Héloïse de Castelnau

ESILV A3 TD.D

21/03/2021

Rapport de TP de Datascience et IA

Approximation de fonction par Algorithme génétique

Sommaire

INTRODUCTION

Le projet donné consiste en la détermination des coefficients de Weierstrass, fonction qui nous indiquerait la température d'une étoile inconnu.
Il nous faut déterminer ces derniers en se basant sur un échantillon de valeurs données.

CONSTRUCTION DE L'ALGORITHME

1. Espace de recherche

L'espace de recherche se définie comme l'ensemble des états atteignables ou solutions depuis l'état initial par n'importe quelle séquence d'action. Dans notre cas c'est en premier lieu un triplet de paramètres abc : $[a; b; c]$ auxquels nous devons intégrer leur espace de définition. Donc trouver :

$$[a; b; c]$$

$$\mathrm{A} = \{\, \mathrm{a} \in \mathrm{R} \mid \mathrm{a} \in]0, 1[\,\}$$

$$\mathrm{B} = \{\, \mathrm{b} \in \mathrm{N} \mid \mathrm{b} \in [1, 20]\,\}$$

$$\mathrm{C} = \{\, \mathrm{c} \in \mathrm{N} \mid \mathrm{c} \in [1, 20]\,\}$$

Ensuite nous devrons intégrer à notre algorithme un Population, soit un ensemble d'individus. Nous devrons alors chercher au sein de cette population, les individus qui correspondent le mieux à la fonction obtenue avec les valeurs données dans le fichier csv. Comment obtenir une valeur nous permettant de comparer ces individus ? c'est le rôle de la fonction fitness.

2. Fonction Fitness

La fonction fitness mesure la qualité de l'individu, exprimée ici sous forme d'un nombre. Nous cherchons l'écart entre la fonction issue des valeurs données et la fonction issue des valeurs obtenues en créant notre individu, en valeur absolue. Nous aurons ainsi point par point selon i : un indicateur temporel, l'écart entre t(i) la température obtenue sur la courbe et t2(i) celle obtenue avec notre calcul de la température selon le triplet $[a; b; c]$. Cette fonction nous sera utile pour pouvoir effectuer au sein de notre population un tri des individus selon leur « coût » (sélection) et donc théoriquement avec les valeurs les plus proches de celles demandés. Mais il se peut que ces dernières ne soit, par exemple qu'un triplet ou les valeurs de b et c sont proches de la réalité et notre a diffère. Comment éviter ce problème ? on s'inspire de la génétique et on effectue des croisements et mutation.

3. Opérateurs de croisement et mutation

Les opérateurs de croisement et mutation. Le croisement tout d'abord , permet un mélange des particularités des individus choisis par la reproduction et la mutation altère aléatoirement les particularités d'un individu.

Ici on effectue un croisement en créant 2 nouveaux individus à partir de 2 initiaux et en interchangeant les coefficients au sein des triplets de chacun des parents. Par exemple :

$$Ind1 : [a; b; c] + Ind2[a'; b'; c'] \rightarrow [a'; b; c'] + [a; b'; c]$$

On observe un taux de croisement de 0.7 dans la nature, nous instaurons alors afin de nous rapprocher de la réalité un croisement seulement si un taux random se trouve en dessous de cette valeur. Après plusieurs tests, j'ai observé que cette condition ne rajoutais rien à notre sélection, sinon s'en éloignait. J'ai donc décide de la supprimer.

Pour la mutation, on mute aléatoirement un ou plusieurs individus, eux aussi sélectionné aléatoirement. L'objectif est de ne pas tomber dans l'eugénisme en ne sélectionnant que les individus à fitness élevée ce qui nous ferait tomber dans un minimum global.

