Optimizing fault search in power grid outages through Reinforcement Learning

Master's degree in *Data Science and Scientific Computing*

CANDIDATE:

Angela Carraro

SUPERVISOR:

Antonio Celani, ICTP

CO-SUPERVISORS:

Andrea Zancola, AcegasApsAmga

Luca Bortolussi, UniTS

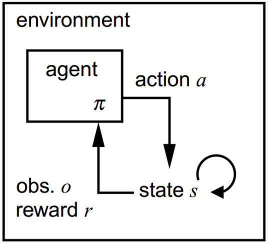
DSSC - UNITS, SISSA, ICTP

Aim of the project

We want to optimize the fault search in power grid outages through Reinforcement learning.

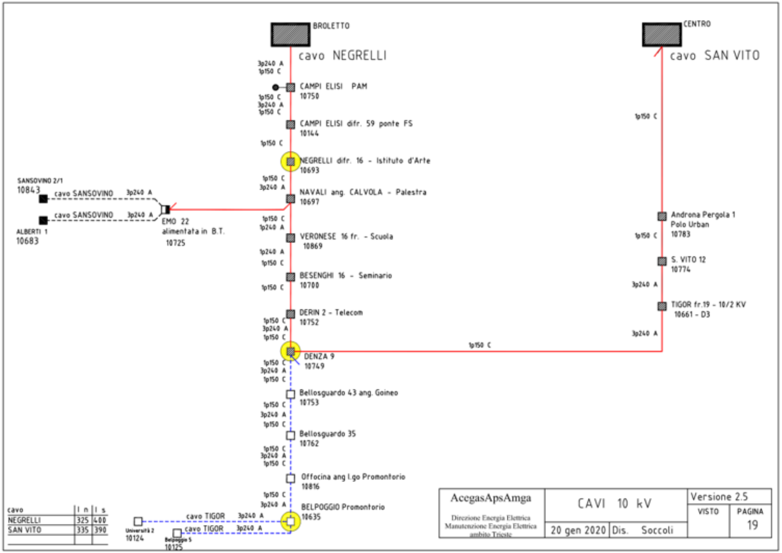
To do so, we use a policy gradient method in a Partially Observable Markov Decision Process, or POMDP. In particular, we perform a gradient descent in the policy space, using a parametrized policy specifically chosen so to it does not depend on the position of the fault, which is a hidden state variable.

Schematic representation of a POMDP agent interacting with the environment.

