Matthieu
Tom

Janvier 2023

SAE 1.02
Comparaison d'algorithmes

UVSQ université PARIS-SACLAY
IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY

1. Le travail demandé

Après avoir implémenté le jeu Diamants sur Python dans la SAÉ 1.01, notre devoir cette fois-ci était de créer une intelligence artificielle pouvant jouer au jeu.
Bien que nous soyons libres sur la manière de procéder, nous avons trouvé que cette SAÉ était compliqué. En effet, non seulement on devait fournir une IA fonctionnelle, mais en plus, elle devait être la meilleure possible. En cela, il fallait trouver le plus vite possible l'alchimie parfaite pour répondre à ces besoins, pour un jeu mélangeant aléatoire, habilité et ruse.

Ainsi, nous sommes heureux de vous présenter après deux semaines de travail notre IA nommée Picsou. L'IA Picsou, qui tout comme le célèbre personnage de Disney, est audacieux et malin pour arriver à ses fins.

2. Approches

Nous avions déjà quelques idées par rapport à ce que c'est une intelligence artificielle. Tom avait fait son exposé de grand oral en terminal sur ce sujet, et Matthieu avait vu une vidéo YouTube expliquant le fonctionnement d'une IA jouant à Super Mario World.

Afin de concevoir une IA capable de jouer correctement au jeu Diamant, nous avons réfléchi étape par étape aux facteurs clé pour la réalisation de ce projet. Grâce à la SAÉ 1.01 Implémentations, nous connaissions déjà bien les règles et le déroulement du jeu.

Donc, premièrement, pour la réalisation de ce projet, il était nécessaire de comprendre comment le moteur du jeu qui nous était donné fonctionne, comment il intègre les règles du jeu et le déroulement de la partie afin de créer un modèle décisionnel efficace.

Ensuite, nous devions décider quel genre d'IA voulions nous faire, soit une IA qui joue grandement avec l'aléatoire, soit qui analyse certains détaille pour savoir s'il est intéressant de sortir de la grotte, il est ainsi possible de créer, soit une IA qui prend beaucoup de risque ou une IA prudente pour sécuriser ce qu'elle a récupéré.

Puis décider de comment l'implémenter, soit qui suit des conditions fixes, par exemple si l'IA possède dans la manche plus de vingt diamants, on la fait rentrer pour sécuriser un maximum, ou bien, on laisse l'IA décider ces choix par elle-même, c'est-à-dire qu'on va lui donner des informations sur l'état du jeu et elle va faire des connexions entre ces valeurs et va faire son choix.

Nous, nous avons décidé d'essayer de réaliser une IA qui s'adapte elle-même aux différentes situations et qui prend une décision grâce à un prétendu système neuronal que va posséder l'IA.

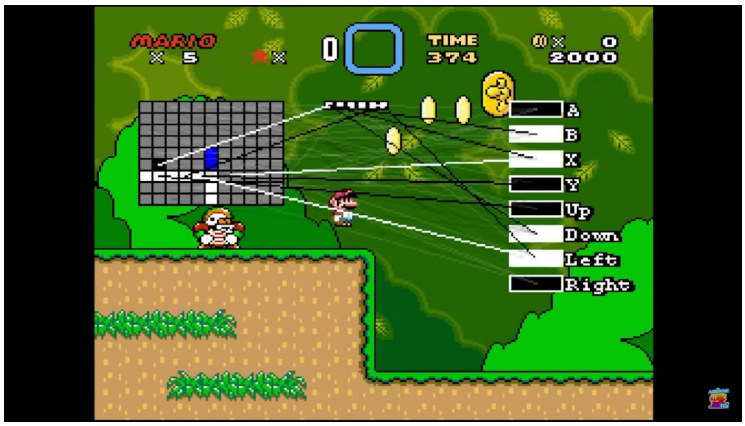
Pour ce faire, il est important de collecter et d'analyser les données du jeu. Cela peut inclure des informations sur les cartes restantes dans le jeu, les points gagnés et perdus, le nombre de dangers en jeu, etc. Ces données vont être utilisées pour entraîner et guider le réseau de neurones à prendre des décisions en fonction de situations de jeu spécifiques.

