Corrigé du
Devoir surveillé de Reconnaissance de Formes – I3 Informatique

Benoît Decoux – 8 Janvier 2002

Durée : 2 heures - Tous documents autorisés

Remarque préliminaire : ce corrigé est détaillé, d'où sa longueur. Tous ces détails n'étaient pas demandés lors du DS.

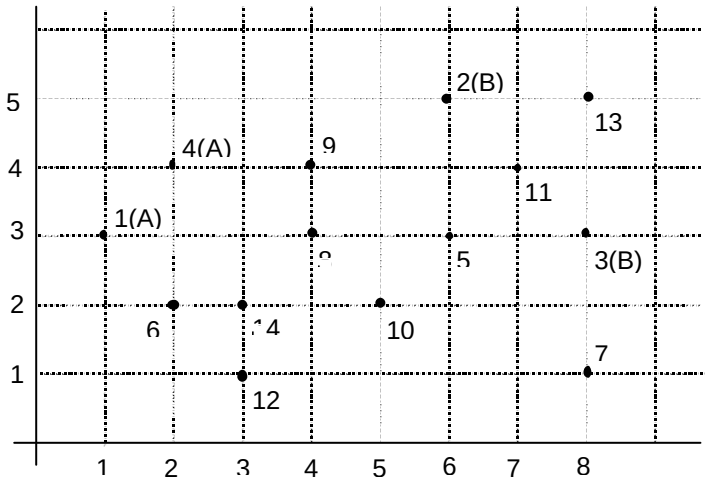
I) Exercices

Exercice 1 : Méthode des k plus proches voisins (kPPV)

Dans la figure 1, les points représentent un ensemble de vecteurs de dimension 2, appartenant à 2 classes appelées A et B. L'ordre de sélection des vecteurs est indiqué par les indices situés à côté de chacun. Les points 1 à 4 sont déjà classés ; on applique donc l'algorithme en commençant avec le point 5.

1) Appliquer la méthode des kPPV avec k=3. Ecrire la classe résultante à côté de chaque point. Préciser la démarche en prenant quelques points comme exemples. (2 points)

Pour chaque point à classer (à partir du 5e), on cherche quels sont les 3 autres points les plus proches de lui déjà classés, et on regarde quelle est la classe la plus représentée parmi ces 3 points.



Par exemple pour le point 6, les 3 points les plus proches déjà classés sont 1 (classe A), 4 (A) et 5 (B). On lui affecte donc la classe A.

La classification résultant de l'application de l'algorithme des kPPV avec k=3 est donc :

5B – 6A – 7B – 8A – 9A – 10A – 11B – 12A – 13B – 14A

2) Montrer par un exemple (tiré de la figure 1 ou inventé) que le résultat de la classification dépend de l'ordre de présentation des exemples. (1 point)

Dans la figure 1, si le point tiré en 10e position avait été tiré en 8e position, il aurait été affecté à la classe B, au lieu de la A précédemment, car les 3 points les plus proches déjà

classés auraient été les 5 (classe B), 6 (A) et 7 (B). De même pour le nouveau point tiré en 8e position.

Autre exemple : si le point tiré à la 9e position avait été tiré en 8e, il aurait été associé à la classe B (au lieu de la A), de même que le nouveau point tiré à la 9e position.

Exemple un peu différent : si le 6e point avait été tiré en 10e et inversement, le 8e point aurait été affecté à la classe B au lieu de la A (et le nouveau 10e point serait affecté à B au lieu de A).

3) Donner un algorithme en pseudo-code (c'est à dire pouvant être programmé directement) implémentant les kPPV avec k=1. Les exemples initiaux seront choisis aléatoirement. Cet algorithme pourra faire appel à des routines extérieures (2 points)

On peut s'inspirer du programme que l'on a utilisé en TP.

Fonctions utilisées :

alea0a1() : retourne un réel aléatoire compris entre 0 et 1
classe_ppv(vec1, vec2, nb_vec_cl) : retourne la classe du plus proche voisin du vecteur pointé par 'vec1', 'vec2' étant un pointeur sur la liste des 'nb_vec_cl' vecteurs déjà classés
chargement_vecteurs_a_classer(vec) : chargement dans la liste de vecteurs pointée par 'vec' des vecteurs à classer à partir d'un fichier
copier_vec_liste(vec1, vec2) : copie du vecteur pointé par 'vec1' dans la liste de vecteurs pointée par 'vec2' (par exemple en fin de liste)
retirer_vec_liste(vec, num) : retire le nume vecteur de la liste de vecteurs pointés par 'vec'

