DS Python 3BIM 2016-2017

Sergio Peignier

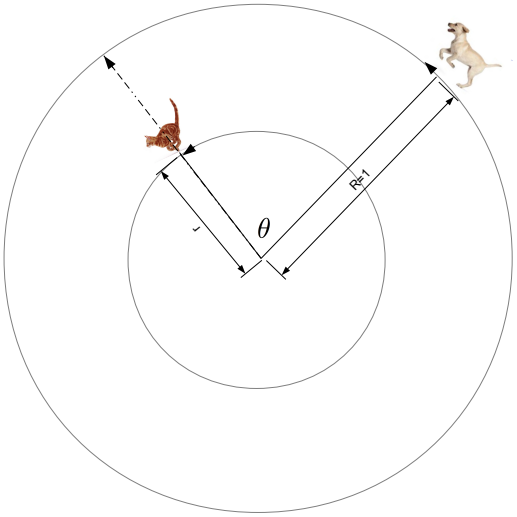
1 Le chat, le chien et l'algorithme évolutif

Un chat se trouve au centre d'un cercle de rayon 1 et un gros chien méchant tourne en suivant la circonférence du cercle (il ne peut pas quitter la circonférence). La vitesse maximale du chat est quatre fois plus petite que celle du chien ($v_{chien} = 4v_{chat}$ et $v_{chat} = 1$ par exemple). Le chat veut quitter le cercle mais le chien a très faim et fera de son mieux pour manger le chat.

Le chat, mauvais en math (il sait quand même que le périmètre d'un cercle de rayon R vaut $2\pi R$), mais bon programmeur, décide d'utiliser un algorithme génétique : Il va faire évoluer une population de chats *in silico* de telle sorte à trouver une stratégie pour sortir. Le chat reçoit un indice supplémentaire, on lui dit qu'il peut arriver à quitter le cercle si :

— Il court en décrivant un cercle de rayon r autour du centre.

— Lorsque le chien est à un angle θ de lui, il doit quitter sa trajectoire et courir directement vers la sortie.



Avec cette information le chat se dit que ses chats *in silico* peuvent avoir 2 gènes : $r \in [0, 1/4[$ et $\theta \in [0, \pi]$. Pour ne pas se compliquer la vie il se dit que ses chats se reproduisent de manière asexuée et que la population de chats sera constante et égale à $N = 50$. Les nouveaux chats ont pour gènes les gènes de leur parent auxquels on rajoute un petit bruit gaussien de moyenne 0 et de variance $\sigma_r^2 = 0.01$ et $\sigma_\theta^2 = 0.02 \times \pi$ respectivement (mutations). De plus seulement le meilleur chat (celui avec la plus grande fitness) de la génération précédente se reproduit N fois (sélection). Afin de calculer la fitness des chats on définit t_{chat} le temps mis par le chat pour atteindre le cercle à partir du moment où il commence à courir tout droit et on définit t_{chien} le temps mis par le chien pour atteindre le même point, la fitness du chat est alors définie $f_{chat} = t_{chien} - t_{chat}$ (plus grande cette différence, plus de marge de temps a le chat pour quitter le cercle). Notez que les deux temps ne dépendent que de r et de θ. L'idée est d'alterner une étape de séléction, puis une étape de mutation pendant $T = 100$ générations. Une fois que son programme a fini de s'executer, le chat prend la meilleure stratégie de la dernière génération et l'applique.

Remarque Les gènes ne doivent pas sortir de leur intervalles de définition (numpy.clip ?).

Question 1 : Que se passe-t-il si $r = 1/4$ et si $r > 1/4$? Pourquoi ne sert-il à rien de considérer ces cas ?

Question 2 : Que se passe-t-il si $2\pi > \theta > \pi$? Pourquoi ne sert-il à rien de considérer ce cas ?

Question 3 : Choisir une structure de code. Justifier.

Question 4 : Le premier individu aura comme génome $r = 0$ et $\theta = 0$.

Question 5 : Faire un graphique qui représente la fitness du meilleur chat en fonction des générations.

Question 6 (bonus) : Pas besoin de faire un algorithme de ce type ! Donner la réponse analytique à ce problème. Afficher cette solution sur le graphe précédent.

2 Bach et l'entropie

2.1 Bach

Question 1 : Charger le jeu de données `https://archive.ics.uci.edu/ml/datasets/Bach+Choral+Harmony` à quoi correspond ce jeu ? Éviter des problèmes en utilisant sep=`'\s*,\s*'` (pour ne pas prendre en compte les espaces).

