**Apprentissage et reconnaissance (GIF-4101 / GIF-7005)**
**Département de génie électrique et de génie informatique**
**Automne 2009**

UNIVERSITÉ LAVAL

# EXAMEN FINAL

---

Instructions : – Aucune documentation sauf aide mémoire d'une page recto-verso manuscrit
– Ce questionnaire comporte 6 pages et 5 questions
– Durée de l'examen : 2 heure 30 minutes

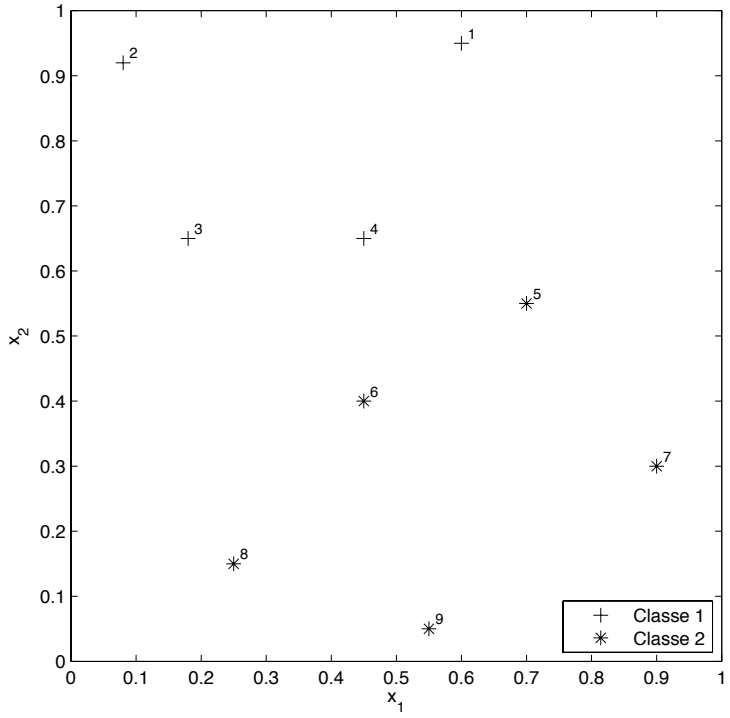
Pondération : Cet examen compte pour $35\%$ de la note finale.

---

1. Soit le jeu de données $\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^{9}$ présenté ci-bas.

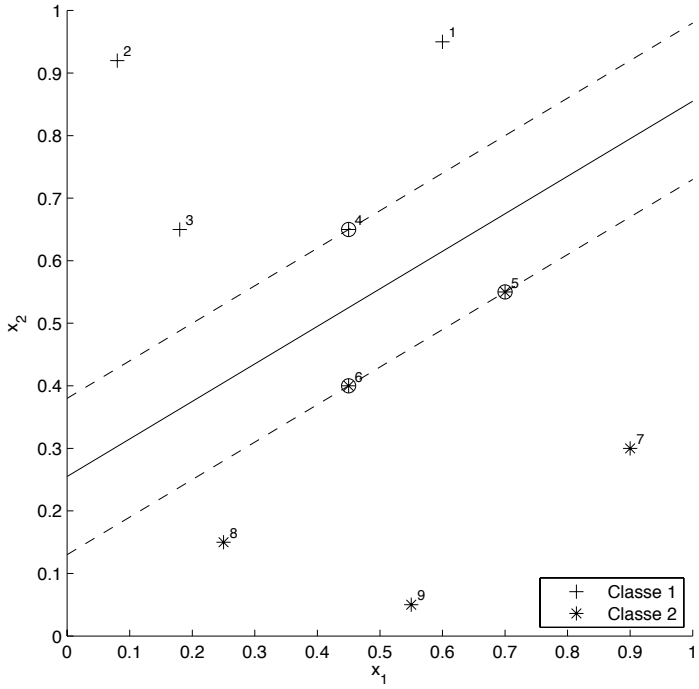
$$\mathbf{x^1} = \begin{bmatrix} 0.6 \\ 0.95 \end{bmatrix}, \quad r^1 = -1, \quad \mathbf{x^2} = \begin{bmatrix} 0.08 \\ 0.92 \end{bmatrix}, \quad r^2 = -1, \quad \mathbf{x^3} = \begin{bmatrix} 0.18 \\ 0.65 \end{bmatrix}, \quad r^3 = -1,$$
$$\mathbf{x^4} = \begin{bmatrix} 0.45 \\ 0.65 \end{bmatrix}, \quad r^4 = -1, \quad \mathbf{x^5} = \begin{bmatrix} 0.7 \\ 0.55 \end{bmatrix}, \quad r^5 = 1, \quad \mathbf{x^6} = \begin{bmatrix} 0.45 \\ 0.4 \end{bmatrix}, \quad r^6 = 1,$$
$$\mathbf{x^7} = \begin{bmatrix} 0.9 \\ 0.3 \end{bmatrix}, \quad r^7 = 1, \quad \mathbf{x^8} = \begin{bmatrix} 0.25 \\ 0.15 \end{bmatrix}, \quad r^8 = 1, \quad \mathbf{x^9} = \begin{bmatrix} 0.55 \\ 0.05 \end{bmatrix}, \quad r^9 = 1$$

La figure suivante trace ces points en deux dimensions.