Structure de données 'vecteur' :

```
vec {   n_cl            /*numéro de la classe du vecteur*/
        dim             /*dimension du vecteur (=nombre de composantes)*/
        comp            /*pointeur sur les composantes*/
}
```

Variables :

nb_vec : nombre de vecteurs dans la base de données
nb_cl : nombre de classes
nb_deja_cl : nombre de vecteurs déjà classés utilisés pour initialiser l'algorithme

Algorithme :

```
vec1=creation_vecteurs_donnee(nb_vec, taille)   /*création de la liste initiale (vide) dans laquelle les vecteurs sont rangés*/
vec2=creation_vecteurs_donnee(nb_vec, taille)   /*création de la liste contenant les vecteurs déjà classés*/
chargement_vecteurs_a_classer(vec1)
pour i variant de 1 à nb_cl             /*choix de vecteurs déjà classés pour initialiser l'algorithme*/
{
    pour j variant de 1 à nb_deja_cl    /*on prend 'nb_deja_cl' vecteurs déjà classés pour chacune des classes*/
    {
        faire
            num=nb_vec*alea0a1()        /*'num' : aléatoire compris entre 0 et 'nb_vec'-1*/
        tant que (vec1+num)->n_cl != j
        copie_vec_liste(vec1+num, vec2+i*nb_deja_cl+j)  /*on met ce vecteur classés dans la liste pointée par 'vec2'*/
        retirer_vec_liste(vec1, num)            /*on retire ce vecteur de la liste des vecteurs non-classés*/
        nb_vec-- ;                              /*on décrémente d'une unité le compteur de vecteurs pas encore classés*/
    }
}
pour i variant de 1 à nb_vec            /*recherche du plus proche voisins pour tous les vecteurs restant à classer*/
{
        num=nb_vec*alea0a1()
        n=classe_ppv(vec1+num, vec2, i+nb_deja_cl*nb_cl)
        si n=vec1->n_cl
            afficher_texte(« classification correcte »)     /*ou incrémenter un compteur des classifications correctes*/
        sinon
            afficher_texte(« classification non correcte »)  /*idem pour compteur des classifications incorrectes*/
}
```

Exercice 2 : Nuées dynamiques et apprentissage compétitif non supervisé

1) Montrer que dans le cas des nuées dynamiques comme dans le cas de l'apprentissage compétitif non supervisé, le vecteur représentant le plus proche d'un vecteur à classer se déplace dans la direction de ce dernier. (2 points)

Cas des nuées dynamiques

Dans la version de base de l'algorithme des nuées dynamiques, chaque vecteur représentant d'une classe est constitué par la moyenne des vecteurs déjà affectés à cette classe.

Or une moyenne peut s'exprimer de manière récursive : la moyenne de n+1 points peut s'exprimer en fonction de la moyenne de n points. La moyenne de n+1 vecteurs x_i est définie par :

$$m_{n+1} = \frac{\sum_{i=1}^{n+1} x_i}{n+1}$$

$$\Leftrightarrow \quad (n+1).m_{n+1} = \sum_{i=1}^{n+1} x_i$$

$$\Leftrightarrow \quad (n+1).m_{n+1} = \sum_{i=1}^{n} x_i + x_{n+1}$$

$$\Leftrightarrow \quad (n+1).m_{n+1} = n.\frac{\sum_{i=1}^{n} x_i}{n} + x_{n+1}$$

$$\Leftrightarrow \quad (n+1).m_{n+1} = n.m_n + x_{n+1}$$

où m_n est la moyenne de n vecteurs, soit :

$$\Leftrightarrow \quad m_{n+1} = \frac{n.m_n + x_{n+1}}{n+1}$$

$$\Leftrightarrow$$

La moyenne de n points est équivalent au barycentre de ces points, s'ils sont tous pondérés par le même coefficient.

Il reste à démontrer que le vecteur joignant le point m_n au point m_{n+1} est colinéaire au vecteur joignant le point m_n au point x_{n+1}. En reprenant l'expression ci-dessus :

$$m_{n+1} - m_n = \frac{n.m_n + x_{n+1}}{n+1} - m_n$$

on obtient :

$$m_{n+1} - m_n = \frac{1}{n+1}\left(x_{n+1} - m_n\right)$$

En notation vectorielle, si W_1 est le point représentant m_n, W_2 le point représentant m_{n+1} et X le point représentant x_{n+1}, le vecteur reliant W_1 à W_2 est défini par :

$$\vec{W_1W_2}$$

que l'on peut écrire également

$$\vec{W_1O} + \vec{OW_2}$$

ou encore

$$\vec{OW_2} - \vec{OW_1}$$

où $\overrightarrow{OW_2}$ représente le vecteur reliant l'origine du repère O et le point W_2, et $\overrightarrow{OW_1}$ l'équivalent pour le point W_1.

