DS09

Intelligence Artificielle – Application en médecine

CCP – PSI 2019

1 Présentation

2 Analyse des données

Question 1 *Écrire une requête SQL permettant d'extraire les identifiants des patients ayant une « hernie discale ».*

Correction
```
SELECT idpatient FROM MEDICAL WHERE etat=''hernie discale'';
```

Question 2 *Écrire une requête SQL permettant d'extraire les noms et prénoms des patients atteints de « spondylolisthésis ».*

Correction
```
SELECT nom, prenom FROM PATIENT
   JOIN MEDICAL
   ON PATIENT.id = MEDICAL.idpatient
   WHERE MEDICAL.etat=''spondylolisthésis'';
```

Question 3 *Écrire une requête SQL permettant d'extraire chaque état et le nombre de patients pour chaque état.*

Correction
```
SELECT etat, COUNT(DISTINCT idpatient) FROM MEDICAL
   GROUP BY etat
```

Question 4 *Citer un intérêt d'utiliser la bibliothèque de calcul numérique Numpy quand les tableaux sont de grande taille.*

Correction Les tableaux Numpy ont une taille fixe dans l'espace mémoire. Cela doit permettre de réduire l'espace utilisé et les temps de calcul.

Question 5 *Déterminer la quantité de mémoire totale en Mo (*$1\,\text{Mo} = 1\,000\,000$ octets*) nécessaire pour stocker le tableau et le vecteur des données si* $N = 100\,000$*. On supposera que les données sont représentées en suivant la norme usuelle IEEE 754.*

Correction

La taille prise par le tableau `data` est $100\,000 \times 6 \times 32/8 = 2{,}4\,\text{Mo}$.
La taille prise par le vecteur `etat` est $100\,000 \times 1 = 0{,}1\,\text{Mo}$.

Question 6 *Écrire une fonction* `separationParGroupe(data,etat)` *qui sépare le tableau data en 3 sous-tableaux en fonction des valeurs du vecteur etat correspondantes (on rappelle que celui-ci contient les valeurs 0, 1 et 2 uniquement). La fonction doit renvoyer une liste de taille 3 de sous-tableaux. Les sous-tableaux ne seront pas nécessairement représentés par un type 'array'; 'liste de listes', 'liste d'array' conviennent également.*

Correction Le tableau data est de la forme [[0,1,2,3,4,5],[6,7,8,9,10,11],...]

```
def separationParGroupe(data,etat) :
    N = len(etat)
    normal = []
    hernie = []
    spondy = []
    for i in range(N) :
        if etat[i] == 1 :
            normal.append(data[i])
        elif etat[i] == 2 :
            hernie.append(data[i])
        elif etat[i] == 3 :
                spondy.append(data[i])
    return normal, hernie, spondy
```

Autre possibilité :

```
def separationParGroupe(data,etat) :
    return [[data[i] for i in range(len(data)) if etat[i]==j] for j in range(3)]
```

Question 7 *Définir les arguments ARGS1, ARGS2, ARGS3 ainsi que la condition TEST définis dans le script précédent permettant d'obtenir la figure 2.*

Correction

```
ax1 = plt. subplot ( n, n, i*n+j+1)
ax1.scatter(groupes [k][:,i], groupes[k][:,j], marker=mark[k])
ax1.hist(data[:,j])

# Test
if i != j:
```

Question 8 *Préciser l'utilité des diagrammes de la diagonale ainsi que celle des diagrammes hors diagonale.*

Correction Les diagrammes sur la diagonale donnent la répartition des patients en fonction de l'attribut j. Les autres diagrammes donnent un attribut en fonction d'un autre, ils permettent de voir s'il y a un lien de corrélation entre 2 attributs. De plus ils permettent de visualiser les répartitions de ces attributs en fonction de l'état de l'individu.

3 Apprentissage et prédiction

3.1 Méthode *k*-NN

Préparation des données

Question 9 *Proposer une expression de x_{normj} un élément du vecteur X_{norm} en fonction de l'élément x_j du vecteur X correspondant et de* $\min(X)$ *et* $\max(X)$.

