The Plant Watering Problem
RL Project report

March 2020

Marco Garcia (marco.garcia-macias@polytechnique.edu)
Christian Bile (ezanin-christian-prince-carlos.bile@polytechnique.edu)

Abstract

In this report we present our attempts by modelling and searching for reinforcement learning strategies to bring novel solutions to the plant growing problem. This problem is presented as an optimization problem aiming to maximize a revenue growing different plants in a garden in time horizon, subject to the rain forecasting, the height and the water level. The main objective of this project is to explore different configurations of this problem and try to optimize the water usage during gardening and maximize the total number of grown plants in a certain period of time. This report contains the challenges and questions, that we faced with theoretical solutions and experiments for some of them as we did not have the time to write the code and do exhaustive tests.

1 Introduction

Many of the real world tasks consist of sequential decision-making nature where an agent observes the state of a system or the environment and decides on an action for that specific state. This action has a direct influence on the state and changes it to another for the next time step. In reinforcement learning, this agent learns through the interaction with the dynamics of this environment to maximize its long-term rewards so it can act optimally. However, the involved environment may be non-stationary or noisy, this means that the next state from taking the same action from a specific state may not necessarily be the same all the time having a stochastic nature. In our plant problem we have this type of stochastic nature, where depending on the rain, the quality of the soil and the actions of the agent there are numerous possibilities for the next states. Such stochastic changes need that the autonomous agent can continually track the environment characteristics and adapt or change the learnt actions in order to ensure efficient system operation.

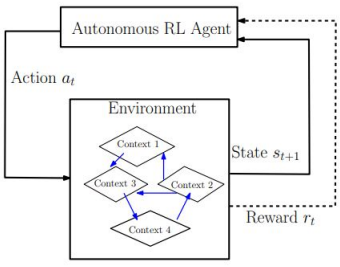


Figure 1: Schema showing a dynamically varying environment

2 The Problem

We consider a farmer who is growing a plant. The plant is growing when there is water, being in a noisy rainy environment. We consider a discrete-time evolution, on a daily basis. Each day:

- the farmer has the possibility to water or not the plant.
- the farmer observes whether there is rain or not.
- the farmer observes how much watered the plant is.
- the plant may nor not grow depending on its watering status. The plant does not grow well if it has too little, or too much water.

When a plant reaches a specific height H, it is replaced by a new one and a gain of G is received. Watering a plant has a cost of C.

2.1 Variables

The variables to consider for this problem are the following:

- $h \in \mathcal{H} = \{0, 1, \dots, H\}$ represents the height of the plant.
- $w \in \mathcal{W} = \{0, 1, 2, 3\}$ represents the watering status of the plant (0 being dry, 3 being overwatered).
- $c \in \{0, 1\}$ represents the context of the day: rainy $c = 1$ or not $c = 0$.
- $a \in \{0, 1\}$ represents the action of the farmer. Watering $a = 1$ or not $a = 0$.
- $\mathcal{S} = \mathcal{H} \times \mathcal{W}$ denote the space of controlled random variables.

2.2 Dynamics

The dynamics of the system is given by the four transition probability distributions $(\mathbf{p}_{c,a})_{r,a}$, where $\mathbf{p}_{c,a} : \mathcal{S} \to P(\mathcal{S})$, depending on the rain variable and the chosen action. Introducing the clip operator $|x|_{a:b} = \max(\min(x, b), a)$ and the Dirac distribution $\delta(h)$ that puts mass 1 to h and 0 to other points, these distributions are defined as follows for each $h \in \{0, 1, \dots, H-1\}$,

$$\begin{aligned}
\mathbf{p}_{c,a}(h, w=0) &= ((1-q_0))\,\delta\left(|h-1|_{0:H}\right) + q_0\delta(h)) \times \mathbf{p}^{\mathcal{W}}_{c,a}(w), \quad q_0 = 0.7 \\
\mathbf{p}_{c,a}(h, w=1) &= ((1-q_1)\,\delta(h) + q_1\delta\left(|h+1|_{0:H}\right)) \times \mathbf{p}^{\mathcal{W}}_{c,a}(w), \quad q_1 = 0.6 \\
\mathbf{p}_{c,a}(h, w=2) &= ((1-q_2)\,\delta(h) + q_2\delta\left(|h+1|_{0:H}\right)) \times \mathbf{p}^{\mathcal{W}}_{c,a}(w), \quad q_2 = 0.9 \\
\mathbf{p}_{c,a}(h, w=3) &= ((1-q_3)\,\delta\left(|h-1|_{0:H}\right) + q_3\delta(h)) \times \mathbf{p}^{\mathcal{W}}_{c,a}(w), \quad q_3 = 0.5
\end{aligned}$$

where $\mathbf{p}^{\mathcal{W}}_{c,a}(w) = (1-q)\delta\left(|c+a+w-1|_{0:3}\right) + q\delta\left(|c+a+w|_{0:3}\right)$ is introduced to capture the effect of the context and action on the watering index, with $q = 0.3$, which model a soil/capture that does not retain water much. For $h = H$, the transition is given by $\mathbf{p}_{c,a}(H, w) = \delta(0) \times \delta(w)$ to model the reset (introduction of a new plant with initial height $h = 0$). The dynamics is completely specified by the parameters $(H, q_0, q_1, q_2, q_3, q)$

2.3 Reward

The reward function is deterministic and given by $\mathbf{r}(h, w, c, a) = GI\{h = H\} - aC$. It is completely specified by the parameters (H, G, C).

2.4 Rain forecast

The rain variable c_t at time t is randomly generated with probability $\theta_t \in [0,1]$, that is $c_t \sim \mathcal{B}(\theta_t)$, where θ_t. The sequence $(c_t)_t$ is generated at time 1 for all next time steps.
There are three different situations (hereafter denoted (E), (P) or (C)) regarding the knowledge of the learner at time t :

- (E) The learner is given the exact rain forecast $c_t, c_{t+1}, \ldots, c_{t+L}$ for a (possibly infinite) look-ahead L.
- (P) The learner is given the exact rain parameter forecast $\theta_t, \theta_{t+1}, \ldots, \theta_{t+L}$ for a (possibly infinite) look-ahead L.c_t is not observed before time t.
- (C) The learner is given high probability confidence sets given by values $\theta^-_{t'}, \theta^+_{t'}, \alpha_{t'}$ for all $t' \in \{t, \ldots, t+L\}$, such that $\theta_{t'} \in \left[\theta^-_{t'}, \theta^+_{t'}\right]$ holds with probability at least $\alpha_{t'} \in [0.5, 1]$. Now, θ_t is unknown and c_t is not observed before time t.

For simplicity, we assume that θ_t can only take three values, $\{0.2, 0.5, 0.8\}$, and consider θ_t is piecewise constant with each epoch having size K.

