RDF : TP9 : arbre de décision

binôme : Benjamin Ruytoor et Aurore Allart
date : 15 mars 2013

Introduction

Dans ce TP, nous allons aborder les arbres de décision. Ces arbres permettent de savoir si, par exemple, un objet fait parti d'une classe ou d'une autres, en fonction de ces caractéristiques.

1. Question de bon sens

◦ Question a

Je pense que N vaut 15 = 2^4-1. Car si B choisit 8 comme première valeur à tester, on tombe sur un arbre binaire. Et donc, en fonction de la valeur de x, B aura 4 propositions au maximum à faire. Par exemple, si x = 1 :
B propose 8, A répond « x est plus petit »
B propose 4, A répond « x est plus petit »
B propose 2, A répond « x est plus petit »
B propose 1, A répond enfin « gagné ! ».

◦ Question b

B comprend par les indices de A, que la réponse se trouve dans les feuilles de l'arbre binaire : c'est dire soit 1, 3, 5, 7, 9, 11, 13 ou 15. En partant de ce principe, B peut se concentrer uniquement sur ces valeurs. Donc si part, du nombre 7 ou 9, cela lui permet de trouver en 3 choix.

En connaissant N et sachant qu'il faudra 4 propositions pour gagner, on peut facilement en déduire que la réponse se troupe dans les feuilles.

2. Variante du jeu du pendu

◦ Chargement de la base de noms

Avec la commande : n = size(noms,1); on récupère le nombre de ligne contenue dans le fichier, qui dans notre cas, est le nombre de mots. N est égale à 283.

◦ But du jeu

Dans le jeu du pendu, on a comme réponse soit oui, soit non. Donc on a un arbre binaire. On a n = 283. En regardant, les puissances de 2, on trouve 2^8 = 256, qui est proche de n. Cela nous donne n = 2^8+ 27, ce n'est pas suffisant pour être sur de trouver en 7 questions le mot. Il nous faut une question supplémentaire, donc on obtient n < 2^9, où p = 9, c'est à dire 9 questions. Là, le programme sera sûr de trouver le mot.

◦ Que fait ce code ?

La matrice 'mat' contient, à l'exécution, des valeurs soit 0 soit 1, si la lettre est présente au moins une fois dans le mot, on a un 1. C'est un tableau de 35 lignes, car de 0 à 9, c'est les chiffres qui sont énumérés, et de 10 à 35, ce sont les lettres. Ce qui donne la lettre 'a' est à la ligne 10, et la lettre 'z' est à la ligne 35.

◦ Quelle lettre est la plus représentée ?

La lettre la plus représenter dans les noms des animaux est la lettre 'e', elle est présente 184 fois. Voici le code permettant de savoir quelle lettre est la plus représentée :

```
//Q4
mat=mat(10:$,:);
matsum=zeros(size(mat,1),1);
for i=1:size(mat,1)
   matsum(i,1)=sum(mat(i,:));
end
[mattri,K]=gsort(matsum);//tri les lettres
mattri=[K,mattri];//fusion :indice de la lettre, occurrence de la lettre
```

e	184
a	150
o	140
r	130
i	124
n	118
u	107
c	86
l	82
t	82
p	61
m	53
s	53
h	49
g	41
b	36
d	33
v	19
f	15
y	13
q	12
k	5
z	5
x	4
j	2
w	1

- **Entropie**

Le calcul en Scilab de l'entropie est le suivant :

```
iX = zeros(26,1);
for j=1:size(mattri,1)
    inv = ((n-mattri(j,2))/n)^((n-mattri(j,2))/n);//calcul si la lettre n'est pas dans le mot
    p = (mattri(j,2)/n)^(mattri(j,2)/n);//calcul si la lettre est dans le mot
    iX(j,1) = -(log2(p) + log2(inv));
end
```

Cela nous donne :

e	0,9339
a	0,9974
o	0.9999
r	0.9952
i	0.9889
n	0.98
u	0.9567
c	0.886
l	0.8684
t	0.8684
p	0.752
m	0.6957
s	0.6957
h	0.6648
g	0.5969
b	0.5497
d	0.5195
v	0.3552
f	0.299
y	0.2689
q	0.2532
k	0.1281
z	0.1281
x	0.1071
j	0.0607
w	0.0339

- **Première question**

Le programme doit choisir la lettre 'o', pour maximiser l'information. Car pour la lettre 'o', l'entropie est de 0.9999, qui est le maximum du tableau. En voici le code :

```
[m,k] = max(iX,'r');
firstLetter = K(k);
```

- **Partage**

Voici la fonction partage :

```
function [A, B, i]=partage(I)
   rel=mat(:,I);
   nrel=size(rel,"c");
   hrel=zeros(1,size(rel,1));
   for k=1:size(rel,1)
      hrel(k)=sum(rel(k,:));
   end;
   //calcul de i par l'entropie
   entropierel = -(log((hrel./nrel).^(hrel./nrel)) + log((1-hrel./nrel).^(1-hrel./nrel)));
   [m,i]=max(entropierel);

   //calcul de A
   A=[];
   for k=1:n
      A = [A,I(k) & (mat(i,k) == 1)];
   end

   //calcul de B
   B=[];
   for k=1:n
      B = [B,I(k) & (mat(i,k) == 0)];
   end
endfunction
```