En raisonnant de la même façon pour le vecteur reliant X à W_1, on peut l'écrire sous la forme

$$\overrightarrow{OX} - \overrightarrow{OW_1}$$

Il n'y a plus qu'à voir que ces vecteurs (W_1W_2' et XW_1') se retrouvent dans les 2 côtés de l'expression (1), et donc que W_1W_2' et $\overrightarrow{XW_1}$ sont bien colinéaires puisque $W_1W_2' = \alpha \overrightarrow{XW_1}$.

Cas de l'apprentissage compétitif non-supervisé

Dans le cas de l'apprentissage compétitif non-supervisé, lors de la présentation d'un nouveau vecteur d'entrée lors de l'apprentissage, le vecteur poids le plus proche de ce vecteur d'entrée est modifié de la quantité :

$$\Delta w_{ij} = \alpha.(x_i - w_{ij}).y_j$$

c'est à dire

$$w_{ij}(t+1) = w_{ij}(t) + \alpha.(x_i(t) - w_{ij}(t)).y_j$$

(i est l'indice de la ième composante du vecteur d'entrée x ; j est l'indice du neurone auquel appartient ce vecteur poids).

Comme on est en apprentissage compétitif, la sortie de neurone dont le vecteur poids est le plus proche du vecteur d'entrée est mise à 1, les autres à 0. On obtient donc :

$$w_{ij}(t+1) = w_{ij}(t) + \alpha.(x_i(t) - w_{ij}(t))$$

pour le neurone vainqueur et pas de changement des poids pour les autres.

Ce que l'on cherche à démontrer c'est que le nouveau vecteur ($w_{ij}(t+1)$) est obtenu par un déplacement de l'ancien ($w_{ij}(t)$) vers la nouvelle entrée ($x_i(t)$).

En d'autres termes, on veut montrer que le vecteur joignant le point $w_{ij}(t)$ à $w_{ij}(t+1)$ est colinéaire au vecteur joignant $w_{ij}(t)$ à $x_i(t)$. Ce qui est bien le cas puisque :

$$w_{ij}(t+1) - w_{ij}(t) = \alpha.(x_i(t) - w_{ij}(t)) \qquad (1)$$

En notation vectorielle, si W_1 est le point représentant $w_{ij}(t)$, W_2 le point représentant $w_{ij}(t+1)$ et X le point représentant $x_i(t)$, le vecteur reliant W_1 à W_2 est défini par :

$$W_1W_2'$$

On peut raisonner comme dans le cas des nuées dynamiques pour montrer que l'on a bien :

$$W_1W_2' = \alpha \overrightarrow{XW_1}.$$

2) Dans la figure 2, les cercles pleins représentent un ensemble de vecteurs de dimension 2, que l'on cherche à classer dans l'une de 2 classes possibles A et B. Les 2 cercles vides représentent les valeurs initiales de 2 vecteurs prototypes représentant chacun une classe. Déduire graphiquement de la question 1) les coordonnées approximatives des vecteurs prototypes résultant de l'application de l'algorithme des nuées dynamiques à tous les vecteurs (représenter le déplacement des prototypes au moyen de flèches). (2 points)

Considérons par exemple que le cercle vide situé le plus haut représente le prototype initial de la classe A, et l'autre la classe B.

L'algorithme commence avec le point 1. Le prototype le plus proche de lui est celui de la classe A ; ce dernier se rapproche donc du point 1, ses nouvelles coordonnées deviennent égales à la moyenne de ces 2 points, c'est à dire (2, 4). Ecrivons ce résultat de la manière suivante :

point 1 $\rightarrow$ prototype A (2 ; 4)

Pour les autres points on obtient (ces calculs n'étaient pas demandés, mais cela permet de connaître la solution exacte permettant de vérifier le résultat graphique) :