(a) (5pts) Supposons que l'on veut classer ces données avec un classifieur de type Séparateur à Vastes Marges (SVM) utilisant un noyau linéaire ($K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$), sans marge floue. Dans votre cahier de réponse, tracez les données du jeu $\mathcal{X}$, les marges géométriques maximales obtenues avec le SVM, l'hyperplan séparateur correspondant, et encerclez les données agissant comme vecteurs de support.

**Solution :**




(b) (10pts) Donnez les valeurs des poids $\mathbf{w}$ et biais $w_0$ correspondant au discriminant linéaire maximisant les marges géométriques tracées en a).

**Solution :** Trois données sont identifiées comme vecteurs de support : $\mathbf{x}^4$, $\mathbf{x}^5$ et $\mathbf{x}^6$. Ceci nous donne trois équations et trois inconnus.

$$\begin{aligned}
\mathrm{h}(\mathbf{x}^4) &= w_2 x_2^4 + w_1 x_1^4 + w_0 = 0.65 w_2 + 0.45 w_1 + w_0 = -1 \\
\mathrm{h}(\mathbf{x}^5) &= w_2 x_2^5 + w_1 x_1^5 + w_0 = 0.55 w_2 + 0.7 w_1 + w_0 = 1 \\
\mathrm{h}(\mathbf{x}^6) &= w_2 x_2^6 + w_1 x_1^6 + w_0 = 0.4 w_2 + 0.45 w_1 + w_0 = 1
\end{aligned}$$

La résolution de ce système d'équation peut se faire de la façon suivant.

$$\begin{aligned}
\mathrm{EQ1} - \mathrm{EQ3} \quad &: \quad (0.65 - 0.4) w_2 + (0.45 - 0.45) w_1 + (1 - 1) w_0 = -1 - 1 \\
&\qquad 0.25 w_2 = -2 \quad \Rightarrow \quad w_2 = -8 \\
\mathrm{EQ2} - \mathrm{EQ1} \quad &: \quad (0.55 - 0.65) w_2 + (0.7 - 0.45) w_1 + (1 - 1) w_0 = 1 + 1 \\
&\qquad -0.1 \times (-8) + 0.25 w_1 = 2 \\
&\qquad 0.25 w_1 = 2 - 0.8 = 1.2 \quad \Rightarrow \quad w_1 = 4.8 \\
\mathrm{EQ1} \quad &: \quad 0.65 w_2 + 0.45 w_1 + w_0 = -1 \\
&\qquad 0.65 \times (-8) + 0.45 \times (4.8) + w_0 = -1 \\
&\qquad w_0 = -1 + 5.2 - 2.16 = 2.04
\end{aligned}$$

La résolution de ce système d'équations nous donne les valeurs suivantes.

$$w_2 = -8, \; w_1 = 4.8, \; w_0 = 2.04$$

2. Les réseaux *Radial Basis Function* (RBF) sont parfois considérés comme des réseaux de neurones de type perceptron multi-couches. En effet, on peut définir un réseau RBF comme un perceptron multi-couches avec une couche cachée, où chaque neurone sur la couche de sortie utilise une fonction de transfert linéaire, $\mathrm{f}(a) = a$. Sur la couche cachée, on retrouve des neurones d'un type

particulier, paramétrés par les variables $\mathbf{m}_i$ et $s_i$ (plutôt que $\mathbf{w}$ et $w_0$), où la sortie $y_j^t$ du $j$-ième neurone de la couche cachée pour une donnée $\mathbf{x}^t$ est calculée selon l'équation suivante.

$$y_j^t = \exp\left(-\frac{\|\mathbf{x}^t - \mathbf{m}_j\|^2}{2s_j^2}\right) = \exp\left(-\frac{\sum_{i=1}^{D}(x_i^t - m_{j,i})^2}{2s_j^2}\right)$$

(a) (10pts) Développez les équations de $\Delta w_{j,i},\ j = 1, \ldots, K$, permettant d'effectuer un apprentissage des poids $\mathbf{w}_j$ et biais $w_{j,0}$ de la couche de sortie du réseau RBF à $K$ neurones par une rétropropagation des erreurs.

**Solution :** Sur la couche de sortie, les poids sont ajustés par l'équation suivante.

$$\Delta w_{j,i} = -\eta \frac{\partial E(\mathbf{x}^t)}{\partial w_{j,i}}$$

Selon le chaînage des dérivées, on peut développer les équations suivantes :

$$\begin{aligned}
\frac{\partial E^t}{\partial w_{j,i}} &= \frac{\partial E^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{j,i}} \\
\frac{\partial E^t}{\partial w_{j,0}} &= \frac{\partial E^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{j,0}}
\end{aligned}$$