Il est également important de développer une méthode pour évaluer les risques et les opportunités lorsque l'IA prend des décisions. Cela peut tenir compte de la répartition des pièges et des trésors dans la pile restante ou de l'état actuel du jeu comme combien y a-t-il de joueurs qui continuent l'exploration, combien de reliques ont été trouvées et quels sont les prochaines cartes qui peuvent sortir.

En résumé, afin de développer une IA efficace pour le jeu Diamant, nous avons remarqué qu'il était nécessaire de comprendre les règles et les objectifs du jeu, de collecter et d'analyser les données du jeu, de développer une méthode d'évaluation des risques et des opportunités et de mettre en place une IA qui prend en compte tous ces paramètres pour décider du meilleur choix à faire.

3. Création de l'IA

Notre projet se base en grande partie sur le travail du vidéaste Laupok sur YouTube. Il avait créé une intelligence artificielle pouvant jouer à Super Mario World. L'IA devait analyser la situation, et en fonction de ce qu'il remarque, faire une action. Il avait fait une première vidéo où il vulgarise le sujet, et une autre où il explique son code.



L'IA jouant à Super Mario World.

En fonction de l'environnement de Mario, cela va affecter les connexions qui vont actionner un bouton ou non.

Source : "j'ai créé une IA qui joue à mario toute seule", Laupok (YouTube)

Initialement, nous pensions seulement nous inspirer de l'idée et faire une sorte de réseau neuronal très simple. L'IA voit ce qu'il se passe, et en fonction on tire des nombres. On les additionne, et plus c'est élevé, plus on a de chance de partir.
Mais après réflexion, la décision que prendrait l'IA relèverait de l'aléatoire. De plus, en fonction des combinaisons, on aurait pu arriver à un résultat imprévu.

L'approche de Laupok nous paraissait intéressante et nous ne voulions pas l'abandonner. Mais par la difficulté de cerner la fabrication d'une tel idée, nous avons pris le choix de reprendre son projet et de le réadapter pour Diamants.

Cependant, Super Mario World et Diamants sont deux jeux très différents. D'autant plus que nous n'avions pas les mêmes objectifs avec Laupok.

Voici un extrait des différences qu'on devait prendre en compte.

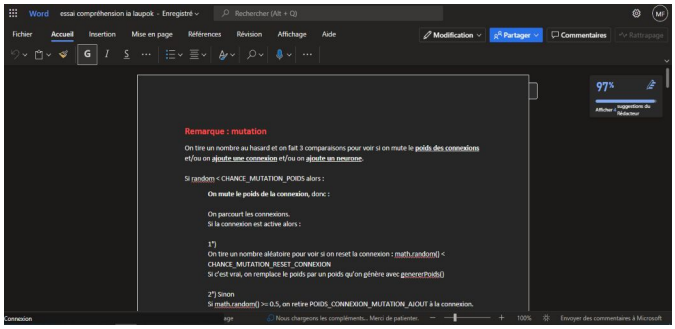
Super Mario World (niveau 1)	**Diamants**
L'IA est programmée en LUA (langage pour un émulateur Super Nintendo)	L'IA doit être programmée en Python
L'IA doit prendre des décisions en temps réel	L'IA doit répondre uniquement quand c'est son tour.
Peu voir pas d'aléatoire dans le niveau 1	L'aléatoire est la base du jeu Diamants
1 joueur	4 joueurs
L'objectif est de terminer le niveau	L'objectif est d'avoir le plus de diamants de ses adversaires.
Une population d'individus est limité à 100 individus.	Une population d'individus est limité à 4 individus.

Le LUA est très différent de Python. Par exemple, le premier indice d'une liste n'est pas 0, mais 1. On peut difficilement définir ce qu'est une liste en LUA. Ca semble être à la fois une liste comme en Python, mais ça ressemble aussi à la programmation orientée objet. Bien que le langage ne semble pas faire l'objet.

```
local lePlusFort = newReseau()
for i = 1, #laPopulation, 1 do
    if lePlusFort.fitness < laPopulation[i].fitness then
        lePlusFort = copier(laPopulation[i])
    end
end
```

Une boucle for en lua

La première étape, était de comprendre le code. Bien qu'il y ait la vidéo d'explication, le mieux était de comprendre en analysant le code.
Matthieu s'occupait de cette partie. Il écrivait le fonctionnement de l'IA dans un document texte, étape par étape.



Etude du fonctionnement de l'IA.
8 pages de faite pour environs 1 journée et demie de travail.