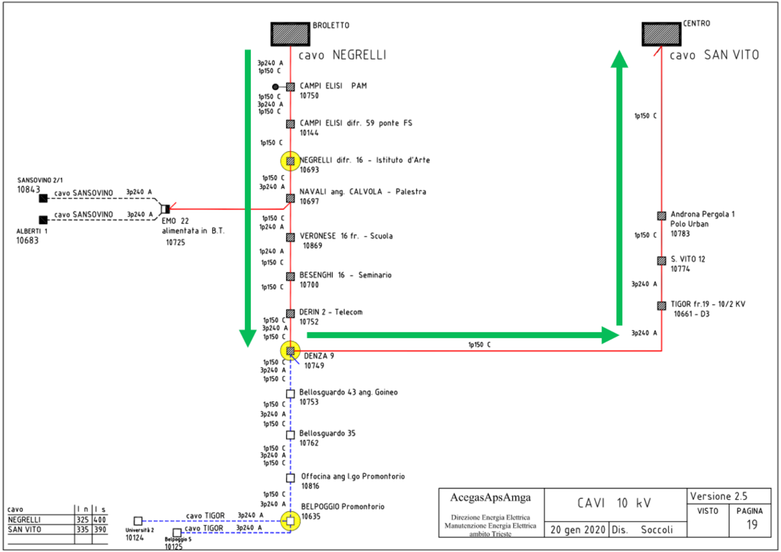


The problem

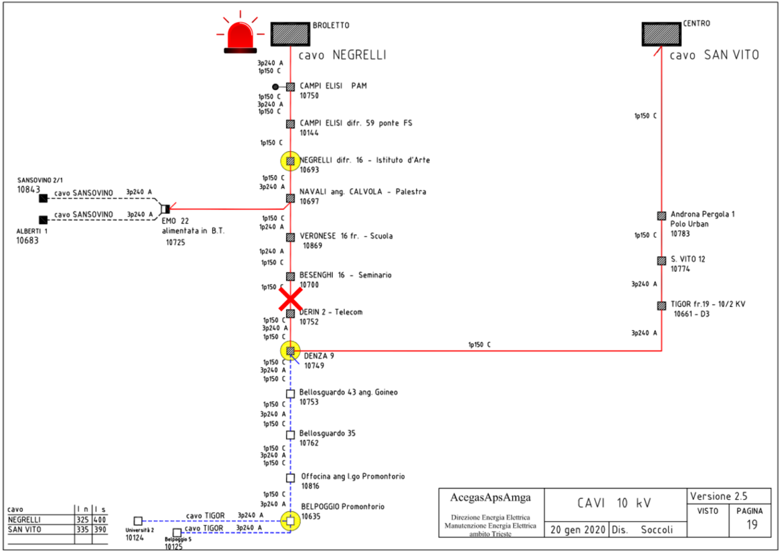
Example of a power line in medium voltage



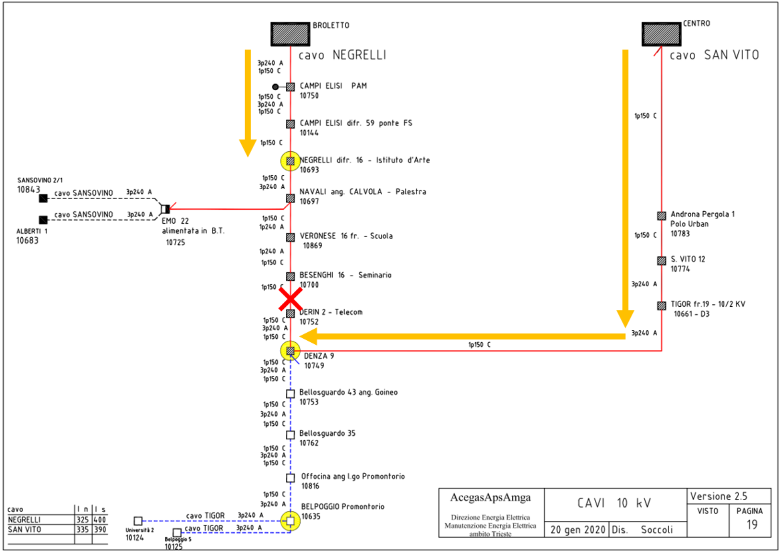
The energy flow in the standard set-up



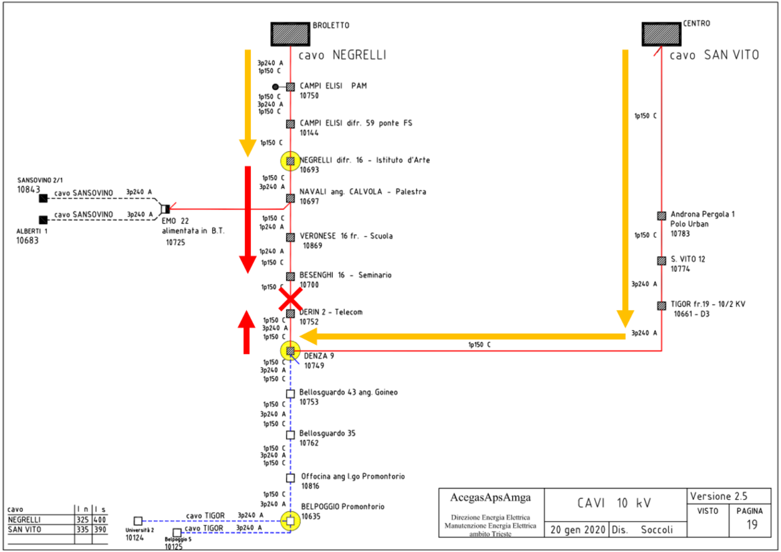
Failure scenario



Failure scenario



Failure scenario



The mathematical framework

The POMDP

Since we don't know where the fault is, we have a problem with **partially observable states**: a partially observable Markov decision process (POMDP).

We are given a set of disconnected substations, $\mathcal{C}$, between two remotely controlled substations, which are already reconnected, so are not included in this set.