On calcule les valeurs des dérivées partielles comme suit :

$$\begin{aligned}
\frac{\partial E^t}{\partial e_j^t} &= \frac{\partial}{\partial e_j^t}\frac{1}{2}\sum_{j=1}^{K}(e_j^t)^2 = e_j^t \\
\frac{\partial e_j^t}{\partial y_j^t} &= \frac{\partial}{\partial y_j^t} r_j^t - y_j^t = -1 \\
\frac{\partial y_j^t}{\partial a_j^t} &= \frac{\partial x}{\partial x} = 1 \\
\frac{\partial a_j^t}{\partial w_{j,i}} &= \frac{\partial}{\partial w_{j,i}}\sum_{i=1}^{R} w_{j,i}y_i^t + w_{j,0} = y_i^t \\
\frac{\partial a_j^t}{\partial w_{j,0}} &= \frac{\partial}{\partial w_{j,0}}\sum_{i=1}^{R} w_{j,i}y_i^t + w_{j,0} = 1
\end{aligned}$$

Ce qui nous donne pour la couche de sortie :

$$\begin{aligned}
\Delta w_{j,i} &= -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial w_{j,i}} = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{j,i}} \\
&= \frac{\eta}{N}\sum_{t=1}^{N} e_j^t y_i^t \\
\Delta w_{j,0} &= -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial w_{j,0}} = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{j,0}} \\
&= \frac{\eta}{N}\sum_{t=1}^{N} e_j^t
\end{aligned}$$

(b) (15pts) Développez les équations de $\Delta m_{j,i}$ et $\Delta s_j$ permettant d'effectuer un apprentissage des paramètres $\mathbf{m}_j$ et $s_j$ de la couche cachée de neurones du réseau RBF, par une rétropropagation des erreurs.

**Solution :** Pour mettre à jour les valeurs de $\mathbf{m}_j$ par le descente du gradient, on utilise l'équation suivante.

$$\Delta m_{j,i} = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial m_{j,i}} = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial m_{j,i}}$$

$$\begin{aligned}
\frac{\partial E^t}{\partial y_j^t} &= \frac{\partial}{\partial y_j^t}\frac{1}{2}\sum_{k=1}^{K}(e_k^t)^2 = \sum_{k=1}^{K} e_k^t\frac{\partial e_k^t}{\partial y_j^t} \\
&= \sum_{k=1}^{K} e_k^t\frac{\partial e_k^t}{\partial a_k^t}\frac{\partial a_k^t}{\partial y_j^t} \\
&= \sum_{k=1}^{K} e_k^t\frac{\partial(r_k^t - \mathrm{f}(a_k^t))}{\partial a_k^t}\frac{\partial\left(\sum_l w_{k,l}y_l^t + w_{k,0}\right)}{\partial y_j^t} \\
&= \sum_{k=1}^{K} e_k^t(-1)w_{k,j} = -\sum_{k=1}^{K} e_k^t w_{k,j}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial y_j^t}{\partial m_{j,i}} &= \frac{\partial}{\partial m_{j,i}}\exp\left(-\frac{\sum_{l=1}^{D}(x_l^t - m_{j,l})^2}{2s_j^2}\right) \\
&= \exp\left(-\frac{\sum_{l=1}^{D}(x_l^t - m_{j,l})^2}{2s_j^2}\right)\frac{(-1)(-1)2(x_i^t - m_{j,i})}{2s_j^2} \\
&= y_j^t\frac{x_i^t - m_{j,i}}{s_j^2}
\end{aligned}$$

Ce qui résulte en l'équation suivante pour la mise à jour des $\mathbf{m}_j$ :

$$\begin{aligned}
\Delta m_{j,i} &= -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial m_{j,i}} \\
&= \frac{\eta}{N}\sum_{t=1}^{N}\left[\sum_{k=1}^{K} e_k^t w_{k,j}\right]\left[\frac{x_i^t - m_{j,i}}{s_j^2}\right]y_j^t
\end{aligned}$$

Pour la mise à jour des valeurs des $s_j$, on procède de façon similaire.

$$\Delta s_j = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial s_j} = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial s_j}$$

$$\begin{aligned}
\frac{\partial y_j^t}{\partial s_j} &= \frac{\partial}{\partial s_j}\exp\left(-\frac{\sum_{i=1}^{D}(x_i^t - m_{j,i})^2}{2s_j^2}\right) \\
&= \exp\left(-\frac{\sum_{l=1}^{D}(x_l^t - m_{j,l})^2}{2s_j^2}\right)(-1)\frac{(-2)\sum_{l=1}^{D}(x_l^t - m_{j,l})^2}{2s_j^3} \\
&= \frac{\sum_{l=1}^{D}(x_l^t - m_{j,l})^2}{s_j^3}\exp\left(-\frac{\sum_{l=1}^{D}(x_l^t - m_{j,l})^2}{2s_j^2}\right)
\end{aligned}$$