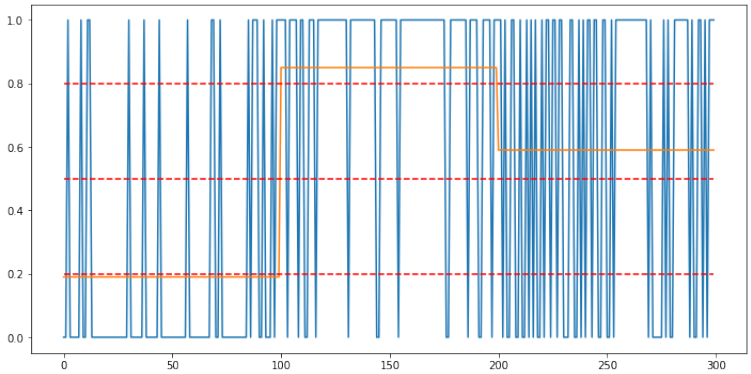


Figure 2: piece-wise generated rain with K=100, T=300 we can see that the rain(blue) follows a $\mathcal{B}(\theta_t)$ with $\theta = 0.2$, 0.8 and 0.5 (red)

2.5 Watering level observation

There are two situations of watering level observation:

- **Fully-observed model**: w_t is systematically observed at time t. It coresponds to the situation described in the previous sections.
- **In the Partially-observed**: observing w_t is an action o that comes with observation costs C'. It has no effect on the dynamics. Hence in this modified problem, the action becomes $(a, o) \in \{0,1\}^2$, the dynamics is $\mathbf{p}_{c,(a,o)} = \mathbf{p}_{c,a}$, and the reward function is $\mathbf{r}(h, w, c, (a, o)) = \mathbf{r}(h, w, c, a) - oC'$

In that case, the system is fully parameterized by $(H, q_0, q_1, q_2, q_3, q, G, C, C')$.

3 Experiments set up

For our experiments we used python, with the help of external libraries, to implement the garden environments, as long as the dynamics and rain forecast, from scratch. The code was submitted as notebook file together with this report for further information. In this section, we underline some of the solutions of implementation, and eventual alternatives, to model the problem for the experiments and learning process.

3.1 The environment

We represent the environment as a garden object $\mathcal{G}$ containing plants represented as uniquely identified objects with the height and water level attributes. The possible interactions between the environment and the agent (the farmer) are:

- **The transition distribution:** given a state $s_t \in \mathcal{S} = \mathcal{H} \times \mathcal{W}$ of a plant, and an action $(a_t, o_t) \in \{0,1\}^2$ of the agent at a timestep t, the environment returns the possible next transitions from the dynamics as long as the rewards obtainable from each.
- **Transition to a new state:** given a current state $s_t \in \mathcal{S} = \mathcal{H} \times \mathcal{W}$ of a plant, and an action $(a_t, o_t) \in \{0,1\}^2$ of the agent at a timestep t, the environment returns the reward and the new state s_{t+1} from the transition distribution.

3.2 The reward

We implemented the reward function is implemented as $\mathbf{r}(h_{t+1}, w_{t+1}, c_t, (a_t, o_t)) = \mathbf{r}(h_{t+1}, w_{t+1}, c_t, a_t) - o_t C' - I\{h_{t+1} = h_t\}C''$, where $C'' \geq 0$ is a penalization cost for not watering enough the plant to make it grow. In that case, the system is fully parameterized by $(H, q_0, q_1, q_2, q_3, q, G, C, C', C'')$.

3.3 The policy

We decided that the optimal policy π to learn, should be parametrized in the rain state observed c and be deterministic: $\pi_c : \mathcal{S} \to \mathcal{A}$, or $\pi_c : \mathcal{H}(\mathcal{S}) \to \mathcal{A}$ when history dependent.

3.4 The rain forecast (C)

The interval $\left[\theta_t^-, \theta_t^+\right]$ containing the real $\theta_t \in \{0.2, 0.5, 0.8\}$ with probability $\alpha_t \in [0.5, 1]$ is computed by generating randomly θ_t^- and θ_t^+ from a Beta distribution, making sure it contains θ_t and the uniform probability in this interval is α_t.

3.5 The Partially-observed model

One of the main challenges faced during this project, was to find a proper implementation of the partially-observed model for the reinforcement learning strategies. We came up with the following final solution:

- **Solution:** The transition distribution of an action $(a_t, 0)$ taken by the agent, is the mixture distribution $\mathbf{p}_{c,a}(h)$ of the 4 transitions $\mathbf{p}_{c,a}(h, w)$ with same weights to increase the uncertainty of the current w_t and the possible next states. During the exploration, the agent tries to find what is the height of the plant that is delicate enough to pay the cost C' by setting $o = 1$ in order to maximize the reward, while at exploitation time , the policy learned is used to make action.

$$\mathbf{p}^{\mathcal{W}}_{c,a}(w) = \sum_{c=0}^{c=1} \sum_{w=0}^{W} \mathbf{p}^{\mathcal{W}}_{c=c,a}(w = w) P_{c=c} P_{w=w}$$

$$\mathbf{p}_{c,a}(w) = \sum_{w=0}^{W} \mathbf{p}_{c,a}(w) P_{w=w}$$

  where we can use the probabilities θ instead of the real value of the rain and some probabilities $P_{waterLvl}$ in order to calculate our transition probabilities without the need to observe w, we can choose pessimistic probabilities by assigning more probability to the level w=3 or choose equal probabilty for every level of water

3.6 Instance

For our experiments we instanced a garden of plants in $|\mathcal{S}| = |\mathcal{H}| \times |\mathcal{W}| = 6 \times 4$ possible states, with exhaustive parametrization and empirical simulations of the gain G and costs C, C', C'' and learning parameters like γ, the time horizon T, the maximum iteration, the learning rate λ and the ϵ threshold, using the Values iterations, Policy Iteration, Q-Learning and DeepQL methods. We took the code of some of these methods from the lab sections and the internet and modified them to make them more suitable to our problem. The code of the Q-Learning was implemented from scratch and has the only difference that we decrease the $\epsilon \in [0, 1]$ exploitation threshold, and the λ learning rate at each time step $t \in T$. More precisely we define the following functions:

- $V(\mathcal{S}, \{\theta, c\}, \gamma, T)$ as the discounted value function iterating over T steps for each state given the rain forecast.
- $V^{opt}(\mathcal{S}, \{\theta, c\}, \gamma, \epsilon, T)$ the discounted Q-value function for each state given the rain forecast, and the policy at the last iteration over T steps and a ϵ desired maximum bound of the difference between the values of the current step and the previous one.
- $Pol(\mathcal{S}, \{\theta, c\}, \gamma, \epsilon, T, I)$ the policy iteration function which returns the suboptimal policy at the last iteration over T steps of rain forecast and a ϵ desired maximum bound of the difference between the values of the current step and the previous one in a maximum iteration of I times $V(.)$. This function is computationally expensive and was often used to verify the correctness of the V^{opt}.
- $QL(\mathcal{S}, \{\theta, c\}, \gamma, \epsilon, \lambda, T, I)$ the Q-learning algorithm with I iterations on the states with T steps of rain forecast and an initial ϵ probability of exploration-exploitation, and initial value of the learning rate parameter λ.

