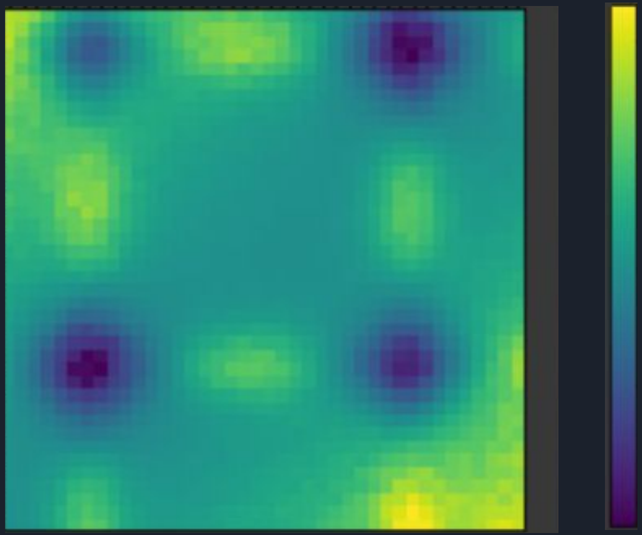


Figure: Year 1's real reward distribution using 40 by 40 points

## Sol 1: BO with prior knowledge

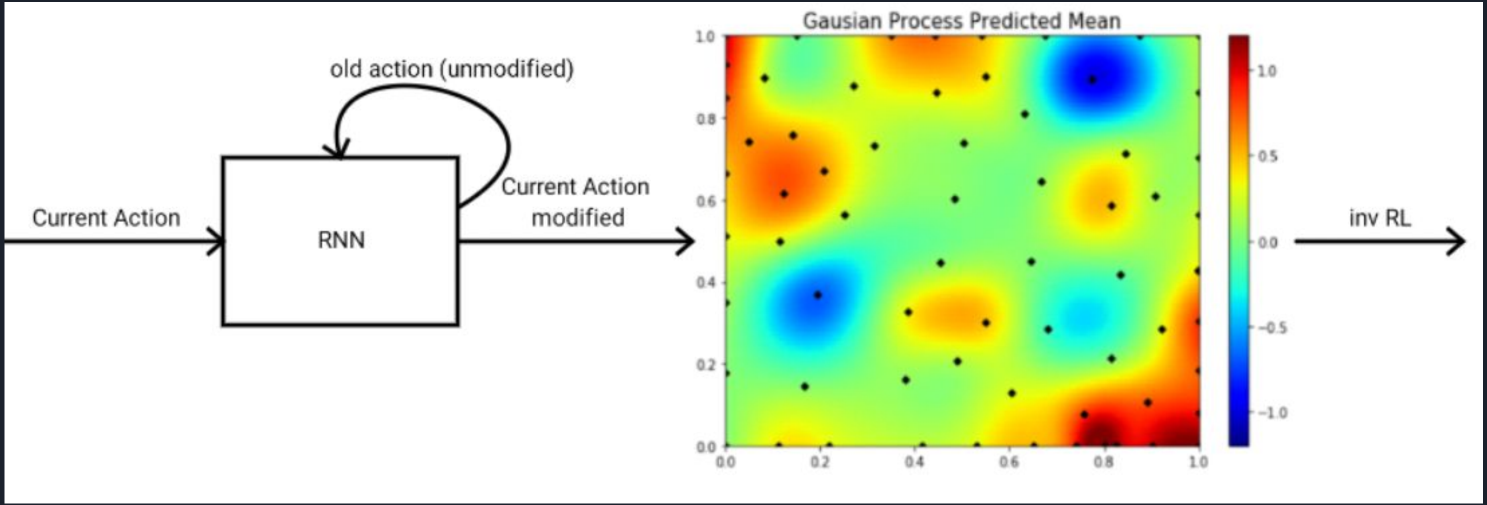


Figure: Architecture of Solution 1: a RNN node combined with the Bayesian Optimisation Alg