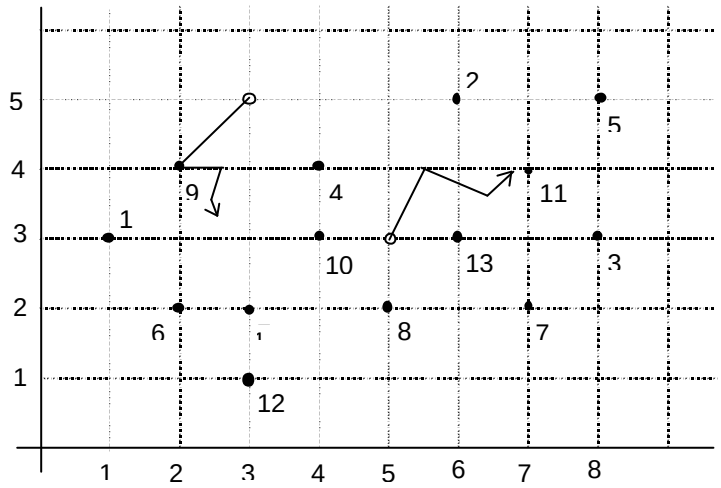
point 2 → prototype B (5,5 ; 4)
point 3 → prototype B (6,33 ; 3,67)
point 4 → prototype A (2,67 ; 4)
point 5 → prototype B (6,74 ; 4)
point 6 → prototype A (2,5 ; 3,5)
point 7 → prototype A (2,6 ; 3,2)

Pour le point 8 il n'était pas évident de trouver le prototype le plus proche de lui, graphiquement. C'est pourquoi les 2 possibilités sont acceptées. En fait quand on effectue le calcul (non demandé), on trouve que c'est le prototype B qui est le plus proche de lui.

Ensuite les 2 prototypes ne se déplacent plus que très peu, on peut donc considérer que les coordonnées approximatives des 2 prototypes sont :

A (2,6 ; 3,2) et B (6,74 ; 4)

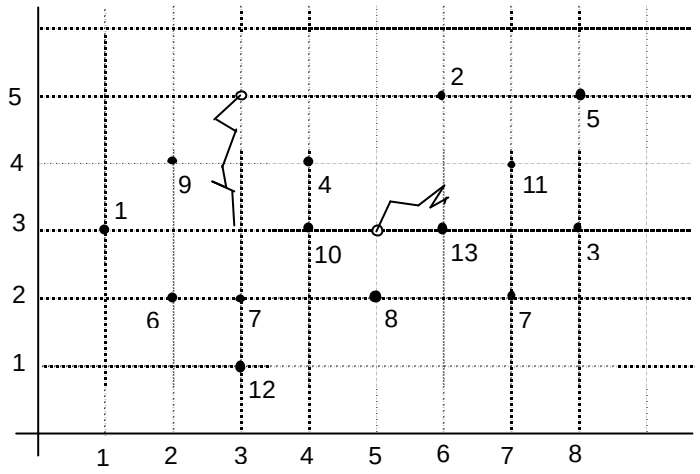
Graphiquement, ce résultat correspond aux déplacements suivants (pour plus de clarté on s'est arrêté au point 7) :



3) Même chose avec l'apprentissage compétitif non supervisé, avec un coefficient d'apprentissage égal à 0,2. (1 point)

Remarque : les composantes des vecteurs représentés sur la figure possèdent des valeurs entières, mais peuvent prendre des valeurs réelles.

L'apprentissage compétitif non supervisé, même s'il est très proche des nuées dynamiques, peut donner des résultats différents puisque les déplacements initiaux des prototypes sont plus petits que dans le cas des nuées dynamiques : ils sont toujours égaux à α (coefficient d'apprentissage) fois la distance entre le point à classer et le prototype le plus proche. Ici α est égal à 0,2 donc ce déplacement est égal au 5^e de la distance. Ce qui donne à peu près :



Cette fois-ci c'est l'affectation du point 10 qui n'est pas facile à déterminer graphiquement, les 2 solutions sont donc acceptées. Dans le graphique ci-dessus on suppose qu'il est plus près du prototype A que du B.

Les coordonnées approximatives résultant de l'algorithme sont donc à peu près :

A (3 ; 3) et B (6 ; 3,5)

Remarque : il y a 2 points numérotés 7, c'est une erreur.