- The state is $s = (x_g, v_k, \{v\})$, where x_g is the position of the fault, $v_k \in \mathcal{C}$ is the substation in which the technician is, and $\{v\}$ is the set of the still disconnected substations after the technician operates in the current substation v_k. We have that the variable x_g is **hidden**, while the variables v_k and $\{v\}$ are **observable**.
- The observation is $o = (v_k, \{v\})$. We define it as a function of s: $o(s) : s = (x_g, o) \mapsto o$.
- The action is the intervention we do in the specific substation we decide to visit, so $a \in \mathcal{C}$. Actually, since we visit only disconnected substations, we have that $a \in \{v\}$.
- The next state is $s' = (x_g, v_{k+1} = a, \{v'\})$, where $\{v'\}$ is the set of disconnected substations after the technician operates in substation v_{k+1}. We have that $\{v'\} \subseteq \{v\} \setminus a$, so $\{v\}$ decreases after each action, thus the process surely terminates.

- The reward is the **cost** of going in a certain substation (time *in seconds*) multiplied by the number of disconnected users.

  $d_{v_k,v_{k+1}} \longrightarrow$ time *in seconds* to go from the substation v_k to the next substation v_{k+1}.

  $n_k \longrightarrow$ number of users still disconnected before operating in the substation v_{k+1}.

  We have $n_k = \sum_{v \in \{v\}} u_v$, where u_v is the number of users underneath substation v.

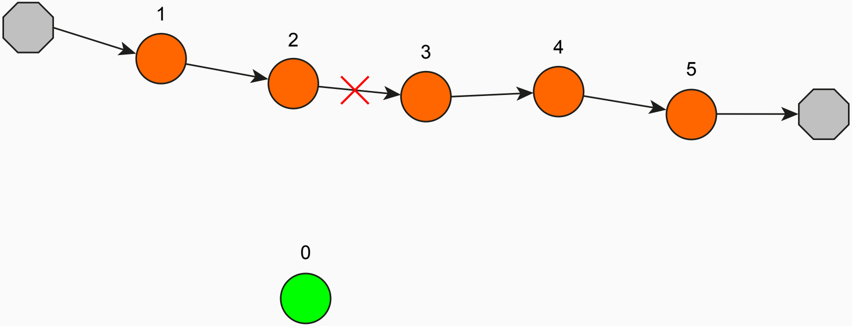
$$r\Big(s = (x_g, v_k, \{v\}), a\Big) = d_{v_k,a} \cdot n_k = d_{v_k,a} \cdot \sum_{v \in \{v\}} u_v \,. \tag{1}$$

For now, in the cost we will ignore the cost of establishing if the fault is before or after the substation in which the technician is, which is complicated and might rise the total cost significantly. This is due to a lack of data. We are developing a Telegram bot to collect them, so to be able to estimate this cost.

The POMDP

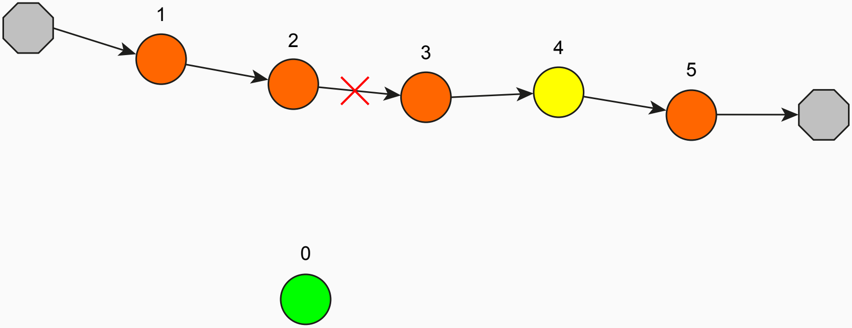
When the fault occurs, the technician can be everywhere: at home if it is the middle of the night, at the company, be around, etc. So we introduce an extra dummy substation, called substation 0, that is the position of the technician when the fault occurs.
So the initial state is always $s_0 = (\, x_g, o_0 = (0, \mathcal{C})\,)$, thus we have $|x_g| = 2|\mathcal{C}| + 1$ initial states, one for every possible position of the fault.

Instead, the terminal state is of the form $s_t = (x_g, v_k, \varnothing)$, where we have that, if the fault is on a cable, v_k will be one of the two substations at the ends of that faulty cable, so we would have two terminal states, while if the fault is in a substation, v_k would be that exact substation, so the terminal state would be only one.



