Optimisation of medical trajectory to avoid diagnostic wandering in rare diseases.

Master MVA, Ecole Normale Supérieure Paris Saclay,

Internship at CMAP, Ecole Polytechnique.

Pierre Clavier

Supervisors : Erwan Le Pennec and Stéphanie Allassonnière.

Optimisation de trajectoire médicale pour minimiser l'errance diagnostique.

Version Française

Résumé

Un patient atteint d'une maladie rare en France doit en moyenne attendre deux ans avant d'être diagnostiqué. Cette errance médicale est fortement préjudiciable tant pour le système de santé que pour les patients dont la pathologie peut s'aggraver. Cette errance est due à la multitude de symptômes possibles caractérisant ces maladies et au fait qu'elles touchent souvent plus d'un organe. Cela se traduit par des consultations "erratiques" de différents spécialistes. L'objectif de cette thèse de Master est de proposer une méthode permettant de mieux guider ses patients afin de diminuer le temps nécessaire pour le diagnostic. On s'appuiera pour cela des techniques issues du Reinforcement Learning pour le choix des examens à faire/spécialistes à contacter en fonction du passé et l'utilisation d'un simulateur déduit des parcours patients mis à disposition pour l'étude. Pour cela on utilisera notament des méthodes basées sur le Distributional Reinforcement Learning (DRL) où on modélise la distribution des récompenses et pas la moyenne. Enfin, on s'intéressera à des méthodes de réduction des risques en cherchant des trajectoires de patient ayant une variance plus faible toujours dans le cadre du DRL.

Optimisation of medical trajectory to avoid diagnostic wandering in rare diseases.

English version

Abstract

A patient suffering from a rare disease in France has to wait an average of two years before being diagnosed. This medical wandering is highly prejudicial both for the health system and for the patients whose pathology may worsen. This is due to the multitude of possible symptoms that characterise these diseases and the fact that they often affect more than one organ. This results in "erratic" consultations in different medical departments or specialists. The objective of this Master thesis is to propose a method to better guide patients in order to reduce the time needed for diagnosis. This will be achieved by using Reinforcement Learning techniques for the choice of examinations to be carried out or services to be contacted according to the past and the use of a simulator deduced from the patient pathways made available for the study. For this we will use methods based on Distributional Reinforcement Learning (DRL) where we model the distribution of returns and not the average. Finally, we will look at risk reduction methods with the aim of finding patient trajectories with a lower variance using again DRL framework.

Acknowledgements

I would like to warmly thank my two tutors and professors, Erwan Le Pennec and Stephanie Allassonnière for their support and help during my Master Thesis. I particularly appreciated the quality of their supervision, their advises and support. A thought for all the teachers of the MVA master's degree with whom I was able to have exchanges which taught me a lot through the quality of their courses despite the health conditions this year. I would like to pay special attention to my parents and my sister who have supported me in my studies since the beginning and who have encouraged me to persevere and go in the direction that interests me the most without any other considerations. Thanks to them for giving me time to find out what I want to work for during the 7 last years after the baccalaureate.

Many thanks also to Stephanie's other students with whom I was able to have scientific and informal conversations : Vianney, Clement, Clément, Fleur and Solange. I would like also to thanks Linus who will work with me in HeKA team for the next three year. Finally, I would like to warmly thank Julia, with whom I spent a superb year at the MVA in all the projects I carried out with her.

Notations

- S_t, A_t, R_t : State, Action, and reward at time t of one trajectory. Capital letter denotes random variables.
- s_t, a_t, r_t : Realizations of previous random variables.
- $\mathcal{A}$: The action space.
- $\mathcal{S}$: The state space.
- $\tau = (s_0, a_0, r_0, s_1, \ldots)$: A trajectory of state and chosen action.
- $0 < \gamma \leq 1$: Discount factor, a penalty to uncertainty of future rewards.
- $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$: Return or discounted future reward.
- $p\left(s', r \mid s, a\right)$: Transition probability of getting to the next state s' from the current state s with action a and reward r.
- $\pi(a \mid s)$: Stochastic policy (the agent strategy); $\pi_\theta(.)$ is a policy parameterized by θ.
- $\mu(s)$: A deterministic policy; we can also label this as $\pi(s)$, but using a different letter gives better distinction so that we can easily tell when the policy is stochastic or deterministic without further explanation.
- $V\big(s\big)$: State-value function measures the expected return of state s $V_w\big(.\big)$ is a value function parameterized by $\boldsymbol{w}$. The value of state s when we follow a policy π : $V_\pi\big(.\big) = \mathbb{E}_{a\sim\pi}\big[G_t | S_t = s\big]$.
- $Q\big(s, a\big)$: Action-value function is similar to $V(s)$, but it assesses the expected return of a pair of state and action (s, a). When we follow a policy π ; $Q\big(s, a\big) = \mathbb{E}_{a\sim\pi}\big[G_t | S_t = s, A_t = a\big]$.
- $A\big(s, a\big) = Q\big(s, a\big) - V\big(s\big)$: Advantage function for action a at state s.

Contents

1 Introduction

1.1 Context and Motivation of the Problem

The Erradiag report from the Rare Disease Alliance showed that in France patients with a rare disease (disease whose prevalence is less than 1/2000 according to European threshold) will have to wait an average of 2 years between the onset of the first symptoms and the diagnosis of their disease. For more than a quarter of them, this period is more than 5 years. This lapse of time, known as diagnostic erraticism is a scourge for the health system because of the additional economic costs, e.g. multiplication of unnecessary medical examinations and treatments and human costs such as aggravation of the pathology due to inadequate care. Patients suffering from rare diseases are particularly affected by diagnostic erraticism due to the the multifactorial nature of these diseases: a rare disease often affects several different organs and diagnosis therefore requires a multidisciplinary approach. Furthermore, a practitioner who does not work in an expert centre will probably never see most rare diseases in the course of his or her career. Although rare diseases are by definition uncommon, there are many patients, this is why 3 million people are affected in France and between 263 and 446 million worldwide (Nguengang-Wakap et al., 2019). This fact is due to the very large number of existing rare diseases: no less than 9000 are listed in OrphaNet 3, the portal dedicated to rare diseases.

France is a pioneer in the fight against rare diseases, launching the first National Rare Disease Plan in 2005, with the creation of rare disease reference centres (CRMR). These multidisciplinary centres aim to provide better care for patients suffering from rare diseases. The report Erradiag reminds us that, despite the progress made following the creation of these centres, a significant number of patients are still referred very late. A quarter of patients wait more than 4 years after the first symptoms before the search for a diagnosis is finally initiated. A better orientation of patients in the health system is therefore necessary. With this in mind, the idea of creating reliable medical specialist recommendations emerged. Based on the patient's history of recent medical examinations, we would like to be able to recommend a specialist to go and see in order to get the best diagnosis. To to reach our goal, several elements are required. First of all, we need patient health data as precise as possible. In a second step, we will look at a system which recommend a specialist to consult. A *parametric statistical model* will be used for this purpose. By using the data of the patient's pathway, which are sequential data and thus have a temporality, we will seek through *Reinforcement Learning* framework to model our recommendations and estimate the parameters of our model. Finally, the path of a patient from one doctor to another will also be modelled through a statistical model that takes into account the patient's history. These 3 elements allow for relevant recommendations that we hope will be as useful as possible for specialists. Finally, a validation of the recommendations by expert knowledge is necessary.

Recently, Medical data, in the form of expert databases such as Orphanet or patient pathway data shave been made available and consultations/reimbursements with the appearance of the electronic health record have been structured and are now more accessible for research, making it possible to improve patient pathway models.

Note that this Master Thesis, although not yet using real medical data, will be followed by a PhD Thesis which will be also joint-work with the AFM-Telethon databases and with the help of medical doctors identified by the AFM (e.g. the AIM neuromuscular disease specialists). As a first step, help on which pathologies to start the analysis from in our model is also needed. This focus will also come from our partnerships. Finally, the interpretability of the results is fundamental at the end of the training of the reinforcement algorithm. Once the policy of the algorithm has been learned (i.e. the estimated optimal decision tree), it must have a medical reality and be validated by physicians. This validation will also be conducted with our clinical partners.

1.2 Purpose of the thesis

With the advent of Reinforcement Learning (RL), which has taken a big step forward with Deep Reinforcement Learning and the ability to master games such as Atari, chess or go Mnih et al. (2013a); Schrittwieser et al. (2019), more and more algorithms have emerged with increasingly impressive performance. The performance of deterministic (DDPG and TD3 Lillicrap et al. (2015); Fujimoto et al. (2018)) or stochastic policies (A2C, PPO, SAC... Schulman et al. (2017); Mnih et al. (2016a); Haarnoja et al. (2018))for environments with continuous or discrete state and action spaces is constantly improved.

Machine Learning and not especially RL have also been used for identify rare disease patient such as in Colbaugh et al. (2018a,b) but not using patient pathway. Moreover RL have been used in Healthcare in many applications such as in Dynamic Treatment Regime Chakraborty and Murphy (2014) for Chronic Diseases or for Automated Medical Diagnosed. Medical diagnosis is a mapping process from a patient's information such as treatment history, current signs and symptoms to an accurate clarification of a disease Ling et al. (2017). However, RL has only been widely used in the diagnosis of rare diseases. Remi besson's thesis Besson (2019); Logé et al. (2020) use DQN for decision making strategy for antenatal sonographic screening of fetal abnormalities for example. In this thesis, the aim is to adapt and continue this work for adults using pathway of patients. *Can we adapt this work and modelisation to our problem with patient who are adults ?*

Another main issue is the variability of policies. Indeed in the classical problem we are looking for the maximum of the expectation of the return given a policy. However the variance of the return is not taken into account whereas it is of capital importance considering human life. It is not acceptable to have one large return and another very small one. We would like

to reduce this variability as much as possible. The Distributional RL framework comes into play insofar as if one has the distribution return, it is then possible to easily calculate the variance and bring it into play in the optimisation problem. So the questions are how do we reduce the variance in our policy and is it possible to do in a sense *Safe Reinforcement Learning with lower variability ?*

1.3 Thesis disposition

The Master thesis is divided into four parts: the first one gives a first idea of modelisation of Patient Pathway for medical recommendation. A modelisation via Bayesian Network of our environment will be conducted and the question of how use prior knowledge to calibrate a model will be discussed. An accurate learning of our environment is of capital importance in the success of our problem and the question of how model trajectories is a cornerstone to give relevant recommendations.

In the second part, a state of the art on the learning of RL and its applications will be proposed. We will explain the idea of different algorithms used, their strength and drawbacks by focusing on our problem where *action-state space is discrete.* A focus on both value function algorithms such as DQN or variant and policy gradient algorithm such as A2C, PPO etc will be conducted.

Then in the third part, we will address the issue of Distributional Reinforcement Learning from a theoretical and practical point of view. C51, QRDQN and IQN algorithms principal will be developed to understand how use DRL in practice.

Finally, we will try to tackle the problem of *Variance Control.* We will present a new algorithm with theoretical justification on convergence of the algorithm and an implementation on our problem. Our implementation of this algorithm achieve same performance than state of the art algorithm while controlling variance to get more stable policy for patient pathway.

2 Modelisation of the environment

2.1 The data

The data we will receive is from the *Health Data Hub*, a new French data centre that centralises the reimbursements of French patients. We have only received a sample of dummy data but for the start of the thesis we will have the full data set.

The *AFM-Telethon*, the French association against myopathies will also gives us a data base of patients containing which type of rare disease they have and what specialist diagnosed them. Valuable help from medical specialists will also be needed in our work to calibrate our models.

Finally *Orphanet data base*, (Weinreich et al., 2008) allow the estimation of cumulative point prevalence of rare diseases in French population(Wakap et al., 2020). These data could be also helpful to calibrate the learning of the model.

2.2 Modelisation of our problem

We denote $\{C_i\}_{i=1}^n$ the consultation for every specialist and rare diseases $\{D_j\}_{j=1}^K$. We could include some medical medical examination such as scanner. The state S_t for a patient is just for example $s_t = \{C_3, \bar{C_6}, \ldots, C_2\} \in \mathcal{S}$ which is the list of specialist/medical examination that patient has already visited. The value C_i stands if doctor i have been consulted, $\bar{C_i}$ otherwise. We model the end of the patient pathway by a state S_{end} which is a terminal state when the disease is diagnosed. The action space $\mathcal{A}$ corresponds to the recommendation of specialist to visit at next step.

Given $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}_+^n$ the cost to get a consultation or exam of specialist i, we define the reward , $\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$

$$r_{t+1}^i := r\left(s_t, a_t\right) = -\alpha_i$$

For simplicity at the beginning and without expert knowledge,we will take $\forall i, \alpha_i = 1$.Then the goal of the algorithm is to find the optimal policy wich maximize the expectation over state and action given the policy :

$$\pi^\star = \arg\max_\pi E_{\tau\sim\pi}\left[\sum_{t=1}^{I} \gamma^{t-1} r_t^i \mid s_0\right].$$

where I is a stopping time which correspond when the disease is diagnosed and $\tau = \{s_0, a_0, s_1, a_1 ...\}$, the trajectory given the policy π. The discount factor $\gamma \in (0, 1)$ in our simulation is chosen closed to 1 as we have relatively short trajectories of the order of 10.

The modelling of the state space can be a little different, in fact here we consider that we can only see a specialist once and that the order of the specialists does not count. We can also order the choice of doctors. The number of choices for the number of doctors to be consulted remains generally low. In a first modelling we choose to model 5 doctors and then in a second one 10 different doctors. Then, we need to model the patient pathway to be able to simulate fake data needed to train RL algorithms. Indeed, most of RL algorithms have big sample complexity and in medical field, it is not possible to get as many sample as we want. This lead to the need of a *simulator of data* to be able to interact with its.

If we are able to sample from our learned environment, we will be able to find a better policy. Therefore, we enter in the framework of *Model-Based RL* where we have to learn the transitions of our model by using data. This contrast to *Model-Free RL* where we are able to interact with our environment such as in video games or robotic field sometimes. When we first learn transitions and then find the best policy, we call this task *planning*. Finally, the use of a sample to learn both a better policy and the model transitions is called *Hybrid Model-Based/Model-Free Imagination*. In our case we will try to plan using transitions learned from our data.