Exercice 3 : Fonction ET avec un neurone

1) Proposer un neurone avec tous ses paramètres réalisant la fonction logique ET. (2 points)

On suppose qu'il s'agit de la fonction ET à 2 entrées. La dimension des entrées est donc 2. On utilise un neurone à seuil possédant 3 liens d'entrée : le premier dont l'entrée est une constante, par exemple 1, et les 2 autres reliés aux 2 entrées variables que l'on notera x_1 et x_2. La sortie de ce neurone est définie par :

$$y = s(w_0 + w_1 x_1 + w_2 x_2)$$

où s est la fonction seuil :

$$s(x) = \begin{cases} 1 \text{ si } x \geq 0 \\ 0 \text{ si } x < 0 \end{cases}$$

Ce neurone possède 2 états de sortie possibles. Ces 2 états peuvent correspondre à 2 classes, dont la frontière est définie dans le plan (x_1, x_2) par :

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

soit :

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$

qui est l'équation d'une droite de pente $-\frac{w_1}{w_2}$ et d'ordonnée à l'origine $-\frac{w_0}{w_2}$.

Graphiquement, on peut vérifier qu'une droite de pente –1 et d'ordonnée à l'origine 1,5 (par exemple) réalise une séparation correcte des 2 classes (les 2 classes sont 0 et 1, les 2

sorties possibles de la fonction ET). Il y a 3 paramètres donc une infinité de solutions possibles. On a donc :

$$-\frac{w_1}{w_2} = -1 \text{ et } -\frac{w_0}{w_2} = 1{,}5$$

Un exemple de solution (obtenue en fixant un des paramètres, par exemple w_2 à 1) est :

w_0=-1,5 w_1=1 w_2=1

2) Donner la structure du réseau le plus simple permettant de résoudre ce problème automatiquement par apprentissage supervisé, ainsi que le pseudo-code permettant cet apprentissage (les poids seront initialisés à 0). (2 points)

Pour résoudre ce même problème par apprentissage compétitif supervisé, on a besoin de 2 neurones de sortie car on a 2 classes. Pour chaque classe, un seul représentant suffit donc les 2 neurones de sortie sont suffisants. Le vecteur poids de chacun codera le représentant de l'une des 2 classes.

Soient :
$\{x_1, x_2, x_3, x_4\}$ les 4 vecteurs à classer
w_{ij} la i^e composante du j^e vecteur représentant (=poids synaptique de le i^e entrée du neurone j)

début
initialiser à 0 le représentant de chacune des 2 classes
répéter jusqu'à convergence (définie par taux d'erreur nul)
pour chacune des 4 entrées à classer faire
j←indice de la classe du représentant le plus proche de cette entrée
si j est l'indice de la classe correcte de l'entrée x_i
ajouter à w_{ij} : $\Delta w_{ij} = +\alpha.(x_i - w_{ij})$
si j n'est pas l'indice de la classe correcte de l'entrée x_i
ajouter à w_{ij} : $\Delta w_{ij} = -\alpha.(x_i - w_{ij})$
fin pour
fin

3) Sachant qu'en logique, la fonction OU exclusif peut être obtenue par la combinaison de la fonction NON-OU et de la fonction ET au moyen de la fonction OU, donner la structure d'un réseau de 3 neurones disposés en 2 couches permettant d'obtenir cette fonction. (1 point)

Dans le cas de la fonction OU, la seule chose qui change par rapport à la fonction ET est l'ordonnée à l'origine qui peut être par exemple égale à 0,5. On a donc :

$$-\frac{w_1}{w_2} = -1 \text{ et } -\frac{w_1}{w_2} = 0{,}5$$

avec comme exemple de solution (en fixant w_2 à 1, par exemple) :

w_0=-0,5 w_1=1 w_2=1

Pour la fonction NON-OU, il suffit d'inverser les poids obtenus pour la fonction OU.

Une autre solution consiste à utiliser une fonction OU en sortie de laquelle on ajoute un inverseur, sous la forme d'un neurone possédant un unique lien d'entrée affecté d'un poids égal à –1 (par exemple), et d'un seuil faiblement négatif, obtenu avec une 2^e entrée constante,

égale à 1 et affectée d'un poids 0,1 (par exemple).L'inconvénient de cette solution est de rajouter un neurone supplémentaire.

Une fois les 3 fonctions de base connues (OU, NON-OU et ET), la fonction OU-exclusif est obtenue en reliant les sorties des fonctions ET et NON-OU aux 2 entrées de la fonction OU.

4) Aurait-on pu utiliser un réseau de neurones utilisant l'apprentissage compétitif supervisé pour résoudre ce problème de classification ? Justifier la réponse. (1 point)

Oui, mais il aurait fallu au moins 2 représentants pour la classe 0, qui est de part et d'autre de la classe 1.