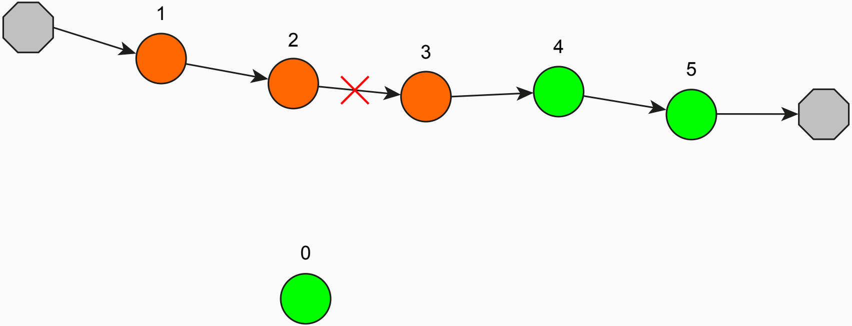
The fault has just occurred. We are in substation 0 and all the substations are disconnected (orange). Initial state: $s_0 = (2\text{-}3, 0, \mathcal{C} = \{1, 2, 3, 4, 5\})$.

Example



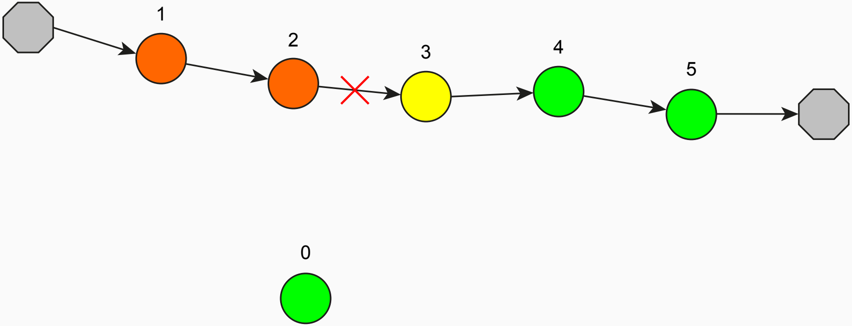
We visit substation 4 (yellow). Action: $a_0 = 4$.

Example



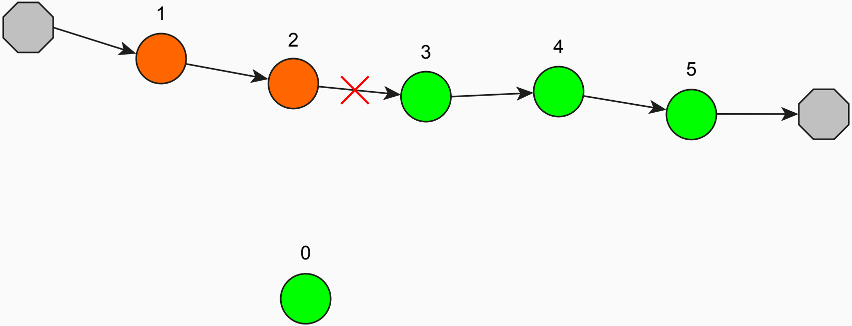
We reconnect substations 4 and 5 (green). State: $s_1 = (2\text{-}3, 4, \{1, 2, 3\})$.

Example



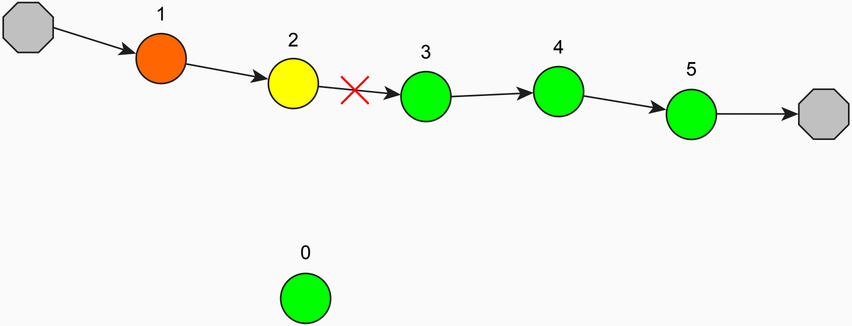
We visit substation 3 (yellow). Action: $a_1 = 3$.