Algorithm 1 Model-Free Learning algorithm

repeat
  Sample from environment $D = (s, a, r, s)$
  Use D to update the policy $\pi(s, a)$
until Convergence of π

Algorithm 2 Planning with learnind transitions

repeat
  Sample from environment $D = (s, a, r, s)$
  Use D to learning the $T_a(s, s')$
  Use T to update the policy $\pi(s, a)$ using planning
until Convergence of π

Algorithm 3 Hybrid Model-Based/Model-Free Imagination

repeat
  Sample from environment $D = (s, a, r, s)$
  Use D to update the policy $\pi(s, a)$
  Use D to learning the $T_a(s, s')$
until Convergence of π

2.3 Modelisation of patient pathway used as simulator

In this chapter we study the problem of building a model of the environment on which our algorithms will be trained. It is a crucial aspect for many applications, since it is often impossible to deploy in real life an algorithm that has not yet been optimized due to the costs generated by bad decisions. In order to be able to sample our patients, we need to learn the transisions between the different specialists and the probabilities of being diagnosed given all state. In his Thesis, Besson (2019) compute the different probabilities of transition using a Barycenter between expert knowledge usins a *maximum entropy principle and maximum likelihood of the data.* In our problem, the amount of data present will be greater so the use of a Bayesian Network Learning. (Heckerman et al., 1995) is relevant in our problem. The aim here is to estimate the *probabilities of conditional transitions* knowing the path followed.

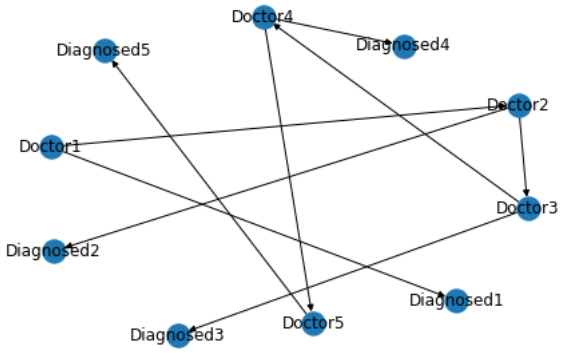

Figure 1: Path between "Doctors" and "get diagnosed" state

2.3.1 Bayesian Network Learning

Over the last decade, Bayesian Networks (BNs) have become an increasingly popular Artificial Intelligence approach. BNs are a widely used method in the modelling of uncertain knowledge. A Bayesian network (also known as a Bayes network, Bayes net, belief network, or decision network) is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG). Bayesian networks are ideal for taking an event that occurred and predicting the likelihood that any one of several possible known causes was the contributing factor. For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

More formally, the aim of Learning a Bayesian Network (BN) called $\mathcal{B}$ is given real data $\mathcal{D}$ from a probability distribution that contains independence assumptions, to encode into a graph $\mathcal{G}$ that model these independence assumptions. By definition, BNs are adapted to

model conditional Independence given some other nodes. We will cover three core subparts of BNs that are : *Parameter Learning, Structure Learning, Inference.*

More formaly, BNs are composed by a :

- A set of random variables : $\mathbf{Y} = \{Y_i\}_{i=1}^n$,
- A DAG that encodes independence assumptions using parents relationship denoted Pa,
- Conditional probability trees (CPTs) : $P\left(Y_i \mid \mathrm{Pa}_{Y_i}\right)$,
- A Joint distribution which is factorized given parents Pa :

$$P(\mathbf{Y}) = P\left(Y_1, \ldots, Y_n\right) = \prod_{i=1}^{n} P\left(Y_i \mid \mathrm{Pa}_{Y_i}\right).$$

Usually, Domain Experts are encode into the structure of the graph and parameters are fitted using data available. Learning is also possible when some data is missing using EM algorithm for example. Usually, Y_i are multinomial or Gaussian, depending on discrete or continous state space but her we only considering *multivariate variable* Y_i.

First Simple Modelisation for Sumulation : In a fist approach, we consider 5 doctors for which are 'neurologist ', 'cardiologist' ,'dermatologist', 'oncologist', 'radiologist'. Moreover, we defined a DAG in Fig. 1 which represent the Patient Pathway. "Doctor 2" in the dag represent the state where 2 doctors have been visited by the patient. So there is C_5^2 state in the random variable doctor 2. This is the same for doctor 3 where 3 doctors are visited etc. For simplicity we assume that is equivalent in term of state to have visited a neurologist and radiologist or the opposite way.

For CPTs between doctor states, we pick *randomly transitions from one specialist to another one.* For CPT from doctors to the state "get diagnosed" we have choose 3 diseases with are present with proportion $\{1/3, 1/3, 1/3\}$ in our population. They are defined as follow :

- **Disease 1**: Need to have visited Neuro, cardio and dermato tho get 0.8 to be diagnosed and if the patient have visited the 5 doctors, he/she is automatically diagnosed with probability 1.
- **Disease 2**: We get 0.6 to be diagnosed if patient have seen "oncolo" and 0.9 if he has seen oncolo and radio. If he has visited the 5 doctors, he/she is automatically diagnosed with probability 1.
- **Disease 3**: Patient has probability 0.2 to be diagnosed if he has seen a cardiologist. Same, he/she is diagnosed if he has seen the 5 doctors.

If the patient has seen all the doctors, he/she is automatically diagnosed, which is a utopian but we need a stopping condition for modelling. A *second modelisation* that can be described in Annex 7.1 takes into account 10 doctors and 5 diseases to learn more complex policies and observe differences in the learning of the algorithms.

2.3.2 Example of sampled trajectories

We use *forward sampling* across the BN with discrete variable to get samples. We have remove data letting Nan value once once the diagnosis is made before "doctor5" state, the terminal state. The use of "1" in Fig. 2 stands for if the disease have been diagnosed, "0" otherwise.

	Doctor1	Diagnosed1	Doctor2	Diagnosed2	Doctor3	Diagnosed3	Doctor4	Diagnosed4	Doctor5	Diagnosed5
0	oncolo	0	oncolo,radio	0	neuro,oncolo,radio	0	neuro,dermato,oncolo,radio	0.0	0.0	1.0
1	radio	0	cardio,radio	0	neuro,cardio,radio	0	neuro,cardio,dermato,radio	1.0	NaN	NaN
2	oncolo	0	dermato,oncolo	0	dermato,oncolo,radio	0	neuro,dermato,oncolo,radio	0.0	0.0	1.0
3	oncolo	0	neuro,oncolo	0	neuro,cardio,oncolo	0	neuro,cardio,oncolo,radio	0.0	0.0	1.0
4	cardio	0	cardio,radio	0	cardio,oncolo,radio	0	cardio,dermato,oncolo,radio	0.0	0.0	1.0

Figure 2: An example of Patient Pathway for disease 1

Once we have simulated data, the aim is to see if, with a *fixed structure*, we are able to estimate the parameters from Θ of the model using these data. There is a field of research in structure learning in Bayesian networks but here we work with a fixed structure because it has a physical meaning that cannot be changed.

2.3.3 Learning Parameters of the BN given the DAG

We are interested in learning the parameters Θ of the Bayesian network that define the transitions between doctors. Two main families of approaches are possible, *Maximum likelihood learning and Bayesian learning* of parameters.

Maximum Likelihood Approach : This is the simple approach when we have no clue or prior knowledge before learning of parameters. However, we usually overfit the data and this approach has a lack of regularity. For each discrete variable Y_i:

$$\forall (a,b) \in \mathcal{S}, \quad \hat{P}_{MLE}\left(Y_i = a \mid \mathrm{Pa}_{Y_i} = b\right) = \frac{\mathrm{Count}\left(Y_i=a,\ \mathrm{Pa}_{Y_i}=b\right)}{\mathrm{Count}\left(\mathrm{Pa}_{Y_i}=b\right)}$$

In the graph 3, we have plotted the error between fixed parameters of a the BN of Fig. 1 and parameters of another BN with learned parameters in function of the sample size. As we can see, the more the sample size of the data increase, the more the error (L2 norm) between true and learned parameters decrease.

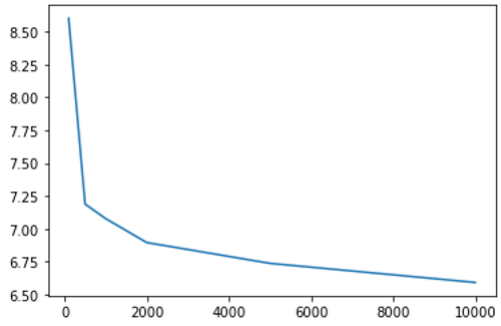

Figure 3: L2 error between True and Learned Parameters of a BN using MLE in function of sample size used to learn parameters.

Bayesian Approach : In the Bayesian setting, we need to fix priors on random variables that will guide the learning of parameters. Exploiting priors, we use prior knowledge about parameters. The consequence of this approach is :

- Beliefs before experiments are used.
- It helps to deal with unseen data.
- It Bias us towards "simpler" models.

Relevant prior for multivariate distribution is Dirichlet distribution has it is a *conjugate prior*

$$\theta \sim \text{Dirichlet}\,(\alpha_1, \ldots, \alpha_k) \sim \prod_i \theta_i^{\alpha_i - 1}$$

.
When estimating the parameters of a categorical or multinomial distribution, simple and very used practice is to add the notion of *pseudocounts* to the observed counts in the data. The consequence of this process is additive smoothing and regulation of the maximum-likelihood estimate in order to avoid overfitting. Moreover, it prevents too sharp assignments to the parameters that are due most of the time to a lack of data that would be used to estimate that parameter.

More formally, a psuedocount c is a pre-determined value that is added to the counts of occurrence of each category/class in the data when estimating the parameters of a multinomial. To give an simple example, if we are trying to estimate the parameter θ that a coin lands heads and we have observed counts H and T for heads and tails respectively, then our maximum likelihood estimate is :

$$\hat{\theta} = \frac{H}{H+T}$$

However, when using psuedocount value of c, then our estimate would be :

$$\begin{aligned}\hat{\theta} &= \frac{H+c}{(H+c)+(T+c)} \\ &= \frac{H+c}{2c+T+H}\end{aligned}$$

Adding equal psuedocounts to each outcome, we push our estimate of the parameter closer to an uniform distribution rather than the maximum likelihood estimate of the parameter. A common psuedocount across classes implies a hypothetical scenario in which we have already observed equal numbers of class before observing the data. To summarize, the larger the psuedocount, the more data will be needed to push the estimate of the parameters away from the uniform distribution.

Relation between Psuedocounts and Maximum a Posteriori (MAP) estimate : Given a counts vector $\mathbf{x}$ generated by a multinomial distribution, the posterior distribution $\boldsymbol{\Theta}$ with a Dirichlet prior is a Dirichlet. Considering :

$$\boldsymbol{\Theta} \sim \text{Dir}\,(\alpha_1, \ldots, \alpha_d)$$

the posterior after observing $\mathbf{x}$ is :

$$\boldsymbol{\Theta} \mid \mathbf{x} \sim \text{Dir}\,(x_1 + \alpha_1, \ldots, x_d + \alpha_d)$$

Then the MAP estimator is :

$$\theta_i^{\text{MAP}} = \frac{x_i + \alpha_i - 1}{\sum_{j=1}^{d} (x_j + \alpha_j - 1)}$$

So using psuedocounts of c_i added to each x_i when estimating our parameters as follows :

$$\hat{\theta}_i = \frac{x_i + c_i}{\sum_{j=1}^{d} (x_j + c_j)}$$

is the same as estimating the MAP assuming a Dirichlet prior parameterized by $(c_1 + 1, c_2 + 1, \ldots, c_k + 1)$.

Conducting the same experience than in Fig. 3 on synthetic data, we observe same results using simple prior with psuedocount of 1 of every node.

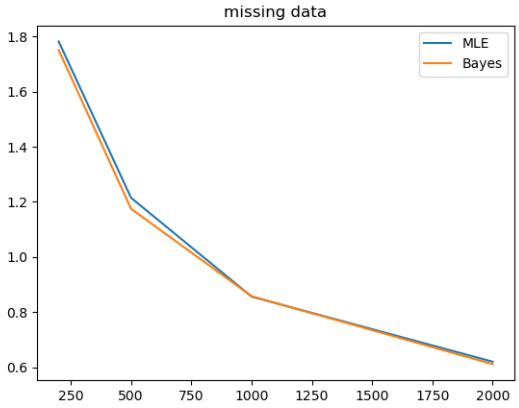

Figure 4: Frobenius Norm Between learned and True Parameters

With this Bayesian framework, we will be able to incorporate prior knowledge using psuedocount in our models with the help of specialist to calibrate our model.

3 Classical Reinforcement Learning

In this chapter, we develop an optimization formulation for the task of building a decision support tool for the diagnosis of rare diseases using patient pathway. We aim to minimize the number of consultations necessary to achieve a state where the patient's disease is diagnosed. To solve this optimization task, we investigate several reinforcement learning algorithms both based on *value function estimation or policy-based algorithms.* In our case, we would like to obtain a *stochastic policy* in the sense that it gives us the doctors with the highest probability of giving a good diagnosis. This is why we will focus on a stochastic rather than deterministic policy. Indeed, we prefer to give doctors a list of different specialists to consult who are relevant rather than a single specialist.

3.1 State of the art

Reinforcement Learning popularised by Bertsekas et al. (2000); Sutton and Barto (2018) has received increasing attention in recent years with improvement in Chess and Go games (Mnih et al., 2013b; Schrittwieser et al., 2019). Whether for discrete or continuous environments, algorithms are capable of learning ever finer policies. We can distinguish two main branches, the algorithms based on the learning of *value functions* and those based directly on the *optimisation of a policy.*

For algorithms based on Value estimation such as Deep Q Networks, (Mnih et al., 2013a), they have showed excellent results in discrete environment. Moreover, Prioritized experience replay (Schaul et al., 2015) has allow better performance with Q Learning. Finally some algorithms have allow DQN to be more stable such as Double Q-Learning (Guez and Van Hasselt, 2015).

On the other hand, to tackle the curve of dimentionality for large state-action spaces and for continuous action spaces, policy based methods have been developed in parallel. Actor-Critic methods (Mnih et al., 2016a) have been developed and take advantage of value function estimation while optimizing the policy as the same time. Then the idea of Trust Region Emerged with TRPO and PPO (Schulman et al., 2015a, 2017) algorithms lead to more stable performance. Finally, Soft Actor Critic algorithms (Haarnoja et al., 2018) improve exploration in Policy Gradient algorithms a have been a success in recent years.

The problem of safety in Reinforcement Learning have been studied and is quite closed to our as in both case we try to constraint the policy we learn. Achiam et al. (2017) have developed and algorithm called CPO which learn policy under contraints. Some other algoritms take care of safety during exploration such as in Dalal et al. (2018). In our problem the main difference is that we want to constraint Variance, however no paper have develop method for this.

