Algorithme ID3 : classification d'animaux

On souhaite écrire un programme capable de déterminer le type d'un animal en posant une suite de questions binaires (réponse oui ou non). Voici un exemple d'exécution :

```
milk ? (répondre y/n) : n
feathers ? (répondre y/n) : n
backbone ? (répondre y/n) : y
fins ? (répondre y/n) : n
aquatic ? (répondre y/n) : n
Ton animal est un reptile
```

Les catégories possibles sont :

numéro	catégorie
0	mammifère
1	oiseau
2	reptile
3	poisson
4	amphibien
5	insecte
6	autre invertébré

Pour ce faire, on a collecté une base de données (`zoo.data`) de 100 animaux dans laquelle chaque animal est décrit par une suite de 15 attributs binaires (par exemple : possède-t-il des cheveux ? est-il un prédateur ? etc). Les attributs sont décrits plus précisément dans le fichier `zoo.names`. De plus chaque animal de cette base de donnée est étiquetée par une valeur entre 0 et 6 qui correspond à sa catégorie. Nous sommes donc dans le cadre de l'**apprentissage supervisé**. Voici les premières lignes de ce fichier :

```
aardvark,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,1
antelope,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,1
bass,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,4
...
```

Dans le fichier `id3_zoo.ml` on définit le type `dataset` pour représenter un tel jeu de données :

```
type dataset = {
    exemples: bool array list; (* valeur binaire des attributs *)
    etiquettes: int list; (* étiquette de chaque exemple *)
    attr_str: string array; (* nom des attributs *)
    etq_str: string array; (* nom des étiquettes *)
```

```
};;
```

Ainsi, les exemples d'apprentissage sont représentés par des tableaux de booléens qui sont regroupés dans la liste `ds_zoo.exemples`. Les étiquettes correspondantes sont enregistrées à part dans la liste `ds_zoo.etiquettes`. Les champs `attr_str` et `etq_str` sont de simples tableaux qui servent à retrouver le nom des attributs et des étiquettes à partir de leur numéro.

1 Arbres de décision

Les arbres de décision seront codés par le type :

```
type 'a abd =
    | Feuille of 'a
    | Noeud of int * 'a abd * 'a abd
;;
```

Dans un `Noeud` le paramètre entier est le numéro de l'attribut considéré par le noeud.

Question 1

Écrire une fonction
`classer : 'a abd -> bool array -> 'a`
qui prend en entrée un arbre de décision et un exemple non étiqueté et qui retourne l'étiquette prédite.

On vous fournit également une fonction `mode_interactif` qui permet de lancer un programme interactif qui prédit la classe d'un animal sur la base d'un arbre de décision.

2 Quelques fonctions utiles

Dans le cadre de l'algorithme d'apprentissage ID3, nous avons besoin de programmer quelques petites fonctions utiles.

Question 2

Écrire une fonction
`meme_valeur : 'a list -> bool`
testant si toutes les valeurs d'une liste sont identiques.

Question 3

Écrire une fonction
`max_liste : int list -> int`
renvoyant la valeur maximale d'une liste d'entiers non vide.

Question 4

Écrire une fonction
`decompte : int list -> int -> int array`
telle que `decompte l maxval` renvoie un tableau `tab` tel que `tab.(i)` est le nombre d'occurrences de `i` dans la liste `l`. Les valeurs de `l` seront comprises entre 0 et `maxval`.

Par exemple `decompte [3;1;3;2] 3` renvoie `[| 0; 1; 1; 2 |]`.

Question 5

Écrire une fonction
`majoritaire : int list -> int`
prenant en entrée une liste non vide d'entiers positifs et renvoyant la valeur majoritaire de la liste.

3 Calculs d'entropie

On suppose déjà écrite la fonction :

```
(* calcule le log en base 2 *)
log2 : float -> float
```

On rappelle que si L est une liste de valeurs, alors l'entropie de cet ensemble est :

$$H = \sum_{x \in L} -\frac{n_x}{n_{total}} \log_2\left(\frac{n_x}{n_{total}}\right)$$

Question 6

Écrire la fonction
`entropie : int list -> float`
calculant l'entropie d'un ensemble de valeurs entières.

Par exemple `entropie [1; 2; 1; 2]` vaut `1.0` car il y a équiprobabilité avec 2 résultats possibles. Autre exemple : `entropie [1; 1; 1; 1; 5; 5; 7; 12]` doit renvoyer `1.75`.

On vous fournit la fonction `separe` qui prend en entrée un ensemble d'exemples étiquetés `(exemples, etiquettes)` et un numéro d'attribut `k` et qui sépare ces données en deux : `(ex_false, et_faux)` et `(ex_true, et_true)` qui sont respectivement les exemples pour lesquels l'attribut `k` vaut `false` et `true`.

On rappelle que si S est un ensemble d'exemples et A un attribut, le **gain** de l'attribut A est défini par

$$G(S,A) = H(S) - \frac{n_{A=false}}{n_{tot}} H(S_{A=false}) - \frac{n_{A=true}}{n_{tot}} H(S_{A=true})$$

Question 7

Écrire une fcontion
`gain : bool array list -> int list -> int -> float`
telle que `gain exemples etiquettes k` retourne le gain d'information de l'attribut d'indice k.

On en déduit facilement une fonction `plus_discriminant` déterminant le numéro d'attribut le plus discriminant parmi un ensemble d'attributs donné.

4 Algorithme ID3

On souhaite implémenter l'algorithme ID3 tel qu'il a été présenté en cours.

Question 8

Compléter la fonction
`id3 : dataset -> int abd`
construisant un arbre de décision binaire à partir d'un jeu de donnés étiquetées.

Vérifier son bon fonctionnement en construisant l'arbre de décision associé au jeu de données `ds_zoo` puis en lançant le mode interactif.