## Sol 2: Independent actions

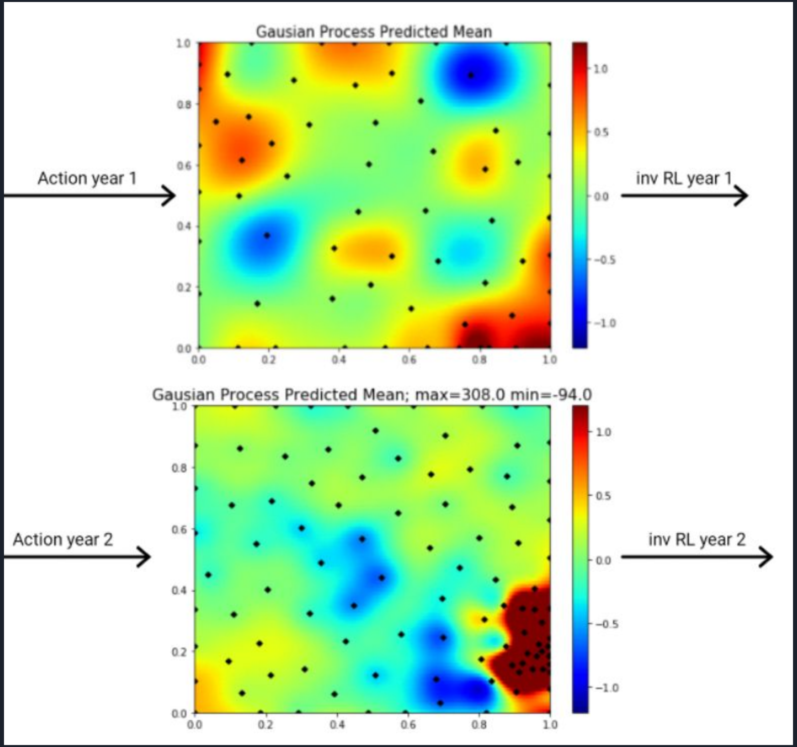
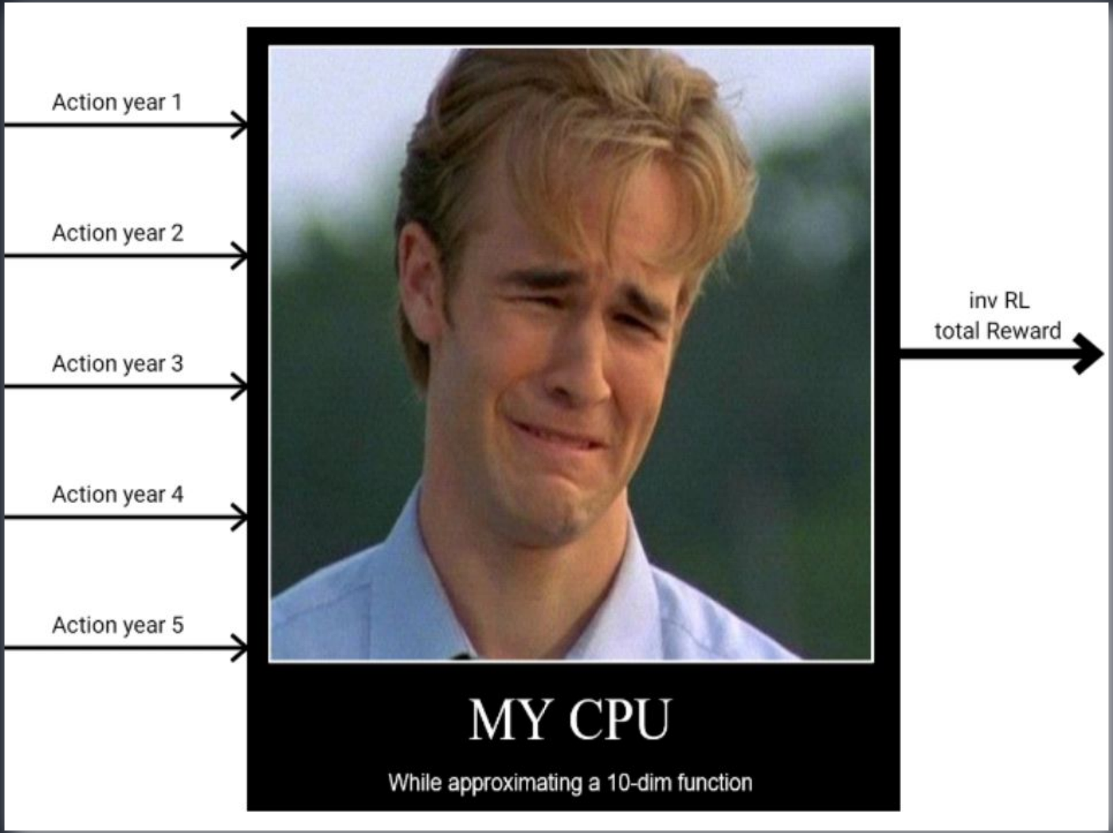