3.2 Framework and Notations

Definition 3.1 (Markov Decision process)

We define a Markov decision process (MDP), given by its action space $\mathcal{A}$ and state space $\mathcal{S}$, state dynamics, reward function and discount factor. In a MDP, the state dynamics can be written as :

$$P\left(s_{t+1} \mid s_t, a_t\right).$$

The return of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$ at time t is defined as :

$$G_t = \sum_k \gamma^k r_{t+k}.$$

If given a complete trajectory (episode) that terminates, the returns can be computed recursively using

$$G_t = r_t + \gamma G_{t+1} \ .$$

Definition 3.2 (Value Function)

The value function $V^\pi : \mathcal{S} \to \mathbb{R}$ of a policy π is the expectation of the cumulative (discounted) future rewards starting from a point $s_0 = s$

$$V_\pi\big(s\big) := \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t r\left(s_t, a_t\right) \mid s_0 = s\right]$$

where the trajectory τ is generated under the policy π. T can be a random stopping time corresponding to the first time of arrival at a so-called terminal state. It can also be $T = \infty$ for infinite horizon problems. Often, we will write the value function at a state s_t, where it is implied we are at the t-th step in a trajectory, as

$$V\left(s_t\right) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t'=t}^{T} \gamma^{t'-t} r\left(s_{t'}, a_{t'}\right) \mid s_t\right]$$

We can remark that this is coherent with the notion of cumulative (discounted) future rewards which defined the value function, but not with the notation. In the finite horizon setting, *a* more correct notation would be to write $V\left(t, s_t\right)$ and make the dependence on the starting time t of the discounting explicit.

Given deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ and associated decision rule $d^\pi(s) = \pi(s)$, the dynamic programming principle leads to a dynamic programming equation called the *Bellman equation:*

$$V_\pi\big(s\big) = r(s, \pi(s)) + \gamma \sum_{s'\in\mathcal{S}} p\left(s, \pi(s), s'\right) V_\pi\big(s'\big)$$

which is a *fixed-point condition*. The Bellman equation can be used to evaluate a policy π, that is compute its associated value function V^π. The Bellman operator $\mathcal{T}^\pi$ can be defined as an operator on a function space such as :

$$\mathcal{T}^\pi v(s) := r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p\left(s, \pi(s), s'\right) v\left(s'\right)$$

This operator is interesting as it is contractant, ensuring the existence of a solution. In fact, it has the following four properties:

- contractive: $\left\|\mathcal{T}^\pi V_1 - \mathcal{T}^\pi V_2\right\|_\infty \leq \gamma \left\|V_1 - V_2\right\|_\infty$
- monotonicity: if $V_1 \leq V_2$, then $\mathcal{T}^\pi V_1 \leq \mathcal{T}^\pi V_2$
- offset: for any $c \in \mathbb{R}, \mathcal{T}^\pi V + cI = \mathcal{T}^\pi V + \gamma cI$
- unique fixed point: V_π is the unique fixed point of the Bellman operator $\mathcal{T}^\pi$.

Definition 3.3 (Optimal Value function)

Given a set of policies Π, the optimal value function satisfies

$$V^* = \max_\pi V_\pi$$

for every state $s \in \mathcal{S}$. An optimal policy π^* is one that satisfies the maximum. Strictly speaking, we are taking a maximal policy with respect to a partial ordering on policies hat compares them by looking at their respective value functions:

$$\forall s \in \mathcal{S}, \quad \pi_1 \leq \pi_2 \Longleftrightarrow V_{\pi_1}(s) \leq V_{\pi_2}(s)$$

With the same manner, the optimal value function V^* obeys a dynamic programming principle called the optimal Bellman equation :

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p\left(s, a, s'\right) V^*\left(s'\right) \right\}$$

We also get fixed-point condition, which can once again be expressed in terms of an operator called the Bellman optimal operator:

$$\mathcal{T} v(s) := \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p\left(s, a, s'\right) v\left(s'\right) \right\}$$

Then, we have that the optimal value function satisfies the equation

$$\mathcal{T} V^* = V^*$$

Definition 3.4 (Q Function or action state value function)

The action-state value function of a policy π is the function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined by

$$Q_\pi(s,a) := \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r\left(s_t, a_t\right) \mid s_0 = s, a_0 = a \right]$$

where the trajectory τ is generated under the policy π. We can notice that Q_π has an link to the value function V^π. Indeed, $\forall s \in \mathcal{S}$, it holds that :

$$V_\pi(s) = \mathbb{E}_{a\sim\pi(s,)}\left[Q_\pi(s,a)\right] = \sum_a \pi(s,a) Q_\pi(s,a)$$

There's also a reverse equality :

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[r(s,a) + V_\pi\left(s_{t+1}\right) \mid s_t = s\right] = r(s,a) + \sum_{s'} p\left(s,a,s'\right) V_\pi\left(s'\right)$$

Definition 3.5 (Optimal action value function)

Optimal policies also share the same optimal action-state value function

$$Q^* = \max_\pi Q_\pi$$

for all states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. There is also a link also between optimal values function :

$$V^*(s) = Q^*\left(s, \pi^*(s)\right) = \max_{a\in\mathcal{A}} Q^*(s,a)$$

One we have define some properties of Value and action-value function, algorithms can be derived to find optimal policy. When all transitions are known, we can use Dynamic programming algorithms such as *Value Iteration or Policy Iteration.*

3.3 Value iteration

This algorithm use the fact that Bellman Operator is a γ- contraction. So given all rewards and transitions probabilities, it is possible to use these operator to converge to an optimal policy. With this algorithm each iteration is very efficient but convergence is asymptotic. Moreover it work only for discrete state and action spaces. The full environment need to be known so in practice is is not very use except for specific cases.

Algorithm 4 Value iteration

for k=1,....K **do**
  $V_k \leftarrow \mathcal{T}^* V_{k-1}$
end for
for $s \in \mathcal{S}$ **do**
  $\pi_K(s) \leftarrow \text{argmax}_{a \in A} \left\{ r(s,a) + \gamma \sum_{s' \in S} p\left(s,a,s'\right) V_K\left(s'\right) \right\}$
end for
return Policy π_K, value V_k

3.4 Policy Iteration

In this algorithm, we take advantage of the action-state function to iteratively update the policy with greedy improvement. It is composed of two step : *evaluation and improvement.* In evaluation step (E), we compute the value function of each state action or state. In improvement step (I) we improve with a greedy manner the policy π. This gives the following sequence :

$$\pi_0 \xrightarrow{E} Q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi^\star \xrightarrow{E} Q^\star$$

Knowing transitions, the Q function can be computed exactly. The main difficulty is when we don't have access to all transitions of the model. In this case, learning requires sampled episodes of the environment and an estimation of a value or action-value function. In practice it is also possible to approximate Q value using gradient descent.

Algorithm 5 Policy Iteration Algorithm

for k=1,...K **do**
  Compute value function Q_{π_k}
  Then set $\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} Q_{\pi_k}(s,a)$
end for
return π_K

In the following section, we try to solve MDP problem with an incremental manner using Reinforcement Learning algorithms when transactions are not known and the dimension not to big.

3.5 Tabular case Reinforcement Learning

These methods are efficient when the dimensionality of the action-state space is not to big.

3.5.1 Monte Carlo estimation

The idea of Monte Carlo estimation is simple, generates episodes τ_i under a policy π starting at some state s_0, and compute their empirical returns :

$$R(\tau_i) = \sum_{t=0}^{T_i} \gamma^t r_{t,i}$$

Then, the value estimate is then the empirical mean :

$$\hat{V}_\pi(s_0) = \frac{1}{n} \sum_{i=1}^{n} R(\tau_i)$$

With beginning state s_0 fixed so we are only estimating its value. Monte Carlo estimation is interesting but does not allow online learning and require full trajectories which can be sometimes very difficult in complex environment.

3.5.2 Temporal difference estimation

The following method called $TD(0)$ estimation is based on V_π which satisfies the Bellman equation. This means that the temporal difference error of an estimate $\hat{V}_\pi$ of V_π, defined after each transition and reward sampled s_t, r_t, s_{t+1} is :

$$\delta_t = r_t + \gamma \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t).$$

This quantity should be small for the estimator $\hat{V}_\pi$ to be good. Indeed, the quantity δ_t is a (biased) estimate of so-called Bellman error, which measures how the discrepancy between $\hat{V}_\pi(s)$ at some state s and the Bellman image $\mathcal{T}^\pi \hat{V}(s)$ so these quantities should be equal. To learn the estimator $\hat{V}^\pi$, TD (0) updates are performed as :

$$\begin{aligned} \hat{V}_\pi(s_t) &\leftarrow \hat{V}_{\mathrm{x}}(s_t) + \alpha_t \delta_t \\ &= (1 - \alpha_t) \hat{V}_\pi(s_t) + \alpha_t \left(r_t + \gamma \hat{V}_\pi(s_{t+1}) \right) \end{aligned}$$

with α_t a learning rate. This method introduces bias but it has less variance than Monte Carlo estimation. Moreover, this method does not require the entire trajectory to be known to perform an update, allowing for fully *online learning*. In order to get less biased estimate of the Bellman error, the use of several rewards to compute the temporal difference can be efficient. So the n -step TD(n) is

$$\delta_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n \hat{V}_\pi(s_{t+n}) - \hat{V}_\pi(s_t).$$

The Q -function update is then done as usual. This can be rewritten using the n -step return:

$$G_{t:t+n} = \sum_{k=0}^{n-1} \gamma^k r_{t+k}$$

leading to

$$\delta_t^{(n)} = G_{t:t+n} - \hat{V}_\pi(s_t).$$

3.5.3 $TD(\lambda)$ estimation

To create a tradeoff between bias and variance in estimation of value function, we can define TD(λ) which is between incremental MC and TD(n). The scheme is defined for $0 < \lambda < 1$ as :

$$\hat{V}_\pi(s_t) \leftarrow \hat{V}_\pi(s_t) + \alpha \sum_{t'-t}^{T} (\gamma\lambda)^{t'-t} \delta_{t'}$$

where $(\delta_t)_t$ is the sequence of temporal differences errors of the trajectory. In theory, this exact scheme does require the entire trajectory be known before performing an update. This can be seen more explicitly by introducing the λ -return :

$$G_t^\lambda = (1-\lambda) \sum_{n-1}^{\infty} G_{t:t+n}$$

But in practice, function approximators can be used to not use full trajectory.

3.6 Policy Improvement via Temporal Difference estimation

3.6.1 SARSA

Once we get an estimation of using previous policy improvement methods, we ca use policy improvement to obtain a better policy. For example we can define an exploration policy using the softmax function on Q and a temperature parameter τ :

$$\pi_Q(s,a) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a' \in A} \exp\left(Q\left(s,a'\right)/\tau\right)}$$

Considering transition (s_t, a_t, r_t, s_{t+1}), we can take the next action a_{t+1} according to π_Q and update using the TD(0) scheme: compute the temporal difference of the Q -function

$$\delta_t = r_t + \gamma \hat{Q}\left(s_{t+1}, a_{t+1}\right) - \widehat{Q}\left(s_t, a_t\right)$$

and then update $\hat{Q}$

$$\begin{aligned} \hat{Q}\left(s_t, a_t\right) &\leftarrow \hat{Q}\left(s_t, a_t\right) + \alpha_t \delta_t \\ &= \left(1 - \alpha_t\right) \hat{Q}\left(s_t, a_t\right) + \alpha_t \left(r_t + \hat{Q}\left(s_{t+1}, a_{t+1}\right)\right). \end{aligned}$$

With this scheme, we ensure the policy continues to improve by decreasing the temperature τ to 0 and becoming more and more greedy in our action selection (keeping the same τ reduces the algorithm to some kind of softmax policy evaluation). In each update, the algorithm uses the entire state-action transition $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$. The use of the value candidate $\hat{Q}$ to define the policy used to select the action means that SARSA is an on-policy learning algorithm.

3.6.2 Q-Learning

Q learning has been for many year state of the art in Reinforcement learning for discrete environment. There exits many variant of this algorithm but the simpler in tabular case is based on the following idea. The principal here is to introduce the optimal TD error for the Q value function :

$$\delta_t = r_t + \gamma \max_{a'} \hat{Q}\left(s_{t+1}, a'\right) - \hat{Q}\left(s_t, a_t\right)$$

and updating the Q estimate using that:

$$\hat{Q}\left(s_t, a_t\right) \leftarrow \hat{Q}\left(s_t, a_t\right) + \alpha_t \delta_t.$$

We can notice that this is not the same as using the greedy action to update : this is upper-bounding the Q-value TD error and then using that upper bound to update the value. So Q-learning is an off-policy learning algorithm, whatever exploration policy we use.

Algorithm 6 Q-learning algorithm

while Trajectories non terminate **do**
  Chose a_t according to exploration policy and get (r_t, s_{t+1})
  Compute $\delta_t = r_t + \gamma \max_{a'} \hat{Q}\left(s_{t+1}, a'\right) - \hat{Q}\left(s_t, a_t\right)$
  $\hat{Q}\left(s_t, a_t\right) \leftarrow \hat{Q}\left(s_t, a_t\right) + \alpha_t \delta_t$
end while
for $s \in \mathcal{S}$ **do**
  $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} \hat{Q}(s, a)$
end for
return π

3.7 Approximate Reinforcement Learning

When the state and action space is too large, the calculations are usually intractable and it is necessary to switch to parametric approximations of the value functions. We define $v_\theta \approx V_\pi$ and $q_\theta \approx Q_\pi$.

3.7.1 Approximation of evaluation via Monte carlo

Considering sample trajectories $(s_{0,i}, a_{0,i}, r_{0,i}, \ldots)$, we want the approximator to match the empirical returns

$$R_i = \sum_{t=0}^{T_i} \gamma^t r_{t,i} = v_\theta\left(s_{0,i}\right) + \varepsilon_i$$

Our objective is to minimize the empirical error :

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_i \left(v_\theta\left(s_{0,i}\right) - R_i\right)^2 .$$

3.7.2 Approximation of evaluation via TD

To use approximation via TD, we need to target the bootstrapped returns :

$$G_t = r_t + \gamma v_\theta\left(s_{t+1}\right)$$

and the parameter is updated according to :

$$\theta \leftarrow \theta - \alpha_t \left(v_\theta\left(s_t\right) - G_t\right) \nabla_\theta v_\theta\left(s_t\right).$$

We can notice that this is here a semi-gradient update (where we minimize the MSE for a sample and where the derivative wrt $v_\theta\left(s_{t+1}\right)$ is ignored).