Question 2 : Faire un dictionnaire qui a comme clé l'accord utilisé et comme valeur, le nombre de fois que Bach utilise cet accord (indice : Counter du module Collections).

Question 3 : Calculer les probabilités d'utilisation de chaque accord grâce au dictionnaire de fréquences calculé précédement.

Question 4 : Quels sont les 10 achords les plus utilisés ? Faire graphique qui représente le nombre de fois que chaque accord est utilisé (en rangeant les accords du plus utilisé au moins utilisé).

Question 5 : Faire un graphique qui représente le nombre de fois que chaque accord est utilisé comme premier accord de la chanson (en rangeant les accords du plus utilisé au moins utilisé).

Question 6 : Faire une fonction qui prend en entrée un accord et qui renvoie les accords que bach utilise juste après et à quelle fréquence. Quels sont les 3 accords les plus utilisés après "D_M" ?

2.2 Entropie

L'entropie est une mesure provenant de la théorie de l'information qui permet de quantifier l'incertitude d'une variable aléatoire. Elle fut définie par Shannon en 1948. Par définition une variable aléatoire qui prend toujours la même valeur a une entropie nulle (son degré d'incertitude est égal à 0). Intuitivement une pièce qui a la même probabilité de fournir pile ou face correspond à un choix binaire, on a donc besoin d'un bit (0 ou 1) pour décrire un choix parmis deux choix équiprobables, le niveau d'incertitude sera de 1 bit. Soit une variable aléatoire X prenant n valeurs possibles (e.g., $n = 2$ dans le cas d'une pièce de monnaie : pille ou face), soit P_i la probabilité d'apparition de la valeur X_i, alors l'entropie de cette variable aléatoire est :

$$H(X) = \sum_{i=1}^{n} P_i \times log_2(\frac{1}{P_i}) = -\sum_{i=1}^{n} P_i \times log_2(P_i)$$

Pour la suite des exercices nous allons prendre l'entropie normalisée par le nombre de valeurs possibles :

$$H(X) = -\frac{1}{n}\sum_{i=1}^{n} P_i \times log_2(P_i)$$

Question 1 : Faire une fonction qui calcule l'entropie d'une séquence envoyée en paramètre. La séquence correspond aux valeurs possibles que prend une variable aléatoire et on estime les probabilités en calculant les fréquences de chaque état dans la séquence (e.g. une liste de 0 ou de 1 pour pile ou face).

Question 2 : Faire une fonction qui calcule l'entropie moyenne d'une liste de séquences envoyée en paramètre (pondérée par le nombre d'instances dans chaque sequence)

Question 3 : Générer une liste contenant la longueur de chaque chanson.

Question 4 : Générer une liste contenant l'entropie des accords de chaque chanson.

Question 5 : Générer une liste contenant l'entropie des basses de chaque chanson.

Question 6 : Générer une liste contenant l'entropie des intensités utilisées dans chaque chanson (attribut 'meter').

Question 7 : Tracer les graphiques suivants :
- Entropie des accords et des basses en fonction de la longueur
- Entropie des basses en fonction des entropies des accords
- Entropie des intensités en fonction de la longueur.

Pouvez vous en déduire quelque chose ?

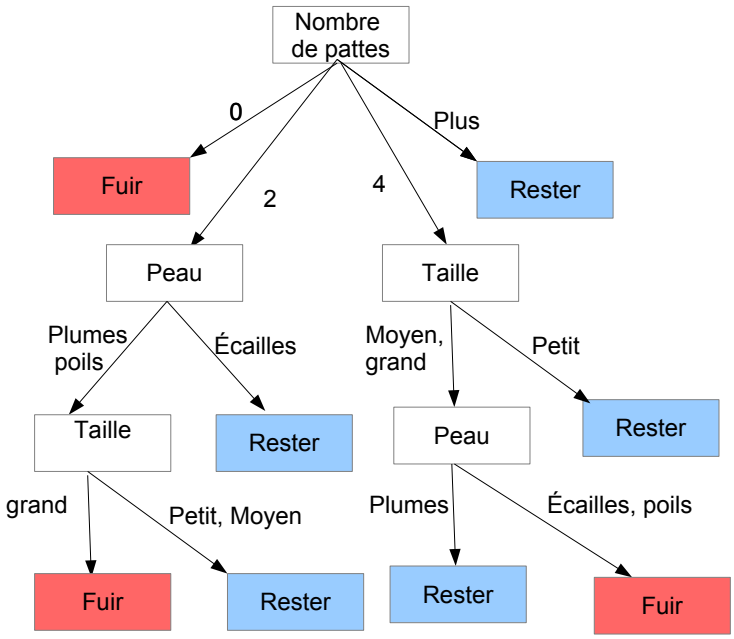
3 Arbre de décision

3.1 Définition

L'arbres de décision correspond à un des algorithmes de machine learning les plus simples. Cet algorithme prend en entrée un vecteur de valeurs d'attributs et produit en sortie une décision. Dans cet exercice nous allons uniquement considerer le cas où les attributs sont discrets et les sorties sont bianires (True ou False). Afin de mieux comprendre l'algorithme lisez l'exemple suivant :