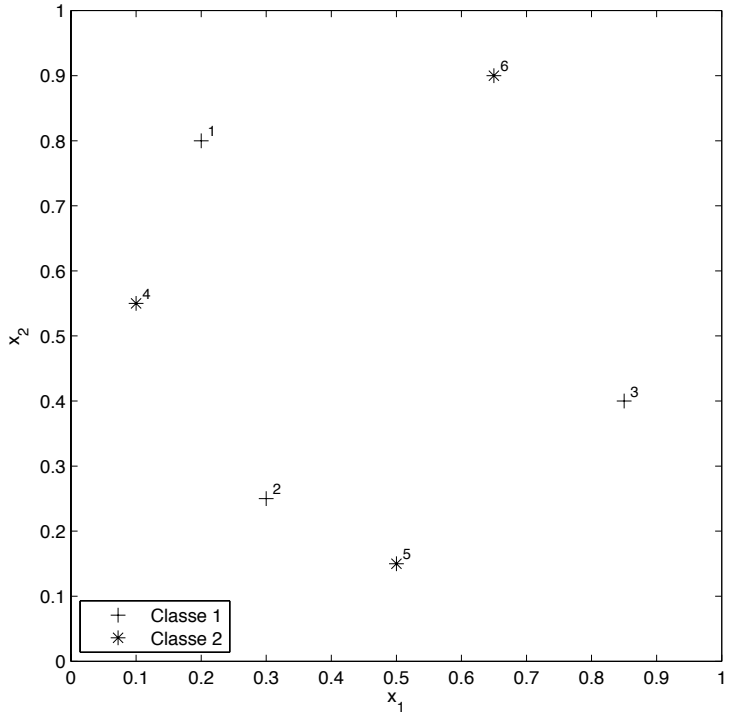
Ce qui résulte en l'équation suivante pour la mise à jour des $s_j$ :

$$\begin{aligned}\Delta s_j &= -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial s_j} \\ &= \frac{\eta}{N}\sum_{t=1}^{N}\left[\sum_{k=1}^{K} e_k^t w_{k,j}\right]\left[\frac{\sum_{l=1}^{D}(x_l^t - m_{j,l})^2}{s_j^3}\right] y_j^t\end{aligned}$$

3. Soit le jeu de données $\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^{6}$ suivant, comportant deux classes ($r^t \in \{-1, 1\}$).

$$\mathbf{x^1} = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, \quad r^1 = -1, \quad \mathbf{x^2} = \begin{bmatrix} 0.3 \\ 0.25 \end{bmatrix}, \quad r^2 = -1, \quad \mathbf{x^3} = \begin{bmatrix} 0.85 \\ 0.4 \end{bmatrix}, \quad r^3 = -1$$
$$\mathbf{x^4} = \begin{bmatrix} 0.1 \\ 0.55 \end{bmatrix}, \quad r^4 = 1, \quad \mathbf{x^5} = \begin{bmatrix} 0.5 \\ 0.15 \end{bmatrix}, \quad r^5 = 1, \quad \mathbf{x^6} = \begin{bmatrix} 0.65 \\ 0.9 \end{bmatrix}, \quad r^6 = 1$$

La figure suivante trace ces points en deux dimensions.




(a) (5pts) Supposons que l'on a entraîné un SVM avec ces données et que l'on a obtenu les valeurs de $\alpha^t$ et $w_0$ suivantes, en utilisant la matrice de Gram $G(\mathcal{X})$ donnée ci-bas, calculée avec un noyau de type gaussien.

$$G(\mathcal{X}) = \begin{bmatrix} 1.00 & 0.54 & 0.31 & 0.87 & 0.36 & 0.65 \\ 0.54 & 1.00 & 0.52 & 0.77 & 0.90 & 0.34 \\ 0.31 & 0.52 & 1.00 & 0.31 & 0.69 & 0.56 \\ 0.87 & 0.77 & 0.31 & 1.00 & 0.53 & 0.43 \\ 0.36 & 0.90 & 0.69 & 0.53 & 1.00 & 0.31 \\ 0.65 & 0.34 & 0.56 & 0.43 & 0.31 & 1.00 \end{bmatrix}, \ \boldsymbol{\alpha} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}, \ w_0 = -0.0042$$

Effectuez le classement des données de $\mathcal{X}$ selon ces valeurs de $\alpha^t$ et $w_0$ et donnez le taux d'erreur de classement correspondant.

**Solution :** Le classement avec un SVM se fait en utilisant l'équation suivante.

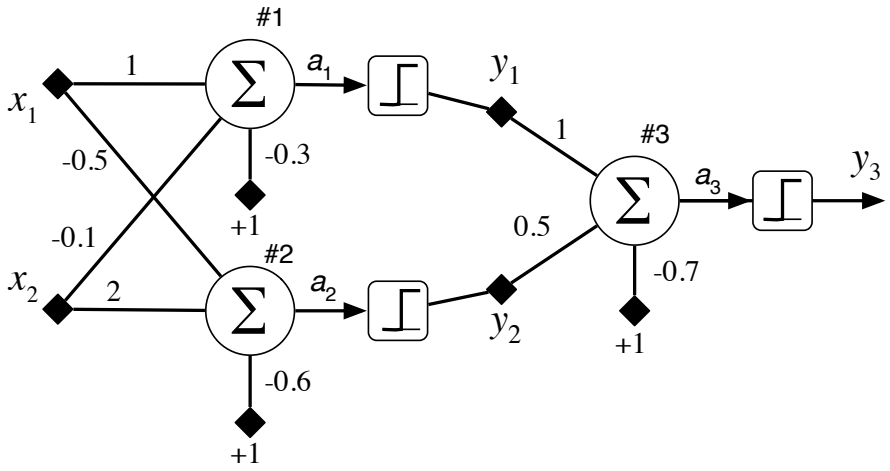
$$\mathrm{h}(\mathbf{x}) = \sum_{t=1}^{N} \alpha^t r^t K(\mathbf{x}^t, \mathbf{x}) + w_0$$

