# RDF TP 10-11 : arbre de décision et reconnaissance de visages

binôme : Benjamin Ruytoor et Aurore Allart
date : 27 mars 2013

## Introduction

Nous avons vu dans le TP précédent, l'arbre de décision sur le problème du pendu. Dans celui-ci, nous allons illustrer l'utilisation d'arbres de décision dans la reconnaissance de visage dans une base de données de 10 exemples de 40 visages différents.

- ### Préparation des données

La taille de chaque visage est de 33 pixels sur la largeur et 40 pixels en hauteur. On le calcule par cet appel :

```
[tailleVisageL,tailleVisageC] = size(allFaces);
tailleVisageL = tailleVisageL / 20; //33
tailleVisageC = tailleVisageC / 20; //40
```

- ### Méthodologie

Il faut utiliser l'entropie sur la couleur du pixel pour choisir le « meilleur » pixel à chaque itération.

- ### Différences avec le jeu du pendu

Il faut parcourir chaque pixel de l'image et calculer l'entropie de ce pixel == 1 pour toutes les images sélectionnées. En voici le code :

```
//fonction sur le calcul de l'entropie pour chaque pixel
//utilisé dans la fonction de partage
//le paramètre stacked est la pile d'image qui contient les images qui nous intéressent
//contient le pixel ou non
// retourne un tableau de la taille de l'image
function entrop=entropie(stacked)
   entrop = zeros(40,33);
   res = 0;
   for jj = 1:40
      for ii = 1:33
         for kk = 1 : 400
            res = res  + stacked(jj,ii,kk);
         end
         res = res/400;
         entrop(jj,ii) = -log2(res^res) - log2((1-res)^(1-res));
      end
   end
endfunction
```

Par exemple, pour le pixel 1,1, on obtient une entropie de 0,0637.

- **Réduction d'entropie**

Pour calculer classCount, il suffit de parcourir chaque pixel de l'image et de regarder dans quelle image, triée par classe, le pixel est là. On obtient pour chaque pixel un tableau de taille 40 (pour les 40 images) qui contient le nombre d'images qui ont leur pixel est à 1. Voici le code de cette fonction :

```
//fonction sur le calcul de l'entropie
//prend en parametre la pile d'image voulu
// retourne un tableau de la taille de l'image e
function count = entropieClass(stacked)
   count = zeros(1,40);
   for jj = 1 : 40
      for ii = 1: 33
         for kk = 1:400
            if stacked(jj,ii,kk)==1 then
               count(1, ceil(kk/10)) = count(1, ceil(kk/10))+ 1;
            end
         end
      end
   end
endfunction
```

Par exemple, pour le pixel 1,1, on obtient : 1 dans la classe 15 et 2 dans la classe 29. Ce qui nous fait 3 images contenant le pixel à 1.

- **Sélection de pixel**

```
//fonction partage
//prend en paramètre le tableau des images voulues
function [A, B, i]=partage(I)
   stackedFacesI=stackedFaces(:,:,I);
   entrop = zeros(size(stackedFacesI,"r"),size(stackedFacesI,"c"),size(stackedFacesI,3));
   //calcul de l'entropie
   entrop = entropie(stackedFacesI);
   [i1,i2] = find(entrop==max(entrop));
   //retourne le premier pixel avec une entropie maximum
   i = [i1(1) i2(1)];
   //retourne le tableau contenant l'indice des images à true quand le pixel = 1
   A = [];
   for k=1:400
      A = [A,I(k) & (stackedFaces(i1,i2,k) == 1)];
   end
   //retourne le tableau contenant l'indice des images à true quand le pixel = 0
   B =  [];
   for k=1:400
      B = [B,I(k) & (stackedFaces(i1,i2,k) == 0)];
   end
endfunction
```

- **Que fait le code suivant ?**

```
I=([1:400]>0);
[A,B,k]=partage(I);
```

Ce code retourne dans A le tableau contenant l'indice des images avec leur pixel = 1, dans B, avec leur pixel = 0 et enfin dans k, les coordonnées du pixel qui nous a permis de partager le tableau des images en 2 tableau de taille équivalente. Ce qui nous donne pour le pixel k =(32,10) :
sum(A) = 200 ;
pour les 20 premiers tableaux de A = 1,2,3,5,6,7,8,9,10 a true et le reste à false.
sum(B) = 200 ;
et pour les 20 premiers tableaux de B = 4, 11 à 20 a true et le reste à false.

On constate que le partage est équitable pour la première passe. Ensuite si on désire continuer les calculs, on peut suivant l'image à tester, prendre soit le tableau de A soit le tableau de B.

## Conclusion

L'utilisation de l'entropie est pratique pour partager 2 classes, mais il devient plus compliquer quand on désire travailler sur plusieurs classes, dans notre cas, 40 visages. Cependant, il est possible de trouver un algorithme avec l'entropie et le calcul pour chaque pixel, des classes contenant ce pixel. Grâce à cette méthode, on peut dire dans quelle classe appartient l'image avec un seuil d'erreur minimale.