Université de Montréal

IFT3150 - Projet Informatique

Rapport de projet

---

auteur:

Éric Pfleiderer (20048976)

superviseure:

Morgan Craig, DMS

Date de remise: 30 avril 2020

1 Résumé

L'objectif de cette recherche est de développer une approche intégrative de la biologie computationnelle et de l'apprentissage par renforcement. Plus spécifiquement, on implémente l'algorithme d'apprentissage **AlphaZero** [1] dans un contexte d'essais cliniques virtuels [2] où on tente d'optimiser le traitement combiné d'immunothérapie et de virothérapie contre le mélanome. On s'intéresse principalement à l'implémentation de l'algorithme et aux modifications nécessaires à son application dans un contexte pratique. On cherche aussi à valider l'approche en terme de ressources calculatoires.

2 Introduction

Avant la joute historique de Kasparov et Deep Blue en 1996, une croyance populaire était que l'intuition de l'homme lui permettait de maitrîser des domaines d'applications qui lui seraient autrement innaccessibles. La défaite de Kasparov contre le superordinateur d'IBM invalide cette hypothèse et remet en question le rôle de l'homme et de l'ordinateur. Si une machine peut développer des perfomances surhumaines aux échecs, peut-elle devenir meilleur que l'homme dans la conduite automobile? Ou dans la reconnaissance d'émotion? Or, l'intelligence de Deep Blue appartient à un paradigme qui est aujourd'hui majoritairement obselète; le comportement de l'ordinateur était modélisé par un groupe d'ingénieur et d'expert d'échecs à l'aide, entre autre, de règles symboliques et de recherches brutes. Le développement d'un tel intelligence est extrèmement couteux et est limité par le savoir humain transmit à la machine. Depuis la popularisation de l'apprentissage machine, la méthode la plus populaire consiste plutôt à apprendre une représentation implicite des règles à l'aide d'exemples et/ou de recherches heuristiques. Ce paradigme fait ses preuves en 2016, lorsque **AlphaGo** remporte la victoire contre le champion du monde à Go, Lee Sedol, un jeu avec un espaces d'états vastement plus important que celui des échecs.

Dans le cadre de cette recherche, on désire appliquer l'algorithme d'apprentissage par renforcement de DeepMind dans le contexte de la biologie computationnelle; on cherche à façonner une approche intégrative permettant l'optimisation de thérapies combinées d'immunothérapie et de virothérapie au sein d'une cohorte virtuelle atteinte du mélanome. On s'intéresse aux modifications nécessaires à apporter à **AlphaZero** et, en deuxième lieu, à la validité de l'approche en terme de ressources nécessaires.

3 Théorie

3.1 Modèle

Le modèle mathématique employé dans le cadre des essais cliniques virtuels est tiré de Cassidy & Craig [2]. Il simule deux cycles cellulaires parallèles comportant chacun $j+2$ phases ou compartiments, où j dépend des paramètres typiques du patient. Le premier cycle inclue les cellules susceptibles d'être éliminées par le système immunitaire, tandis que le deuxième inclue les cellules qui développent une résistance à la réponse immunitaire. Chaque cycle comporte des cellules *dormantes* $\mathbf{Q}$ qui transitent vers la phase *Gap 1* $\mathbf{G}_1$. Les cellules $\mathbf{G}_1$ transitent à leur tour vers une série de j compartiments intermédiaires $\mathbf{A}_i$ qui cyclent vers la phase *dormante* $\mathbf{Q}$.

La réponse immunitaire est modélisée par 3 termes: le nombre de cytokines $\mathbf{C}$, le nombre de phagocytes $\mathbf{P}$ et le nombre de virus $\mathbf{V}$. Les termes $\mathbf{C}$ et $\mathbf{V}$ peuvent être influencées par l'administration de GM-CSF (immunothérapie) et de T-VEC (virothérapie). Le taux de variation instantanné de chaque quantité est modélisé par le système d'équations différentielles ordinaires suivant:

$$
\begin{aligned}
\frac{\partial \mathbf{Q}(t)}{\partial t} &= 2(1-\mu)\kappa_{tr}\mathbf{A}_j(t) - a_1\mathbf{Q}(t) - d_1\mathbf{Q}(t) - \psi_Q(\mathbf{U}(t))\mathbf{Q}(t) \\
\frac{\partial \mathbf{G}_1(t)}{\partial t} &= a_1\mathbf{Q}(t) - a_2\mathbf{G}_1(t) - d_2\mathbf{G}_1(t) - \psi_G(\mathbf{U}(t)) - \eta(\mathbf{V}(t))\mathbf{G}_1(t) \\
\frac{\partial \mathbf{A}_1(t)}{\partial t} &= a_2\mathbf{G}_1(t) - k_{tr}\mathbf{A}_1(t) - [\hat{d}_g + \eta(\mathbf{V}(t)) + \psi_G(\mathbf{U}(t))]\mathbf{A}_1(t) \\
\frac{\partial \mathbf{A}_i(t)}{\partial t} &= \kappa_{tr}(\mathbf{A}_{i-1}(t) - \mathbf{A}_i(t)) - [\hat{d}_g + \eta(\mathbf{V}(t)) + \psi_G(\mathbf{U}(t))]\mathbf{A}_i(t),\ i \in [2, j] \\
\frac{\partial \mathbf{Q}_{1,R}(t)}{\partial t} &= 2\mu\kappa_{tr}\mathbf{A}_j(t) + 2\kappa_{tr}\mathbf{A}_{j,R}(t) - a_1\mathbf{Q}_R(t) - d_1\mathbf{Q}_R(t) \\
\frac{\partial \mathbf{G}_{1,R}(t)}{\partial t} &= a_1\mathbf{Q}_R(t) - a_2\mathbf{G}_{1,R}(t) - d_2\mathbf{G}_{1,R}(t) - \eta(\mathbf{V}(t))\mathbf{G}_{1,R}(t) \\
\frac{\partial \mathbf{A}_{1,R}(t)}{\partial t} &= a_2\mathbf{G}_{1,R}(t) - k_{tr}\mathbf{A}_{1,R}(t) - [\hat{d}_g + \eta(\mathbf{V}(t)) + \psi_G(\mathbf{U}(t))]\mathbf{A}_{1,R}(t) \\
\frac{\partial \mathbf{A}_{i,R}(t)}{\partial t} &= \kappa_{tr}(\mathbf{A}_{i-1,R}(t) - \mathbf{A}_{i,R}(t)) - [\hat{d}_g + \eta(\mathbf{V}(t)) + \psi_G(\mathbf{U}(t))]\mathbf{A}_{i,R}(t),\ i \in [2, j] \\
\frac{\partial \mathbf{I}(t)}{\partial t} &= -\delta\mathbf{I}(t) + \eta(\mathbf{V}(t))[\mathbf{G}_1(t) + \mathbf{N}(t)] \\
\frac{\partial \mathbf{V}(t)}{\partial t} &= \alpha\delta\mathbf{I}(t) - \omega\mathbf{V}(t) - \eta(\mathbf{V}(t))[\mathbf{G}_1(t) + \mathbf{N}(t)] \\
\frac{\partial \mathbf{C}(t)}{\partial t} &= C_{prod}(\mathbf{U}(t)) - \kappa_{elim}\mathbf{C}(t) \\
\frac{\partial \mathbf{P}(t)}{\partial t} &= \varphi(\mathbf{C}(t)) - \gamma_p\mathbf{P}(t)
\end{aligned}
$$

Figure 1: Voir [2] pour la liste des paramètres et de leurs moyennes.

La taille de la tumeur est estimée en sommant tous les compartiments $\mathbf{Q}$, $\mathbf{G}_1$ et $\mathbf{A}$, ainsi que les cellules infectées $\mathbf{I}$. On utilise le solveur numérique *ode* de la librairy *scipy* pour solutionner le système.

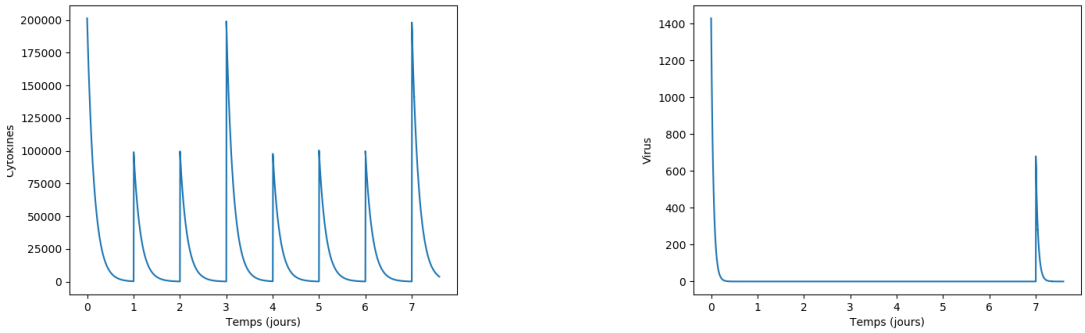
L'administration de doses durant le traitement est modélisée par une exponentielle décroissante à partir du temps d'administration t_i:

$$Dose(t) = \sum_{i=1}^{N} \frac{k_a F Admin_i(t)}{vol} \exp^{-k_a(t-t_i)}$$

où k_a est le taux d'absorption, F est la biodisponibilité et $Admin_i(t)$,

$$Admin_i(t) = \begin{cases} 0 \; if \; t < t_i, \\ Dose_i \; if \; t \geq t_i \end{cases} \quad ,$$

est la dose à administrer en quantité de cytokines ou de virus. À titre d'exemple, on affiche une thérapie combinée de 7 jours, où on a la possibilité d'administrer 1 ou 2 doses. L'immunothérapie est administrée quotidiennement, alors que l'administration de la virothérapie est hebdomadaire.



(a) Immunothérapie quotidienne

(b) Virothérapie hebdomadaire

Figure 2: Thérapie combinée de GM-CSF et de T-VEC en fonction du temps.

3.2 Essais cliniques virtuels

Une cohorte de patients virtuels est définie en variant certains paramètres du modèle autour d'une valeur moyenne:

Paramètre	Moyenne	Description
a_1	1.183	Phase *dormante* vers interphase (1/jour)
d_1	0	Taux de mortalité de la phase *dormante* (1/jour)
a_2	1.758	Phase interphase ver phase active (1/jour)
d_2	0.539	Taux de mortalité de la phase interphase (1/jour)
τ	0.8354	Durée attendue du cycle cellulaire (jour)
k_p	0.05	Phagocyte-cellule tumorale taux de contact (1/jour)
$k_{q,s}$	10	Constante de digestion cellulaire pour les phagocytes
k_{cp}	4.675	Taux de production maximal de phagocytes (10^{10} cellules/jour)

Table 1: Paramètres typiques et leur moyenne à travers les cohortes virtuelles.

La valeur de chaque paramètre est choisi semi-aléatoirement de sorte que le résultat soit inclue dans l'intervalle $[\mu - \frac{\mu}{10}, \mu + \frac{\mu}{10}]$, où μ est la valeur moyenne du paramètre à travers la cohorte. En pratique, on échantillonne une distribution normale avec une moyenne μ et un écart-type $\mu/30$.

L'essai clinique s'étend sur une durée de 180 jours; les premiers 75 jours correspondent à la phase thérapeutique, alors que les autres correspondent à une phase d'observation. Chaque patient est traité par une combinaison d'immunothérapie quotidienne et d'une virothérapie hebdomadaire. Lors de chaque administration de GM-CSF ou de T-VEC, on doit choisir la dose appropriée en fonction des paramètres typiques du patient et de son état courant. Les doses légales sont entre 0 et 4 fois la dose standard.

À la fin de la période d'observation, on détermine la taille de la tumeur et le nombre de doses totales administrées afin d'évaluer si le traitement combinée est une réussite. On discute des critères de réussite à la section 5.

3.3 Réseau de neurones

Dans le contexte de notre essai clinique, **AlphaZero** doit décider de la dose optimale à administrer en fonction des paramètres typiques du patient et de son état courant. Cette décision peut être modélisée comme un problème de classification à travers k classes, où chaque classe correspond à une dose légale. Simultanément, **AlphaZero** doit évaluer la position courante selon certains critères de réussite personalisés; il s'agit ici d'un problème de régression bornée. Le réseau de neurone au coeur de l'algorithme doit donc suivre une architecture *many-to-many*, soit un réseau avec plusieurs entrées et plusieurs sorties. L'implémentation du réseau est explicitée davantage à la section 4 et est fortement inspirée par l'article *Mastering the game of Go*[1].