Figure: Partial architecture of Solution 2: 5 independents Bayesian Optimisation objects are used

## Sol 3: Correlated actions



## Sol 4: BO with Ensemble Learning

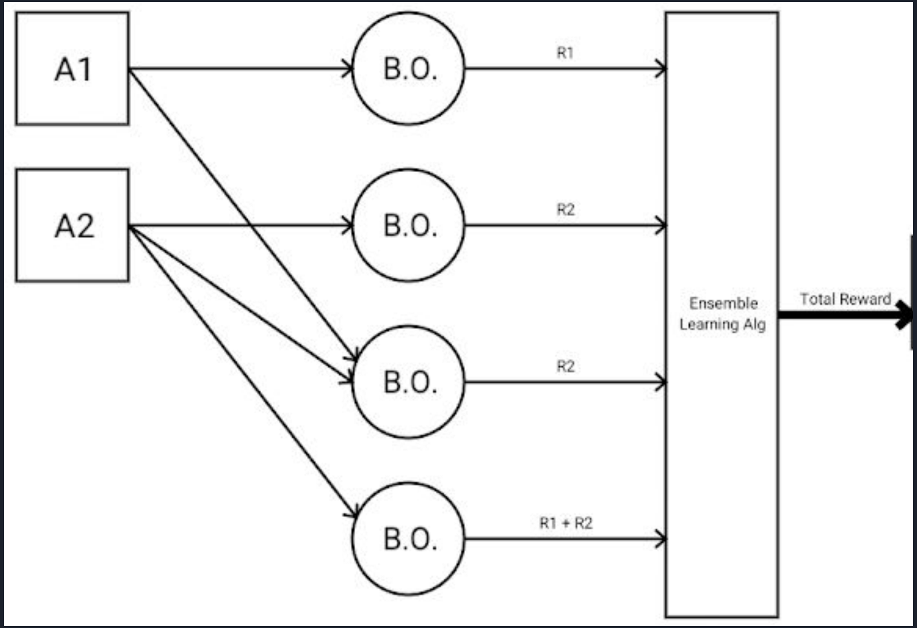


Figure: BO combined with a Ensemble Learning Alg

## Sol 4: BO with Ensemble Learning

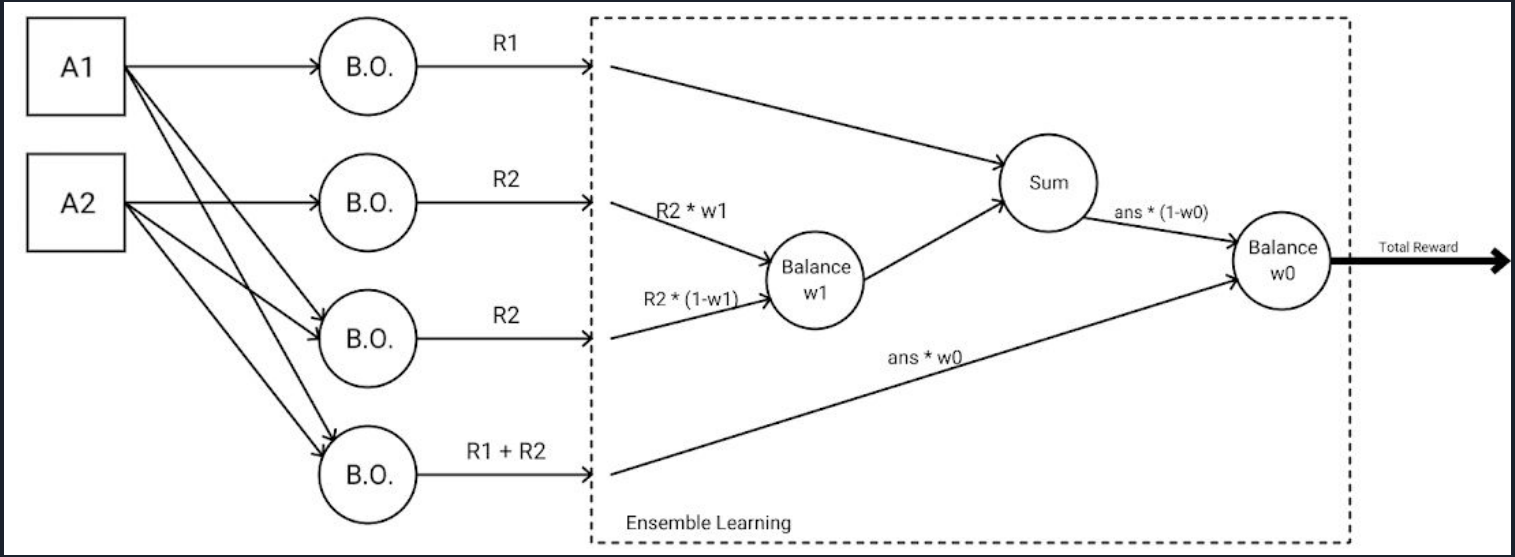


Figure: BO combined with a Afterward Balance Network

## Results

| | Environnement 1 | | Environnement 2 | |
|---|---|---|---|---|
| Random Search (Baseline) | 170 | 100 % | 200 | 100 % |
| BO with prior knowledge | 490 | 290 % | 300 | 150 % |
| Independent actions | 250 | 150 % | 500 | 250 % |
| Correlated actions | 400 | 235 % | 200 | 100 % |
| BO with Ensemble Learning * | - | 200 % | - | 140 % |

## Sequence Breaking + Q-Learning

# Problems summary:

**Problems**

- Continuous actions space
- Sequential decisions making problem
- 20 episodes

**Solutions**

- Discrete actions space
- Q-Learning
- Breaking the sequence of actions

# Continuous action space

Action in range [0,1]

By experiment, Reward([0.1,0.3]) is close to Reward([0.1231, 0.32124])

