Algorithmes Évolutionnaires — Master 2 MIAGE IA^2
Travaux dirigés N° 6 : neuroévolution

Andrea G. B. Tettamanzi
Université côte d'Azur
andrea.tettamanzi@univ-cotedazur.fr

Année universitaire 2023/2024

Résumé

Dans cette séance, nous allons développer un algorithme évolutionnaire du type NEAT pour optimiser un réseau de neurones. Pour ce faire, nous utiliserons les *frameworks* DEAP et Keras.

1 Introduction

Pour appliquer les algorithmes évolutionnaires aux réseaux de neurones, il y a deux possibilités : soit on code soi-même les primitives du réseau de neurones, soit on s'appuye sur un frameworks, comme Keras, qui fournit toutes ces primitives et plein de choses en plus. Dans les deux cas, il y a des avantages et des inconvénients : si on décide de coder soi-même, il faudra investir du temps dans le développement, mais on aura un contrôle et une compréhension parfaits sur le code produit ; si on opte pour un framework, in faudra investir du temps pour apprendre à l'utiliser, mais on aura accès à tout ce qu'il y a de plus récent et performant en matière d'apprentissage profond.

2 Consignes

1. Réfléchissez à votre politique *make or buy* : vous êtes libres de choisir si utiliser Keras, développer votre propre code pour les réseaux de neurones, ou utiliser une autre bibliothèque/API.
2. Développez un algorithme évolutionnaire du type de NEAT (vu en cours) pour trouver une architecture de réseau de neurones *feed-forward*. En gros, vous devrez établir une représentation pour la structure et les poids de connexion du réseau, coder les opérateurs génétiques de mutation et recombinaison, ainsi que la fonction de fitness, qui sera basée sur la fonction de perte du réseau de neurones.
3. Choisissez un jeu de données, par exemple sur le site `https://www.kaggle.com`. Veillez à en choisir un où les variables sont toutes numériques. Il est conseillé d'utiliser le même que vous aviez choisi lors du TD N° 5.
4. S'il vous reste du temps, vous pouvez utiliser l'apprentissage du réseau (*backpropagation* ou similaire) comme opérateur de recherche locale (mutation Lamarckienne) et comparer les résultats de votre algorithme avec ou sans cet opérateur.

Rendez votre code et vos observations par courriel, dans un archive zippé.