Dimitri Charneux
Rémi Kroll

Rendu TP: arbre de décision et reconnaissance de visages

Introduction

Dans le cadre de ce TP, nous allons chercher à reconnaître des visages grâce à l'utilisation d'un arbre de décision. Nous utilisons l'entropie pour construire l'arbre. La différence avec le TP précédent est que nous avons plusieurs visages qui appartiennent à la même personne. Ce qui implique une gestion de classe : chaque classe sera définie par une personne. Il y a dix visages différents par classe. Nous allons devoir construire un arbre sur la base d'une image d'apprentissage contenant 40 classes. Puis dans un deuxième temps, exécuter l'algorithme de l'arbre en lui passant en entré une image contenant un visage. Il faudra vérifier si le visage sera reconnu à partir des visages d'apprentissage.

Données d'apprentissage

Nous disposons d'une image source contenant 10 exemples de 40 visages. Cette image est en niveau de gris. Nous avons utilisé une fonction permettant de binariser une image source. Cette fonction est la suivante :

```
rdfReadGreyImage <- function (nom) {
      image <- readImage (nom)
       if (length (dim (image)) == 2) {
             image
       } else {
            channel (image, 'red')
      }
}

      nom <- "allFaces.png";
      image <- rdfReadGreyImage (nom)
```

Préparation des données

Dans notre image AllFaces en N&B nous souhaitons obtenir une image par visage. Pour connaître la taille d'une image, il faut diviser la taille de l'image par le nombre de visages pour chaque dimension (en largeur et en hauteur). Nous avons alors une taille de 660/20 = 33 sur 800/20 = 40.

Nous pouvons maintenant créer un tableau de trois dimensions pour empiler les visages.

```
# création de stackedFaces :
# Empilement des visages 40x33 dans une matrice stackedFaces 40x33x400
# Attention, exécution possiblement longue ~30 sec, pas de panique...)

initStackedFaces <- function(){

  stackedFaces= array(0, dim=c(40,33,400)) # numLignes, numColonnes, numFaces

  for(i in 0:19){
    for(j in 0:19){
      stackedFaces[,,(i*20 + j + 1)] = image[(1+i*33):((i+1)*33),(1+j*40):
((j+1)*40)]
    }
  }
  stackedFaces
}
```

Méthodologie et différences avec le jeu du pendu

Pour choisir le meilleur pixel à chaque itération, on calcule la moyenne de chaque pixel de chaque classe puis nous calculons l'entropie par classe en fonction de ces moyennes . Pour cela nous avons créé la fonction suivante :

```
entropie <- function(stackedFaces, etiquettes, booleans, branche){
  entropies = array(0, dim=c(40,33))
  probabilite = array(0, dim=c(40,40,33))
  if(branche == 1){
    indexB = sum(booleans)
  }else{
    indexB = length(booleans) -  sum(booleans)
  }
  for(x in 1:40){
    for(i in 1:40){
      for(j in 1:33){
        somme = cpt = 0
        for(z in 1:indexB){
          if(booleans[z] == branche && etiquettes[z] == x){
            somme = somme + stackedFaces[i, j, z]
            cpt = cpt +1
```

```
          }
          proba = somme / cpt
        }

        probabilite[x, i, j] = proba
      }
    }
  }
  for(i in 1:40){
    for(j in 1:33){
      proba = (sum(probabilite[, i, j]) / 40)
      probaInv = 1- proba
      entropies[i, j] = - (log2(proba^proba) + log2(probaInv^probaInv))
    }
  }
  entropies
}
```

Dans un premier temps, nous calculons l'indice indexB qui correspond au nombre d'élément dans la branche souhaité.

Ensuite, dans la première boucle, nous effectuons le calcul de la moyenne de chaque pixel pour chaque classe. Nous ne pouvons pas appeler la méthode « sum » car nous ne connaissons pas le nombre d'éléments présents dans chaque classe. Nous devons donc calculer cette somme nous même en testant si l'image est bien dans la classe et dans la branche souhaité.

Enfin, il nous reste plus qu' à calculer l'entropie pour chaque classe, ce qui est fait dans la seconde boucle.

Voici un exemple pour la première itération de la fonction. Nous obtenons quatre pixels qui ont chacun une entropie de 1 :

	row	col
1	10	7
2	7	25
3	33	26
4	3	31

Réduction d'entropie

Pour calculer la réduction de l'entropie, nous avons créé les tableaux « classCount » et « classRatio » contenant respectivement pour chaque classe le nombre d'éléments présents dans les sous-ensembles et le ratio de ces éléments. Nous avons ensuite créer un tableau « etiquettes » qui nous permet de savoir à quelle classe

appartient une image.

```
classCount = array(0, dim=c(2, 40))
classRatio = array(0, dim = c(2,40))


for(i in 1:400){
  if( i %% 10 == 0){
    etiquettes[i] = as.integer(i/10)
  }else{
    etiquettes[i] = as.integer(i / 10) +1
  }

  if(stackedFaces[entropieMax[1,1], entropieMax[1,2],i] < 0.5){
    boolean[i] = 0
    classCount[1,etiquettes[i]] = classCount[1,etiquettes[i]] + 1
  } else {
    boolean[i] = 1
    classCount[2,etiquettes[i]] = classCount[2,etiquettes[i]] + 1
  }
}
for(i in 1:40){
  classRatio[1,i] = classCount[1, i]/10
  classRatio[2,i] = classCount[2, i]/10

}
```

Nous avons rencontré des difficultés pour intégrer la réduction de l'entropie. Nous pensons pouvoir la réalisé de cette manière Pour chaque pixel, nous allons calculer l'entropie des deux sous-arbres. Ensuite, nous utilisons la formule suivante vu en cours :

```
H - (pA*H_A) - (pB*H_B)
```

Pour utiliser cette formule nous avons déjà intégré le tableau « classRatio » qui nous donne les probabilités pA et pB. Il reste plus qu'à trouver la variation la plus élevé pour choisir le pixel qui sépare le mieux les différentes classes.

Induction de l'arbre

Pour créer l'arbre de décision associé à la recherche de visages, Nous commençons par calculer les entropies associé à chaque pixels. Ensuite, nous séparons les images en deux ensembles séparer par le pixel trouver grâce à l'entropie.

Ensuite, nous recommençons l'opération pour chaque ensembles jusqu'à ce que les images présentes dans les sous-ensembles appartiennent à la même classe.

Nous n'avons pas réussi à terminer cette méthode dans les temps car l'exécution

prenait trop de temps. Avec un seul nœud, nous avons bien les deux sous-ensembles voulu.

Conclusion

Nous avons pu voir que la reconnaissance de visage à partir de plusieurs photos de différents visages n'est pas une chose aisée. Il faut réussir à séparer notre ensemble de visage en différents groupes représentant les différentes personnes et ne pas les séparer par image individuelle. Nous pensons avoir fait une erreur car notre exécution est beaucoup trop longue. Nous aurons apprécier être un peu plus guidé dans la manière de calculer l'entropie en prenant en compte différentes classes.