3.7 Parameters influences

After different experiments, which results are shown and explained in the next sections, we came to a conclusion of the influences of following the parameters of the systems and algorithms used:

- The gain **G** can lead to a policy of not growing anything if the maximum reward achievable per plant in time horizon T is lower or too close to the average regret.
- The cost **C** of watering can lead to a policy of not growing anything if it too high. Also, it can lead to the same wrong policy in regret minimization based strategies, as the average regret will be negative at each decision of not watering ($a = 0$).
- The cost $\mathbf{C}'$ of observing the water level, when it is too high, it can lead to a policy of watering blindly plants in a critical water level.
- The cost $\mathbf{C}''$ of not growing a plant can help to push the agent to grow plants, but the result may not be the maximum reward expected, and can be negative sometime when the optimal policy for a given configuration is the not-watering/not-growing plants.
- The soil capture q : a very small value of q makes very difficult to retain water from rain and from the actions, this causes more costs in dry contexts, causing the agent to never exit from state 0.
- The discount factor γ measures the importance that we give to the values of the transition states (the accumulated discounted reward in those states), a lower γ incurs a decision highly influenced by the current rewards obtainable from the transition state.
- The time horizon **T** Very small time horizons end up be costly for the farmer, since depending on the rain and on the soil capture, the average time to grow a plant in optimal conditions (always water level = 2 would require at least H time steps

4 Case 1: Fully-observed model and rain forecast (E)

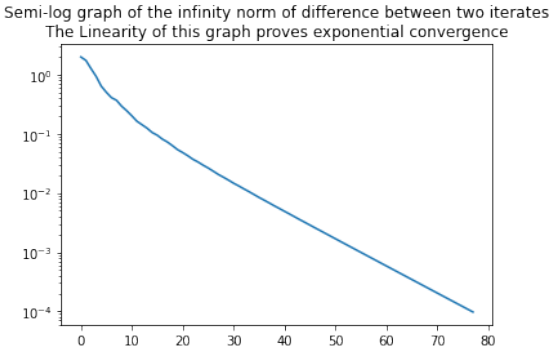
4.1 Always rain days or Always dry days

This configuration assumes that the the rain forecast is constant through in a infinity time steps, and can be either a constant raining environment or a dried one. For this situation, we can find easily a optimal policy by using exact methods such as Value Iteration or Policy Iteration since the optimal policy is no longer parametrized in the rain state $\pi_c(\mathcal{S}) = \pi(\mathcal{S})$. Therefore, we can learn this optimal policy by values iteration in a sufficient time horizon for every state and for every action, with corresponding transition probabilities and rewards that become deterministics and known. However, the algorithm can converge to a suboptimal policy π^* because of the system parameters and the algorithm parameters like the discount factor γ. For example, in a context where there's no rain, an intuitive optimal policy would suggest the action to water the plant that has been just sowed (initial state) to make it grow, but this is not always the case as shown in the results below, because of the system parameter like the game gain G and the soil capture $q = 0.3$ which influences a lot the dynamics and the action decision.

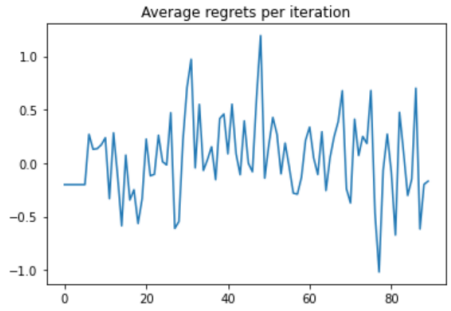
4.1.1 Dry days (c = 0) results

Here we present of the results of the experiments for few configurations. An intuitive optimal policy would be in this case **[1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]**, as we need to water every day plants with a water level below the critic threshold to make them grow and maximize the reward. The parameters that matter the most in this case is the gain and costs values.

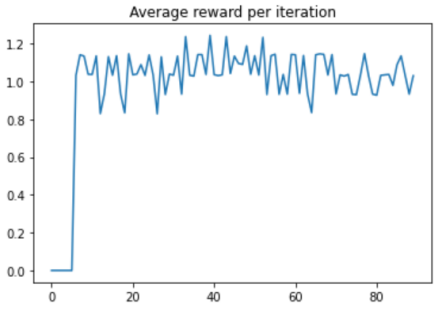
- *Configuration 1:* $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 1000$



(a) Semi-log graph showing convergence of the value iteration for *configuration 1*



(b) Average regret per iteration



(c) Average reward per iteration

Figure 3: Results for *configuration 1*

The optimal policy $\pi^*(\mathcal{S})$ returned by V^{opt} for this type of configuration is: **[1, 1, 0, 0, 1, 1, 0, 0,**

1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0] .
By following this policy when there is no rain, we obtained an average reward = 57.55 with an average grown plants of 12.94 in approximately 79.4 steps

- *Configuration 2:* $T = 100, G = 1, C = 0.2, C' = 0, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 1000$

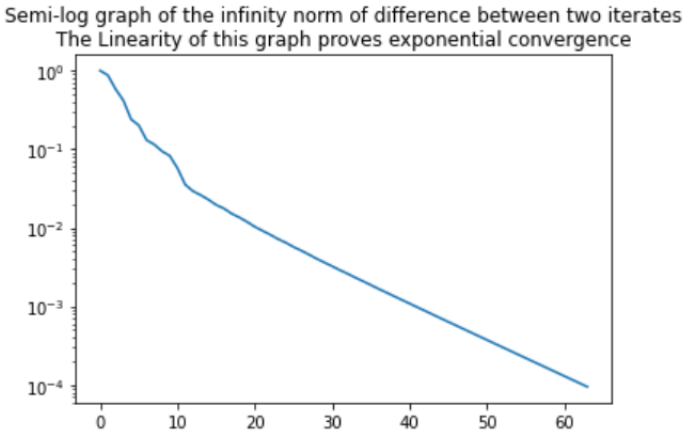


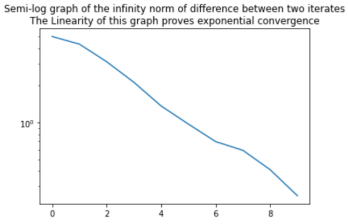
Figure 4: Semi-log graph showing convergence of the value iteration for *configuration 2*

The optimal policy $\pi^*(\mathcal{S})$ returned by V^{opt} for this type of configuration is: **[0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]** in 64 steps. The average reward from this policy obviously 0 if we start from the initial state since we never grow anything.