Dans le cas présent les résultats seraient les suivants.

$$\begin{aligned}
\mathrm{h}(\mathbf{x}^1) &= (0.1 \times -1 \times 1.00) + (0.1 \times -1 \times 0.54) + (0.1 \times -1 \times 0.31) \\
&\quad +(0.1 \times 1 \times 0.87) + (0.1 \times 1 \times 0.36) + (0.1 \times 1 \times 0.65) - 0.0042 \\
&= -0.1 - 0.054 - 0.031 + 0.087 + 0.036 + 0.065 - 0.0042 = -0.0012 \\
&\quad \Rightarrow \mathbf{x}^1 \in C_1 \\
\mathrm{h}(\mathbf{x}^2) &= (0.1 \times -1 \times 0.54) + (0.1 \times -1 \times 1.00) + (0.1 \times -1 \times 0.52) \\
&\quad +(0.1 \times 1 \times 0.77) + (0.1 \times 1 \times 0.90) + (0.1 \times 1 \times 0.34) - 0.0042 \\
&= -0.054 - 0.1 - 0.052 + 0.077 + 0.09 + 0.034 - 0.0042 = -0.0092 \\
&\quad \Rightarrow \mathbf{x}^2 \in C_1 \\
\mathrm{h}(\mathbf{x}^3) &= (0.1 \times -1 \times 0.31) + (0.1 \times -1 \times 0.52) + (0.1 \times -1 \times 1.00) \\
&\quad +(0.1 \times 1 \times 0.31) + (0.1 \times 1 \times 0.69) + (0.1 \times 1 \times 0.56) - 0.0042 \\
&= -0.031 - 0.052 - 0.1 + 0.031 + 0.069 + 0.056 - 0.0042 = -0.0312 \\
&\quad \Rightarrow \mathbf{x}^3 \in C_1 \\
\mathrm{h}(\mathbf{x}^4) &= (0.1 \times -1 \times 0.87) + (0.1 \times -1 \times 0.77) + (0.1 \times -1 \times 0.31) \\
&\quad +(0.1 \times 1 \times 1.00) + (0.1 \times 1 \times 0.53) + (0.1 \times 1 \times 0.43) - 0.0042 \\
&= -0.087 - 0.077 - 0.031 + 0.1 + 0.053 + 0.043 - 0.0042 = -0.0032 \\
&\quad \Rightarrow \mathbf{x}^4 \in C_1 \\
\mathrm{h}(\mathbf{x}^5) &= (0.1 \times -1 \times 0.36) + (0.1 \times -1 \times 0.90) + (0.1 \times -1 \times 0.69) \\
&\quad +(0.1 \times 1 \times 0.53) + (0.1 \times 1 \times 1.00) + (0.1 \times 1 \times 0.31) - 0.0042 \\
&= -0.036 - 0.09 - 0.069 + 0.053 + 0.1 + 0.031 - 0.0042 = -0.0152 \\
&\quad \Rightarrow \mathbf{x}^5 \in C_1 \\
\mathrm{h}(\mathbf{x}^6) &= (0.1 \times -1 \times 0.65) + (0.1 \times -1 \times 0.34) + (0.1 \times -1 \times 0.56) \\
&\quad +(0.1 \times 1 \times 0.43) + (0.1 \times 1 \times 0.31) + (0.1 \times 1 \times 1.00) - 0.0042 \\
&= -0.065 - 0.034 - 0.056 + 0.043 + 0.031 + 0.1 - 0.0042 = 0.0148 \\
&\quad \Rightarrow \mathbf{x}^6 \in C_2
\end{aligned}$$

Donc, deux données sont mal classées, soit $\mathbf{x}^4$ et $\mathbf{x}^5$, pour un taux d'erreur de classement de $0.333$

(b) (5pts) Supposons le réseau de type perceptron multi-couches suivant, avec une couche cachée à deux neurones et une couche de sortie à un neurone.




Les poids $\mathbf{w}$ et biais $w_0$ des trois neurones du réseau sont les suivants.

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ -0.1 \end{bmatrix},\ w_{1,0} = -0.3,\ \mathbf{w}_2 = \begin{bmatrix} -0.5 \\ 2 \end{bmatrix},\ w_{2,0} = -0.6,\ \mathbf{w}_3 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix},\ w_{3,0} = -0.7$$

Les trois neurones du réseau utilisent une fonction de transfert binarisant les valeurs selon la valeur du signe de $a_i$.

$$\mathrm{f}(a) = \begin{cases} 1 & \text{si } a \geq 0 \\ 0 & \text{autrement} \end{cases}$$

Effectuez le classement de ces données de $\mathcal{X}$ selon cette paramétrisation de perceptron multi-couches et donnez le taux d'erreur de classement correspondant.