4. Processus de Sélection

Le processus de sélection est inspiré du processus de sélection naturelle fait en sorte que les chromosomes les mieux adaptés se reproduisent plus souvent et contribuent davantage aux populations futures. Ici nous effectuons une sélection avant notre croisement. On a déjà préalablement classé notre population selon sa fitness ici « valeur_population ». Nous prenons alors une partie des individus les mieux classés puis une minorité des individus les moins bien classés, afin de ne pas tomber dans l'eugénisme et effectuions croisement puis mutation sur cette population avant de la faire augmenter.

5. Population

La population initiale est l'ensembles des individus crées ici aléatoirement. Après avoir agencé une sélection auprès de nos individus nous effectuons premièrement un remplacement générationnel avec une même taille de de population mieux « adaptés ». Ici nous partons d'une population de 35 individus, c'est la taille de population qui me permettait d'obtenir en un temps raisonnable et selon un moindre de nombre de génération un résultat, il faut en moyenne une dizaine de génération pour obtenir une fitness en dessous de 0.05

ANALYSE DES RESULTATS

Nous pouvons effectuer en premier lieux différents tests de lancée du programme. On obtient le tableau suivant :

Combinaison [a ;b ;c]	Nombre de générations	Temps d'exécution	Fitness
0.312;15;3	8	0:00:00.919319	0.048
0.379;15;3	12	0:00:01.323243	0.045
0.346;15;15	6	0:00:00.748996	0.049
0.309;15;2	7	0:00:00.753493	0.043
0.339;15;8	6	0:00:00.631515	0.048
0.371;15;5	13	0:00:01.644410	0.05
0.342;15;16	7	0:00:00.777370	0.049

On obtient une moyenne de temps d'exécution sur 7 générations de : 0.82833514285 secondes.

On obtient des valeurs de « c » du triplet qui diffèrent beaucoup, on s'en préoccupe peu car elle influe de très peu la fitness étant le nombre de fois qu'on somme notre fonction pour obtenir la fonction de Weierstrass. J'ai d'abord effectué le calcul du temps d'exécution sur mon ordinateur et ai obtenue des valeurs supérieurs à 1.5s donc bien moins pertinentes. Collab est donc ici plus rentable.

J'ai par ailleurs du repenser tout mon tri de la population. J'avais implémenté un tri basé sur le tri-fusion et me retrouvais avec des durées qui ne me laissait même pas la possibilité d'accéder au résultat de mon algorithme. C'est en reprenant le dernier td et le CM que j'ai décidé d'utiliser la méthode sort() et lambda.

Lors de la conception de l'algorithme j'avais d'abord commencé a faire des fonctions les unes à la suite des autres et par manque de clarté et en ayant des problèmes lors de l'exécution finale(je me perdais facilement dans ce que je devais appeler) j'ai du faire des classes. Autre difficulté qui m'a permise de comprendre comment créer par exemple des individus à partir de liste.

Ensuite j'ai du intégrer ma fonction fitness, j'ai d'abord voulu créer une fonction qui me ferait l'écart de celle calculée avec les points individus et celle obtenue grâce au tableau de valeurs donnés. J'ai du y intégrer une valeur absolu de cette soustraction ayant des valeurs négatives, car on ne sait pas quelle valeur est supérieure à laquelle.

J'a aussi essayé pour le croisement de séparer ma population en 4 et d'y intégrer un croisement entre mon premier et mon dernier quart après classement de valeurs ainsi que mon premier et mon deuxième. Mais n'y intégrant jamais ma population la « mieux classé » dans ma génération suivante je n'obtenais pas de valeurs stables.

C'est en effet lors de l'exécution de ma boucle finale que je me suis rendue compte que je devais faire des efforts sur les tests de mes fonctions qui on souvent des problèmes d'incrémentations et que je manquais parfois de rigueur algorithmique, bloquant parfois 20 minutes parce que je comprenais pas que j'appelais une liste et non un Individu. Par exemple lors de la création de listes comprenant les individus croisés et mutés, il m'a fallu comprendre l'utilisation d'une classe pour les transformer en population.