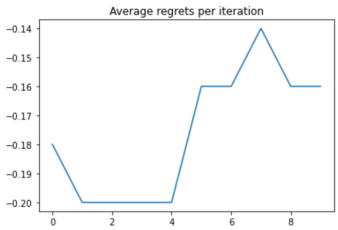
4.1.2 Rainy days (c = 1) results

Here we present of the results of the experiments for few configurations. An intuitive optimal policy would be in this case **[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]**, as the constant rain state is enough to grow the plants. The parameters that matter the most in this case is how much we iterate through the values in order to get as close as possible to this policy.

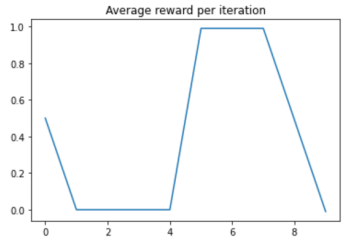
- *Configuration 3:* $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 10$



(a) Semi-log graph showing convergence of the value iteration



(b) Average regret per iteration

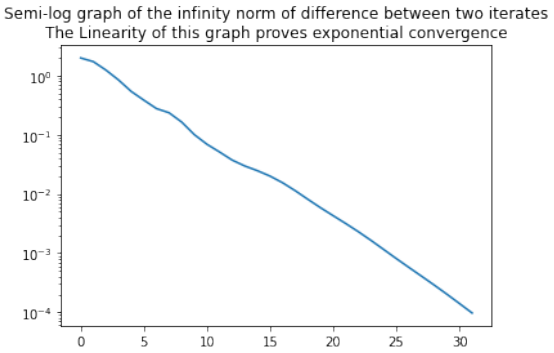


(c) Average reward per iteration

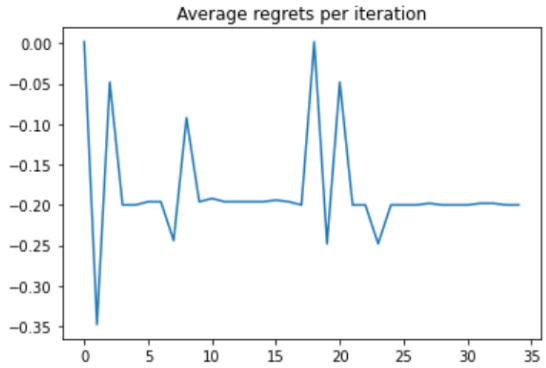
Figure 5: Results for *configuration 3*

The optimal policy $\pi^*(\mathcal{S})$ returned by V^{opt} for this type of configuration is: **[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]** in I steps with a residual difference of Q-Values of $0.2522780979183388 > \epsilon$.

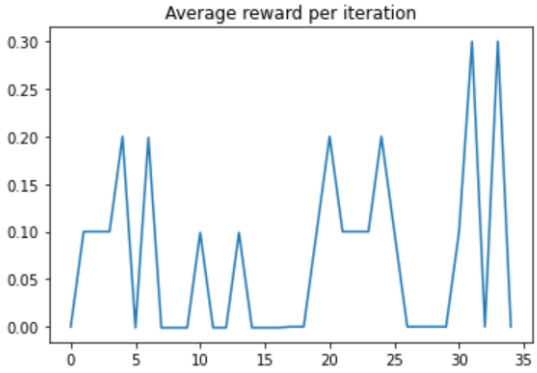
- *Configuration 4:* $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 1000$



(a) Semi-log graph showing convergence of the value iteration



(b) Average regret per iteration



(c) Average reward per iteration

Figure 6: Results for *configuration 4*

The optimal policy $\pi^*(\mathcal{S})$ returned by V^{opt} for this type of configuration is: **[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]**. We obtained an average reward = 2.55 with an average grown plants of 0.51 in approximately 9 steps.

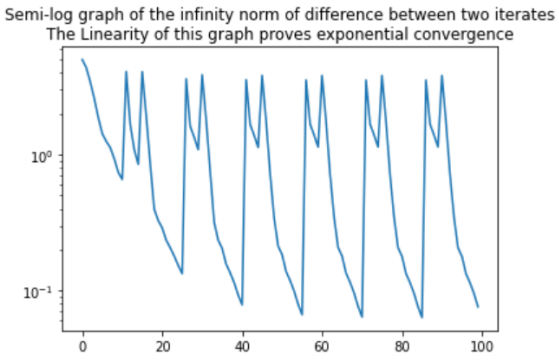
4.2 Changing rain forecast ($c_t = 0$ if $(tmod15) \leq 10$ and $c_t = 1$ else)

This configuration assumes that the the rain forecast is no longer constant through in an infinity time steps, but piece-wise constant. For this situation, an optimal policy returned by the methods $V^{opt}(.)$ and $Pol(.)$ is parametrized in the rain state $\pi_c(\mathcal{S})$, meaning that the actions of the agent is influenced by the observed rain state c_t. However, the algorithm converge to a suboptimal policy π_c^* because of the noise in the environment, the system configuration and the algorithm parameters like the discount factor γ. The convergence of the algorithm is less optimal and the difference between the current time steps values and the previous ones oscillates in a range that represent the breakpoints of the piecewise constant of rain states, as shown in the example below.

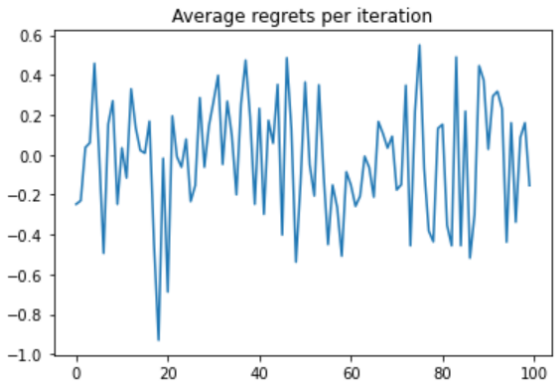
- *Configuration 5:* $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 100$

  The optimal policy returned by V^{opt} for this type of configuration is: $\pi^*_{c=0}(\mathcal{S})$ = **[1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]**, and $\pi^*_{c=1}(\mathcal{S})$ = **[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]**

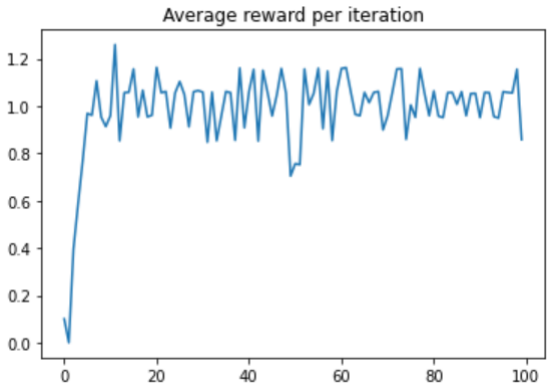
As we see for this configuration, the policy of $\pi^*_{c=0}(.)$ is equal to the policy of *configuration 1* where the rain



(a) Semi-log graph showing convergence of the value iteration



(b) Average regret per iteration



(c) Average reward per iteration

Figure 7: Results for *configuration 5*

forecast is constant and equals to 0, which is the policy of only watering in time step t the plants that do not have a critical watering level. Also, when $c = 1$, the policy $\pi^*_{c=1}(.)$ seems to be optimal as it suggests to only water the plant in a very low water level as the soil capture of the water is low $q = 0.3$.