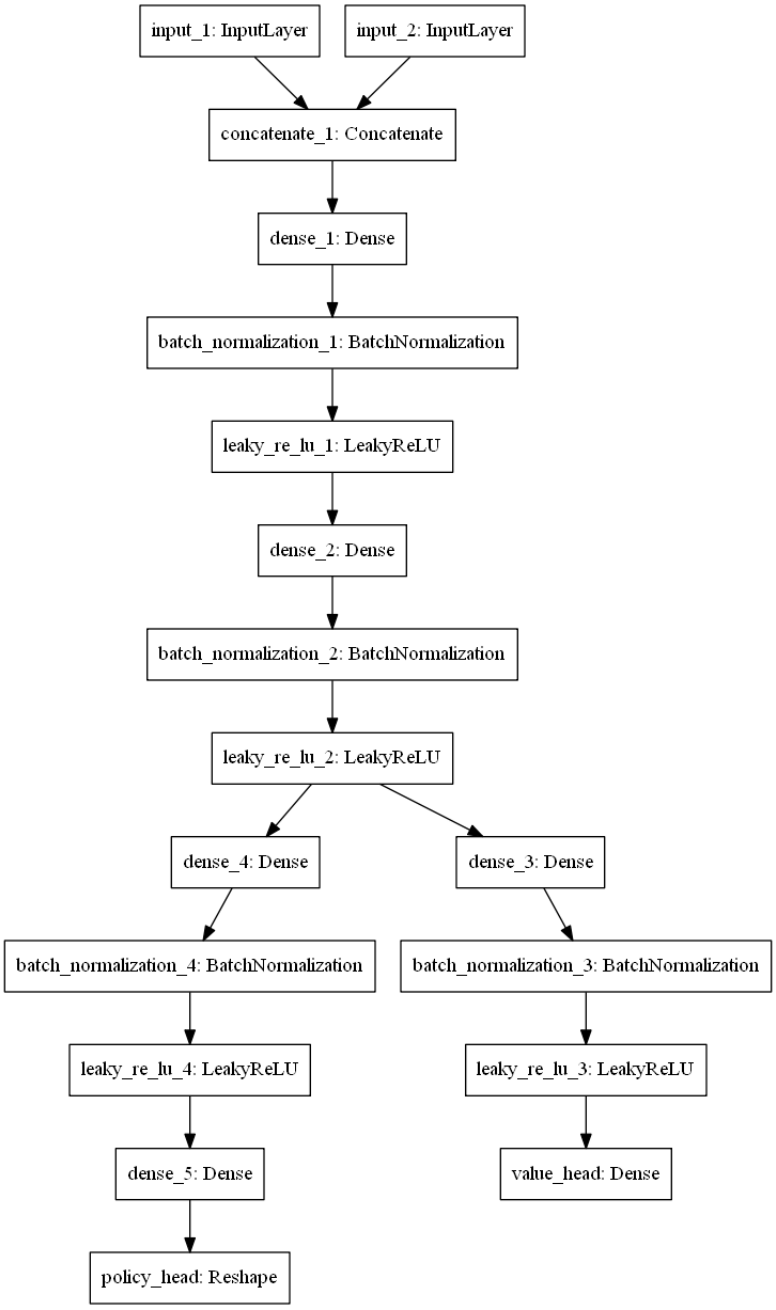


Figure 3: Exemple de réseau de neurone utilisé pour l'implémentation de l'algorithme AlphaZero.

3.4 Recherche arborescente Monte Carlo

Supposons le défi suivant: ayant accès à une quantité limité d'argent et à k machines à sous ayant chacun leur propre distribution de probabilité de livrer une récompense, comment pourrait-on maximiser le retour attendu? Quelle stratégie est optimale? Le *K-armed bandit problem* est un exemple classique d'apprentissage par renforcement qui démontre le dilemne entre l'exploration et l'exploitation et qui permet aussi de modéliser notre contexte d'essai cliniques; les actions à performer consistent à choisir la dose optimale à administrer, tandis que la récompense correspond à la mort ou la survie du patient virtuel suite à une période d'observation. Le protocole *AlphaZero* emploie la recherche arborescente Monte Carlo, ou MCTS, pour naviguer un tel dilemne.

Les MCTS sont des algorithmes de recherche heuristiques qui fonctionnent en explorant un espace d'état et en bâtissant un graphe équivalent. Chaque noeud du graphe possède les statistiques suivantes, qui sont constamment mises à jour lors de l'exploration:

1. N(s, a), le nombre de visite de l'action a, à partir de la position s.
2. W(s, a), la valeur totale attendue si on choisi l'action a, étant à la position s.
3. Q(s, a), la valeur moyenne attendue si on choisi l'action a, étant à la position s.
4. P(s, a), la probabilité à priori de sélectionner l'action a, étant à la position s.