Un rat (et oui, encore un rat !) doit apprendre à distinguer si un animal qui approche est dangereux ou pas. Pour ce faire il a accés à un certain nombre d'attributs charactérisant l'animal en question :
- Sur combien de pattes marche l'animal en question (0, 2, 4 ou plus)
- Peau de l'animal (poils, plumes ou écailles)
- Taille relative de l'animal (petit, moyen, grand)

Dans notre cas la décision sera tout simplement de Fuir ou de Rester. On donne à totre indicatif un arbre de décision possible :



À chaque noeud de l'arbre un test est effectué par rapport à un des attributs. Les branches partant de chaque noeud correspondent alors aux issus possibles de ce test. Les feuilles de l'arbre (derniers noeuds) donnent la sortie à fournire, étants donnés les résultats des tests précédents.

Question 1 : Quelle décision sera prise par le rat s'il rencontre un aigle ? Expliquer.

Question 2 : Proposer une manière simple d'encoder l'arbre et écrire l'encodage correspondant à l'arbre fourni en exemple (uniquement pour la partie de l'arbre qui caractérise les animaux avec 0 ou 2 pates).

3.2 Inférence d'arbre de décision à partir d'exemples

Imaginons maintenant qu'on a enregistré plusieurs exemples (attributs et décision à prendre). Nous allons essayer d'inférer un arbre de décision qui soit le plus petit possible et qui décrive le mieux possible notre série d'exemple. Pour ce faire nous allons utiliser un algorithme glouton, à chaque étape l'algorithme évalue quel est le meilleur test à faire (celui qui conduit à la meilleure séparation possible des exemples en fonction de la décision à prendre) et rajoute le noeud correspondant au test choisi. L'algorithme refait l'étape en question avec les données filtrées dans chacune des branches issues du dernier test. L'algorithme s'arrête lorsque tous les cas ont été traités ou lorsque sa taille maximale esta atteinte (paramètre de l'algorithme).

Imaginons que notre rat receuille, auprès de ses amis rats, un certain nombre d'exemples d'animaux rencontrés et la meilleure décision à prendre dans chaque cas. Maintenant notre rat procède méthodiquement, il se demande quelle est l'attribut le plus discriminant, il essaye la taille, le nombre de pattes, ... tous les attributs un par un. Ensuite il choisi de prendre en compte le teste qui découpe les exemples en groupes d'exemples plus pures (le cas idéal étant : que des animaux dangeureux d'une part et inoffensif de l'autre). Ensuite il traite les sous problèmes un par un en considérant les attributs restants. Il procéde ainsi jusqu'à ce que tous les exemples ont été parfaitement classifié ou jusqu'à ce que la taille maximale de l'arbre est atteinte (paramètre de l'algorithme). Si la taille maximale est atteinte mais les groupes formés ne sont pas totalement purs, la décision majoritaire dans le groupe est prise. La pureté des découpages est évaluée en calculant l'entropie des décisions à prendre de chaque groupe et en prenant : 1-entropie.

Question 1 : Proposer une structure de code. Justifier.

Question 2 : Programmer la fonction d'apprentissage (récursive ?). Si vous n'avez pas codé al fonction entropie, ce n'est pas grave imaginez qu'il existe une fonction entropie qui prend en entrée une liste de groupes et renvoie leur entropie.

Question 3 : Entrainer l'arbre avec les exemples fournis en p.j.

Question 4 : Programmer une fonction qui prend en entrée une liste d'attributs et propose une décision en sortie. Si votre arbre n'a pas été crée (si votre fonction d'apprentissage ne marche pas) prenez comme arbre celui de l'exemple.

Question 5 : Lancer la fonction de décision sur l'exemple suivant : pattes = 2, taille = grand, peau = écailles.

4 Bonus

Question 1 : Faire un programme qui vérifie qu'un nombre envoyé en paramètres est bien un nombre premier.

Question 2 : Soit p un nombre premier tel que $p > 3$, démontrer que p est toujours divisible par 24.