3.7.3 Approximation of optimization via Q-Learning

To approximate Q-learning which tries to minimize the TD error upper bound, we reformulate the function approximation step as targeting a bootstrap estimate of the value

$$G_t = r_t + \gamma \max_{a' \in \mathcal{A}} q_\theta\left(s_{t+1}, a'\right)$$

In the online case, we target the bootstrapped loss

$$\mathcal{L}(\theta) = \left(q_\theta\left(s_t, a_t\right) - G_t\right)^2$$

and perform updates with a semi-gradient such as :

$$\theta \leftarrow \theta - \alpha_t \left(q_\theta\left(s_t, a_t\right) - G_t\right) \nabla_\theta q_\theta\left(s_t, a_t\right).$$

The main issue here is that this method can sometimes diverge or takes many time to converge. Indeed, oscillations, over-estimation effects, problem of scaling and correlation between samples can occurs. To deal with this issue, some solutions to stabilize the learning have been proposed such as : *Target Network, double Q-learning, reward normalization or Experience replay* (Mnih et al., 2013b). Experience replay use old sample to be more sample

efficient and the use of a Target Network. The target network's parameters are not trained, but they are periodically synchronized with the parameters of the main Q-network. The idea is that using the target network's Q values to train the main Q-network will improve the stability of the training. Here an algorithm that we will use with experience replay and target network ideas :

Algorithm 7 Q-learning with Experience Replay and Target Network

Input: Inital parameters $\theta, \bar{\theta}$ and target update interval C.
for $i = 1, ..., n$ **do**
  Set initial state $s_{0,i}$
  while termination **do**
    take action $a_{t,i}$ and get $r_{t,i}, s_{t+1,i}$
    Store the transition $e_{t,i} = (s_{t,i}, a_{t,i}, r_{t,i}, s_{t+1,i})$ in buffer $\mathcal{D}$
    $t \leftarrow t + 1$
    Update network q_θ from mini batch using target-boostrapped returns
    Every C steps update $\bar{\theta} \leftarrow \theta$
  end while
end for

Finally Double Q-Learning is used to limit over-estimation in the learning. The scheme of Double Q-Learning is the following :

- Let $a_2^* = \text{argmax}_a \, q_{\theta_2}(s_{t+1}, a)$
- Set the return bootstrap $G_t = r_t + \gamma q_{\theta_1}(s_{t+1}, a_2^*)$
- Update: $\theta_1 \leftarrow \theta_1 - \alpha_t \left(q_{\theta_1}(s_t, a_t) - G_t\right) \nabla_{\theta_1} q_{\theta_1}(s_t, a_t)$
- Repeat by alternating θ_1 and θ_2.

3.8 Policy Gradient Method

The aim of reinforcement learning is to find an optimal strategy for an agent to obtain optimal sum of rewards. The main idea of policy gradient methods target at modeling and optimizing the policy directly in contrast to Q-Learning where we use value function. The policy is modeled with a parameterized function respect to θ, denoted $\pi_\theta(a|s)$. The value of the reward (objective) function depends on this policy and then various algorithms can be applied to optimize θ for the best reward. Usually we define a policy performance metric denoted J. Very often this metric is simply the value function :

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]. \tag{1}$$

where $\tau = \{s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T\}$ and $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$ We will sometimes denote $J(\pi_\theta)$ as $J(\theta)$ as we are maximizing over the policy parameter θ. Optimization methods seek to maximize performance according to θ, so their updates approximate gradient ascent in J :

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J\left(\boldsymbol{\theta}_t\right)}$$

where $\widehat{\nabla J\left(\boldsymbol{\theta}_t\right)}$ is a stochastic estimate whose expectation approximates the gradient of the performance measure with respect to its argument $\boldsymbol{\theta}_t$. All methods that follow this general scheme are referred to policy gradient methods. Moreover, methods that learn approximations to both policy and value functions are often called *actor-critic methods*, where the term *actor* is a reference to the learned policy, and the term *critic* refers to the learned value function.

Very often, policy-based methods are useful in the continuous space. Because there is an infinite number of actions and/or states to estimate the values for and hence value-based approaches are way too expensive computationally in the continuous space. For example, in generalized policy iteration, the policy improvement step $\arg\max_{a \in A} \pi(s, a)$ requires a full scan of the action space, suffering from the curse of dimensionality. However it is also very useful for discrete environments and gives good very good results also. The following property allow us to derive algorithms based on policy gradient.

Proposition 3.1 (Gradient under a parametric law)

Given a set of probability models $\left\{P_\theta : \theta \in \Theta \subseteq \mathbb{R}^d\right\}$ on a set $\mathcal{X}$ and a function $f : \mathcal{X} \to \mathbb{R}$, we get that

$$\nabla_\theta \mathbb{E}_{X \sim P_\theta}[f(X)] = \mathbb{E}_{X \sim P_\theta}\left[f(X) \nabla_\theta \log P_\theta(X)\right]$$

This property is interesting when deriving estimators of the derivatives in optimization problems with stochastic objectives. Generalization to the case where f also depends on θ is straightforward. This can be shown either by either writing the expectation as an integral, or by a change of measures with a Radon-Nikodym derivative.
The following property allow us to derivate the gradient of equation 1 which is :

$$\nabla_\theta J\left(\pi_\theta\right) = \mathbb{E}_{\tau \sim \pi_\theta}\left[R(\tau) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta\left(s_t | a_t\right)\right]$$

Moreover, there are other ways of writing and derivate the policy gradient, such as in the book of Sutton and Barto (2018).

$$\nabla_\theta J\left(\pi_\theta\right) = \mathbb{E}_\pi\left[Q^{\pi_\theta}(s, a) \nabla \pi_\theta(s, a)\right].$$

A key point according to Schulman et al. (2015b) is that there are several different related expressions for the policy gradient, which have the form :

$$\nabla_\theta J\left(\pi_\theta\right) = \mathbb{E}\left[\sum_{t=0}^{T} \Psi_t \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]$$

where Ψ_t may be one of the following:

- $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
- $Q^{\pi}(s_t, a_t)$: state-action value function.
- $\sum_{t'=t}^{\infty} r_{t'}$: reward following action a_t.
- $A^{\pi}(s_t, a_t)$: advantage function.
- $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
- $r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$: TD residual.

A major breakthrough in performance is to use advantage function which gives very good results in terms of variance reduction. The estimator GAE, Schulman et al. (2015b) of advantage function is currently very use especialy on continuous-control setting. Now, we will present the most famous algorithms based on policy gradient in the next section.

3.8.1 The REINFORCE algorithm

REINFORCE (or Monte-Carlo policy gradient) algorithm principle relies on an estimated return by Monte-Carlo methods using episode samples to update the policy parameter θ. This algorithm works due to the fact that the expectation of the sample gradient is equal to the actual gradient :

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_\pi \left[Q^\pi(s,a) \nabla_\theta \ln \pi_\theta(a \mid s) \right] \\ &= \mathbb{E}_\pi \left[G_t \nabla_\theta \ln \pi_\theta (A_t \mid S_t) \right] \quad , \text{ because } Q^\pi(S_t, A_t) = \mathbb{E}_\pi \left[G_t \mid S_t, A_t \right] \end{aligned}$$

This method relies on a full trajectory as many Monte Carlo methods.

Algorithm 8 REINFORCE Algorithm

Initialize the policy parameter θ at random.
Generate one trajectory τ on policy π_θ :
for $t = 1, ...T$: **do**
  Estimate the the return G_t
  Update policy parameters $\theta \longleftarrow \theta + \alpha^t \gamma G_t \nabla\theta \ln \pi_\theta(A_t|S_t)$
end for

We can improve this algorithm by subtract a baseline to reduce the variance of gradient estimation while keeping the bias unchanged. This can be done using advantage function for example.

3.8.2 Actor-Critic method

Actor-critic algorithms are based on two models, which may optionally share parameters in certain cases :

- Critic updates the value function parameters w and depending on the algorithm it could be action-value $Q_w(a|s)$ or state-value $V_w(s)$.
- Actor updates the policy parameters *theta* for $\pi_\theta(a|s)$, in the direction suggested by the critic.

It is relevant to learn the value function in addition to the policy, since knowing the value function can assist the policy update, such as by reducing gradient variance in REINFORCE policy gradients, and that is the main idea of Actor-Critic method does.

Algorithm 9 Actor-Critic Algorithm

Initialize s, θ, w at random and sample $a \sim \pi_\theta(a|s)$
for $t = 1, ...T$ **do**
  Collect sampled $r_t \sim R(s, a)$ and $s' \sim P(s'|s, a)$
  Update the policy parameters : $\theta \leftarrow \theta + \alpha_\theta Q_w(s, a) \nabla_\theta \ln \pi_\theta(a \mid s)$;
  Compute the correction (TD error) for action-value at time t :
  $\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s, a)$
  Use it to update the parameters of action-value function :
  $w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s, a)$
  Update $a \leftarrow a'$ and $s \leftarrow s'$.
end for

The more famous algorithms based on this principle are A2C and A3C (Mnih et al., 2016a) which are respectively synchronous and asynchronous version of this algorithm running different agent in parallel. These two algorithms gives us stochastic policy but other algorithm could result in deterministic policy such as DPG and its variants, DDPG and D4PG (Silver et al., 2014; Lillicrap et al., 2015). In this framework action are chosen with a deterministic manner such as $a = \mu(s)$.

A very interesting idea to improve training stability is that we should *avoid parameter updates that change the policy too much at one step.* This crucial idea coupled with the use of a surrogate function lead to two famous algorithms used, TRPO and PPO. Before this, we will delve into Off Policy algorithm to understand the surrogate function in TPRO and PPO.

3.8.3 Short description of Off Policy Algorithm

Off policy is the term used when we refer to the use of sample from another distribution to compute gradient. The *behavior policy* for collecting samples is a known policy (predefined

just like a hyperparameter), labelled as $\beta(a \mid s)$. The objective function J is the expectation of the value function can be defined according tho its behavior policy:

$$J(\theta) = \sum_{s \in \mathcal{S}} d^{\beta}(s) \sum_{a \in \mathcal{A}} Q_{\pi}(s,a)\pi_{\theta}(a \mid s) = \mathbb{E}_{s \sim d^{\beta}} \left[\sum_{a \in \mathcal{A}} Q_{\pi}(s,a)\pi_{\theta}(a \mid s) \right]$$

where $d^{\beta}(s)$ is the stationary distribution of the behavior policy β (we can define $d^{\beta}(s) = \lim_{t \to \infty} P\left(S_t = s \mid S_0, \beta\right)$) and Q_{π} is the action-value function estimated with regard to the *target policy* π and not the behavior policy.

Given that the training observations are sampled by $a \sim \beta(a \mid s)$, we rewrite J as :

$$\begin{aligned}
&\nabla_{\theta} J(\theta) = \nabla_{\theta} \mathbb{E}_{s \sim d^{\beta}} \left[\sum_{a \in \mathcal{A}} Q_{\pi}(s,a)\pi_{\theta}(a \mid s) \right] \\
&= \mathbb{E}_{s \sim d^{\beta}} \left[\sum_{a \in \mathcal{A}} \left(Q_{\pi}(s,a)\nabla_{\theta}\pi_{\theta}(a \mid s) + \pi_{\theta}(a \mid s)\nabla_{\theta} Q_{\pi}(s,a) \right) \right] \quad \text{derivativing with product rule} \\
&\overset{(i)}{\approx} \mathbb{E}_{s \sim d^{\beta}} \left[\sum_{a \in \mathcal{A}} Q_{\pi}(s,a)\nabla_{\theta}\pi_{\theta}(a \mid s) \right] \\
&= \mathbb{E}_{s \sim d^{\beta}} \left[\sum_{a \in \mathcal{A}} \beta(a \mid s) \frac{\pi_{\theta}(a \mid s)}{\beta(a \mid s)} Q_{\pi}(s,a) \frac{\nabla_{\theta}\pi_{\theta}(a \mid s)}{\pi_{\theta}(a \mid s)} \right] \\
&= \mathbb{E}_{\beta} \left[\frac{\pi_{\theta}(a \mid s)}{\beta(a \mid s)} Q_{\pi}(s,a) \nabla_{\theta} \ln \pi_{\theta}(a \mid s) \right]
\end{aligned}$$

The part $\pi_{\theta}(a \mid s)\nabla_{\theta} Q^{\pi}(s,a)$ on the line with (i) can be ignored because if we use an approximated gradient with the gradient of Q ignored, we still guarantee the policy improvement and eventually achieve the true local minimum. This is justified in the proof here Degris et al. (2012). We can also recognize an importance weight with the quotient $\frac{\pi_{\theta}(a|s)}{\beta(a|s)}$.

3.8.4 TRPO (Trust Region Policy Optimization).

Trust region policy optimization denoted TRPO (Schulman et al., 2015b) enforce a KL divergence constraint on the size of policy update at each iteration. So big changes of policies are not possible and lead to more stable learning of policy. Before this, we will explain how derive the objective of TRPO using off-policy formulation.

Considering the case when we are doing off-policy RL, the policy β used for collecting trajectories on rollout workers is different from the policy π to optimize for. The objective function called L here in an off-policy model measures the total advantage over the state visitation

distribution and actions, while the mismatch between the training data distribution and the true policy state distribution is compensated by importance sampling estimator.

$$
\begin{aligned}
L(\theta) &= \sum_{s \in \mathcal{S}} d^{\pi_{\theta_{\text{old}}}} \sum_{a \in \mathcal{A}} \left(\pi_\theta(a \mid s) \hat{A}_{\theta_{\text{old}}}(s,a) \right) \\
&= \sum_{s \in \mathcal{S}} d^{\pi_{\theta_{\text{odd}}}} \sum_{a \in \mathcal{A}} \left(\beta(a \mid s) \frac{\pi_\theta(a \mid s)}{\beta(a \mid s)} \hat{A}_{\theta_{\text{old}}}(s,a) \right) \\
&= \mathbb{E}_{s \sim d^{\pi_{\text{old}}}, a \sim \beta} \left[\frac{\pi_\theta(a \mid s)}{\beta(a \mid s)} \hat{A}_{\theta_{\text{old}}}(s,a) \right]
\end{aligned}
$$

The notation θ_{old} is the policy parameters before the update, $d^{\pi_{\theta_{\text{old}}}}$ is defined in the same way as above; $\beta(a \mid s)$ is the behavior policy for collecting trajectories. Estimated advantage $\hat{A}(.)$ is used rather than the true advantage function $A(.)$ because the true rewards are usually unknown.