L'unité centrale de la MCTS est la simulation Monte Carlo, composée de trois phases:

1. Sélection

   La phase de sélection consiste à parcourir un arbre Monte Carlo existant en profondeur et à choisir une feuille. Le choix d'action pour passer d'un noeud à l'autre durant le parcours est déterminé par la fonction

   $$\arg\max_a(Q(s_t, a) + U(s_t, a)), \tag{1}$$

   où $Q(s_t, a)$ est un terme d'exploitation et $U(s_t, a)$ est un terme d'exploration. $Q(s_t, a)$ est initialisé à 0 lors de la première visite et est ensuite mise à jour durant les propagations arrières futurs. Le terme d'exploration $U(s_t, a)$ prend la forme

   $$U = c_{PUCT} P(s, a) \sqrt{\frac{N(s, b)}{1 + N(s, a)}} \tag{2}$$

   où c_{PUCT} est une constante gérant le compromis exploration-exploitation. La probabilité $P(s, a)$ est initialisée lors de la première visite selon l'évaluation du réseau de neurones actuel. Puisque l'évaluation du réseau est pratiquement aléatoire durant les phases initiales de l'entraînement, le terme d'exploration U débute avec un comportement aléatoire qui perd éventuellement de sa stochasticité lors des phases plus avancées de l'entraînement.

2. Évaluation et expansion

   Si la feuille sélectionnée n'est pas terminale, on l'évalue à l'aide du réseau de neurones et, pour chaque action légale, on simule l'action et on ajoute l'état résultant à l'arbre Monte Carlo.

   Par contre, si la feuille sélectionnée est terminale, on l'évalue selon les règles du contexte. Puisque la fonction (1) est particulièrement bien adaptée pour des récompenses symétriques bornées par $[-1, 1]$, on score la feuille terminale similairement à un jeu; +1 pour une victoire, 0 pour une partie nulle et -1 pour une défaite. La définition d'une victoire, défaite ou partie nulle est explicité à la section 5.

3. Propagation arrière

   Durant la propagation arrière, la valeur obtenue durant l'évaluation est propagée aux noeuds visités durant la sélection afin de mettre à jour leurs statistiques.

Chaque recherche Monte Carlo implique une multitude de simulations. Les résultats de ces recherches sont utilisés pour guider l'entraînement du réseau de neurone en produisant des cibles d'entraînement, ou *labels*.

4 Implémentation

Le projet est implémenté en *python* à l'aide, entre autres, des libraires *numpy*, *scipy*, *tensorflow* et *keras*. L'entièreté du projet et de son historique est accessible sur le dépôt Github de l'auteur [3]. Les modules principaux sont les suivants:

1. Module **insilico**

   Le module **insilico** comporte toutes les fonctionnalités reliées au modèle mathématique de Cassidy & Craig [2].

   (a) Classe **State**

   La classe **State** livre les fonctionnalités nécessaires à la simulation d'une tumeur ou patient. Elle inclue le système d'équations différentielles **??**, un solveur numérique (*ode* de *scipy*), ainsi que l'ensemble de l'information et des méthodes nécessaire pour avancer la simulation d'un pas de temps.

   (b) Classe **Environment**

   La classe **Environment** est un *wrapper* de la classe **State**. Elle sert d'interface entre la classe **Agent** et la classe **State**. Elle facilite la manipulation et l'exploration de l'espace d'états durant l'entraînement.

2. Module **MCTS**

   Le module MCTS livre les fonctionnalités nécessaires pour bâtir et parcourir un arbre Monte Carlo. Notre implémentation suit une stratégie dite *epsilon-greedy*; lorsqu'on est positionné à la racine d'un arbre Monte Carlo, on modifie l'équation (2) et on y introduit une combinaison linéaire entre les probabilités à priori P(s, a) et un bruit stochastique $D(\alpha)$ provenant d'une distribution de Dirichlet. Le terme d'exploration U prend alors la forme

   $$U = c_{PUCT}[(1-\epsilon)P(s,a) + \epsilon D(\alpha)]\sqrt{\frac{N(s,b)}{1+N(s,a)}}. \tag{3}$$

   Cette modification permet d'introduire un comportement aléatoire à la fonction (1) qui ne s'atténue pas lorsque l'entraînement progresse et que les probabilités à priori P(s, a) perdent de leur stochasticité.