Pour la classe 1, une seul représentant aurait suffit, initialisé de manière quelconque (c'est l'avantage de l'apprentissage supervisé, on n'a pas besoin de mettre des contraintes au niveau de l'initialisation, puisqu'on impose qu'un vecteur représente une classe donnée, l'algorithme éloigne ou rapproche ce vecteur des vecteurs de sa classe selon le cas).

Pour la classe 0, il aurait fallu imposer des contraintes, pour que les deux vecteurs représentants soient de part et d'autre de la classe 1.

Le réseau doit donc se composer de 2 couches : la première est une couche de compétition dans laquelle un seul neurone est actif lors de la présentation de l'une des 4 entrée : le neurone vainqueur. Sa sortie est mise à 1 celle et celle des autres à 0. La classe 1 est codée par un seul neurone dans cette couche : l'entrée est classée dans la classe 1 quand ce neurone est actif. Pour la classe 0, dans la couche de compétition il y a 2 neurones dont les vecteurs poids constituent les 2 représentants de la classe. Les sorties de ces 2 neurones sont reliées à l'entrée d'un fonction OU.

Exercice 4 : Classification bayesienne

1) Comment peut-on obtenir une densité de probabilité aléatoire gaussienne, à partir d'un générateur aléatoire dont la densité de probabilité est constante (comme ceux que l'on trouve dans les compilateurs C par exemple) ? (1 point)

Il suffit de générer plusieurs nombres aléatoires (par exemple une dizaine) avec le générateur aléatoire du compilateur C (rand()) et de les additionner. Le résultat est une variable aléatoire définie par une densité de probabilité gaussienne.

Il faut bien sûr générer de nombreuses valeurs ainsi pour voir qu'elles suivent une densité de probabilité gaussienne.

Cette propriété résulte du *théorème central limite,* selon lequel la distribution d'observations résultant de plusieurs causes non corrélées, où aucune ne prédomine sur une autre, tend vers une distribution gaussienne (ce qui est le cas de très nombreuses données réelles). Soient n variables aléatoires indépendantes x_i. Leur somme est une variable aléatoire de moyenne η égale à la somme des moyennes η_i des x_i, et de variance σ égale à la somme de leurs variances σ_i :

$$\eta = \eta_1 + \eta_2 + ... + \eta_n$$

et

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 + ... + \sigma_n^2$$

Alors la variable $\frac{x-\eta}{\sigma^2}$ est une variable aléatoire gaussienne centrée réduite, c'est à dire de moyenne nulle (=centrée sur 0).

2) Soient A et B deux ensembles de vecteurs à 2 dimensions appartenant à deux classes différentes :

$$A=\left\{b_1\begin{pmatrix}3\\3\end{pmatrix},b_2\begin{pmatrix}3\\4\end{pmatrix},b_3\begin{pmatrix}4\\3\end{pmatrix},b_4\begin{pmatrix}4\\4\end{pmatrix}\right\} \text{ et } B=\left\{a_1\begin{pmatrix}1\\1\end{pmatrix},a_2\begin{pmatrix}1\\2\end{pmatrix},a_3\begin{pmatrix}2\\1\end{pmatrix}\right\}$$

Ces deux classes peuvent être modélisées par les deux gaussiennes bi-dimensionnelles suivantes :

$$G_A(x)=\frac{1}{1{,}57}\exp\left[-\frac{\begin{pmatrix}x_1-3{,}5\\x_2-3{,}5\end{pmatrix}^T\begin{pmatrix}0{,}25&0\\0&0{,}25\end{pmatrix}\begin{pmatrix}x_1-3{,}5\\x_2-3{,}5\end{pmatrix}}{2}\right]$$

$$G_B(x)=\frac{1}{1{,}2}\exp\left[-\frac{\begin{pmatrix}x_1-1{,}33\\x_2-1{,}33\end{pmatrix}^T\begin{pmatrix}0{,}22&-0{,}11\\-0{,}11&0{,}22\end{pmatrix}\begin{pmatrix}x_1-1{,}33\\x_2-1{,}33\end{pmatrix}}{2}\right]$$

Indiquer comment ces gaussiennes peuvent être obtenues à partir des données (sans refaire les calculs mais on donnant les expressions littérales de leurs paramètres et en expliquant comment ces derniers sont obtenus) (2 points)

Une gaussienne multidimensionnelle est définie par :

$$G(x)=\frac{1}{(2\pi)^{n/2}\sqrt{|C|}}\exp\left[-\frac{(x-x_c)^T C^{-1}(x-x_c)}{2}\right]$$

- n est la dimension des données (=2 ici)
- x_c est le centre de la gaussienne : elle est obtenue en calculant la moyenne des vecteurs de données (il s'agit d'une valeur estimée) :

$$x_c=\frac{\sum_{i=1}^{N}x_i}{N}$$

où N est le nombre d'exemples de valeurs pour la variable x (exemple : 50 pour chacune des 4 variables des Iris d'une des 3 classes).