5 Case 2: Fully-observed model and rain forecast (P)

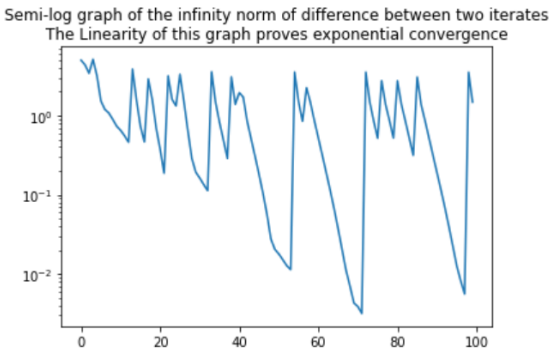
In this configuration where the rain forecast c_t is a random variable $c_t \sim \mathcal{B}(\theta_t)$ at each time step, we add one more "layer" of noisiness in the environment studied in the case in Section 4.2 because the rain forecast has rate of change that is unknown, we just know that at each time step t, $c_t = 1$ with probability θ_t. Thus, the change can be continuous or piecewise.

5.1 rainy days with constant probability θ

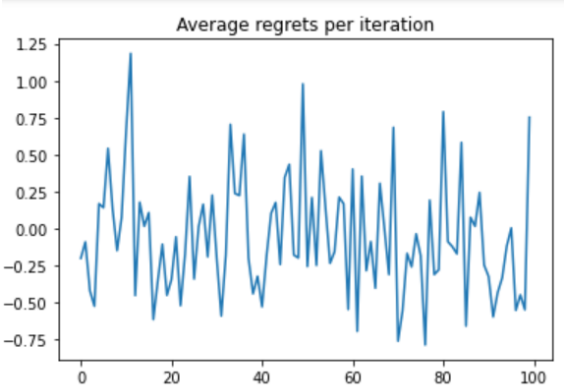
Using the same RL strategies of the previous case and the same system configuration of the experiment in Section 4.2, with the only difference that the value c_t is estimated (observed) at time step t from a Bernoulli distribution with constant probability θ generated random before starting the game, we tried to figure out whether it is possible to learn a suboptimal policy similar to the one found there. The result below, shows a convergence to the same suboptimal policy even with the more noisy environment.

- *Configuration 6:* $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 100$

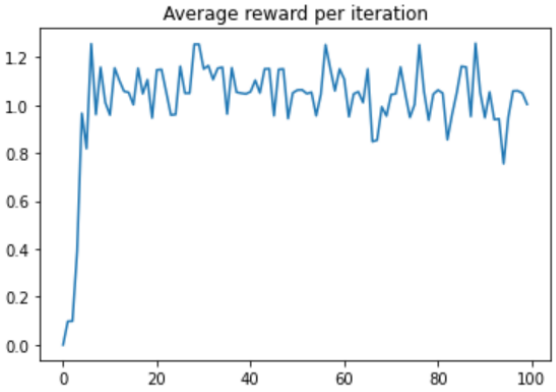
  The optimal policy returned by V^{opt} for this type of configuration is: $\pi^*_{c=0}(\mathcal{S}) =$ **[1, 1, 0, 0, 1, 1, 0,**



(a) Semi-log graph showing convergence of the value iteration



(b) Average regret per iteration



(c) Average reward per iteration

Figure 8: Results for *configuration 6*

0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0], and $\pi^*_{c=1}(\mathcal{S}) =$ **[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]**

We explain this behavior by analysing the different values of $\theta \in \{0.2, 0.5, 0.8\}$:
When having the value of a constant θ it is possible to calculate the transitions probabilities if we have understainding of the dynamics, In this case, without changing the dynamics we can calculate the transition probabilities given a rain probability θ:

$$\mathbf{p}^{\mathcal{W}}_{c,a}(w) = (1-q)\delta\left(|c+a+w-1|_{0:3}\right) + q\delta\left(|c+a+w|_{0:3}\right)$$

can become :

$$\mathbf{p}^{\mathcal{W}}_{\theta,a}(w) = \mathbf{p}^{\mathcal{W}}_{c=0,a}(w)(1-\theta) + \mathbf{p}^{\mathcal{W}}_{c=1,a}(w)(\theta)$$

So we can calculate all the transition probabilities given a rain probability and then do the Value Iteration and try to see if there is convergence to a suboptimal parametrized policy $\pi^*_\theta(.)$.
Using the same system configuration as before, $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 100$, we have the following results for different values of θ:

- $\theta = 0.2$
  The optimal policy for this type of configuration is: **[1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]**.
  This policy is equal to the one returned when there is no rain at all as the value of θ corresponds to a probability of 0.8 to have $c_t = 0$, meaning that 80% of the time it's not raining in an infinite time

horizon. The issue comes in when it rains and we water too much a plant, and this problems is reflected in the obtained average reward = 44.91 with an average grown plants of 10.04 in approximately 9 steps, which is lower than the one expected in an environment when it's not raining at all.

- $\theta = 0.5$
  The optimal policy for this type of configuration is: **[1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]**.
  This policy tries to respond optimally to the change of $c_t = 0$ 50% of the time in an infinite time horizon. When there is always rain, we obtained an average reward = 39.10 with an average grown plants of 8.34 in approximately 11 steps.

- $\theta = 0.8$
  The optimal policy for this type of configuration is: **[0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]**.
  As for value 0.2, this policy is the same obtained as for a constant value of $c_t = 1$, 80% of the time in an infinite time horizon. When there is not always rain, we obtained an average reward = 25.35 with an average grown plants of 5.12 in approximately 16 steps.