Correction En utilisant une correction linéaire où les x_j sont « étalés » entre 0 et 1, on a $x_{\text{normj}} = a\,x_j + b$ avec $x_{\text{normj}} = 0$ pour $x_j = \min(X)$ et $x_{\text{normj}} = 1$ pour $x_j = \max(X)$.

On a donc $0 = a\min(X) + b$ et $1 = a\max(X) + b$ soit $b = -a\min(X)$ et $1 = a(\max(X) - \min(X)) \Leftrightarrow a = \dfrac{1}{\max(X)-\min(X)}$. On a donc $b = -\dfrac{\min(X)}{\max(X)-\min(X)}$.

Par suite, $x_{\text{normj}} = \dfrac{1}{\max(X)-\min(X)}x_j - \dfrac{\min(X)}{\max(X)-\min(X)}$.

Au final, $x_{\text{normj}} = \dfrac{x_j - \min(X)}{\max(X)-\min(X)}$.

Question 10 *Écrire une fonction* `separationParGroupe` *qui retourne les valeurs du minimum et du maximum d'un vecteur X passé en argument. La fonction devra être de complexité linéaire.*

Correction

```
def min_max(X):
    mini = X[0]
    maxi = X[0]
    N = len(X)
    for i in range (N):
        if X[i]<mini :
            min = X[i]
        if X[i]>maxi :
            maxi = X[i]
    return min,max
```

Question 11 *Écrire une fonction* `min_max(X)` *qui parcourt les N lignes du tableau data et calcule les distances euclidiennes entre le n-uplet z et chaque n-uplet x du tableau de données connues (x représente une ligne du tableau). La fonction doit renvoyer une liste de taille N contenant les distances entre chaque n-uplet x et le n-uplet z.*

Correction On considère que le valeurs de data ont été normalisées.

```
def distance(z,data):
    res = []
    for X in data :
        d = 0
        for i in range(len(z)):
            d=d+(z[i]-X[i])**2
        res = res.append(math.sqrt(d))
    return res
```

Détermination des k plus proches voisins

Question 12 *Expliquer ce que font globalement les parties 1 (lignes 3 à 7), 2 (lignes 10 à 12) et 3 (lignes 15 à 21) de l'algorithme. Préciser ce que représentent les variables locales T, dist, select, ind.*

Correction Les lignes 3 à 7 permettent, pour un individu z, de lui associer la distance d'un individu contenu dans data et de lui associé l'état de l'individu. Cette liste est alors triée en fonction des distances. Les k premiers individus de T sont donc les k individus les plus proches de z.

La partie 2 permet de compter, parmi les k plus proches voisins de z, le nombre d'individus par état.

La partie 3 permet de déterminer dans quel état est probablement l'individu z.

De fait, on a :

- T : liste triée contenant des couples (distance, état) ;
- dist : liste des distances entre z et data ;
- select : select[i] est le nombre de patient à l'état i parmi les k plus proches voisins ;
- ind :représente l'indice i où select[i] est le plus grand.

Validation de l'algorithme

Question 13 *Indiquer l'information apportée par la diagonale de la matrice. Exploiter les valeurs de la première ligne de cette matrice en expliquant les informations que l'on peut en tirer. Faire de même avec la première colonne. En déduire à quoi sert cette matrice.*

Correction La diagonale permet de savoir le nombre de patients détectés à l'état i qui sont réellement à l'état i. La première ligne permet de connaître le nombre de patients à l'état 0 qui ont été détectés à l'état 0 (première ligne, première colonne), à l'état 1 (première ligne, deuxième colonne) et à l'état 3 (première ligne, troisième colonne).

On peut tenir le même raisonnement pour la première colonne. Cette matrice permet de détecter les erreurs faites par l'algorithme k-NN.

Question 14 *Commenter la courbe obtenue et critiquer l'efficacité de l'algorithme.*

Correction

- On nombre de voisins compris entre 8 et 11 semble optimal.
- La détermination de k semble empirique.

- Plus k est grand, plus l'algorithme est lent.

