

# Apprentissage par renforcement profond

Nathan Barloy

Telecom Nancy

Université de Lorraine

Email : nathan.barloy@telecomnancy.eu

Aurélien Delage

Telecom Nancy

Université de Lorraine

Email: aurelien.delage@telecomnancy.eu

Olivier Buffet

and Vincent Thomas

INRIA

Equipe : Larsen

E-mail : olivier.buffet@loria.fr

and vincent.thomas@loria.fr

**Résumé—The abstract goes here.**

## I. INTRODUCTION

Les débuts de l'intelligence artificielle - ou du moins la perception de l'éventualité de créer, un jour, des machines intelligentes qui surpasseraient les humains - peuvent être attribués aux travaux d'Alan Turing. Il proposa en 1936 une théorie formelle de ce qui sera plus tard un ordinateur. Une Machine de Turing est un concept abstrait, qui consiste en un ruban sur lequel se déplace une tête d'écriture, permettant de réaliser des opérations simples (arithmétiques, par exemple) [2].

14 ans plus tard, Alan Turing présente le concept de Test de Turing. Il concerne directement l'intelligence artificielle en proposant un test qui permettrait de mesurer la qualité de l'intelligence de la machine. L'expérience consiste à faire discuter un humain et une machine, l'humain ne sachant pas s'il discute avec un autre humain ou une machine, et simplement à lui demander à la fin de la discussion, s'il pense avoir parlé avec un humain ou avec une machine. A ce jour, aucune intelligence artificielle n'a **réellement** réussi à passer le Test de Turing [3] [6].

Cela commence à nous amener à notre sujet. Depuis ces définitions formelles d'Alan Turing, la qualité des intelligences artificielle n'a eu de cesse de croître.

En effet, on peut citer la célèbre victoire de l'algorithme *Deep Blue* contre M.Garry Kasparov [4], champion du monde d'échec, en 1996. On peut aussi citer la victoire d'*AlphaGo* battant en 2017 le champion du monde du jeu de Go Lee Sedol. Sa version adaptée au jeu d'échec est aussi particulièrement impressionnante tant dans l'originalité du jeu proposé que dans l'incapacité des maîtres d'échec de comprendre réellement, et d'anticiper les coups de l'algorithme [5].

Et voilà notre sujet: il s'avère que ces deux algorithmes ont été développés par Google<sup>tm</sup> en utilisant des techniques de **Reinforcement Learning**, ou autrement dit : **Apprentissage par Renforcement Profond**.

Dans ce PIDR, nous nous proposons d'appliquer les techniques de base d'Apprentissage par Renforcement Profond à des jeux de type Atari.

## A. Machine Learning, Réseaux de neurones

Le Machine Learning (ou Apprentissage Automatique) consiste, entre autre..., à faire apprendre à un ordinateur à réagir, à résoudre un problème, en réaction à un certain jeu de données en entrée. Une des techniques principales du Machine Learning est l'utilisation de réseaux de neurones, et c'est ce qui est utilisé pour l'Apprentissage par Renforcement Profond. Nous allons donc nous concentrer dessus pour cette partie.

### 1) Réseaux de Neurones: Qu'est-ce qu'un réseau de neurone?

Commençons par dire - de manière très schématique - que c'est un système, composé de couches et de noeuds, qui permet de répondre quelque chose (une solution) à un jeu de données en entrée (un problème). Disons aussi qu'ils réalisent des performances particulièrement impressionnantes sur certains problèmes.

Disons aussi qu'outre les performances sur les jeux, concernant la voiture autonome, ou tout autre application des réseaux de neurones, il est à noter que les algorithmes utilisant des réseaux de neurones tendent à percevoir la **sémantique des mots et des idées**. Par exemple, l'équation roi-homme+femme est résolue par l'algorithme en donnant la solution: reine. Les mots ne sont plus des chaînes de caractères, ils sont des objets sémantiques en espaces de très grande dimension [7].