Ce n'était pas quelque chose d'évident.
Il fallait savoir ce qu'était une génération d'individu, une population, un individu, une espèce.
Ainsi, une population est un ensemble d'individus. Cependant, parmi ces individus, on peut remarquer des différences. Pour les différencier, ils sont classés par espèce.
L'algorithme génétique se repose sur la théorie de l'évolution. Plus une espèce est puissante au sein d'une population, plus elle peut faire d'enfants pour la prochaine génération d'individus. Une espèce qui ne fait pas d'enfants est une espèce qui va être éliminé.
Chaque enfant est issu en principe de deux parents d'une même population, ainsi, une partie de ses gènes viendront du premier parent, et le reste du deuxième.
Mais l'enfant apporte ses particularités à l'espèce, il mute donc.

L'individu est un système de neurone. Il est composé de neurones et de synapses, ici appelé connexion. Un neurone contient une valeur, et un identifiant unique.
Il y a trois type de neurones : les neurones d'INPUT, CACHER et d'OUTPUT.
Les neurones d'INPUT sont celles qui représentent la situation du jeu. Par exemple, un neurone est consacré au nombre de diamants que possède le joueur.
Ce nombre est associé à la valeur du neurone.
Les neurones CACHER contenant le résultat d'un ou plusieurs calculs, ce qui fait qu'elles peuvent influer sur le réseau de neurone.
Mais également les neurones d'OUTPUT, neurones contenant les valeurs finales et qui permettent de faire une action.
Ces neurones sont connectés par des connexions, qui possède un poids.
On multiplie ce poids par la valeur du neurone à l'entrée de la connexion, et le calcul est stocké dans le neurone qui le suit.

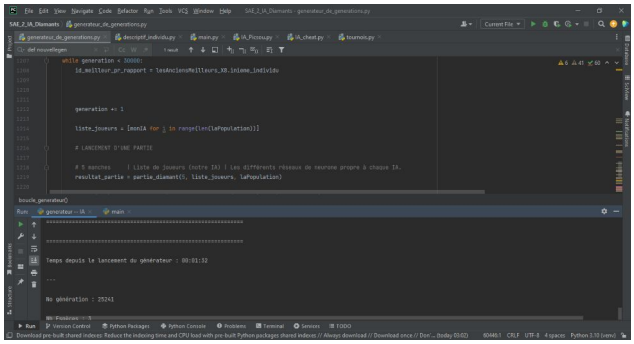
Une fois avoir compris le principe et réussit à implémenter notre version de l'IA de Laupok, il a fallu créer les neurones d'entrée.
Pour cela, nous avons dû créer des petites fonctions qui renvoi des valeurs ou une probabilité que tel ou tel évènements arrive. Nous avons donc eu besoin de scruter les évènements du jeu mais également les autres joueurs.
Ainsi, dans notre fichier IA_Picsou, nous gardons les informations concernant la partie. Pour les caractéristiques des joueurs, nous avons créé une classe spéciale : Joueur_X8.
Chaque objet de cette classe représente un joueur, l'IA y compris. On y marque le nombre de diamants que possède le joueur, son historique de gain.

Certaines de ces informations peuvent ne pas être utilisé. Cependant, nous avons préféré les implémentés en amont en cas de besoin.
Pour mettre à jours les réseaux de neurones, nous avons repris la fonction majReseau qui appel les diverses fonctions pour mettre leurs valeurs dans les neurones.

Pour faire les calculs, nous avons la fonction feedforward qui parcourt la liste de connexions. Au préalable, elle réinitialise tous les neurones CACHER et d'OUTPUT, et calcul toutes les connexions active.
Elle prend la valeur du neurone d'entrée, le multiplie par son poids, et l'ajoute à la valeur du neurone de sortie.
Le fait qu'une connexion soit active ou non dépend de la mutation de l'individu.
Certaines mutations rajoutent au réseau des neurones. Ces neurones étant placer au milieu d'une connexion, elle la divise en deux nouvelles connexions. L'ancienne connexion n'est donc plus utilisée, mais reste existence au cas où l'IA essayerait de la recréer.

Enfin nous avons decision_plan_A qui récupère les valeurs des neurones d'OUTPUT. Le premier neurone correspond à l'action de partir, et l'autre de rester. On les compare et en fonction du résultat, l'IA reste ou part.