5.2 Rainy days with constant probability θ DeepQL

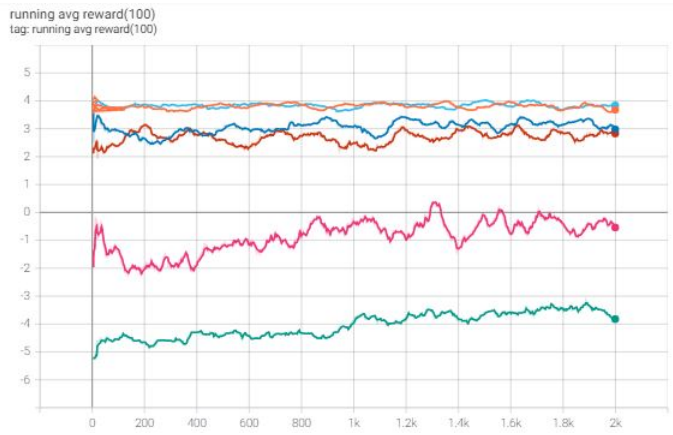


Figure 9: training agent with different $\theta \in \{0.2, 0.5, 0.8\}$

For the deep QL we used a network with 2 hidden layers of 100 units, we perform relu activation function and a sigmoid final activation, our learning rate is 0.0001 and the target network is updated every 50 steps, the replay buffer needs min 100 observations and max 10000 and we train for 100 episodes of T=100

- $\theta = 0.2$
  Using 100 episodes of $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0$ by following this policy when there is always rain, we obtained an average reward = 4.84 with an average grown plants of 2.1 in approximately 47 steps

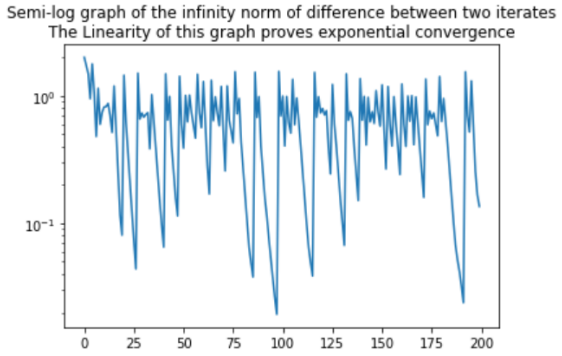
- $\theta = 0.5$
  Using 100 episodes of $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0$ by following this policy when there is always rain, we obtained an average reward = 17.98 with an average grown plants of 4.63 in approximately 21 steps

- $\theta = 0.8$
  The optimal policy for this type of configuration is: Using 100 episodes of $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0$ by following this policy when there is always rain, we obtained an average reward = 18.75 with an average grown plants of 4.82 in approximately 20 steps

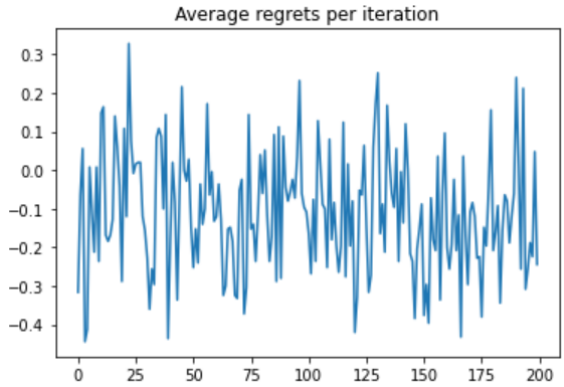
5.3 Changing θ ($\theta_t = \theta$ if $(tmod15) \leq 10$ and $\theta_t = \theta'$ else)

For this configuration, we converge to the same policy, but with more iterations, as the forecast result in time horizon T is the same as Section 4.2 with the approach in Section 5.1 with the difference that θ is piecewise constant.

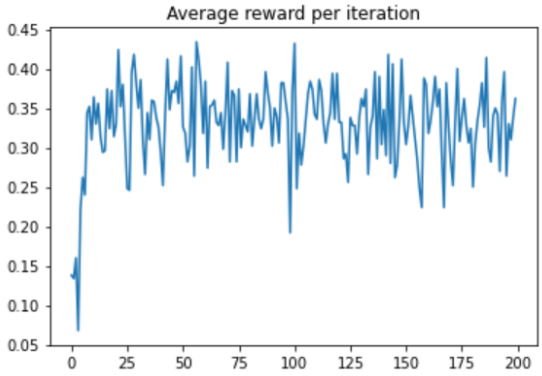
- *Configuration 7:* $T = 100, G = 5, C = 0.1, C' = 0, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 200$



(a) Semi-log graph showing convergence of the value iteration



(b) Average regret per iteration



(c) Average reward per iteration

Figure 10: Results for *configuration 7*

The optimal policy returned by V^{opt} for this type of configuration is: $\pi^*_{c=0}(\mathcal{S}) =$ **[1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]**, and $\pi^*_{c=1}(\mathcal{S}) =$ **[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]**

5.4 Rainy days with piece-wise constant θ ACKTR method

For this type of context we decided to train a ACKTR (Actor-Critic using Kronecker-Factored Trust Region) model, it is a Policy Gradient method with the trust region optimization, one deep neural network is used to estimate the policy and another network to estimate the advantage function. We trained this network with a K = 50, T = 100 for episodes=1000

6 Case 3: Fully-observed model and rain forecast (C)

This configuration assumes that the θ parameter is unknown over the time steps, we only know that it is contained in an interval $\left[\theta_t^-, \theta_t^+\right]$ with probability $\alpha_t \in [0, 1]$. This adds another layer of noise to the