*Nous recommandons très vivement au lecteur de regarder la référence citée ci-dessous, ainsi que la série de vidéos de vulgarisation qui l'accompagne, présentant de manière formelle la perception de la sémantique des mots en tant qu'objets d'espaces de grandes dimension, que l'on cherche à appliquer en espaces de dimension raisonnable, permettant l'analyse algorithmique...*

Un réseau de neurone tend donc à réfléchir, penser comme un humain. Ceci est la production d'un réseau de neurones de Google<sup>tm</sup> à qui l'on a demandé de penser, de créer un tableau artistiques.

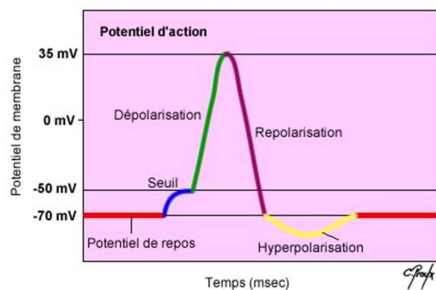


De même, la startup *DeepL* a réussi à produire un algorithme traduisant des livres de manière stupéfiante. Leur IA semble avoir appris la sémantique des mots [9]

## 2) Comment fonctionne un réseau de neurones?:

Premièrement, un neurone informatique est construit pour fonctionner de la même façon qu'un neurone humain. Le réseau de neurone fonctionne alors comme le réseau de neurones humain.

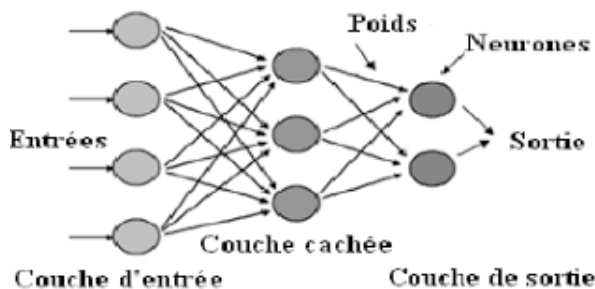
Un neurone n'est rien de plus qu'une cellule du cerveau humain recevant des impulsions électrique au cours du temps. Si le potentiel électrique est supérieur à un seuil, ce neurone est alors excité [8]:



Source : [http://www.cours.pharmacie.com/physiologie/systeme\\_nerveux.html](http://www.cours.pharmacie.com/physiologie/systeme_nerveux.html)

C'est alors la combinaison de ces neurones interconnectés en réseau qui permet la puissance de l'intellect humain.

Un réseau de neurone informatique a pour but d'imiter ce comportement. Un réseau peut être représenté schématiquement de la façon suivante :



On remarque que toutes les valeurs de la couche  $i-1$  entrent en entrée de tous les neurones de la couche  $i$  ( $i \neq 1$ ). A chacune de ces valeurs est associé un poids.

Ecrivons les équations, ce sera plus simple.

Notons  $n_i^j$  le neurone  $i$  de la couche  $j$ ,  $\forall i \in [0, p], \forall j \in [0, n]$ , en supposant qu'il y a  $n$  couches, et  $p$  neurones à la couche  $i$ .

Alors on a :

$$n_i^{j+1} = \sum_{k=1}^p (n_k^j \cdot W_k^{j,l}) + b \quad (1)$$

où on note  $W_k^j$  les poids,  $k \in [0, p]$  associés aux  $p$  neurones de la couche  $i$ .

On voit que l'on peut réécrire cette formule sous forme matricielle. Prenons soin de définir correctement les vecteurs et les matrices utilisés.

- $\forall j \in [0, n], n^j \in R^p$
- $\forall j \in [0, n], W^j \in M_{n,p}$
- $\forall j \in [0, n], b \in R^p$

et on a :

$$n^{j+1} = W \cdot n^j + b \quad (2)$$

Les poids pouvant être grand,  $\forall j \in [0, n], n^j \in R^p$ . On aimerait qu'à l'instar du neurone humain, son état soit activé ou non activé, ie que l'on ait :

$$\forall j \in [0, n], n^j \in [0, 1]^p. \quad (3)$$

Ceci se fait en utilisant une fonction d'activation, ie une fonction de  $R \mapsto [0, 1]$  ou  $[-1, 1]$ . Différentes fonctions existent [9], la plus utilisée est la fonction sigmoïde :  $s_a(x) = \frac{1}{1+e^{-x}}$ .