**Solution :** Dans le cas présent les résultats seraient les suivants.

Neurone #1 :

$$\begin{aligned}
a_1^1 &= (\mathbf{w}_1)^T\mathbf{x}^1 + w_{1,0} = [1\ -0.1][0.2\ 0.8]^T + -0.3 = 0.2 - 0.08 - 0.3 = -0.18 \\
y_1^1 &= \mathrm{f}(a_1^1) = 0 \\
a_1^2 &= (\mathbf{w}_1)^T\mathbf{x}^2 + w_{1,0} = [1\ -0.1][0.3\ 0.25]^T + -0.3 = 0.3 - 0.025 - 0.3 = -0.025 \\
y_1^2 &= \mathrm{f}(a_1^2) = 0 \\
a_1^3 &= (\mathbf{w}_1)^T\mathbf{x}^3 + w_{1,0} = [1\ -0.1][0.85\ 0.4]^T + -0.3 = 0.85 - 0.04 - 0.3 = 0.51 \\
y_1^3 &= \mathrm{f}(a_1^3) = 1 \\
a_1^4 &= (\mathbf{w}_1)^T\mathbf{x}^4 + w_{1,0} = [1\ -0.1][0.1\ 0.55]^T + -0.3 = 0.1 - 0.055 - 0.3 = -0.255 \\
y_1^4 &= \mathrm{f}(a_1^4) = 0 \\
a_1^5 &= (\mathbf{w}_1)^T\mathbf{x}^5 + w_{1,0} = [1\ -0.1][0.5\ 0.15]^T + -0.3 = 0.5 - 0.015 - 0.3 = 0.185 \\
y_1^5 &= \mathrm{f}(a_1^5) = 1 \\
a_1^6 &= (\mathbf{w}_1)^T\mathbf{x}^6 + w_{1,0} = [1\ -0.1][0.65\ 0.9]^T + -0.3 = 0.65 - 0.09 - 0.3 = 0.26 \\
y_1^6 &= \mathrm{f}(a_1^6) = 1
\end{aligned}$$

Neurone #2 :

$$\begin{aligned}
a_2^1 &= (\mathbf{w}_2)^T\mathbf{x}^1 + w_{2,0} = [-0.5\ 2][0.2\ 0.8]^T + -0.6 = -0.1 + 1.6 + -0.6 = 0.9\\
y_2^1 &= \mathrm{f}(a_2^1) = 1\\
a_2^2 &= (\mathbf{w}_2)^T\mathbf{x}^2 + w_{2,0} = [-0.5\ 2][0.3\ 0.25]^T + -0.6 = -0.15 + 0.5 - 0.6 = -0.25\\
y_2^2 &= \mathrm{f}(a_2^2) = 0\\
a_2^3 &= (\mathbf{w}_2)^T\mathbf{x}^3 + w_{2,0} = [-0.5\ 2][0.85\ 0.4]^T + -0.6 = -0.425 + 0.8 - 0.6 = -0.225\\
y_2^3 &= \mathrm{f}(a_2^3) = 0\\
a_2^4 &= (\mathbf{w}_2)^T\mathbf{x}^4 + w_{2,0} = [-0.5\ 2][0.1\ 0.55]^T + -0.6 = -0.05 + 1.1 - 0.6 = 0.45\\
y_2^4 &= \mathrm{f}(a_2^4) = 1\\
a_2^5 &= (\mathbf{w}_2)^T\mathbf{x}^5 + w_{2,0} = [-0.5\ 2][0.5\ 0.15]^T + -0.6 = -0.25 + 0.3 - 0.6 = -0.55\\
y_2^5 &= \mathrm{f}(a_2^5) = 0\\
a_2^6 &= (\mathbf{w}_2)^T\mathbf{x}^6 + w_{2,0} = [-0.5\ 2][0.65\ 0.9]^T + -0.6 = -0.325 + 1.8 - 0.6 = 0.875\\
y_2^6 &= \mathrm{f}(a_2^6) = 1
\end{aligned}$$

Neurone #3 :