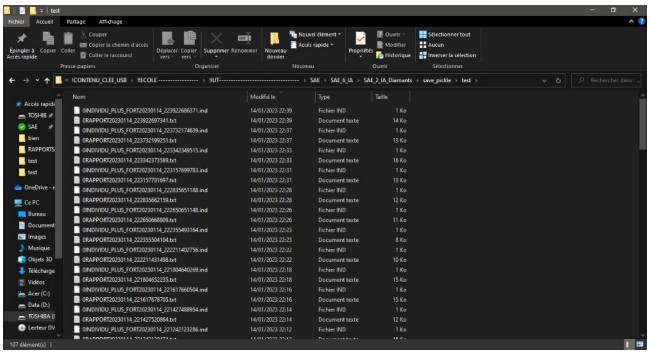
L'IA Picsou dépend d'un individu. Sans individu, elle ne peut pas fonctionner.
Pour nos tests, nous avons dû créer un générateur de population. C'est ce générateur qui s'occupe de la fastidieuse tâche qu'est l'algorithme génétique.
Notre IA ne fait qu'appliquer le réseau de neurone envoyé par ce générateur.



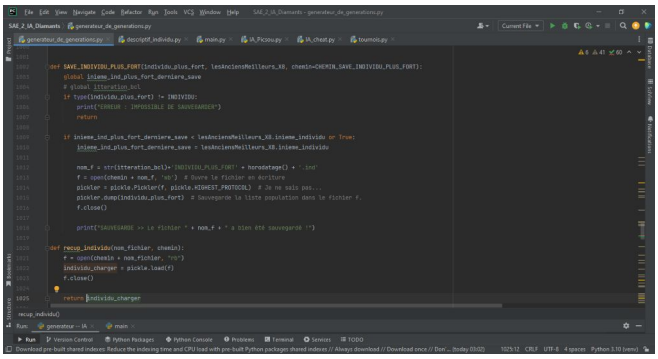
Le générateur de population,
la pièce maîtresse pour l'élaboration
des individus

Nous avons ainsi dû modifier le moteur du jeu pour pouvoir envoyer au chargement de l'IA son individu. A la fin d'une partie, en plus de renvoyer les scores finaux, le moteur nous renvoyait les objets individus et un rapport de la partie.
Chaque ligne du rapport était un str, mis dans une liste qu'on enregistrait dans un fichier texte.

L'une des parties les plus intéressantes du projet était la façon dont nous gardons en mémoire nos individus favoris. Pour cette tâche, nous nous sommes aidée de Pickle, un module Python.
Ce module transforme une variable, une liste, un objet ou tout élément qu'on peut manipuler dans Python sous forme d'un fichier binaire.
Ce fichier peut ensuite être récupéré et réutiliser tel quel, comme si on ne l'avait jamais transformé en fichier binaire.



Nos objets INDIVIDU transformés en fichier binaire.
Le “.ind” n’est pas une vrai extension.
C’est là pour qu’on sache qu’il s’agit d’un individu



Exemples de deux fonctions utilisant Pickle.
En haut, une fonction archivant un individu sous forme de fichier, en bas, une fonction qui la restaure.

Nous avons rencontré différente difficultés avec nos individus :

- Les scores étaient identiques
- Soit ils quittaient tout le temps au premier tour, soit ils succombaient aux dangers.

Nous avons remarqué que paradoxalement, plus il y a de neurone, plus l’IA avait tendance à ne pas être efficace. Nous avons donc dû retirer beaucoup de neurones afin qu’on puisse avoir un résultat convenable.

Malheureusement, au moment d’élire le meilleur individu qui pourra accompagner notre IA, nous nous sommes rendu compte de quelques erreurs.
Par exemple, nos IA manipulaient un deck différent que celui du moteur du jeu.
Nous avons dû recréer des individus en urgence pour pouvoir clôturer le projet.