On remarque que la fonction  $f$  pour passer d'une couche à une autre est une fonction affine, de partie linéaire  $\mathcal{L}(f) = W$  qui n'est pas nécessairement bijective.  $f$  ne l'est alors pas. Ceci montre que l'on peut obtenir des mêmes résultats tout en ayant des poids sur les neurones totalement différents. Cela a une importance quand on test la sensibilité du réseau à des petites perturbations non prévues : si les poids sont trop grands, il y a un risque de propagation de l'erreur et de divergence par rapport à la solution attendue.

On sait maintenant comment réagit un réseau de neurone à un jeu de données en entrée. Mais comment fait-il pour résoudre des problèmes, pour apprendre à donner la bonne solution?

## 3) Apprentissage du réseau de neurone:

Premièrement, les techniques d'Intelligence Artificielle à base de réseaux de neurones sont des techniques dites **supervisées**, ie des techniques dont une partie des données pour l'apprentissage seront utilisées afin de vérifier la validité de l'apprentissage.

Les neurones humains ont la faculté de renforcer leurs interconnexions. En effet, quand on apprend une nouvelle langue, certaines connexions entre neurones se renforcent, et deviennent de plus en plus facile à emprunter, d'où l'apprentissage de la langue. Les réseaux de neurones utilisent le même procédé.

L'idée est de renforcer certaines connexions en donnant un poids plus important entre neurones, ie un coefficient de la matrice W fort.

Pour apprendre l'Anglais, il faut pratiquer. Pour un réseau de neurones, c'est la même chose! On nourrit le réseau avec un grand nombre de données en entrée et de leurs résultat, plusieurs fois, ce qui lui permet d'apprendre les poids de ses matrices W de chaque couche, afin d'avoir une prédiction la plus fiable possible.

#### 4) Technique de rétro-propagation de l'erreur, du gradient.:

A chaque étape (chaque fois qu'on donne un jeu de donnée au réseau de neurones), le réseau doit mettre à jour ses poids. Une des techniques les plus connues pour faire ceci - et que nous utiliserons dans notre projet - est la **Rétro-propagation du gradient**.

Pour l'expliquer, il faut d'abord préciser ce que l'on cherche à faire. Globalement, on cherche à minimiser la différence entre les prédictions du réseau de neurone, et les résultats connus.

Le résultat de sortie sera nécessairement un vecteur  $y_{pred} \in \mathbb{R}^p$ , p pouvant valoir 1.

On définit alors **la fonction de perte**  $L$  comme étant une mesure de la différence entre  $y_{connu}$  et  $y_{pred}$ . On peut alors utiliser plein de fonctions différentes, comme  $\|y_{connu} - y_{pred}\|_1$ ,  $\|y_{connu} - y_{pred}\|_2$ ,  $\|y_{connu} - y_{pred}\|_\infty$ ,... Ce sont des applications différentiables. En effet, si on leur associe un produit scalaire, alors  $\|x\| = \langle x, x \rangle$ , or le produit scalaire étant une application bilinéaire de  $\mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$ , elle est différentiable. En effet, on a, pour toute application bilinéaire  $f$  d'un espace vectoriel  $E$  dans un espace vectoriel  $F$ , la propriété suivante :

$$\exists C > 0, \forall, x \in E^n, \|f((x_1, \dots, x_n))\|_F \leq C \prod_{i=1}^n \|x_i\|_E \quad (4)$$