$$\begin{aligned}
a_3^1 &= (\mathbf{w}_3)^T\mathbf{y}^1 + w_{3,0} = [1\ 0.5][0\ 1]^T + -0.7 = 0 + 0.5 - 0.7 = -0.2\\
y_3^1 &= \mathrm{f}(a_3^1) = 0 \quad \Rightarrow \quad \mathbf{x}^1 \in C_1\\
a_3^2 &= (\mathbf{w}_3)^T\mathbf{y}^2 + w_{3,0} = [1\ 0.5][0\ 0]^T + -0.7 = 0 + 0 - 0.7 = -0.7\\
y_3^2 &= \mathrm{f}(a_3^2) = 0 \quad \Rightarrow \quad \mathbf{x}^2 \in C_1\\
a_3^3 &= (\mathbf{w}_3)^T\mathbf{y}^3 + w_{3,0} = [1\ 0.5][1\ 0]^T + -0.7 = 1 + 0 - 0.7 = 0.3\\
y_3^3 &= \mathrm{f}(a_3^3) = 1 \quad \Rightarrow \quad \mathbf{x}^3 \in C_2\\
a_3^4 &= (\mathbf{w}_3)^T\mathbf{y}^4 + w_{3,0} = [1\ 0.5][0\ 1]^T + -0.7 = 0 + 0.5 - 0.7 = -0.2\\
y_3^4 &= \mathrm{f}(a_3^4) = 0 \quad \Rightarrow \quad \mathbf{x}^4 \in C_1\\
a_3^5 &= (\mathbf{w}_3)^T\mathbf{y}^5 + w_{3,0} = [1\ 0.5][1\ 0]^T + -0.7 = 1 + 0 - 0.7 = 0.3\\
y_3^5 &= \mathrm{f}(a_3^5) = 1 \quad \Rightarrow \quad \mathbf{x}^5 \in C_2\\
a_3^6 &= (\mathbf{w}_3)^T\mathbf{y}^6 + w_{3,0} = [1\ 0.5][1\ 1]^T + -0.7 = 1 + 0.5 - 0.7 = 0.8\\
y_3^6 &= \mathrm{f}(a_3^6) = 1 \quad \Rightarrow \quad \mathbf{x}^6 \in C_2
\end{aligned}$$

Donc, le perceptron multi-couches fait deux erreurs de classement ($\mathbf{x}^3$ et $\mathbf{x}^4$), pour un taux d'erreur de classement de $0.33$.

(c) (5pts) Le classement par des souches de décision est effectué avec l'équation suivante.

$$\mathrm{h}(\mathbf{x}|\theta, \upsilon, \gamma) = \mathrm{sgn}(\theta(x_\gamma - \upsilon)), \quad \theta \in \{-1, +1\},\ \gamma \in \{1, \ldots, D\},\ \upsilon \in \mathbb{R}$$

Soit l'ensemble des trois souches de décisions suivantes, avec les valeurs de paramètre $\beta$ correspondantes, produits par l'exécution de l'algorithme AdaBoost sur le jeu de données $\mathcal{X}$.

$$\begin{array}{lllll}
\text{Souche \#1 :} & \theta_1 = +1, & \upsilon_1 = 0.4, & \gamma_1 = 1, & \beta_1 = 0.5\\
\text{Souche \#2 :} & \theta_2 = +1, & \upsilon_2 = 0.475, & \gamma_2 = 2, & \beta_2 = 0.333\\
\text{Souche \#3 :} & \theta_3 = -1, & \upsilon_3 = 0.15, & \gamma_3 = 1, & \beta_3 = 0.5
\end{array}$$

Effectuez le classement des données de $\mathcal{X}$ avec ces trois souches de décisions combinées selon l'algorithme AdaBoost et donnez le taux d'erreur de classement correspondant.

**Solution :**
Le classement selon chacune des souches de décision serait le suivant.
Souche #1 :

$$\begin{aligned}
\mathrm{h}_1(\mathbf{x}^1) &= \mathrm{sgn}(1 \times (0.2 - 0.4)) = -1 \\
\mathrm{h}_1(\mathbf{x}^2) &= \mathrm{sgn}(1 \times (0.3 - 0.4)) = -1 \\
\mathrm{h}_1(\mathbf{x}^3) &= \mathrm{sgn}(1 \times (0.85 - 0.4)) = 1 \\
\mathrm{h}_1(\mathbf{x}^4) &= \mathrm{sgn}(1 \times (0.1 - 0.4)) = -1 \\
\mathrm{h}_1(\mathbf{x}^5) &= \mathrm{sgn}(1 \times (0.5 - 0.4)) = 1 \\
\mathrm{h}_1(\mathbf{x}^6) &= \mathrm{sgn}(1 \times (0.65 - 0.4)) = 1
\end{aligned}$$

Souche #2 :

$$\begin{aligned}
\mathrm{h}_2(\mathbf{x}^1) &= \mathrm{sgn}(1 \times (0.8 - 0.475)) = 1 \\
\mathrm{h}_2(\mathbf{x}^2) &= \mathrm{sgn}(1 \times (0.25 - 0.475)) = -1 \\
\mathrm{h}_2(\mathbf{x}^3) &= \mathrm{sgn}(1 \times (0.4 - 0.475)) = -1 \\
\mathrm{h}_2(\mathbf{x}^4) &= \mathrm{sgn}(1 \times (0.55 - 0.475)) = 1 \\
\mathrm{h}_2(\mathbf{x}^5) &= \mathrm{sgn}(1 \times (0.15 - 0.475)) = -1 \\
\mathrm{h}_2(\mathbf{x}^6) &= \mathrm{sgn}(1 \times (0.9 - 0.475)) = 1
\end{aligned}$$