Considering a policy $\pi_{\theta_{\text{old}}}$, we want to use the previous samples to get to a better policy π_θ. So we choose β in off line setting as our current policy $\pi_{\theta_{\text{old}}}$. However, there is no guarantee we can trust the behavior in these samples: we can end up stuck in a cycle of states. We can rewrite our objective function as :

$$
L_{\pi_{\theta_{\text{old}}}}(\pi_\theta) = \mathbb{E}_{s \sim d^{\pi_{\theta_{\text{old}}}}} \mathbb{E}_{a \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(s,a)}{\pi_{\theta_{\text{old}}}(s,a)} A^{\pi_{\theta_{\text{old}}}}(s,a) \right] = \sum_{s \in \mathcal{S}} d^{\pi_{\theta_{\text{old}}}}(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) A^{\pi_{\theta_{\text{old}}}}(s,a)
$$

Moreover, we can show that the gradient of $L_{\pi_{\theta_{\text{old}}}}(\pi_\theta)$ wrt θ is equal to the policy gradient of J in this section. So $L_{\pi_{\theta_{\text{old}}}}$ is a local approximation of the policy performance $J(\pi_\theta) - J(\pi_{\theta_{\text{old}}})$ in the neighborhood of π_θ. In practice the advantage function $A^\pi(s,a)$ will be approximated, using a TD(λ) error for instance. So we maximize a lower bound on L_π which is still a local approximation of J using the total variation.

Policy improvement scheme to derive an algorithm. Given the current policy π_k, we get the next one by solving :

$$
\pi_{k+1} \in \underset{\pi'}{\text{argmax}} L_{\pi_k}(\pi') - C \mathbb{E}_{s \sim d^{\pi_k}} \left[D_{\text{TV}}(\pi'(s) \| \pi_k(s)) \right]
$$

where $C > 0$ is a constant, and we search π' in our search space (for instance, parametric). The maximum we get is positive and satisfies

$$
J(\pi') - J(\pi_k) \geq \max_{\pi'} \left\{ L_{\pi_k}(\pi') - C \mathbb{E}_{s \sim d^{\pi_k}} \left[D_{\text{TV}}(\pi'(s) \| \pi_k(s)) \right] \right\} \geq 0.
$$

The maximization problem might be too difficult due to how the total variation distance D_{TV} is defined. We can relax it by lower-bounding again, using Pinkser's inequality to see that :

$$
D_{\text{TV}}(\pi'(s) \| \pi(s)) \leq \sqrt{2 \text{KL}(\pi'(s) \| \pi(s))}.
$$

More details can be found in the original paper. So finally we obtain a problem of the form :

$$\begin{aligned}\pi_{k+1} &= \text{argmax}_{\pi'} L_{\pi_k}(\pi') \\ \text{s.t. } \overline{\text{KL}}(\pi' \| \pi) &= \mathbb{E}_{s\sim d^\pi}\left[\text{KL}\left(\pi'(s) \| \pi_k(s)\right)\right] \leq \delta\end{aligned}$$

This constrained optimization problem is hard to solve, but can be approximated by a quadratically-constrained linear program. The approximate update is:

$$\begin{aligned}\theta_{k+1} &= \underset{\theta}{\text{argmax}} g^\top (\theta - \theta_k) \\ \text{s.t. } &\frac{1}{2}(\theta - \theta_k)^\top F (\theta - \theta_k) \leq \delta\end{aligned}$$

where $g = \nabla_\theta L_{\pi_k}(\pi_\theta)$ and F is the Fisher information matrix. This is solved by :

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^\top F(\theta_k)^{-1} g}} F(\theta_k)^{-1} g$$

This algorithm is very hard to implement and has some stability problems. This is why it lead to the algorithm PPO (Proximal Policy Optimization) (Schulman et al., 2017).

3.8.5 PPO (Proximal Policy Optimization)

Given that TRPO is relatively complicated and we still want to implement a similar constraint, proximal policy optimization (PPO) simplifies it by using a clipped surrogate objective while retaining similar performance. We denote the probability ratio between old and new policies as :

$$r(\theta) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{dd}}}(a \mid s)}$$

Then, the objective function of TRPO becomes:

$$J^{\text{TRPO}}(\theta) = \mathbb{E}\left[r(\theta)\hat{A}_{\theta_{\text{old}}}(s,a)\right].$$

The idea to limit a two big distance between θ_{old} and θ while maximizing $J^{\text{TRPO}}(\theta)$ to avoid to instability is here used with another technique. PPO imposes the constraint by forcing $r(\theta)$ to stay within a small interval around 1 , precisely $[1-\epsilon, 1+\epsilon]$, where ϵ is a hyperparameter.

$$J^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r(\theta)\hat{A}_{\theta_{\text{dd}}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{\text{old}}}(s,a)\right)\right]$$

Here the function $\text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)$ clips the ratio to be no less than $1-\epsilon$ and no more than $1+\epsilon$. Then, the objective of PPO takes the minimum one between the original value and the clipped version and therefore we lose the motivation for increasing the policy update to extremes for better rewards. The objective function is augmented with an error term

on the value estimation (use of an actor-critic network) and an *entropy term* to encourage exploration.

$$J^{\text{CLIP}'}(\theta) = \mathbb{E}\left[J^{\text{CLIP}}(\theta) - c_1\left(V_\theta(s) - V_{\text{target}}\right)^2 + c_2 H\left(s, \pi_\theta(.)\right)\right]$$

PPO demonstrate very nice performance in practice with outstanding simplicity compare to TRPO. PPO and A2C are two very common and efficient algorithms that will compare to other techniques in our problem in the following sections.

4 Distributional Reinforcement Learning (DRL)

4.1 Motivations of Distributional Reinforcement Learning for our problem

The aim of this work is to motivate the distributional reinforcement learning framework in order to minimize the variance in a reinforcement problem. We wish to choose action that are less risky and have less variance because we are dealing with human lives and not with arbitrary values. Variance estimation and reduction in Reinforcement Learning is a capital issue and the framework of Distributional Reinforcement Learning allow us to estimate the variance of the return as we get access to the distribution of the return and no only of the mean which is consider in most of classic Reinforcement Learning problems. Moreover, amazing progress has been made in RL with the advent of Distributional RL, so it is natural to be interested in this type of problem. Here we would like to minimize the variance while maximizing the expectation of the returns.

From an algorithmic point of view, there are many benefits to learning an approximate distribution rather than its approximate expectation. The distributional Bellman operator preserves *multimodality* in value distributions, which we believe leads to more stable learning. Approximating the full distribution also mitigates the effects of learning from a *nonstationary policy.*

Knowing the distribution of the returns, we are able to give an estimator of the variance. So we are interested in the *estimation* of the distribution of returns to get a good estimation of variance. In a second time we will try to make *control* and use this phase of estimation to choose the relevant actions.

Distributional Reinforcement Learning has been a outstanding breakthrough in RL community in the last few years. When Bellemare et al. (2017) has showed that the Bellman operator over value distributions is a contraction in a maximal form of the Wasserstein, this opened up great opportunities for research in RL has it allow to model distribution and not only the mean of the reward. A first algorithm, C51 based on categorical distributions, has emerged and beats the classical methods on discrete enclosures. Then in a second step, an algorithm based on quantile estimation of the distribution was developed: QR DQN Dabney et al. (2017), at the time of its creation, it had more theoretical guaranties than C51. An improvement of the latter was to be interested in representing the quantile function by a neural network as in Dabney et al. (2018). Moreover, DRL has been studied theoretically in Rowland et al. (2019, 2018) , respectively on C51 algorithm or the role of sample and statistics in DRL. Here the question is how use DRL to learn stochastic policies in discrete environment and be able to control the variance of out learned policy as much as possible?

4.2 Notations

As before, we model the agent-environment interactions by a Markov decision process (MDP) with $\mathcal{S}$ and $\mathcal{A}$ the state and action spaces, R the random variable reward function, $P(x' \mid x, a)$ the probability of transitioning from state x to state x' after taking action a, and $\gamma \in [0, 1)$ the discount factor. A policy $\pi(\cdot \mid s)$ maps each state $s \in \mathcal{S}$ to a distribution over $\mathcal{A}$.

Given a fixed policy π, the return, $Z_\pi = \sum_{t=0}^{\infty} \gamma^t R_t$ is a random variable representing the sum of discounted rewards observed along one trajectory of states while following π. Standard RL algorithms estimate the expected value of Z^π, the value function,

$$V_\pi(s) := \mathbb{E}\left[Z_\pi(s)\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, a_t\right) \mid s_0 = s\right].$$

Similarly, many RL algorithms estimate the action-value function :

$$Q_\pi(s, a) := \mathbb{E}\left[Z_\pi(s, a)\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, a_t\right)\right]$$
$$s_t \sim P\left(\cdot \mid s_{t-1}, a_{t-1}\right), a_t \sim \pi\left(\cdot \mid s_t\right), s_0 = s, a_0 = a.$$

In distributional RL the distribution over returns (i.e. the probability law of Z^π), plays the central role and *replaces the value function.*

In classic RL, we approach Q function using the optimality equation

$$Q^*(x, a) = \mathbb{E}R(x, a) + \gamma \mathbb{E}_P \max_{a' \in \mathcal{A}} Q^*\left(x', a'\right).$$

This equation has a unique fixed point Q^*, the optimal value function, corresponding to the set of optimal policies Π^* (π^* is optimal if $\mathbb{E}_{a \sim \pi^*} Q^*(s, a) = \max_a Q^*(s, a)$). We view value functions as vectors in $\mathbb{R}^{\mathcal{S} \times \mathcal{A}}$, and the expected reward function as one such vector. In this context, the Bellman operator $\mathcal{T}^\pi$ and optimality operator $\mathcal{T}$ are :

$$\mathcal{T}^\pi Q(s, a) := \mathbb{E}R(s, a) + \gamma \mathop{\mathbb{E}}_{P, \pi} Q\left(s', a'\right)$$
$$\mathcal{T} Q(s, a) := \mathbb{E}R(s, a) + \gamma \mathbb{E}_P \max_{a' \in \mathcal{A}} Q\left(s', a'\right).$$

In the policy evaluation framework, we are using in the value function V^π associated with a given policy π. The analogue here is the value distribution Z_π. In this section we characterize Z_π and study the behaviour of the policy evaluation operator $\mathcal{T}^\pi$. We emphasize that Z_π describes the intrinsic randomness of the agent's interactions with its environment, rather

than some measure of uncertainty about the environment itself. We will write the distribution of the return of policy π and initial state-action air$(s, a) \in \mathcal{S} \times \mathcal{A}$ as :

$$Z_\pi^{(s,a)} = \text{Law}_\pi \left(\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a \right)$$

Moreover, we denote Z_π for the collection of distributions $\left(Z_\pi^{(s,a)} \mid (s, a) \in \mathcal{S} \times \mathcal{A} \right)$. We would like to define the Belleman Operator for distribution but we need the notion of *push forward.*

Definition 4.1 (Push forward measure)

Let $\nu \in P(\mathbb{R})$ be a probability distribution and $f : \mathbb{R} \to \mathbb{R}$ a measurable function, the pushforward measure $f_\# \nu \in P(\mathbb{R})$ is defined by $f_\# \nu(A) = \nu\left(f^{-1}(A)\right)$, for all Borel sets $A \subseteq \mathbb{R}$.

Intuitively, $f_\# \nu$ is obtained from ν by shifting the support of ν according to the map f. Of particular interest in this thesis will be pushforward measures obtained via an affine shift map $f_{r,\gamma} : \mathbb{R} \to \mathbb{R}$, defined by $f_{r,\gamma}(s) = r + \gamma s$.

Definition 4.2 (Distributional Bellman Operator and Distributional Bellman equation)

The return distribution function Z_π associated with a policy π, defined previously, satisfies the distributional Bellman equation:

$$Z_\pi^{(s,a)} = \left(\mathcal{T}^\pi Z_\pi\right)^{(s,a)} \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}$$

where $\mathcal{T}^\pi : P(\mathbb{R})^{\mathcal{S}\times\mathcal{A}} \to P(\mathbb{R})^{\mathcal{S}\times\mathcal{A}}$ is the distributional Bellman operator, defined by:

$$\left(\mathcal{T}^\pi Z\right)^{(s,a)} = \int_{\mathbb{R}} \sum_{(s',a') \in \mathcal{S}\times\mathcal{A}} \left(f_{r,\gamma}\right)_\# Z^{(s',a')} \pi\left(a' \mid s'\right) p\left(\, \mathrm{d}r, s' \mid s, a\right)$$

for all $Z \in P(\mathbb{R})^{\mathcal{S}\times\mathcal{A}}$. This equation serves as the basis of distributional RL, just as the standard Bellman Operator. Here $a_t \sim \pi\left(\cdot \mid s_t\right)$ but we can define Optimal Bellman Operator called $\mathcal{T}$ with the same manner considering the *controle framework* where $a_t = \arg\max_{a'} \mathbb{E}[Z(S, a')]$.

Definition 4.3 (Wasserstein Metric.)

The p -Wasserstein distance is characterized as the L^p metric on inverse cumulative distribution functions (inverse CDFs). That is, the p -Wasserstein metric between distributions U and Y is given by,

$$W_p(U,Y) = \left(\int_0^1 \left| F_Y^{-1}(\omega) - F_U^{-1}(\omega) \right|^p d\omega \right)^{1/p}$$

where for a random variable Y, the inverse CDF, F_Y^{-1} of Y is defined by:

$$F_Y^{-1}(\omega) := \inf \{ y \in \mathbb{R} : \omega \leq F_Y(y) \}$$

where $F_Y(y) = \Pr(Y \leq y)$ is the CDF of Y.

Proposition 4.1 (Contraction of Distributional Bellman Operator for Wasserstein metric.)