3.2 Méthode de classification naïve bayésienne

Apprentissage

Question 15 *Écrire deux fonctions de complexité linéaire* `moyenne(x)` *et* `variance(x)` *permettant de renvoyer la moyenne et la variance d'un vecteur x de dimension quelconque.*

Correction
```
def moyenne(x):
    return sum(x)/len(x)

def variance(x):
    m =moyenne(x)
    v=0
    for i in x:
        v+=(i-m)**2
    return v/len(x)
```

Question 16 *Proposer une fonction* `distance(z,data)` *qui renvoie une liste composée de doublets* `tri(T)` *pour chaque attribut de la matrice* `T` *en les regroupant selon les valeurs du vecteur* `datatest` *:*

$$\begin{aligned}&[[[\mu_{x_0,y_0},\sigma_{x_0,y_0}],[\mu_{x_1,y_0},\sigma_{x_1,y_0}],\dots,[\mu_{x_5,y_0},\sigma_{x_5,y_0}]]\\&,[[\mu_{x_0,y_1},\sigma_{x_0,y_1}],[\mu_{x_1,y_1},\sigma_{x_1,y_1}],\dots,[\mu_{x_5,y_1},\sigma_{x_5,y_1}]]\\&,[[\mu_{x_0,y_2},\sigma_{x_0,y_2}],[\mu_{x_1,y_2},\sigma_{x_1,y_2}],\dots,[\mu_{x_5,y_2},\sigma_{x_5,y_2}]]].\end{aligned}$$

Correction
```
def synthese(data,etat) :
    res = []
    groupes = separationParGroupe(data,etat)
    for k in range(len(groupes)): #indice etat
        groupes[k] = array(groupes[k])
        l=[]
        for j in range(len(data[0])): #indice attribut
            x = groupes[k][:,j]
            l.append([moyenne(x),variance(x)])
            res.append(l)
    return res
```

Prédiction

Question 17 *Écrire une fonction gaussienne(a,moy,v) qui calcule la probabilité selon une loi gaussienne de moyenne moy et de variance v pour un élément a de* $\mathbb{R}$.

Correction
```
def gaussienne(a,moy,v) :
    return 1/sqrt(2*pi*v)*exp(-(a-moy)**2/(2*v))
```

Question 18 *Déduire de la description de l'algorithme de classification naïve bayésienne une fonction* `etattest` *prenant en argument le vecteur z qui contient le n-uplet de la donnée z, le tableau des données connues data et le vecteur d'appartenance à un groupe de chacune de ces données* `etat`*. Cette fonction renvoie la probabilité d'appartenance à chacun des $nb = 3$ groupes sous la forme d'une liste de trois valeurs. On utilisera la fonction* `separationParGroupe(data,etat)` *et la fonction* `data`.

Correction
```
def probabiliteGroupe(z,data,etat) :
    liste = []
    synt = synthese(data,etat)
```

```
    groupes = separationParGroupe(data,etat)
    for k in range(3) :
        proba = 1
        for j in range(len(data[0])) :
            a = z[j]
            moy = synt[k][j][0]
            v = synt[k][j][1]
            proba *= gaussienne(a,moy,v)
        PY = len(groupes[k])/len(data) # P(Y=yj)
        proba *= PY
        liste.append(proba)
    return liste
```

Question 19 *Écrire une fonction* `moyenne(x)`*, dont vous préciserez les arguments, qui renvoit le numéro du groupe auquel appartient un élément* z*.*

Correction

```
def prediction(z,data,etat) :
    prob = probabiliteGroupe(z,data,etat)
    i,maxi = 0,prob[0]
    for k in range(1,3) :
        if prob[k]>maxi :
            i,maxi = k,prob[k]
    return i
```

Question 20 *Proposer une explication qui justifie cette utilisation du logarithme.*

Correction Utilisation de valeurs comprises entre 0 et 1. Remplacement d'un produit par une somme...

Question 21 *Calculer le pourcentage de réussite de la méthode* k*-NN, à partir de la matrice de confusion pour* $k = 8$*, ainsi que celui de la méthode naïve bayésienne à partir de la matrice ci-dessus. Discuter de la pertinence de chacune des deux méthodes sur l'exemple traité.*

Correction ...