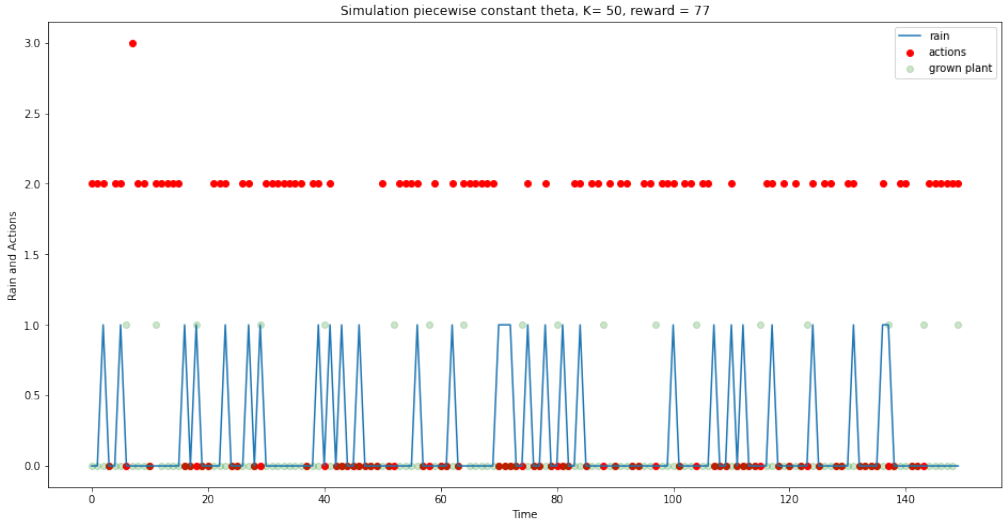


Figure 11: Episode evaluation of T=150 and K=50

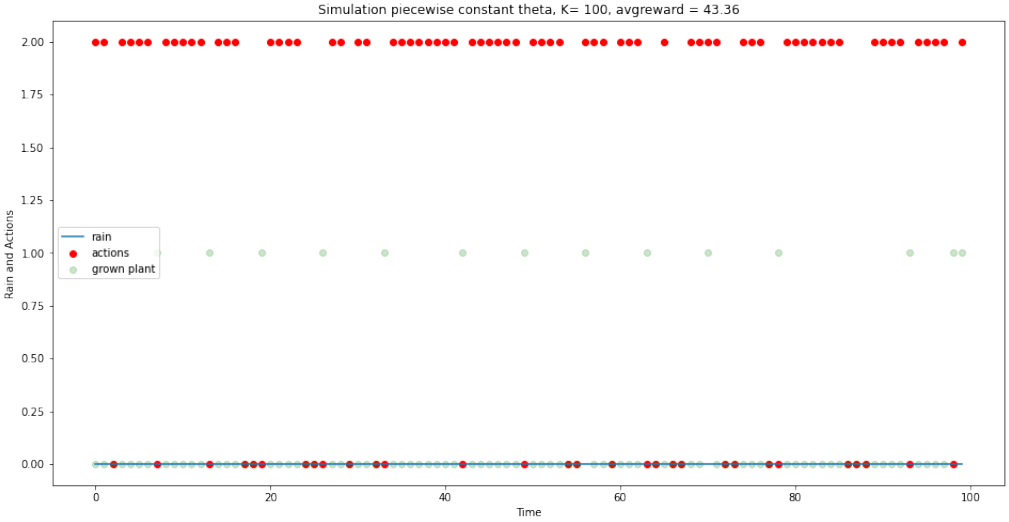


Figure 12: Episode evaluation of T=100 and no rain

environment because as we need to estimate to values now: the value θ_t and $c_t \sim \mathcal{B}(\theta_t)$. In the case of a constant $\left[\theta_t^-, \theta_t^+\right]$ and α_t over the time steps, the suboptimal policy returned by the methods $V^{opt}(.)$ and $Pol(.)$ is parametrized in the rain state $\pi_c^*(\mathcal{S})$ and is the same as the one returned in the case explained in Section 5.3. The worst scenario happens when the observed sequence c_t is 0 or 1 $\forall t \in T$, or for at least 80% of the times, this would converge to a biased policy and will perform worse in another environment where the dynamics of the rain forecast are the opposite. A best scenario would be to have a more heterogeneous sequence of c_t.

7 Case 4: Partially-observed model

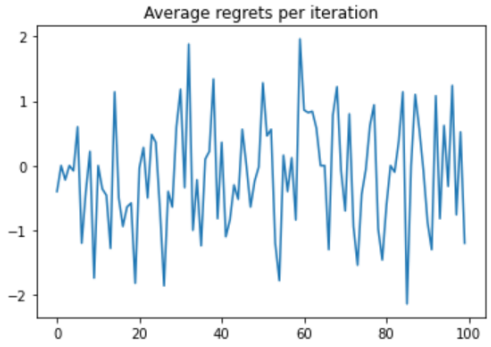
In this configuration the cost of observation of the water level C' is strictly positive, and the agent must choose the optimal action $(a, o) \in \{0, 1\}^2$ to maximize the reward. In Section 3.5, we presented our solution to model the partially-observed learning through the computation of a mixture of probability of the different values of w given only the height. More precisely, for the values based methods, the possible transition

distribution evaluated for each state s_t is the mixture of the distribution given only the height h of the plant and the action a, if the agent chooses to not observe w_t, thus $o = 0$, otherwise the right transition distribution is shown to the agent. The goal is to find the policy $\pi_c : \mathcal{S} \to \{0, 1\}^2$. In this configuration, the exact methods (Value iteration and Policy iteration) do not converge, for this reason we decided to implement the Q-Learning algorithm as defined in Section 3.6, and use it to search a suboptimal policy in this configuration. The algorithm struggles to find a stable policy because of the new constraints added. In the cases stated in Section 5.3 and Section 6, the algorithm does not converge at all to an acceptable policy, which is a policy that allows sowing. Below are the results of two experiments done. A state $s \in$ S such that $\exists a\ \pi_c^*(s) = (a, 1)$, is a state which water level needs to be checked before taking any decision that could lead to an overwatered plant. Therefore, an intuitive optimal policy would set to 1 those states having height close to the maximum, with a water level close to the threshold of overwatering.

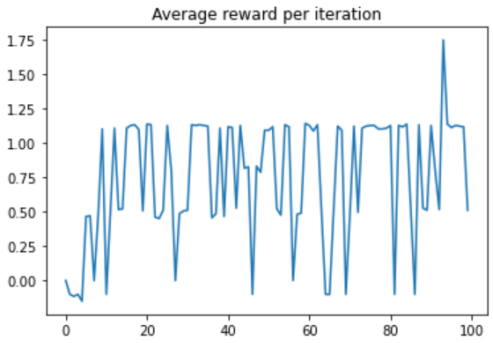
- **No rain ($c_t = 0$) in Partially-observed model:** $T = 100, G = 5, C = 0.1, C' = 0.1, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 200$

```
policy is:
c= 0:
        state(0) = (1, 0)
        state(1) = (1, 0)
        state(2) = (1, 0)
        state(3) = (0, 0)
        state(4) = (1, 1)
        state(5) = (1, 0)
        state(6) = (0, 1)
        state(7) = (0, 0)
        state(8) = (1, 1)
        state(9) = (1, 0)
        state(10) = (0, 0)
        state(11) = (0, 1)
        state(12) = (1, 0)
        state(13) = (1, 0)
        state(14) = (1, 1)
        state(15) = (0, 1)
        state(16) = (1, 1)
        state(17) = (1, 0)
        state(18) = (0, 0)
        state(19) = (0, 0)
        state(20) = (1, 1)
        state(21) = (0, 1)
        state(22) = (1, 0)
        state(23) = (1, 0)
```

(a) Policy



(b) Average regret per iteration



(c) Average reward per iteration

Figure 13: Results using Q-Learning

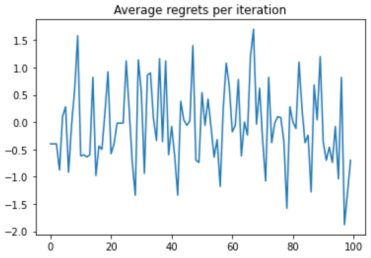
- **Constant $\theta = 0.2$:** $T = 100, G = 5, C = 0.1, C' = 0.1, C'' = 0, \gamma = 0.9, \epsilon = 10^{-4}, I = 200$

```
c= 0:
        state(0) = (1, 1)
        state(1) = (1, 0)
        state(2) = (0, 0)
        state(3) = (0, 1)
        state(4) = (0, 1)
        state(5) = (1, 1)
        state(6) = (0, 0)
        state(7) = (1, 1)
        state(8) = (0, 1)
        state(9) = (1, 0)
        state(10) = (0, 0)
        state(11) = (0, 1)
        state(12) = (1, 0)
        state(13) = (1, 0)
        state(14) = (0, 0)
        state(15) = (1, 1)
        state(16) = (0, 1)
        state(17) = (1, 0)
        state(18) = (1, 0)
        state(19) = (0, 0)
        state(20) = (1, 0)
        state(21) = (1, 0)
        state(22) = (1, 0)
        state(23) = (0, 1)
```