In the context of distributional RL, let $\mathcal{Z}$ be the space of action-value distributions with finite moments:

$$\begin{aligned} \mathcal{Z} = & \{ Z : \mathcal{S} \times \mathcal{A} \to P(\mathbb{R}) \mid \\ & \mathbb{E}\left[|Z(s,a)|^p \right] < \infty, \forall (s,a), p \geq 1 \} \end{aligned}$$

Then, for two action-value distributions $Z_1, Z_2 \in \mathcal{Z}$, we will use the maximal form of the Wasserstein metric introduced by Bellemare et al. (2017),

$$\bar{d}_p \left(Z_1, Z_2 \right) := \sup_{s,a} W_p \left(Z_1(s,a), Z_2(s,a) \right)$$

It was shown that $\bar{d}_p$ is a metric over value distributions. Furthermore, the distributional Bellman operator $\mathcal{T}^\pi$ is a contraction in $\bar{d}_p$, So we get from (Bellemare, Dabney, and Munos 2017). $\mathcal{T}^\pi$ is a γ -contraction: for any two $Z_1, Z_2 \in \mathcal{Z}$,

$$\bar{d}_p \left(\mathcal{T}^\pi Z_1, \mathcal{T}^\pi Z_2 \right) \leq \gamma \bar{d}_p \left(Z_1, Z_2 \right)$$

Definition 4.4 (Cramer mectric)

The Cramér distance ℓ_p between two distributions $Z_1, Z_2 \in P(\mathbb{R})$, with cumulative distribution functions F_{1_1}, F_{1_2} respectively, is defined by:

$$\ell_p \left(Z_1, Z_2 \right) = \left(\int_{\mathbb{R}} \left(F_{Z_1}(s) - F_{Z_2}(s) \right)^p \ \mathrm{d}s \right)^{1/p}$$

Further, the supremum-Cramér metric $\bar{\ell}_p$ is defined between two distribution functions $Z_1, Z_2 \in P(\mathbb{R})^{\mathcal{S} \times \mathcal{A}}$ by :

$$\bar{\ell}_p(Z_1, Z_2) = \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} \ell_p \left(Z_1^{(s,a)}, Z_2^{(s,a)} \right)$$

Proposition 4.2 (Contraction of Distributional Belleman Operator for Cramer distance.)

The Cramér distance possesses the following characteristics (detailed derivation of each can be found in (Bellemare et al., 2017b)):

$$l_P(A+X, A+Y) \leq l_p(X,Y), \quad l_p(cX, cY) \leq |c|^{1/P} l_p(X,Y)$$

Using the above characteristics, the Bellman operator in l_p divergence is

$$\begin{aligned} l_p\left(\mathcal{T}^\pi Z_1(s,a), \mathcal{T}^\pi Z_2(s,a)\right) &= l_p\left(R(s,a) + \gamma P^\pi Z_1(s,a), R(s,a) + \gamma P^\pi Z_2(s,a)\right) \\ &\leq |\gamma|^{1/P} l_p\left(P^\pi Z_1(s,a), P^\pi Z_2(s,a)\right) \\ &\leq |\gamma|^{1/p} \sup_{s',a'} l_p\left(Z_1\left(s',a'\right), Z_2\left(s',a'\right)\right) \end{aligned}$$

Substituting the result into the definition of the maximal form of the Cramér distance yields

$$\begin{aligned} \bar{l}_p\left(\mathcal{T}^\pi Z_1, \mathcal{T}^\pi Z_2\right) &= \sup_{s,a} l_p\left(\mathcal{T}^\pi Z_1(s,a), \mathcal{T}^\pi Z_2(s,a)\right) \\ &\leq |\gamma|^{1/p} \sup_{s',a'} l_p\left(Z_1\left(s',a'\right), Z_2\left(s',a'\right)\right) \\ &= |\gamma|^{1/P} \bar{l}_p\left(Z_1, Z_2\right) \end{aligned}$$

Thus the distributional Bellman operator is a $|\gamma|^{1/p}$ -contraction mapping in the Cramér metric space, which was also proven in Rowland et al. (2019).

On the control setting : The fact that the Ditributional Belleman Operator is a contraction for two metrics is good news and gives us hope that our algorithm will converge through a fixed point algorithm. However, in the control case it has been proven (Dabney et al., 2017) that the operator $\mathcal{T}$ in the space of distribution is not a contraction for any metric space. This is due to the fact that multiple optimal policies can have very different distributions of the total return even though they have all exactly the same expected total return. Several practical issues arise in terms of implementation. How to estimate these distribution? Have we got convergence proof using approximation in algorithms?

4.3 DRL Algorithms and Practical Implementation

4.3.1 Description of DRL algorithms

There are 4 main steps in the implementation of a DRL algorithms which are :

- Distribution parametrisation,
- Stochastic approximation of the Distributional Bellman operator,

- Projection of the Bellman target distribution onto our parametrized distribution,
- Gradient update of parameters via a loss function.

Many distribution parametrization exists. The 3 main families are *Categorical Distribution, Quantile distribution and Gaussian Mixture.* We will details these after.

For all these algorithms we need to define a *projection* of the target which must me consistent with the contraction of the metric.

Stochastic approximation of these Distributional Bellman operator is also needed to use algorithms in practical case where environment is unknown.

Finally we update our distribution of return according to a loss function using Belleman operator for a given metric that is defined previously or another one. To ensure to have convergence of the algorithm a contraction property for a metric space is needed for the projection of the sampled Bellman Operator.

4.3.2 Categorical Algorithms

Categorical Distributional RL also called CDRL algorithms are a type of distributional RL algorithms that focus on approximating distributions with the parametric family of the form

$$\left\{ \sum_{k=1}^{K} p_k \delta_{z_k} \mid \sum_{k=1}^{K} p_k = 1, p_k \geq 0 \forall k \right\} \subseteq P(\mathbb{R})$$

where $z_1 < \cdots < z_K$ are an evenly spaced, fixed set of supports. For evaluation of a policy $\pi : \mathcal{S} \to P(\mathcal{A})$, given a collection of approximations $(Z(s,a) \mid (s,a) \in \mathcal{S} \times \mathcal{A})$, the approximation at $(s,a) \in \mathcal{S} \times \mathcal{A}$ is :

$$Z(s,a) \leftarrow \Pi_{\mathcal{C}} \mathbb{E}_\pi \left[\left(f_{R_0,\gamma}\right)_{\#} Z\left(S_1, A_1\right) \mid S_0 = s, A_0 = a \right].$$

With $\Pi_{\mathcal{C}} : P(\mathbb{R}) \to P\left(\{z_1, \ldots, z_K\}\right)$ is a projection operator defined for a single Dirac delta as

$$\Pi_C\left(\delta_w\right) = \begin{cases} \delta_{z_1} & w \leq z_1 \\ \frac{w - z_{k+1}}{z_k - z_{k+1}} \delta_{z_k} + \frac{z_k - w}{z_k - z_{k+1}} \delta_{k+1} & z_k \leq w \leq z_{k+1} \\ \delta_{z_K} & w \geq z_K \end{cases}$$

and extended affinely and continuously. In the language of operators, the CDRL update may be neatly described as $Z \leftarrow \Pi_{\mathcal{C}} \mathcal{T}^\pi Z$, where we abuse notation by interpreting $\Pi_{\mathcal{C}}$ as an operator on collections of distributions indexed by state-action pairs, applying the transformation (projection) to each distribution.

We saw that the operator $\Pi_C \mathcal{T}^\pi$ is a $\sqrt{\gamma}$ -contraction in the supremum-Cramér distance, this is why by the contraction mapping theorem, repeated CDRL updates converge to a unique limit point, regardless in theory of the initial approximate distributions. (Bellemare et al., 2017; Rowland et al., 2018).

Now we need to care about *Stochastic approximation.* The update $Z \leftarrow \Pi_C \mathcal{T}^\pi Z$ is ost of the time not computable in practice, due to unknown/intractable dynamics. An unbiased approximation to $\left(\mathcal{T}^\pi Z\right)(s,a)$ may be obtained by interacting with the environment to obtain a transition (s,a,r,s',a'), and computing the target

$$\left(f_{r,\gamma}\right)_{\#} Z\left(s',a'\right)$$

It can be shown (Rowland et al., 2018) that the following esimator is unbiased for the CDRL update defined before $\left(\Pi_{\mathcal{C}} \mathcal{T}^\pi Z\right)(s,a)$:

$$\Pi_{\mathcal{C}}\left(f_{r,\gamma}\right)_{\#} Z\left(s',a'\right)$$

Finally, the current estimate $Z(s,a)$ can be moved towards the stochastic target by following the (semi-)gradient of some loss, in analogy with semi-gradient methods in classical RL.
KL loss as Wasserstein gives biased estimate of the mean of distribution but are used in most of DRL papers has loss to update parameters. We are looking for the difference between a distribution and the projection of the push forward of the stochastic approximation of the Distributional Bellman Operator.

$$\mathrm{KL}\left(\Pi_{\mathcal{C}}\left(f_{r,\gamma}\right)_{\#} Z\left(s',a'\right) \| Z(s,a)\right)$$

Other losses, such as the Cramer distance, may also be considered as it is a contraction for the considered projection. Another type of algorithm exists based in Quantile estimation. For this algorithms, we will refer to QDRL for Quantile Distributional Reinforcement Algorithms.

4.3.3 QR-DQN (Quantile Regression DQN)

Quantile DRL (QDRL) algorithms are a type of algorithms which particularity is that they restrict approximate distributions to the parametric family of the form

$$\left\{\frac{1}{K}\sum_{k=1}^{K}\delta_{z_k} \mid z_{1:K} \in \mathbb{R}^K\right\} \subseteq P(\mathbb{R}).$$

For evaluation of a policy $\pi : \mathcal{S} \to P(\mathcal{A})$, given a collection of approximations $(Z(s,a) \mid (s,a) \in \mathcal{S} \times \mathcal{A})$, the approximation at $(s,a) \in \mathcal{S} \times \mathcal{A}$ is updated according to:

$$Z(s,a) \leftarrow \Pi_{W_1} \mathbb{E}_\pi\left[\left(f_{R_0,\gamma}\right)_{\#} Z\left(S_1,A_1\right) \mid S_0 = x, A_0 = a\right]$$

With $\Pi_{W_1} : P(\mathbb{R}) \to P(\mathbb{R})$ is a projection operator defined by

$$\Pi_{W_1}(\mu) = \frac{1}{K} \sum_{k=1}^{K} \delta_{F_\mu^{-1}(\tau_k)}$$

where $\tau_k = \frac{2k-1}{2K}$, and F_μ is the CDF of of μ.

Π_{W_1} is the quantile projection defined as above, and when applied to value distributions gives the projection for each state-value distribution.Given two distributions $Z_1, Z_2 \in \mathcal{Z}$ for an MDP with countable state and action spaces we get :

$$\bar{d}_\infty \left(\Pi_{W_1} \mathcal{T}^\pi Z_1, \Pi_{W_1} \mathcal{T}^\pi Z_2\right) \leq \gamma \bar{d}_\infty \left(Z_1, Z_2\right)$$

Moreover, the inverse cdf $F_\mu^{-1}(\tau)$ may also be characterised as the minimiser (over $q \in \mathbb{R}$) of the quantile regression loss

$$\mathrm{QR}(q; \mu, \tau) = \mathbb{E}_{Z \sim \mu} \left[[\tau \mathbf{1}_{Z > q} + (1 - \tau) \mathbf{1}_{Z \leq q}] \left| Z - q \right| \right].$$

This fact is of capital importance in deriving a stochastic approximation version of the algorithm. Indeed, with the same manner than for CDRL, the update $Z \leftarrow \Pi_{W_1} \mathcal{T}^\pi Z$ is most of the time not computable, due to unknown/intractable dynamics. Instead, a stochastic target may be computed by using a transition (s, a, r, s', a'), and updating each atom location $z_k(s, a)$ at the current state-action pair (s, a) by following the gradient of the QR loss:

$$\nabla_q \, \mathrm{QR} \left(q; \left(f_{r,\gamma}\right)_{\#} Z\left(s', a'\right), \tau_k \right) \Big|_{q = z_k(s,a)}$$

Because the QR loss is affine in its second argument, this yields an unbiased estimator of the true gradient which is not the case for a Wasserstein loss.

$$\nabla_q \mathrm{QR}\left(q; \left(\mathcal{T}^\pi Z\right)(s, a), \tau_k\right)\big|_{q = z_k(s,a)}$$

We are not using Wasesstein loss has it gives biased estimate. More details can be found in annex. Now some paper use Cramer as a loss with Quantile distribution.

4.3.4 IQN (Implicit Quantile Network)

This new algorithms is closed to QRDQN. In the Implicit Quantile Network (IQN) algorithm, Dabney et al. (2018) proposed to approximate the quantile function of $\mathbf{Z}(s, a)$ with a neural network and to learn it using quantile regression.

The quantile function of $\boldsymbol{Z}(s, a)$ can be learned using Neural Network. Denote $\hat{\mathbf{Z}}(s, a)$ the approximated random variable whose quantile function is given by a neural network $\Psi(s, \tau)$

which takes as input a state s and a probability $\tau \in [0,1]$ and returns the corresponding τ -quantile $\hat{\boldsymbol{Z}}_\tau(s,a)$ for each action a. Once a transition observed from our environment (s,a,r,s'), Ψ is trained by sampling $2N$ values $\tau = (\tau_1,\ldots,\tau_N)$ and $\tau' = (\tau'_1,\ldots,\tau'_N)$ with the uniform distribution on $[0,1]$. By inverse transform sampling, sampling $\boldsymbol{\tau}$ amount to sampling N values from $\hat{\mathbf{Z}}(s,a)$ corresponding to $\hat{\mathbf{Z}}_{\tau_1}(s,a),\ldots,\hat{\mathbf{Z}}_{\tau_N}(s,a)\Big)$, and similarly for $\boldsymbol{\tau}'$ and sampling from $\hat{\boldsymbol{Z}}\left(s',\pi\left(s'\right)\right)$ where π is the current policy. From samples,we compute N^2 TD errors in the distributional setting:

$$\delta_{ij} = r + \gamma \hat{\mathbf{Z}}_{\tau'_j}\left(s',\pi\left(s'\right)\right) - \hat{\mathbf{Z}}_{\tau_i}(s,a)$$

Following quantile regression, the following loss function for training the neural network Ψ in (s,a,r,s') is given by:

$$L_{IQN} = \frac{1}{N}\sum_{i\in[N]}\sum_{j\in[N]}\eta^K_{\tau_i}\left(\delta_{ij}\right)$$

where for any $\tau \in (0,1], \eta^\kappa_\tau\left(\delta_{ij}\right) = \left|\tau - \mathbb{I}\left(\delta_{ij}<0\right)\right|\frac{L_k(\delta_{ij})}{\kappa}$ is the quantile Huber loss with threshold κ with $L_\kappa(\delta) = \frac{1}{2}\delta^2$ for $|\delta| \leq \kappa$ or $\kappa\left(|\delta| - \frac{1}{2}\kappa\right)$ otherwise.

It is possible to sample τ with not an uniformly manner if we are looking for risk-adverse or risk-seeking policy using risk-adverse or risk-seeking function to sample. These three presented algorithms are very popular in DRL community and achieve very good results on discrete environment.

