RDF : TP 6 : Classement de caractères manuscrits

binôme : Aurore Allart et Benjamin Ruytoor
date : 5 mars 2013

Introduction

Dans ce TP, nous allons utiliser et évaluer les algorithmes kNN et Naïve Bayes, pour automatiser la reconnaissance et le classement de caractères manuscrits.

L'algorithme kNN est une méthode de classification supervisée en prenant les k plus proches voisins de x. Par exemple, si on a k = 3 (donc 3 chiffres autour de x), et que ces chiffres sont 1,2,2, alors x fait partie de la classe des 2.

La classification de Naïve Bayes est un type de classification probabiliste simple basée sur le théorème de Bayes avec une forte indépendance des hypothèses.

1. Visualiser les Données

Pour visualiser les autres chiffres, par exemple le chiffre 1, on modifie dans les instructions suivantes :

```
loadmatfile('mnist_all.mat');
y = matrix(train0(1,:),28,28);
Matplot(y);
```

la ligne, « y = matrix(train0(1,:),28,28); », en changeant 'train0 ' en 'train1'. Ce qui donne :

```
loadmatfile('mnist_all.mat');
y = matrix(train1(1,:),28,28);
Matplot(y);
```

2. L'algorithme de plus proche voisin ou kNN

Question 1 :

Les résultats de la distance changent si on prend d^2 ou d. Cependant cela n'a pas d'importance pour la suite du programme, car on utilise pas la distance dans un calcul mais juste pour comparer les distances entres eux.

Question 2 : (voir page suivante)

Pour une meilleure lisibilité du code, nous l'avons sur une page. Voici les erreurs moyennes :

- pour k = 1 et n = 200 : e = 9,42 %
- pour k = 3 et n = 200 : e = 10,38 %.
- pour k = 5 et n = 200 : e = 13,6 %.

Question 3 :

Cette approche est bien supervisée, car on utiliser une classification supervisée pour retrouver la forme du chiffre. Une classification supervisée c'est une technique de classement automatique où l'on cherche à produire des règles à partir d'une base de données contenant des « exemples ». Dans notre cas, on a une base de données de 200 images de chaque chiffre, et on teste si x fait parti d'une classe de ses chiffres.

Question 2 :
Voici le code la fonction kNN ainsi que celui du calcul de l'erreur de classement :

```
//fonction kNN
function retour=kNN(x_train, y_train, x_test, k)
   retour=zeros(size(x_test,1),1);
   for i=1 :size(x_test,1)
      //cree un tableau qui stock la classe des k plus proches voisins
      x_tmpTab=ones(k,1).*99999999999;
     // cree un tableau qui stock la valeurs des k plus proches voisins
      y_tmpTab=zeros(k,1);

      for j=1 :size(x_train,1)
         //calcul de la distance
         distance=sqrt(sum((x_test(i,:)-x_train(j,:))^2));
         if(distance<x_tmpTab(k)) then//comparaison avec le voisin le plus éloigné
            for ii=1:k
               if(distance<x_tmpTab(ii)) then
                  x_tmpTab(ii)=distance;
                  y_tmpTab(ii)=y_train(j);
                  break;
               end
            end
         end//fin if

      end
      if(k==1) then
         //classe selectionnee
         selected=y_tmpTab(1);
      else
         m=tabul(y_tmpTab,'i'); //liste le nombre de fois qu'un chiffre apparaît
         selected=m(find(m(:,2)==max(m(:,2))),1);// tire les scores pour prendre le plus récurrent
         if(size(selected,1)<>1) then // si il y a la même récurrence pour différents scores
            selected=y_tmpTab(1);
         end
      end
      retour(i)=selected;
   end
endfunction

//calcul l'erreur de classement
function retour=compare(estim, y_test)
   retour=zeros(size(estim,1),1);
   for i= 1:size(estim,1)
      if(estim(i)<>y_test(i)) then
         retour(i)=1;
      end
   end
endfunction
```

3. L'algorithme Naïve Bayes

Question 1 :

L'hypothèse principale dans cette approche est l'indépendance conditionnelle des attributs.

Question 2 :

```
// calcul de la fréquence des couleurs pour les images en fonction de la classe
function retour=NBtrain(x_train, y_train)
   mfrequence=zeros(10,256);
   for i=1:size(x_train,1)
      for y=1:size(x_train,2)
         mfrequence(y_train(i),(x_train(i,y)+1))=mfrequence(y_train(i),(x_train(i,y)+1))+1;
      end
   end
   retour=mfrequence;
endfunction

//calcul de l'algorithme Naive Bayes
function retour=NBclassement(x_test, train)
   retour=zeros(size(x_test,1),1);
   for i=1:size(x_test,1)
      tmp=tabul(x_test(i,:),'i');
      valeur=zeros(1,10);
      for t=1:10
         for y=1:size(tmp,1)
            valeur(t) = valeur(t) + log(train(t,(tmp(y,1)+1)))*tmp(y,2);
         end
      end
      [k,ki]=gsort(valeur);
      retour(i)=ki(1);
   end
endfunction

//compare les valeurs de y_test et ceux qu'on a trouvé par l'algorithme
function retour=compare(estim, y_test)
   retour=zeros(size(estim,1),1);
   for i= 1:size(estim,1)
      if(estim(i)<>y_test(i)) then
         retour(i)=1;
      end
   end
endfunction

//appel des fonctions précedentes
train=NBtrain(x_train,y_train);
train=train./(size(x_train,2)*200);
estim=NBclassement(x_test,train);
t=compare(estim,y_test);
```

Nous avons trouvé une erreur de 75 %, le code contient certainement des erreurs, mais on ne voit pas où. Pourtant on a suivi ce que vous nous aviez dit.

Question 3 :

Cette approche est supervisée car on se base sur des références de pixels pour classer les pixels de l'image x. Dans la plus part des applications pratiques, l'estimation des paramètres pour les modèles bayésiens naïfs repose sur le maximum de vraisemblance.

Question 4 :

Cette approche est probabiliste car cette algorithme est basée sur le théorème de Bayes, qui est lui même basé sur la théorie des probabilités. La formule de l'algorithme est la suivante :
p((x1,...,xn)|Ci) = multiplication de j = 1 à n de p(xj | Ci).
Il nous faut donc calculer la fréquence de chaque pixel qui est dans l'image. Par conséquence, nous avons bien une probabilité de présence du pixel dans l'image.

Conclusion

On remarque que ces 2 algorithmes ne sont pas infaibles, si les images se ressemblent trop, le programme peut les confondre.