Example



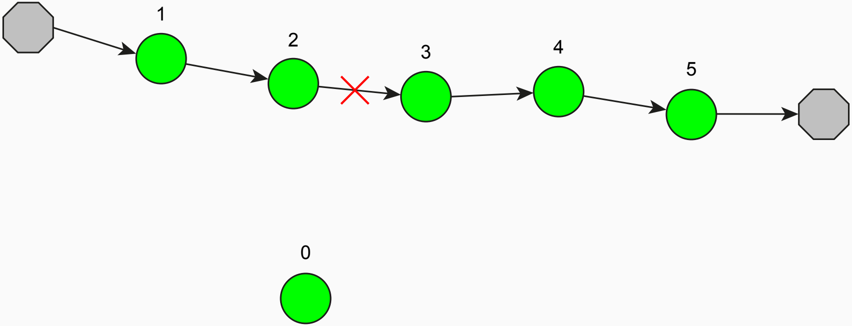
We reconnect substation 3 (green). State: $s_2 = (2\text{-}3, 3, \{1, 2\})$.

Example



We visit substation 2 (yellow). Action: $a_2 = 2$.

Example



We reconnected substations 1 and 2 (green). All the substations are reconnected.
Terminal state: $s_3 = (2\text{-}3, 2, \varnothing)$.

The model

The transition probability

We will perform policy iteration, in particular gradient descent in the policy space. So we **parametrize the policy**, and we impose that the parameters depend only on the observable variables, and we try to find the best policy in this subspace in order to optimize the POMDP.

The system is deterministic, so, given an admissible action a, we will surely perform it and end up in the state to which that action leads.

In our case we have that the transition probability is equal to 1 only when, starting from state $s = (x_g, v_k, \{v\})$, the new substation v_{k+1} of the next state $s' = (x_g, v_{k+1}, \{v'\})$ is equal to the action $a \in \{v\}$ that we took:

$$p(s' \mid s, a) = \mathbb{I}\big(s' = \sigma(s, a)\big) = \begin{cases} 1 & \text{if } v_{k+1} = a \\ 0 & \text{if } v_{k+1} \neq a \end{cases} . \tag{2}$$

The policy

The policy depends only on the observable states, so it doesn't know where the fault is.
Let's define a parameterized policy using the *soft-max (i.e., Boltzmann) distribution*:

$$\pi\Big(a \;\Big|\; o = (v_k, \{v\}), \boldsymbol{\theta}\Big) = \frac{e^{\theta_{o,a}}}{\sum_{b \in \{v\}} e^{\theta_{o,b}}}\,. \tag{3}$$

where $\boldsymbol{\theta}$ are the parameters for each observable o and action a:

$$\boldsymbol{\theta} = (\theta_{o,a})_{o \in \mathcal{O}, a \in \mathcal{A}} = \begin{pmatrix} \theta_{o_1,a_1} & \theta_{o_1,a_2} & \cdots & \theta_{o_1,a_N} \\ \theta_{o_2,a_1} & \theta_{o_2,a_2} & \cdots & \theta_{o_2,a_N} \\ \vdots & & & \vdots \\ \theta_{o_{|O|},a_1} & \theta_{o_{|O|},a_2} & \cdots & \theta_{o_{|O|},a_N} \end{pmatrix} \tag{4}$$

(where $N = |\mathcal{C}|$ is the number of substations, so the number of possible actions).

The policy can not depend on the position of the failure, otherwise we would have automatically solved the problem: the solution would be to go in the substation in which the failure is or at the substations at the ends of the faulty electrical cable.

The action value function or quality of the state-action pair, thanks to the Bellman equation, is:

$$Q_\pi(s,a) = \sum_{s'} p(s'|s,a) \left(r(s,a,s') + \sum_{a'} \pi(a'|o(s'),\boldsymbol{\theta}) Q_\pi(s',a') \right) . \tag{5}$$

Given that the state is $s = (\, x_g, o = (v_k, \{v\})\,)$, the action is $a \in \{v\}$ and the new state is $s' = \sigma(s,a) = (\, x_g, o' = (v_{k+1} = a, \{v'\})\,)$, and given (1) and (2), it becomes:

$$Q_\pi(s,a) = \left(d_{v_k,a} \cdot n_k + \sum_{a' \in \{v'\}} \pi\Big(a' \Big| o\big(\sigma(s,a)\big), \boldsymbol{\theta}\Big)\, Q\big(\sigma(s,a), a'\big) \right) . \tag{6}$$