- C est la matrice de covariance des données, que l'on peut écrire sous la forme :

$$C=\left\{\text{cov}(x_i,x_j), i=1,\ldots,n; j=1,\ldots,n\right\}$$

L'élément situé sur la j^e ligne et la i^e colonne représente la covariance entre les i^e et j^e variables, définie par :

$$\mathrm{cov}(x_i, x_j) = \sum_{k=1}^{N} \frac{(x_{i,k} - x_c)(x_{j,k} - x_c)}{N}$$

La dimension de C est n×n.

- C^{-1} est l'inverse de C
- |C| est le déterminant de C (un scalaire)

Elle peut également être obtenue en faisant le produit, pondéré par 1/*n*, de la matrice des vecteurs (disposés en colonne) et de sa transposée.

3) Effectuer le calcul détaillé de la classification pour une ressemblance maximale. (1 point)

Soit $d_r(x)$ la décision de classification pour une ressemblance maximale, du vecteur *x*, on a :

$$d(x) = y_i \text{ si } p(x \mid y_i) > p(x \mid y_j)$$

avec i,j=1, 2, ..., N et j≠i, où N est le nombre de classes (N=2 ici).
Dans le cas présent chaque classe est définie par une gaussienne multidimensionnelle, qui représente la fonction de densité de probabilité de cette classe :

$$p(x|y_A)=G_A(x)$$
$$p(x|y_B)=G_B(x)$$

où y_A et y_B représentent les classes A et B.

On ne donne pas d'exemple numérique auquel appliquer cette règle, alors on peut reprendre celui de l'année dernière :

$$x\begin{pmatrix}2\\2\end{pmatrix}$$

En appliquant la gaussienne multidimensionnelle à cette valeur on obtient :

Classe A : 0,36

Classe B : 0,8

Avec ce type de décision le vecteur x est donc affecté à la classe A.

4) Même chose pour une classification pour une erreur de classification minimale. (1 point)

Soit $d_e(x)$ la décision de classification pour une erreur minimale, du vecteur *x*, on a :

$$d(x) = y_i \text{ si } P(y_i \mid x) > P(y_j \mid x)$$

c'est à dire, d'après le théorème de Bayes :

$$\Leftrightarrow \qquad d(x) = y_i \text{ si } \frac{p(x \mid y_i)P(y_i)}{\sum_{i=1}^{N} p(x \mid y_i)P(y_i)} > \frac{p(x \mid y_j)P(y_j)}{\sum_{i=1}^{N} p(x \mid y_i)P(y_i)}$$

$$\Leftrightarrow \qquad d(x) = y_i \text{ si } p(x \mid y_i)P(y_i) > p(x \mid y_j)P(y_j)$$

avec i, j=1, 2, ..., N et j≠i

On applique cette règle au même exemple numérique :

$$x\begin{pmatrix}2\\2\end{pmatrix}$$

On connaît déjà $p(x|y_A)$ et $p(x|y_B)$: il reste à connaître les probabilités à priori des classes $P(y_A)$ et $P(y_B)$. C'est simplement les proportions d'exemples appartenant à chacune d'elles :

$$P(y_A)=4/7 \text{ et } P(y_B)=3/7$$

Le résultat est donc :

Classe A : 0,36×4/7=0,21

Classe B : 0,8×3/7=0,34

Avec ce type de décision le vecteur x est à nouveau affecté à la classe B.

Exercice 5 : Reconnaissance de parole par la méthode de comparaison dynamique

On souhaite calculer la distance entre les deux ensembles de vecteurs U et V suivants :

$$U=\left\{u_1\begin{pmatrix}0\\1\end{pmatrix},u_2\begin{pmatrix}1\\1\end{pmatrix},u_3\begin{pmatrix}1\\2\end{pmatrix},u_4\begin{pmatrix}1\\1\end{pmatrix}\right\} \text{ et } V=\left\{v_1\begin{pmatrix}0\\1\end{pmatrix},v_2\begin{pmatrix}2\\1\end{pmatrix},v_3\begin{pmatrix}1\\1\end{pmatrix},v_4\begin{pmatrix}1\\2\end{pmatrix},v_5\begin{pmatrix}1\\1\end{pmatrix}\right\}$$

par la méthode de comparaison dynamique (ou Dynamic Time Warping).