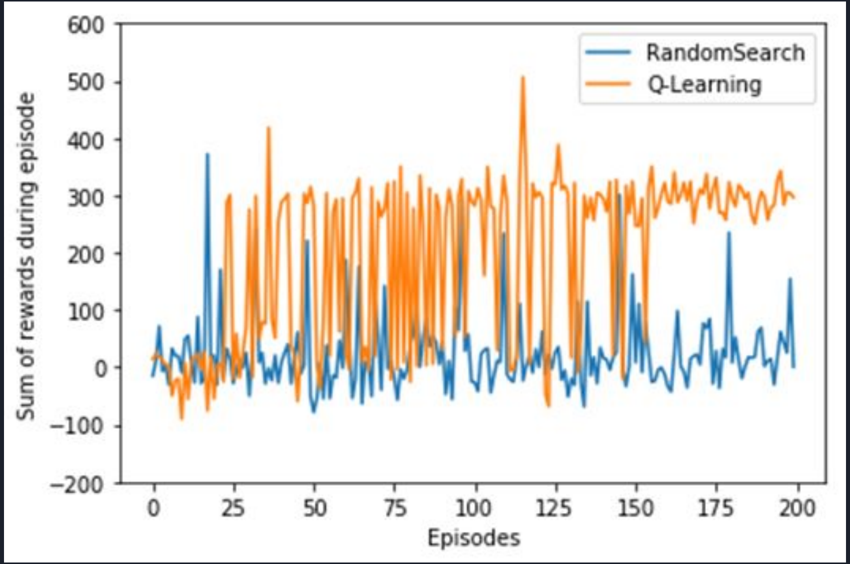
=> **discrete actions space**, limit in combination of (0.1, 0.2, 0.3 ,…,1.0)

Ex: Policy: [[0.1,0.3],[0.3,0.2],[0.5,0.5],[0.6,0.6],[0.7,0.1]]

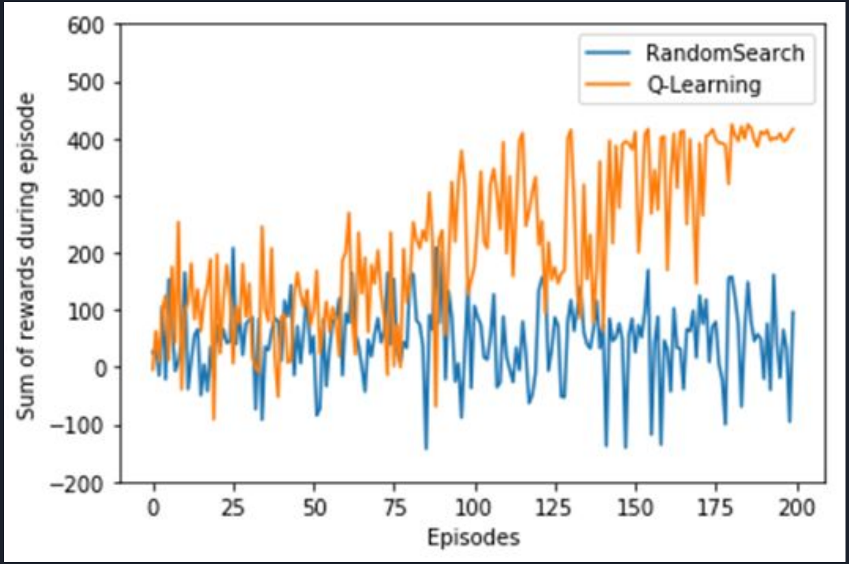
- Advantages: Easy to compute and apply algorithms that work only for discrete actions space
- Disadvantages: cannot reach the maximum reward (global optimum)

# Sequential decisions making problem

•Q-Learning with ε-greedy worked well with 200 episodes but not well with 20 episodes



Environment 1



Environment 2

# 20 episodes

20 episodes = 100 evaluations, each year has 20 evaluations

Approach: Breaking the sequence of actions, finding the maximum immediate reward of a year, ignore the relation between years, ignore future reward.

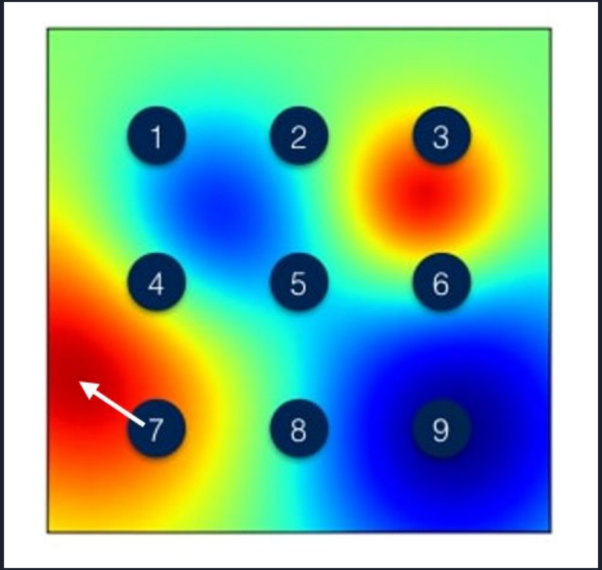
Use grid search and hill climbing to explore environment instead using random and depends on the luck:

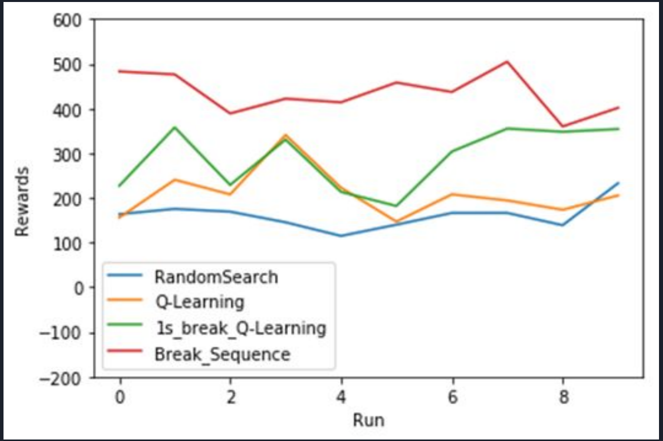
Use 16 evaluations to search the whole space, get the action has the highest reward,

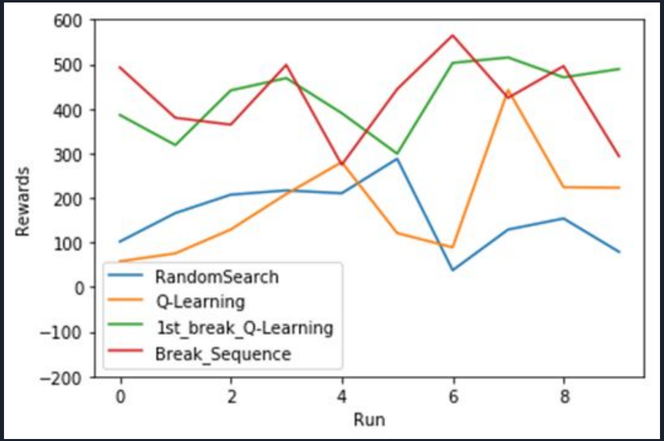
Use 4 evaluations left to search around this action, try to follow the direction that has higher reward

Option 1: Apply this technique for 5 years

**Option 2: Apply this technique  for 1st year and use Q-learning for other years**







## Result

10 independent runs

20 episodes each run

Average reward of 10 runs

Final Submission

| | Environment 1 | | Environment 2 | |
|---|---|---|---|---|
| Random Search (Baseline) | 161.20 | 100 % | 158.991 | 100 % |
| Breaking Sequence for 5 years | 434.36 | 290 % | 423.019 | 267 % |
| Q-Learning | 209.51 | 130 % | 185.055 | 117 % |
| **1st year breaking +Q-Learning** | **289.93** | **180 %** | **428.01** | **269 %** |

## 1st year breaking +Q-Learning

- Advantages
  - Simple, general
  - Improving the results comparing to baseline and simple Q-learning
  - Working well for both long runs and short runs
- Disadvantages:
  - The worst case: Maximum reward in the 1st year leads to very bad rewards in next 4 years
  - Cannot get the maximum reward for the sequence

Challenge result will be published on July 20th, 2019

## References

[1] https://github.com/fmfn/BayesianOptimization

[2] Bent et.al,  "Novel Exploration Techniques (NETs) for Malaria Policy Interventions"

[3] Amor et al.,"Intelligent Exploration for Genetic Algorithms"

Thanks for listening
Q&A