η function

We define $\rho_0(s')$ as the probability of starting in the state s'. In our case

$$\begin{aligned}\rho_0\Big(s = (\,x_g, o = (v_k, \{v\})\,)\Big) &= \mathrm{Pr}(x_g)\mathbb{I}(o = o_0 = (0, \mathcal{C})) \\ &= \frac{1}{2|\mathcal{C}| + 1}\mathbb{I}\big(v_k = 0, \{v\} = \mathcal{C}\big)\,;\end{aligned} \tag{7}$$

so ρ_0 is uniform in x_g since it doesn't depend on the position of the fault.

Besides, we define the average number of time steps that the agent spends in state s' before the process dies as

$$\begin{aligned}\eta_\pi(s') &:= \rho_0(s') + \sum_s \eta_\pi(s) \sum_a \pi(a|o(s), \boldsymbol{\theta})p(s'|s, a) \\ &= \frac{1}{2N + 1}\mathbb{I}\big(v_k = 0, \{v'\} = \mathcal{C}\big) + \sum_{s \in \mathrm{pa}(s')} \eta_\pi(s)\pi(v_{k+1}|o(s))\,,\end{aligned} \tag{8}$$

Notice that both (6) and (8) are linear systems.

The performance measure

Let's define the performance measure $J_\pi(\boldsymbol{\theta})$ as the sum of all the costs we incur summed in time until the process is concluded:

$$J_\pi(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} r(s_t, a_t, s_{t+1})\right] = \sum_s \rho_0(s) \sum_a \pi_{\boldsymbol{\theta}}(a|o(s), \boldsymbol{\theta}) Q_{\pi_{\boldsymbol{\theta}}}(s, a) \tag{9}$$

Thanks to the policy gradient theorem we have that

$$\nabla_{\theta_{o',a'}} J_\pi(\boldsymbol{\theta}) = \sum_s \eta_{\pi_{\boldsymbol{\theta}}}(s) \sum_a Q_{\pi_{\boldsymbol{\theta}}}(s, a) \nabla_{\theta_{o',a'}} \pi(a|o(s), \boldsymbol{\theta})\,. \tag{10}$$

To optimize J we perform a gradient descent on $\boldsymbol{\theta}$ with *learning rate* α:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}_k)\,, \tag{11}$$

Problem: we can compute Q_π and η_π only if we know the position of the fault x_g!

Given the equation for the policy in (3), we have that its derivative is

$$\frac{\partial}{\partial \theta_{o',a'}} \pi\Big(a \mid o = (v_k, \{v\}), \boldsymbol{\theta}\Big) = \delta_{o',o} \left(\delta_{a',a} - \pi(a'|o)\right) \pi(a|o) \tag{12}$$

So, we sum all the values of the gradient that have the same observation!

Therefore, the equation of the gradient becomes

$$\begin{aligned}
\nabla_{\theta_{o',a'}} J_\pi(\boldsymbol{\theta}) &= \sum_s \eta_\pi(s) \sum_a Q_\pi(s,a) \nabla_{\theta_{o',a'}} \pi(a|o(s)) \\
&= \sum_s \eta_\pi(s) \sum_a Q_\pi(s,a) \delta_{o',o(s)} \Big(\left(\delta_{a',a} - \pi(a'|o(s))\right) \pi(a|o(s)) \Big) \\
&= \sum_{x_g} \eta_\pi((x_g, o')) \sum_a Q_\pi((x_g, o'), a) \left(\delta_{a,a'} - \pi(a'|o')\right) \pi(a|o'),
\end{aligned} \tag{13}$$