3. Module **learners**

   Le module **learners** inclue les fonctionnalités reliées à l'apprentissage machine.

   (a) Classe **Learner**

   La classe **Learner** correspond au réseau de neurone employé au coeur d'**AlphaZero**. Il s'agit de la structure du réseau et ne contient aucun algorithme d'apprentissage automatique. On suit l'implémentation de DeepMind [1] en implémentant deux têtes, soit la *policy head*, qui tente de prédire l'action optimale, et la *value head*, qui tente d'estimer la qualité de la position courante. Les entrées du réseau de neurones consistent en la concaténation de deux vecteurs: le premier contient les paramètres typiques du patient, tandis que l'autre contient la solution courante du modèle mathématique.

   (b) Classe **Agent**

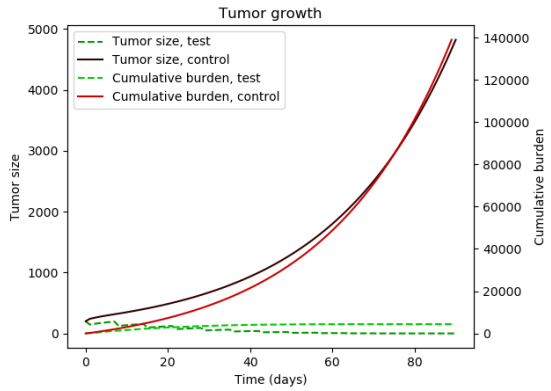
   La classe **Agent** emploie la classe **Learner** et le module **MCTS**. Elle livre les fonctionnalités d'apprentissages par renforcement, c'est-à-dire qu'elle possède un réseau de neurones, peut bâtir et explorer ses propores arbres Monte Carlo et contient toutes les méthodes nécessaire pour l'interaction de ces structures.

4. Module **training**

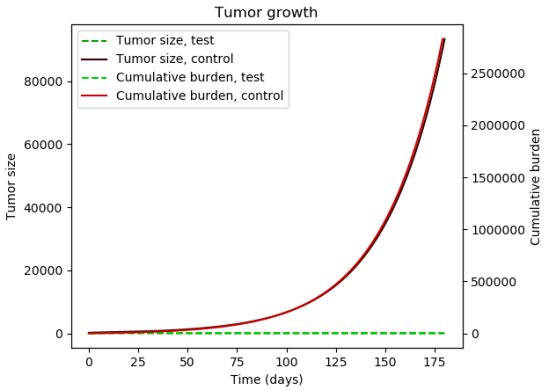
   Le dernier module correspond à la routine d'apprentissage par renforcement. Une instance de la classe **Agent** est mise en compétition successive contre elle-même. Lors de chaque joute, l'agent bâtit un arbre MCTS et utilise les résultats de la recherche pour guider l'entraînement de son réseau de neurones. La nouvelle version de l'agent est seulement préservé si elle bat la génération précédente par une marge prédéfinie.

5 Discussion

Le système **??** possède deux bassins d'attractions qui définissent le comportement de la solution dans l'espace de phases. Il peut atteindre un point d'équilibre correspondant à la persistance de la tumeur, ou encore un point d'équilibre modélisant une extinction. On vérifie l'intégrité et la cohérence de notre implémentation en s'assurant qu'il est possible d'accéder à chaque régime. On lance deux simulations, l'une où le patient demeure non traité et l'une où on administre la thérapie combinée la plus aggressive possible. Les résultats des simulations sont affichés à la figure 4:



(a) Grosseur des tumeurs en fonction du temps pour 80 jours.



(b) Grosseur des tumeurs en fonction du temps pour 180 jours.

Figure 4: Grosseur de la tumeur de deux patients en fonction du temps. Le premier patient, en trait plein, demeure non traité, tandis que le deuxième patient, en trait pointillés, reçoit le traitement légal le plus aggressif possible.

References

[1] **Silver D.**, **Schrittwieser J.**, **Simonyan K.** et **al**, Mastering the game of Go without human knowledge, Nature 550, 354–359 (2017) doi:10.1038/nature24270

[2] **Cassidy T.**, **Craig M.**, Determinants of combination GM-CSF immunotherapy and oncolytic virotherapy success identified through in silico treatment personalization, PLoS Comput Biol 15(11): e1007495

[3] **Pfleiderer E.**, dépôt Github IFT3150, https://github.com/EricPfleiderer/IFT3150