et alors,

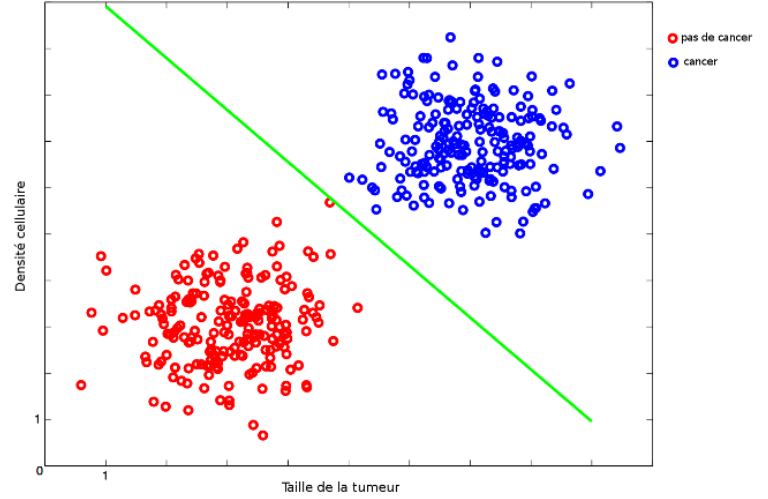
$$\begin{aligned} & \|f(x+h) - f(x) - \sum_{i=1}^n f((x_1, \dots, h_i, \dots, x_n))\| \\ & \leq C \prod_{i=1}^n \|x_i\|_E = o_{h \rightarrow 0_E}(\|h\|) \end{aligned} \quad (5)$$

La fonction de perte est alors différentiable, on peut donc **suivre son gradient** pour minimiser la fonction de perte, ie  $\tilde{x} = x - \alpha \nabla_x(f)$ , avec  $\alpha$  un coefficient arbitraire.

La mise à jour des poids est un peu complexe et lourde en notation, nous envoyons le lecteur sur la page wikipédia l'expliquant [11].

C'est sur la mise à jour des poids selon cette technique que tout l'apprentissage repose.

5) *Resultats, visualisation*: On l'a dit tout à l'heure, un neurone se calcule de la façon suivante (1)  $n_l^{j+1} = \sum_{k=1}^p (n_k^j \cdot W_k^{j,l}) + b$ . Il s'agit d'un produit scalaire auquel on ajoute la constante b. Ceci est **exactement** l'équation d'un hyperplan de dimension p-1 dans un espace de dimension p. On peut alors interpréter le travail d'un neurone dans un réseau de neurone comme la délimitation d'un hyperplan séparant en deux les données en entrée de la couche. On peut alors visualiser le fonctionnement d'un neurone, par exemple pour la classification de données, dans  $\mathbb{R}^2$  [12]:



#### B. Apprentissage par Renforcement, ou Q-learning

### II. CONCLUSION

The conclusion goes here.

### ACKNOWLEDGMENT

The authors would like to thank...

## RÉFÉRENCES

- [1] H. Kopka and P. W. Daly, *A Guide to L<sup>A</sup>T<sub>E</sub>X*, 3rd ed. Harlow, England: Addison-Wesley, 1999.
- [2] <http://zanotti.univ-tln.fr/turing/>
- [3] <https://culture.univ-lille1.fr/fileadmin/lna/lna66/lna66p04.pdf>
- [4] [https://fr.wikipedia.org/wiki/Matchs\\_Deep\\_Blue\\_contre\\_Kasparov](https://fr.wikipedia.org/wiki/Matchs_Deep_Blue_contre_Kasparov)
- [5] <https://www.youtube.com/watch?v=bo5pIUo86BU>
- [6] <https://www.csee.umbc.edu/courses/471/papers/turing.pdf>
- [7] <https://www.youtube.com/watch?v=r4EtRfHSU7E>
- [8] <http://www.cours-pharmacie.com/physiologie/systeme-nerveux.html>
- [9] <https://www.lebigdata.fr/deep-learning-ia-traduction>
- [10] <http://www.statsoft.fr/concepts-statistiques/reseaux-de-neurones-automatisees/reseaux-de-neurones-automatisees.htm>.XGINLlwzZPZ
- [11] [https://fr.wikipedia.org/wiki/Rétropropagation\\_du\\_gradient](https://fr.wikipedia.org/wiki/R%C3%A9tropropagation_du_gradient)
- [12] <https://markdown.data-ensta.fr/s/machine-learning-introductions>