Idea of the algorithm

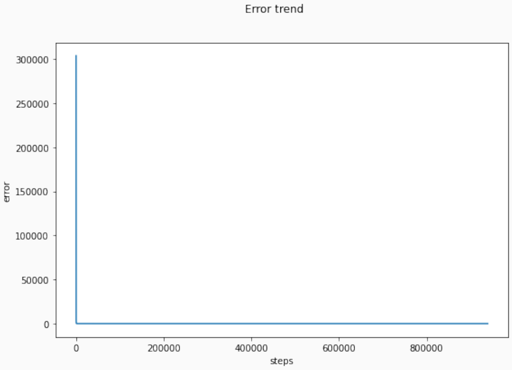
We start from a certain policy, for example *random policy*, in which all the parameters θ are equal to 0: $\boldsymbol{\theta} = \mathbf{0} = (0, 0, \dots, 0)$, so that all actions have an equal probability of being selected

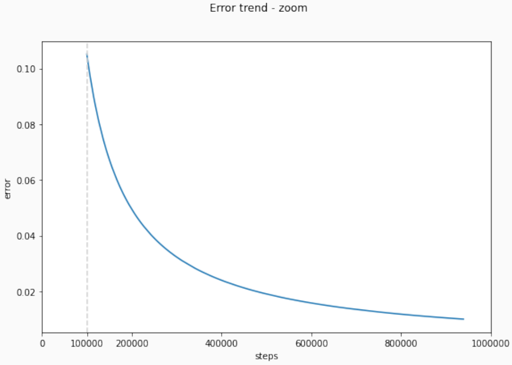
$$\pi\left(a \mid o(s), \boldsymbol{\theta}\right) = \frac{e^{\theta}}{\sum_{b \in \{v\}} e^{\theta}} = \frac{e^{\theta}}{e^{\theta} \sum_{b \in \{v\}} 1} = \frac{1}{|\{v\}|}, \tag{14}$$

Then we compute the gradient using (13):

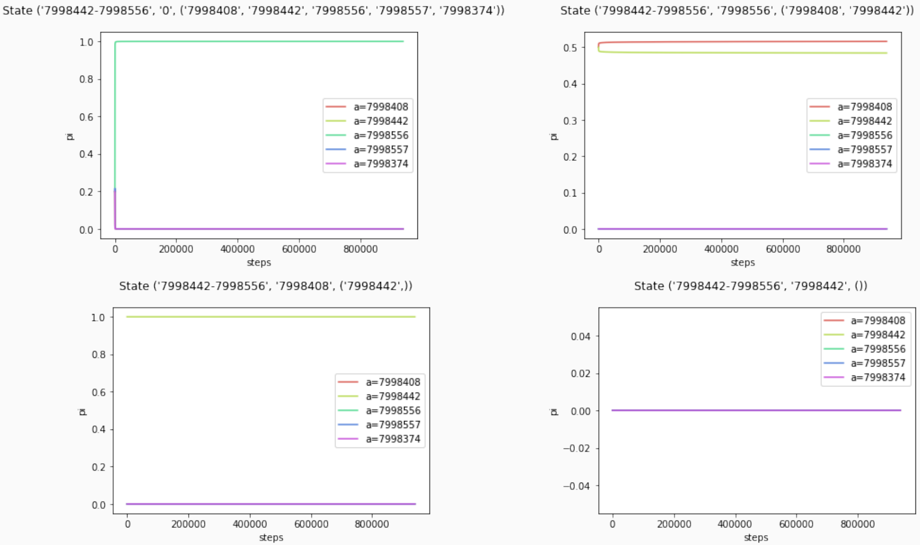
as we saw in (12), the gradients of the policy are simple computations, while to compute Q_π and η_π you have to solve the linear equations (6) and (8) for the current policy, so they have to be solved at each iteration of gradient descend.

Error trend


Error trend
300000
250000
200000
150000
100000
50000
0
error
0
200000
400000
600000
800000
steps


Error trend - zoom
0.10
0.08
0.06
0.04
0.02
error
0
100000
200000
400000
600000
800000
1000000
steps

Policies in a real example



Thank you for your attention!

☺

References

- D. Aberdeen and Jonathan Baxter. “Scaling Internal-State Policy-Gradient Methods for POMDPs”. In: *International Conference on Machine Learning*. 2002.
- Jan Peters and J. Andrew Bagnell. “Policy Gradient Methods”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 774–776. ISBN: 978-0-387-30164-8.
- Matthijs T. J. Spaan. “Partially Observable Markov Decision Processes”. In: *Reinforcement Learning: State of the Art*. Ed. by Marco Wiering and Martijn van Otterlo. Springer Verlag, 2012, pp. 387–414.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts (MA): MIT Press, 2018.