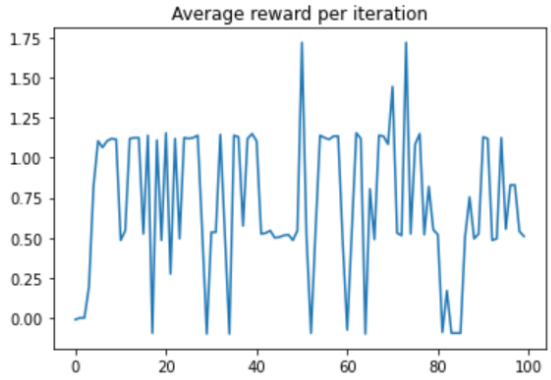
(a) Policy

```
c= 1:
        state(0) = (1, 1)
        state(1) = (1, 0)
        state(2) = (0, 0)
        state(3) = (0, 1)
        state(4) = (0, 1)
        state(5) = (0, 1)
        state(6) = (0, 0)
        state(7) = (0, 0)
        state(8) = (1, 1)
        state(9) = (1, 0)
        state(10) = (0, 0)
        state(11) = (1, 1)
        state(12) = (1, 0)
        state(13) = (1, 0)
        state(14) = (0, 0)
        state(15) = (1, 1)
        state(16) = (0, 1)
        state(17) = (1, 0)
        state(18) = (1, 0)
        state(19) = (0, 0)
        state(20) = (1, 0)
        state(21) = (1, 0)
        state(22) = (1, 0)
        state(23) = (0, 0)
```

(b) Policy



(c) Average regret per iteration



(d) Average reward per iteration

Figure 14: Results using Q-Learning

- $\theta = 0.5$ The optimal policy for this type of configuration is: (1,1),(1,1),(0,1),(0,0),(1,1),(1,1),(0,1),(0,0),(1,1), (1,1),(0,1),(0,0),(1,1),(1,1),(0,1),(0,0),(1,1),(1,1), (0,1),(0,0),(0,0),(0,0),(0,0)]

8 The optimal sowing plant problem

In order to solve this problem, we decided to try to approach: The first approach is based on ACKR method while the second one is based on a history-based value iteration.

8.1 Approach 1

Given an optimal policy π^* that decides to water a plant depending on the current state and a complete forecasting of the rain, we decided to train another agent that receives as input the forecast and decides at which time $d \in \{1, \ldots, T\}$ it should sow.
The action space of this agent is $\{k \in N : k \leq T\}$ and the observation space is $\{0, 1\}^T$ for the context (E), $\{\theta_t\}^T$ for context (P) and $\{\theta_t^+, \theta_t^-\}^T$ for context (C), we define a reward function:

$$r_{sow}(\pi^*, h, w, c) = G[\sum_{t=d}^{T} r(h, w, c, a) > \lambda] - C$$

We decided to train this agent using again ACKR during 4000 episodes of T=100 K=20 in context (E) with = 30. After training we tested in 200 episodes and we got an average reward of 32 which reflects that our agent learnt to select d which allow him to get at least a reward value of 30

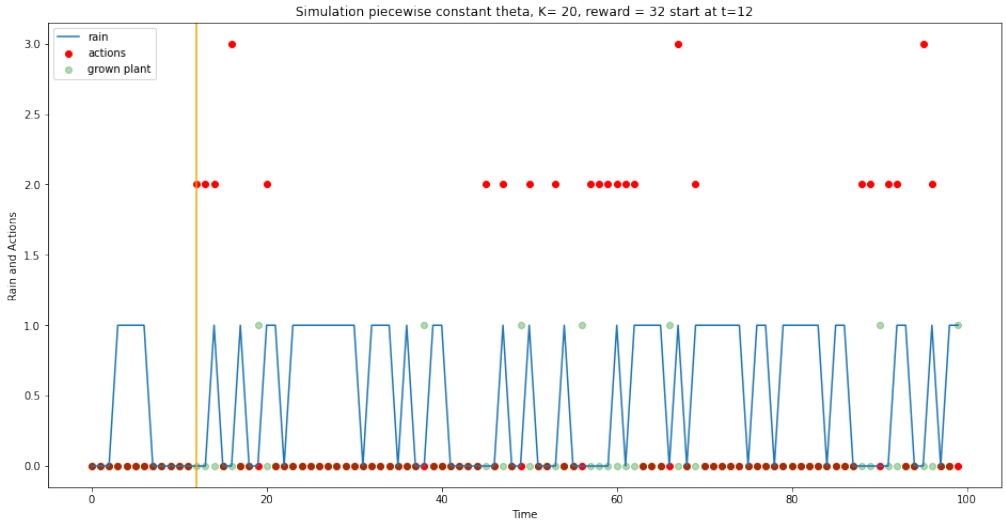


Figure 15: Starting actions at time = 12

8.2 Approach 2

The second approach is based on the intuition that we can start sowing if and only if we have enough water level in the terrain. We need to estimate at a current time step, the water level using the capture index of the watering index $\mathbf{p}^{\mathcal{W}}_{c,a}(w) = (1-q)\delta\left(|c+a+w-1|_{0:3}\right) + q\delta\left(|c+a+w|_{0:3}\right)$, thus we need to keep track of the past raining days as the action of watering is null until we start sowing. If the estimation of w is correct at each timestep, and the optimal policy π^* is not a suboptimal one, and the dynnamics of the system are deterministics, the optimal sowing date would be preceded by at least two raining days. This was almost never the case in our experiments when the environment was getting more and more noisy.

9 Conclusion

In this project, we attempted to bring a novel solution to the plant watering problem through modelling and experiments using Reinforcement Learning strategies. We analysed different situations in order to give an explanation to the underneath phenomena in the results obtained. We decided to start with simple methods to then compare them with novel RL methods by we could not complete fully the project because of the several parallel projects we had and the multiple tests and debugging loops we had on our submitted code. Therefore, we conclude proposing alternative solution that we did not have the time to try on this problem: The non-stationary environment faced in this problem could be addressed with Context Q-Learning, an algorithm that we had the chance to discover during the Lecture project for non-stationary environment. Also, the variability of the rewards/regrets and value iteration shown in many of our results, is the result of having values over the stop threshold as the algorithm is not converging to that value. We can increase that value and therefore obtain a policy that may not be acceptable, which is a policy that does not allow sowing. Remains the The many-plant watering problem and The planning switch-cost problem that we did not have the time to address, thus leave them for a future work.