- **Que fait ce code ?**

```
I=([1:n]>0);
[A,B,i]=partage(I);
secondeLettre = code2str(i); //retourne "o"
motContenantLaSecondeLettre = sum(A); //retourne 140
motNeContenantPasLaSecondeLettre = sum(B); //retourne 143
//3e lettre en prenant comme deuxieme lettre le "o"
[C,D,j]=partage(A);
TLettre1 = code2str(j); //retourne "e"
mCLTL1 = sum(C); //retourne 71
mNCPLTL1 = sum(D); //69
//3e lettre en ne prenant pas le "f" en deuxieme lettre
[E,F,k]=partage(B);
TLettre2 = code2str(k);//retourne "r"
mCLTL2 = sum(E);//retourne 70
mNCPLTL2 = sum(F);//retourne 73
```

- **A vous de jouer**

Choisissez le nom d'un animal...

```
Ce mot contient-il la lettre o ? [o/n]
-->o
Ce mot contient-il la lettre e ? [o/n]
-->o
```

```
Ce mot contient-il la lettre r ? [o/n]
-->n
Ce mot contient-il la lettre n ? [o/n]
-->n
Ce mot contient-il la lettre t ? [o/n]
-->o
Ce mot contient-il la lettre u ? [o/n]
-->n
Ce mot contient-il la lettre a ? [o/n]
-->o
Ce mot contient-il la lettre c ? [o/n]
-->o
L'animal est un(e) cacatoes
```

- **Pour continuer**
  - La profondeur maximale de l'arbre.

```
function maximum=arbre(I, str)
  if (sum(I)>1) then
    [A,B,i]=partage(I);
    maxiA = arbre(A,str+'| ');
    printf("%s%c (%i,%i)\n",str,code2str(i),sum(A),sum(B));
    maxiB = arbre(B,str+'| ');
    maximum = 1 + max(maxiA,maxiB);
  else
    printf("%s -> %s\n",str,noms(I));
    maximum = 0;
  end
endfunction
```

La profondeur maximale de l'arbre est de 10.

  - les mots les plus défavorables.

Voici la fonction qui permet de Les mots qui nécessitent le plus de questions possibles sont :

```
function maximum=arbre(I, str, profondeurActuelle, profondeurMax)
  if (sum(I)>1) then
    profondeurActuelle = profondeurActuelle +1;
    [A,B,i]=partage(I);
    maxiA = arbre(A,str+'| ',profondeurActuelle, profondeurMax);
    printf("%s%c (%i,%i)\n",str,code2str(i),sum(A),sum(B));
    maxiB = arbre(B,str+'| ',profondeurActuelle, profondeurMax);
    maximum = 1 + max(maxiA,maxiB);
  else
    if profondeurActuelle ==profondeurMax then
      global elements;
      elements = [elements noms(I)];
    end
    printf("%s -> %s\n",str,noms(I));
    maximum = 0;
  end
endfunction
```

Avec comme code d'éxécution :

```
global elements;
elements = [];
maximum = arbre(I,' ',0,10);
```

Les noms d'animaux défavorables sont : roussette, tortue, hareng, panthere, merlin, varan.

- le nombre moyen de questions nécessaires pour trouver la solution en suivant cet arbre.

Ce nombre est calculé de cette manière, nbQuestion étant le nombre de question qu'il faut pour chaque mot :

```
function [nbQuestion, maximum]=arbre(I, str, profondeurActuelle, profondeurMax)
   if (sum(I)>1) then
      profondeurActuelle = profondeurActuelle +1;
      [A,B,i]=partage(I);
      [nbQ1,maxiA] = arbre(A,str+'| ',profondeurActuelle, profondeurMax);
      printf("%s%c (%i,%i)\n",str,code2str(i),sum(A),sum(B));
      [nbQ2,maxiB] = arbre(B,str+'| ',profondeurActuelle, profondeurMax);
      maximum = 1 + max(maxiA,maxiB);
      nbQuestion = nbQ1 + nbQ2;
   else
      if profondeurActuelle ==profondeurMax then
         global elements;
         elements = [elements noms(I)];
      end
      nbQuestion = profondeurActuelle;
      printf("%s -> %s\n",str,noms(I));
      maximum = 0;
   end
endfunction
```

Après l'appel de cette fonction, il suffit de diviser la variable de retour par le nombre de mot, cela nous donne :

```
[nbQuestion,maximum] = arbre(I,' ',0,10);
nbQuestion = nbQuestion/n;
```

Et la moyenne est 8,2756 questions par mots.

- A quoi faut-il comparer ce nombre et à quoi est due la différence ?

Ce nombre est à comparer avec le nombre maximum et minimum de question. Sachant que le nombre de question maximum est 10 questions et le nombre minimum est de 8. On constate qu'on ne peut pas trouver tous les mots avec que 8 questions. Avec 8 questions on trouve que 256 mots, alors qu'on en a 283.

Conclusion

Grâce à l'entropie, on prend la lettre qui a le plus de chance de diviser le problème en 2 parties : soit la lettre est dans le mot, soit la lettre ni est pas. On constate aussi, que pour certains mots, 9 questions ne nous suffit pas pour les obtenir.