TP - RdF: Arbres de décision

François Lepan

21 mars 2013

1 Question de bon sens

1.1 Que vaut N ?

N vaut 2^{4+1} car pour quatre propositions on a toujours que deux possibilité et le +1 pour arrivé à la solution final.

1.2 Expliquer le raisonnement de B et calculer la quantité d'information que lui a donnée A par sa réponse

? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?

2 Variante du jeu du pendu

2.1 But du jeu

Si le programme est sûr de trouver la solution en au plus p questions, quelle est la relation entre n et p ? En déduire une valeur numérique minimale pour p.

? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?

2.2 Que fait ce code ?

```
mat = zeros(35,n);

for i = 1 : n
        c = str2code(noms(i));
        mat(c,i) = 1;
end
```

Ce code stock dans *mat*, en colonnes, les lettres du mot qui sont présentes (35 car "a" == 10). Pour cela on met 1 si la lettre est présente (si elle est la deux fois on ne le saura pas).

2.3 Quelle lettre est la plus représentée ?

```
// retourne une matrice de 283 colonnes et 35 lignes contenant
// pour chaque colonne un 1 ou un 0 si il contient la lettre correspondant à
// l'indice (a à pour indice 10)
function mat = presentLettersIn(noms)

    n = size(noms,1);

    mat = zeros(35,n);

    for i = 1 : n
        c = str2code(noms(i));
        mat(c,i) = 1;
    end

endfunction


function h = getNumberOfLetters(noms)

    // on créer une colonne de 283 afin de faire une multiplication
    // entre cette colonne et la matrice de 1 et 0 retournée par presentLettersIn
    // afin de récupérer une seul colonne de 35 contenant pour chaque element
    // la somme des éléments de sa ligne.
    one = ones(size(noms,1),1);

    m = presentLettersIn(noms)

    h = m * one;

endfunction



function letter = mostPresentLettersIn(noms)

    h = getNumberOfLetters(noms);

    // On récupère l'indice de la lettre la plus présente
    [value, mostPresentLetterCode] = max(h);

    // on transforme l'indice en lettre
    letter = code2str(mostPresentLetterCode);

endfunction

// Chargement de la base de noms d'animaux
```

```
exec("rdfAnimaux.txt");

// on récupère la lettre la plus présente dans tous les mots
letter = mostPresentLettersIn(noms)
```

Après exécution du code on obtient la lettre e.

2.4 Entropie

```
function resultat = entropie(noms, n)

    m = getNumberOfLetters(noms);

    resultat = - log((m./n).^(m./n)) - log((1 - m./n).^(1 - m./n))

endfunction
```

Voici la fonction qui permet de calculé l'entropie

2.5 Partage

??????????????????
??????????????????
??????????????????

2.6 Que fait ce code ?

```
I=([1:n]>0);

[A,B,i]=partage(I);code2str(i),sum(A),sum(B)
[C,D,j]=partage(A);code2str(j),sum(C),sum(D)
[E,F,k]=partage(B);code2str(k),sum(E),sum(F)
```

Il affiche la répartition des mots dans l'arbre. Pour le premier partage on à 140 mots dans A et 143 dans B. Pour le partage de A on obtient deux ensembles C contenant 71 mots et D 69 mots. Et enfin pour le partage de B on obtient deux ensembles C contenant 70 mots et D 73 mots.

Si on continu on aura pour chaque noeud le nombre de fils qu'il possède, pour chacun de ses fils le nombre de fils, etc.

2.7 L'arbre

Profondeur

Afin de connaître la profondeur maximale de l'arbre il suffit de rajouter ces lignes en bleu à la fonction *arbre* :

```
function maxim = arbre(I,str)
    if (sum(I)>1) then
```

```
        [A,B,i] = partage(I);
        max1 = arbre(A,str+'| ');
        printf("%s%c (%i,%i)",str,code2str(i),sum(A),sum(B));
        max2 = arbre(B,str+'| ');

        maxim = 1 + max(max1,max2);

    else
        printf("%s -> %s",str,noms(I));
        maxim = 0;
    end

endfunction
```

Le résultat est 10.

Mots défavorables

Afin d'afficher les mots les plus défavorables il faut rajouter ces lignes en bleu à la fonction *arbre* :

```
function maxim = arbre(I,str,currentDeepth,maxDeepth)

    if (sum(I)>1) then

        currentDeepth = currentDeepth +1;

        [A,B,i] = partage(I);

        max1 = arbre(A,str+'| ',currentDeepth,maxDeepth);

        printf("%s%c (%i,%i)",str,code2str(i),sum(A),sum(B));

        max2 = arbre(B,str+'| ',currentDeepth,maxDeepth);

        maxim = 1 + max(max1,max2);

    else

        if currentDeepth == maxDeepth then
            global deepestElements;
            deepestElements = [deepestElements noms(I)];
        end

        printf("%s -> %s",str,noms(I));
```

```
        maxim = 0;
    end

endfunction
```

Ainsi qu'une variable globale : *global deepestElements ;*

```
global deepestElements;
// pour le vider à chaque exécution
deepestElements = [];

maximum = arbre(I,' ',0,10)
```

L'exécution du code précédent nous fournis les animaux suivant : roussette, tortue, hareng, panthere, merlan et varan.

Moyenne

Afin d'avoir en plus le nombre moyen nécesssaire pour trouver la solution en suivant cet arbre il faut ajouter les lignes en bleu à la fonction *arbre* :

```
function [nbQuestion, maxim] = arbre(I,str,currentDeepth,maxDeepth)

    if (sum(I)>1) then

        currentDeepth = currentDeepth +1;

        [A,B,i] = partage(I);

        [nbQuestion1, max1] = arbre(A,str+'| ',currentDeepth,maxDeepth);

        printf("%s%c (%i,%i)",str,code2str(i),sum(A),sum(B));

       [nbQuestion2, max2] = arbre(B,str+'| ',currentDeepth,maxDeepth);

        maxim = 1 + max(max1,max2);
        nbQuestion = nbQuestion1 + nbQuestion2;

    else

        if currentDeepth == maxDeepth then
            global deepestElements;
            deepestElements = [deepestElements noms(I)];
        end

        nbQuestion = currentDeepth;
```

```
        printf("%s -> %s",str,noms(I));
        maxim = 0;
    end

endfunction
```

Ainsi que diviser le nombre totale de question par le nombre d'animaux.

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

moyenne = nbQuestion / n
```

Comparer a P !!!!!!!!!!!!!!!!!!!!!!
??????????????????????
??????????????????????