5 Our contribution : Variance Control using DRL for Medical Pathway

5.1 Description of the problem

The main idea here is to reduce variance of our trajectories as much as possible to avoid that some patient get non acceptable trajectories an a very long time waiting to be diagnosed. We propose two different ideas to control the variance of trajectories. One based on *Policy Iteration Principal* with a new objective function and one based on *Policy Gradient Under Constraint*. Both methods relies on DRL framework. In the first one, we will give a guarantee of convergence and an implementation. The second on is still and idea and need to be explore in depth.

5.2 Variance control using variance penalization

5.2.1 Motivation of the algorithm

A simple first idea is to take into account an estimation of the variance in the formulation of the objective function we are trying to maximize. The idea is to get a trade off between maximizing the expectation and reduce the variance of our policies. Then a natural idea is to write an objective function with both quantities. Then a natural formulation is to find an action that maximize. We define $\xi_\alpha(U)$ the operator such as $\xi_\alpha = \mathbb{E}[U] - \alpha\sqrt{\mathbb{V}[U]} = \mathbb{E}[U] - \alpha\sqrt{(\mathbb{E}[U^2] - \mathbb{E}[U]^2)}$. We are looking for action a^* such that :

$$a^* = \arg\max_a \xi_\alpha Z_\pi^{(s,a)} = \arg\max_a \mathbb{E}[Z_\pi^{(s,a)}] - \alpha\sqrt{\mathbb{V}[Z_\pi^{(s,a)}]} \tag{2}$$

However we must define α carefully to ensure convergence property. A second formulation could be the following to get a certain homogeneity between the square of the expectation and the variance but it is not acceptable has we are only dealing positive quantities but does not take into account negative reward.

$$a^* = \arg\max_a \xi_\alpha^{'} Z_\pi^{(s,a)} = \arg\max_a \mathbb{E}[Z_\pi^{(s,a)}]^2 - \alpha\mathbb{V}[Z_\pi^{(s,a)}] \tag{3}$$

So we will chose formulation 2 for the next of our work. The question now is how chose α and can we get convergence property of this objective function? That we will see in the next section.

5.2.2 Convergence of the algorithm

We know that $\mathcal{T}$ is not a contraction in general in the space of distribution but we are looking for a contraction of this operator in the sup norm under element-wise ξ evaluation. This is

motivated by the fact we know that $\mathcal{T}$ is a γ-contraction in sup-norm under element-wise expectation Bellemare et al. (2017) i.e:

$$\|\mathbb{E}\mathcal{T}U - \mathbb{E}\mathcal{T}V\|_\infty \leq \gamma\|\mathbb{E}U - \mathbb{E}V\|_\infty$$

The fact that it is contraction for a certain metric ensure by the Banach contraction mapping theorem that repeated updates converge to a unique limit point, regardless in theory of the initial approximate distributions.

Theorem 1 *The operator $\mathcal{T}$ is a γ- contraction for the infinite norm using element-wise evaluation with the statistic ξ_α, in other term :*

$$\forall U, V \subseteq P(\mathbb{R}), (s,a) \in \mathcal{S}\times\mathcal{A}, \quad \sup_{s,a} |\xi_\alpha\mathcal{T}U(s,a) - \xi_\alpha\mathcal{T}V(s,a)| \leq \gamma \sup_{a',s'} |\xi_\alpha(U)(s',a') - \xi_\alpha(V)(s',a')|$$

So we get a contraction of $\mathcal{T}$ which ensure use a convergence in control setting which is of capital importance for our algorithm. The sensitivity to α will be also studied empirically.

Proof :

$$\begin{aligned}
&\Big|\xi_\alpha\mathcal{T}U(s,a) - \xi_\alpha\mathcal{T}V(s,a)\Big| = \Big|\sum_{r,s'} p(r,s'|s,a)\Big(\int (r+\gamma u)dU(s', A^U(r,s')) \\
&- \alpha\Big[\int (r+\gamma u)^2 dU(s', A^U(r,s')) - \Big(\int (r+\gamma u)dU(s', A^U(r,s'))\Big)^2\Big]^{1/2} - \int (r+\gamma v)dV(s', A^V(r,s')) \\
&+ \alpha\Big[\Big(\int (r+\gamma v)dV(s', A^V(r,s'))\Big)^2 - \int (r+\gamma v)^2 dV(s', A^V(r,s'))\Big]^{1/2}\Big| \\
&= \Big|\sum_{r,s'} p(r,s'|s,a)\Big(r + \gamma\int udU - \alpha\Big[r^2 + \gamma^2(\int u^2 dU) + 2\gamma r(\int udU) - r^2 \\
&- \gamma^2(\int udU)^2 - 2\gamma r(\int udU)\Big]^{\frac{1}{2}}\Big) \\
&- \Big(r + \gamma\int vdV - \alpha\Big[r^2 + \gamma^2(\int v^2 dV) + 2\gamma r(\int vdV - r^2 - \gamma^2(\int vdV)^2 - 2\gamma r(\int vdV)\Big)\Big]^{\frac{1}{2}}\Big| \\
&= \gamma\left|\sum_{r,s'} p(r,s'|s,a)\left(\int udU - \alpha\Big[(\int u^2 dU) - (\int udU)^2\Big]^{\frac{1}{2}}\right) - \left(\int vdV - \alpha\Big[(\int v^2 dV) - (\int vdV)^2\Big)\Big]^{\frac{1}{2}}\right|
\end{aligned}$$

Choosing $A' = A^U(r,s') = \arg\max_{a'} \mathbb{E}[U] - \alpha\mathbb{V}[U]$ as in control setting. Fist line comes from definition of $\mathcal{T}$. So we get :

$$
\begin{aligned}
&\Big|\xi_\alpha TU(s,a) - \xi_\alpha TV(s,a)\Big| \\
=&\gamma\Big|\sum_{r,s'} p(r,s'|s,a)\Big(\max_{a'}\xi_\alpha(U)(s',a') - \max_{a''}\xi_\alpha(V)(s',a'')\Big)\Big| \\
\leq&\gamma\sum_{r,s'} p(r,s'|s,a)\max_{a'}\Big|\xi_\alpha(U)(s',a') - \xi_\alpha(V)(s',a')\Big| \\
\leq&\gamma\sup_{a',s'}\Big|\xi_\alpha(U)(s',a') - \xi_\alpha(V)(s',a')\Big|
\end{aligned}
$$

Because of triangle inequality and the fact that $|\max_a f(a) - \max_{a'} g(a')| \leq \max_a \mid f(a) - g(a) \mid$.

5.2.3 Description of the algorith

Considering distribution $\theta(s,a)$ represented by N points $\theta = (\theta_1, \ldots, \theta_N)$ and m the size of a minibatch, $\hat{\tau}_i = (i - 0.5)/N$ and as in QRDQN, $\rho_\tau(u) = u(\tau - \delta_{u<0})$ which is the quantile regression (QR) loss of QRDQN, the algorithm is the following :

Algorithm 10 QR-DQN with Variance Control objective

Input: $\gamma, \alpha, \theta, \theta'$, mini-batch (s_k, a_k, r_k, x'_k) for $k = 1 \ldots m$
for k=1,...m **do**
  $a'_k \leftarrow \arg\max_{a'} \xi(\theta')(x'_k, a')$
  $\mathcal{T}\theta_j(s_k, a_k) \leftarrow r_k + \gamma\theta'_j(s'_k, a'_k)$
end for
$\mathcal{L} \leftarrow \frac{1}{m}\sum_{k=1}^{m}\frac{1}{N^2}\sum_{i,j}\rho_{\bar{\tau}_i}(\mathcal{T}\theta_j(x_k, a_k) - \theta_i(x_k, a_k))$
Output $\nabla_\theta\mathcal{L}$.

From a practical point of view, to estimate the distribution, we have follow the algorithm QRDQN with 200 quantiles uniformly spread per action-state. The more quantiles we sample, the more accurate we represent the distribution but increase complexity. Variance in our algorithm is estimated using classic empirical estimator of the variance of our quantiles. Another idea is to use the framework of Constraint Optimization co control the variance estimated via DRL.

5.3 Variance control using Policy Gradient and Constraint RL

One could during control step try to optimize a function or surrogate function of the gradient of Policy Value under constraint that the variance is too big. For example, considering the objective of PPO algorithm :

$$J_{PPO}(\boldsymbol{\theta}) = \sum_{t=0}^{\infty} \min \Big(r_t(\boldsymbol{\theta}) \mathbb{E}_{\boldsymbol{\theta}} \left[\boldsymbol{Z}^{\overline{\boldsymbol{\theta}}}(s_t, a_t) - \boldsymbol{Z}^{\overline{\boldsymbol{\theta}}}(s_t) \right] \\ \text{clip}\left(r_t(\boldsymbol{\theta}), \boldsymbol{\varepsilon}\right) \mathbb{E}_{\boldsymbol{\theta}} \left[\boldsymbol{Z}^{\overline{\boldsymbol{\theta}}}(s_t, a_t) - \boldsymbol{Z}^{\overline{\boldsymbol{\theta}}}(s_t) \right] \Big)$$

where $\overline{\boldsymbol{\theta}}$ is the parameter of the current policy and $r_t(\boldsymbol{\theta}) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\overline{\theta}}(a_t|s_t)}$ .The problem can then be tackled by iteratively solving the following problem with this surrogate function:

$$\max_{\boldsymbol{\theta}} J_{PPO}(\boldsymbol{\theta}) \text{ s.t. } \rho\left(\boldsymbol{Z}^{\boldsymbol{\theta}}\right) \leq d$$

Now using the classic log barrier function, we reformulate problem as an unconstrained problem:

$$\max_{\boldsymbol{\theta}} J_{PPO}(\boldsymbol{\theta}) + \ln\left(d - \rho\left(\boldsymbol{Z}_i^{\boldsymbol{\theta}}\right)\right)$$

Here we are interested in the variance of the problem that can be estimated using quantile or implicit quantile approximation function:

$$\rho\left(\boldsymbol{Z}_i^{\boldsymbol{\theta}}\right) = \sum_{i=1}^{N} \left(\tau_i - \tau_{i-1}\right) \hat{\boldsymbol{Z}}_{\tau_i}(s_0)^2 - \left(\sum_{i=1}^{N} \left(\tau_i - \tau_{i-1}\right) \hat{\boldsymbol{Z}}_{\tau_i}(s_0) \right)^2$$

This idea has not been implemented has yet but raise some technical issues as the gradient

$$\nabla_{\boldsymbol{\theta}} J_{PPO}(\boldsymbol{\theta}) - \frac{\nabla_{\boldsymbol{\theta}} \rho\left(\boldsymbol{Z}^{\boldsymbol{\theta}}\right)}{d - \rho\left(\boldsymbol{Z}^{\boldsymbol{\theta}}\right)}$$

is difficult to compute because of $\nabla_{\boldsymbol{\theta}} \rho\left(\boldsymbol{Z}^{\boldsymbol{\theta}}\right)$ which is the gradient of the critic with respect to the parameters of the actor in PPO. An idea could be to connect actor network to critic network with the non linearity ρ to be able to compute gradient via automatic differentiation.

5.4 Comparison of different algorithms on our problem

In this experiment, the different algorithms presented above were compared, namely PPO, A2C, DQN and a version of QRDQN with and without variance penalisation. We took 200 quantiles for QRDQN and we use a target network updated every 200 steps for both DQN and QRDQN with experience replay trick. Exploration fraction in greedy agorithm is adapted to encourage exploration at the beginning and decrease progressively.

Each point in the graph is the average of himself and the 49 previous one to smooth the trajectories which are quite fluctuating by their stochastic nature. Each of the 50 point use for 1 point is the itself the average of 20 trajectories sampled via the learned algorithm. A complete description of the environment can be found in Annex 7.1.

Interpretation of Results of graphs 5 and 6:

- For our environment, the two versions of QRDQN find the best optimal policy the fastest. Then performances of PPO an DQN are quite similar for this discrete environment and finally A2C is the slowest to reach optimal policy.
- We can see that our version with a variance penalty has a smaller variance than the classical version of QRDQN. At first, the variances are similar and then the variances increase when a better policy is found. Finally, in the penalized version of QRDQN, the variance drops without affecting the mean in a second step. These results are promising and allow us to control the variance of our trajectories in order to make medical trajectory learning more stable.

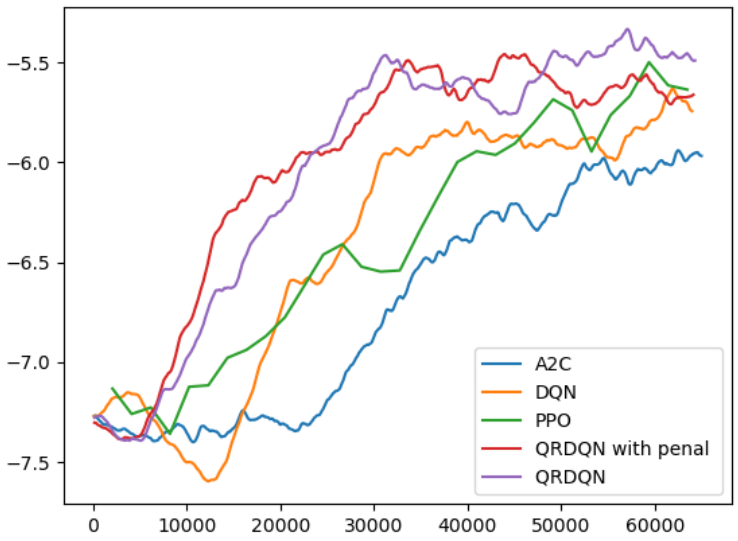

Figure 5: Mean reward of 20 trajectories in function of time-steps

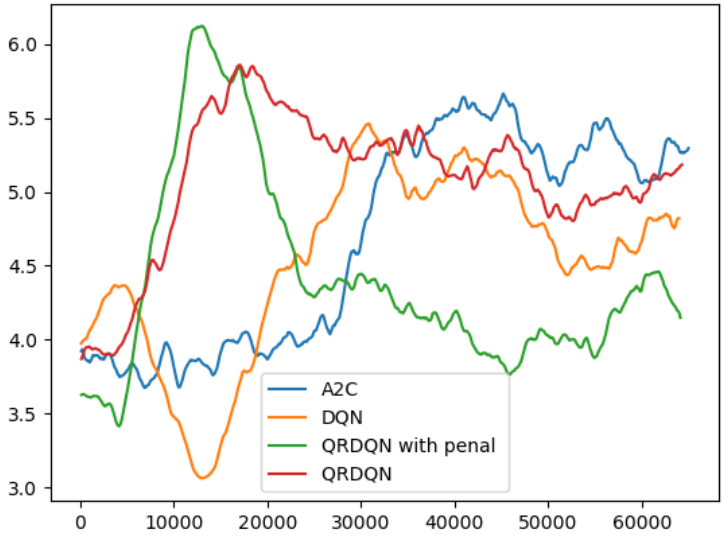

Figure 6: Variance reward of 20 trajectories in function of time-steps

To give a short illustration of distribution in QRDQN, we have represented in Fig. 7 three densities for the state "Neurologist,Cardiologist". Sticks in black are the estimated quantiles and in blue the density is plotted for each possible action which are "Oncologist, Dermatologist and Radiologist". We know that from our environment that the right recommendation should be to consult "dermatologist" has if the patient has visited this combination it has 0.8 to be diagnosed. For the other doctors it is less probable to be diagnosed. This is why the median of the distribution shift to the left.