1) Reproduire le tableau suivant et le remplir avec les distances locales entre les u_i et les v_i (mesure de distance utilisée : distance euclidienne). Remarque : il est inutile de calculer les distances correspondant à la première ligne et la première colonne. (1 point)

Ces ensembles de vecteurs sont très proches de ceux du devoir surveillé de l'année dernière. Seul le vecteur v_4 est différent, et les vecteurs u_4 et v_5 ont été rajoutés.

Par exemple, la distance euclidienne entre v_4 et u_2 est :

$$d(v_4,u_2)=\sqrt{(1-1)^2+(2-1)^2}=1$$

Le tableau des distances locales obtenu est :

	u_1	**u_2**	**u_3**	**u_4**
v_1	/	/	/	/
v_2	/	1	$\sqrt{2}$	1
v_3	/	0	1	0
v_4	/	1	0	1
v_5	/	0	1	0

2) Appliquer l'algorithme DTW à ces données (en expliquant la procédure) et reproduire à nouveau le tableau ci-dessus, rempli avec les distances cumulées. (2 points)

On applique l'algorithme défini par :

$$D(i,j) = \min[(D(i-1,j-1), D(i-1,j), D(i,j-1)] + d(i,j)$$

où d(i,j) est la distance locale entre les vecteurs d'indices i et j. La 1ère ligne et la 1ère colonne sont initialisées par une grande valeur G (voir programme C utilisé en TP), par exemple 100000, sauf la 1ère case qui elle est initialisée à 0.

	$\mathbf{u_1}$	$\mathbf{u_2}$	$\mathbf{u_3}$	$\mathbf{u_4}$
$\mathbf{v_1}$	0	G	G	G
$\mathbf{v_2}$	G	1	$\sqrt{2}+1$	$\sqrt{2}+2$
$\mathbf{v_3}$	G	1	2	2
$\mathbf{v_4}$	G	1	1	2
$\mathbf{v_5}$	G	1	2	1

La distance cumulée entre les 2 ensembles de vecteurs est ici 1.

3) Représenter par une flèche le cheminement de l'algorithme, à rebours à partir de la case d'arrivée (u_4,v_5). (1 point)

Pour trouver le chemin optimal on part de la case d'arrivée, on regarde de quelle case on vient (c'est à dire celle qui correspond à la distance cumulée minimale, en l'occurrence (v_4,u_3)), et ainsi de suite. On obtient le chemin suivant :

$$(u_4,v_5) \rightarrow (u_3,v_4) \rightarrow (u_2,v_3) \rightarrow (u_2,v_2) \rightarrow (u_1,v_1)$$

II) Questions à choix multiples : mettre une croix dans le bon cercle (5 points)

Bonne réponse : +0,5 point ; Pas de réponse : 0 point ; Mauvaise réponse : −0,5 point (une note négative sera convertie en 0)

1) Avec l'affichage de Sammon, 2 exécutions successives de l'algorithme avec les mêmes données donnent le même résultat.
   oui O non Θ
2) Le fait de normaliser les données à apprendre et à reconnaître améliore les résultats de la classification.
   oui Θ non O
3) Dans l'apprentissage supervisé, l'information d'appartenance aux classes est utilisée pendant l'apprentissage.
   oui Θ non O
4) La fonction logique OU-exclusif est un problème linéairement séparable.
   oui O non Θ
5) Un bon taux de reconnaissance sur la base d'apprentissage signifie une bonne généralisation.
   oui O non Θ
6) La densité de probabilité d'une variable aléatoire résultant de plusieurs causes non corrélées tend vers une densité de probabilité gaussienne.
   oui Θ non O
7) La classification bayesienne est une méthode optimale pour des données gaussiennes.
   oui Θ non O
8) L'algorithme de rétropropagation du gradient permet de résoudre les problèmes non-linéairement séparables.
   oui Θ non O
9) Dans les réseaux multicouches auxquels on applique la rétropropagation du gradient, on doit avoir au moins une couche non-linéaire.
   oui Θ non O
10) La matrice de covariance d'un ensemble de vecteurs de dimension n est de taille $n \times n$.
    oui Θ non O