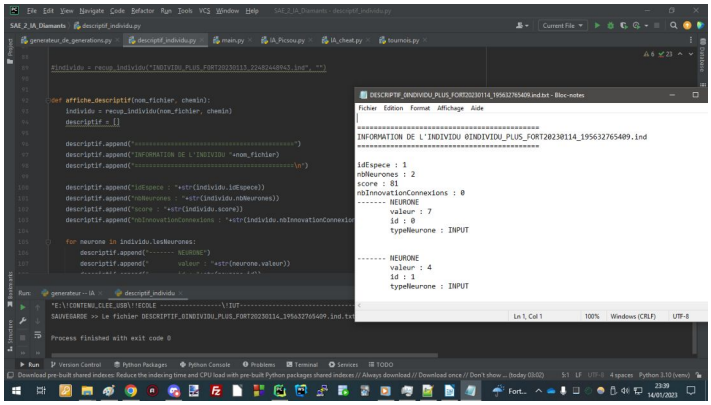
```python
def tournoi(listeIndividu):
    """

    :param listeIndividu: Liste de groupe de 4 toutes les IA du tournoi  16 joueurs
    :return:
    """
    liste_winner_phase1 = []
    for i in range(4):
        print("tournois n°", i)
        liste_winner_phase1.append(mini_tournoi(listeIndividu[i]))

    winner = mini_tournoi(liste_winner_phase1)
    return winner

def mini_tournoi(listeIndividu):
    tournoi_score = [0, 0, 0, 0]
    for i in range (15000):
        score = partie_diamant(5,["IA_PipoX8_EPREPA", "IA_PipoX8_EPREPA", "IA_PipoX8_EPREPA", "IA_PipoX8_EPREPA"],listeIndividu)[0]

        for i in range(len(score)):
            index_best = score.index(max(score))
            score[score.index(max(score))] = -1

            tournoi_score[index_best] += len(score)-i
    return listeIndividu[tournoi_score.index(max(tournoi_score))]
```

Nos fonctions tournois qui ont malheureusement été inutile.

Heureusement, nous avons réussi à trouver le meilleur individu possible.
Nous supposons toutefois que le choix de l’algorithme génétique n’était pas la bonne solution pour cette SAÉ.

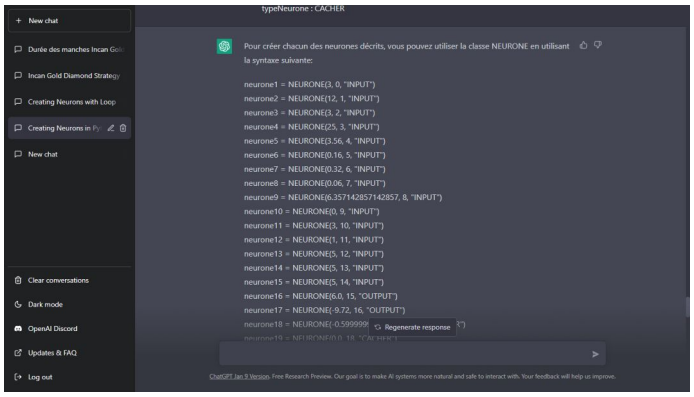
4. Finalisation de l'IA

Pour pouvoir insérer directement l'individu dans le fichier Python, nous nous sommes aidé de ChatGPT, une IA.
Pour cela, nous avons créé une fonction qui affiche le descriptif d'un individu souhaité.



La fonction et le fichier texte qu'elle a généré

Nous lui avons envoyé nos classes en plus de ce descriptif afin qu'il puisse nous donner les lignes pour créer les différents objets souhaités.



Exemple de ce qu'a pu nous renvoyer ChatGPT quand on l'a testé.
Cela nous as simplifié beaucoup le travail.

Notre IA était divisée en plusieurs fichiers. Vu que nous devions rendre qu'un unique fichier, nous avons du tout fusionner en un seul fichier python : IA_Picsou.

C'est ainsi que s'achève notre aventure avec l'intelligence artificielle Picsou.

5. Crédits

« j'ai créé une IA qui joue à mario toute seule », Laupok :
https://www.youtube.com/watch?v=F63GNXGHVwM

« comment setup l'ia + code source review », Laupok :
https://www.youtube.com/watch?v=u5xCl1bSe6o

Code source de son IA :
https://pastebin.com/Jcvdqhqm

Un site montrant un réseau de neurone :
https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=5,2,2,2&seed=0.24975&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

La majorité des fonctions reprise du code de Laupok possède le meilleur nom, et sont crédité la plupart des cas.
Nous les avons réadaptés pour qu'elles puissent fonctionner sur Python, nous avons revu les types de structure et refait les commentaires par ce que nous comprenons.
Les fonctions ont été adapté pour notre IA.