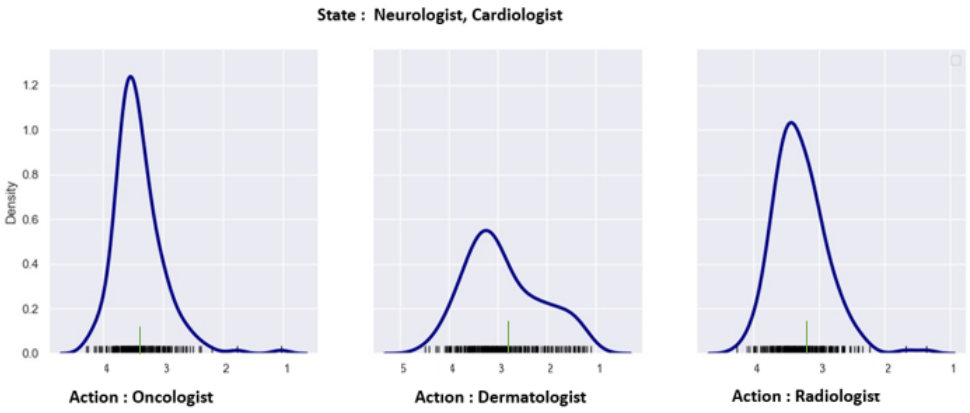

Figure 7: Densities of 3 action-state in function of the number of consultation remaining before being diagnosed. In black, quantiles of the distribution and in green the median of the distribution.

5.5 Sensitivity to α

In our algorithm the choice of α has be explored. In order to understand the influence of this parameter, we trained our algorithm on our problem and varied α. The average reward and variance of 20 trained trajectories were plotted. Every point is the average of the 49 previous one and itself to smooth the curve.

As can be seen, the average performances are quite similar and are not really dependent on α as we do not vary α too much. However, the choice of α is important for variance reduction because when we choose a too small α, the interest of variance reduction is lost in our algorithm as expectation is not regularized. So we must an acceptable value of α to ensure a good variance control of our problem. If α is too big, the algorithm will not be able to find a good policy in term of expectation.

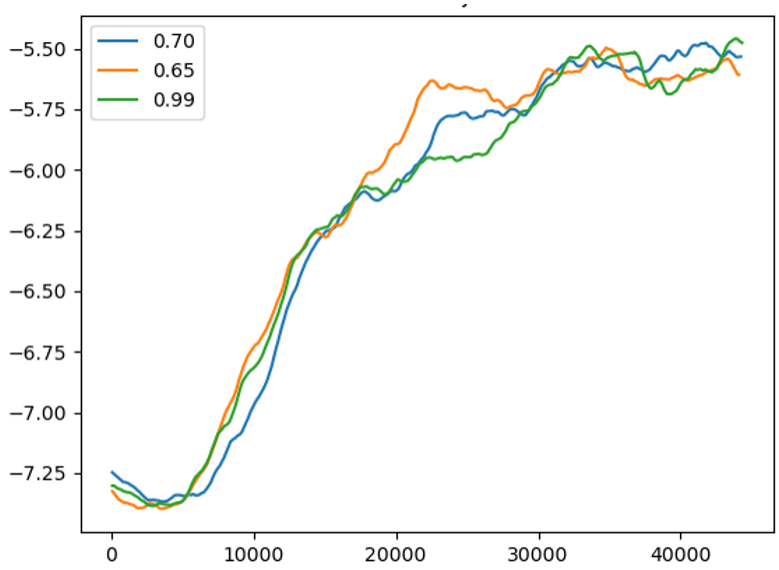

Figure 8: Mean reward for QRDQN with $\gamma\alpha$ penalization

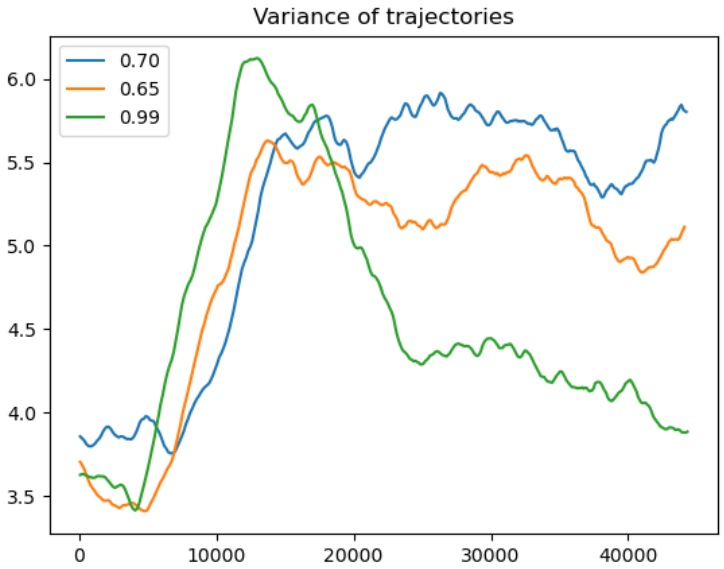

Figure 9: Variance reward for QRDQN with α penalization

6 Conclusions and Perspectives

We were interested in modelling the trajectory of rare disease patients and how to try to find a strategy to minimise diagnostic wandering. Particular attention was paid to reducing the variance of trajectories in learned policies while maintaining good overall performance in terms of expectation. We proposed and implemented an algorithm and justified its convergence to answer our problem. However, it is now a question of running it on real health data once the latter arrive for the beginning of the PhD thesis. It would be interesting to run these algorithms on other normalized environments of MujoCo.

It is also possible to ensure more certain trajectories with risk adverse reinforcement learning. This idea is developed first in Dabney et al. (2018), These two problems are quite similar and are based on the DRL theory as well. It would be interesting to look at the results and compare them to our methods to see if the variance also decreases. Moreover, Theoretical quantitative analysis of our algorithm would be appreciated afterwards.

7 Annexes

7.1 Description of the Environment of simulation

In the environment we consider 10 doctors. which are cardiologist, dermatologist, neurologist, oncolologist, radiologist, generalist doctor, urologist, gastrologist, gynecologist and venerologist.

We do not order the fact to see a cardiologist and a dermatologist and vice versa so there are 1023 states. The action chosen by the algorithm are the next doctor to be consulted.
For CPTs between doctor states, We pick *randomly transitions from one specialist to another one.* For CPT, to get diagnosed we have choose 5 diseases with are present with proportion $(1/5, 1/5, 1/5, 1/5, 1/5)$ in our population. They are defined as follow :

- **Disease 1**: Need to have visited cardio,dermato tho get 0.8 to be diagnosed and if have visited the 10 doctors, we get automatically diagnosed with probability 1.
- **Disease 2**: Need to have visited dermato,neuro tho get 0.8 to be diagnosed and if have visited the 10 doctors, we get automatically diagnosed with probability 1.
- **Disease 3**: Need to have visited radio,generalist tho get 0.8 to be diagnosed and if have visited the 10 doctors, we get automatically diagnosed with probability 1.
- **Disease 4**: Need to have visited generalist,uro,gastro tho get 0.8 to be diagnosed and if have visited the 10 doctors, we get automatically diagnosed with probability 1.

- **Disease 5**: Need to have visited gyneco,venero tho get 0.8 to be diagnosed and if have visited the 10 doctors, we get automatically diagnosed with probability 1.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

Dimitri P Bertsekas et al. *Dynamic programming and optimal control: Vol. 1*. Athena scientific Belmont, 2000.

Rémi Besson. These de doctorat decision support for prenatal ultrasound screening of the fetus anomalies using statistical learning, 10 2019. URL `http://www.theses.fr/2019SACLX037`.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in Neural Information Processing Systems*, 2018-December:8224–8234, 7 2018. URL `http://arxiv.org/abs/1807.01675`.

Bibhas Chakraborty and Susan A Murphy. Dynamic treatment regimes. *Annual review of statistics and its application*, 1:447–464, 2014.

Rich Colbaugh, Kristin Glass, Christopher Rudolf, and Mike Tremblay Volv Global Lausanne Switzerland. Learning to identify rare disease patients from electronic health records. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2018:340–347, 2018a. ISSN 1942597X. URL `https://www.ncbi.nlm.nih.gov/pubmed/`.

Rich Colbaugh, Kristin Glass, Christopher Rudolf, and Mike Tremblay. Robust ensemble learning to identify rare disease patients from electronic health records. volume 2018-July, pages 4085–4088. Institute of Electrical and Electronics Engineers Inc., 10 2018b. ISBN 9781538636466. doi: 10.1109/EMBC.2018.8513241.

Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. arxiv e-prints, page. *arXiv preprint arXiv:1710.10044*, 2017.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search, 2011.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value expansion for efficient model-free reinforcement learning, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

M Ghavamzadeh, S Mannor, J Pineau, A Tamar, Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends R in Machine Learning*, 8:359–492, 2015. doi: 10.1561/2200000049.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. *33rd International Conference on Machine Learning, ICML 2016*, 6:4135–4148, 3 2016. URL `http://arxiv.org/abs/1603.00748`.

Arthur Guez and H Van Hasselt. Deep reinforcement learning with double q-learning. *Association for the Advancement of Artificial Intelligence*, 2015.

David Ha and Jürgen Schmidhuber. World models. *arXiv*, 3 2018. doi: 10.5281/zenodo.1207631. URL `http://arxiv.org/abs/1803.10122http://dx.doi.org/10.5281/zenodo.1207631`.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *35th International Conference on Machine Learning, ICML 2018*, 5:2976–2989, 1 2018. URL `http://arxiv.org/abs/1801.01290`.

David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv*, 5 2020. URL `http://arxiv.org/abs/2005.01643`.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Yuan Ling, Sadid A Hasan, Vivek Datla, Ashequl Qadir, Kathy Lee, Joey Liu, and Oladimeji Farri. Diagnostic inferencing via improving clinical concept extraction with deep reinforcement learning: A preliminary study. In *Machine Learning for Healthcare Conference*, pages 271–285. PMLR, 2017.

Rudolf Lioutikov, Alexandros Paraschos, Jan Peters, and Gerhard Neumann. Sample-based informationl-theoretic stochastic optimal control. pages 3896–3902. Institute of Electrical and Electronics Engineers Inc., 9 2014. doi: 10.1109/ICRA.2014.6907424.

Frédéric Logé, Rémi Besson, and Stéphanie Allassonnière. Optimisation des parcours patients pour lutter contre l'errance de diagnostic des patients atteints de maladies rares, 5 2020. URL `https://hal.archives-ouvertes.fr/hal-02973092`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 12 2013a. URL `https://arxiv.org/abs/1312.5602v1`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013b.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016a.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016b.

Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Proceedings of the International Conference on Electronic Business (ICEB)*, 2018-December:421–429, 6 2020a. URL `http://arxiv.org/abs/2006.16712`.

Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. A framework for reinforcement learning and planning. *arXiv*, 6 2020b. URL `http://arxiv.org/abs/2006.15009`.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 7579–7586, 8 2017. URL `http://arxiv.org/abs/1708.02596`.

S Nguengang-Wakap, DM Lambert, A Olry, C Rodwell, C Gueydan, V Lanneau, Y Le Cam, and A Rath. Estimating global point prevalence of rare diseases: analysis of the orphanet database. In *EUROPEAN JOURNAL OF HUMAN GENETICS*, volume 27, pages 1768–1769. NATURE PUBLISHING GROUP MACMILLAN BUILDING, 4 CRINAN ST, LONDON N1 9XW, ENGLAND, 2019.

Aske Plaat, Walter Kosters, and Mike Preuss. Deep model-based reinforcement learning for high-dimensional problems, a survey. 8 2020. URL `http://arxiv.org/abs/2008.05598`.

Martin Prodel. Process discovery, analysis and simulation of clinical pathways using healthcare data, 4 2017. URL `https://tel.archives-ouvertes.fr/tel-01665163`.

Mark Rowland, Marc Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 29–37. PMLR, 2018.

Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G Bellemare, and Will Dabney. Statistics and samples in distributional reinforcement learning. In *International Conference on Machine Learning*, pages 5528–5536. PMLR, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588:604–609, 11 2019. doi: 10.1038/s41586-020-03051-4. URL `http://arxiv.org/abs/1911.08265http://dx.doi.org/10.1038/s41586-020-03051-4`.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. International Conference on Learning Representations, ICLR, 6 2016. URL `https://sites.google.com/site/gaepapersupp`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Jean Tarbouriech, Runlong Zhou, Simon S. Du, Matteo Pirotta, Michal Valko, and Alessandro Lazaric. Stochastic shortest path: Minimax, parameter-free and towards horizon-free regret. 4 2021. URL `http://arxiv.org/abs/2104.11186`.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. corr abs/1509.06461 (2015). *arXiv preprint arXiv:1509.06461*, 2015.

Stéphanie Nguengang Wakap, Deborah M Lambert, Annie Olry, Charlotte Rodwell, Charlotte Gueydan, Valérie Lanneau, Daniel Murphy, Yann Le Cam, and Ana Rath. Estimating cumulative point prevalence of rare diseases: analysis of the orphanet database. *European Journal of Human Genetics*, 28:165–173, 2020. doi: 10.1038/s41431-019-0508-0. URL `https://doi.org/10.1038/s41431-019-0508-0`.

Théophane Weber, Sébastien Racanière, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 2017-December:5691–5702, 7 2017. URL `http://arxiv.org/abs/1707.06203`.

Steffanie S Weinreich, R Mangon, JJ Sikkens, ME Teeuw, and MC Cornel. Orphanet: a european database for rare diseases. *Nederlands tijdschrift voor geneeskunde*, 152(9):518–519, 2008.