Souche #3 :

$$\begin{aligned}
\mathrm{h}_3(\mathbf{x}^1) &= \mathrm{sgn}(-1 \times (0.2 - 0.15)) = -1 \\
\mathrm{h}_3(\mathbf{x}^2) &= \mathrm{sgn}(-1 \times (0.3 - 0.15)) = -1 \\
\mathrm{h}_3(\mathbf{x}^3) &= \mathrm{sgn}(-1 \times (0.85 - 0.15)) = -1 \\
\mathrm{h}_3(\mathbf{x}^4) &= \mathrm{sgn}(-1 \times (0.1 - 0.15)) = 1 \\
\mathrm{h}_3(\mathbf{x}^5) &= \mathrm{sgn}(-1 \times (0.5 - 0.15)) = -1 \\
\mathrm{h}_3(\mathbf{x}^6) &= \mathrm{sgn}(-1 \times (0.65 - 0.15)) = -1
\end{aligned}$$

Combinaison du classement selon l'équation d'AdaBoost ce fait selon l'équation suivante :

$$\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} \left( \log \frac{1}{\beta_j} \right) \mathrm{h}_j(\mathbf{x})$$

Calcul des $\log(1/\beta_j)$ :

$$\begin{aligned}
\log \frac{1}{\beta_1} &= \log \frac{1}{0.5} = 0.6931 \\
\log \frac{1}{\beta_2} &= \log \frac{1}{0.333} = 1.0986 \\
\log \frac{1}{\beta_3} &= \log \frac{1}{0.5} = 0.6931
\end{aligned}$$

Résultat de la combinaison :

$$\begin{aligned}
\bar{\mathrm{h}}(\mathbf{x}^1) &= (0.6931 \times -1) + (1.0986 \times 1) + (0.6931 \times -1) \\
&= -0.6931 + 1.0986 - 0.6931 = -0.2876 \\
&\Rightarrow \mathbf{x}^1 \in C^1 \\
\bar{\mathrm{h}}(\mathbf{x}^2) &= (0.6931 \times -1) + (1.0986 \times -1) + (0.6931 \times -1) \\
&= -0.6931 - 1.0986 - 0.6931 = -2.4848 \\
&\Rightarrow \mathbf{x}^2 \in C^1 \\
\bar{\mathrm{h}}(\mathbf{x}^3) &= (0.6931 \times 1) + (1.0986 \times -1) + (0.6931 \times -1) \\
&= 0.6931 - 1.0986 - 0.6931 = -1.0986 \\
&\Rightarrow \mathbf{x}^3 \in C^1 \\
\bar{\mathrm{h}}(\mathbf{x}^4) &= (0.6931 \times -1) + (1.0986 \times 1) + (0.6931 \times 1) \\
&= -0.6931 + 1.0986 + 0.6931 = 1.0986 \\
&\Rightarrow \mathbf{x}^4 \in C^2 \\
\bar{\mathrm{h}}(\mathbf{x}^5) &= (0.6931 \times 1) + (1.0986 \times -1) + (0.6931 \times -1) \\
&= 0.6931 - 1.0986 - 0.6931 = -1.0986 \\
&\Rightarrow \mathbf{x}^5 \in C^1 \\
\bar{\mathrm{h}}(\mathbf{x}^6) &= (0.6931 \times 1) + (1.0986 \times 1) + (0.6931 \times -1) \\
&= 0.6931 + 1.0986 - 0.6931 = 1.0986 \\
&\Rightarrow \mathbf{x}^6 \in C^2
\end{aligned}$$

Le classifieur fait une erreur ($\mathbf{x}^5$), pour un taux d'erreur de classement de $0.167$.

(d) (5pts) Supposons finalement que l'on veut utiliser une mixture d'experts pour classifier les données en utilisant les trois classifieurs précédents (SVM, perceptron multi-couches, AdaBoost avec souches de décision), en ayant une mesure de l'expertise pour chaque données du jeu $\mathcal{X}$, pour chacun des classifieurs ($\mathbf{w}(\mathbf{x}) = [w_{SVM}(\mathbf{x})\ w_{PMC}(\mathbf{x})\ w_{Ada}(\mathbf{x})]^T$).

$$\begin{array}{lll}
\mathbf{w}(\mathbf{x}^1) = [0.8\ 0.5\ 0.6]^T, & \mathbf{w}(\mathbf{x}^2) = [0.9\ 0.5\ 0.7]^T, & \mathbf{w}(\mathbf{x}^3) = [0.5\ 0.3\ 0.65]^T \\
\mathbf{w}(\mathbf{x}^4) = [0.65\ 0.3\ 0.6]^T, & \mathbf{w}(\mathbf{x}^5) = [0.7\ 0.4\ 0.7]^T, & \mathbf{w}(\mathbf{x}^6) = [0.65\ 0.45\ 0.7]^T
\end{array}$$

Considérez que les décisions obtenues par les trois classifieurs ont été binarisées au préalable, c'est-à-dire que $\mathrm{h}_{SVM}(\mathbf{x}) \in \{-1, 1\}$, $\mathrm{h}_{PMC}(\mathbf{x}) \in \{-1, 1\}$ et $\mathrm{h}_{Ada}(\mathbf{x}) \in \{-1, 1\}$. Effectuez le classement de ces données de $\mathcal{X}$ selon cette mixture d'experts et donnez le taux d'erreur de classement correspondant.

**Solution :** Le classement par la mixture d'expert se ferait par l'équation suivante.

$$\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} \mathbf{w}_j(\mathbf{x}) \mathrm{h}_j(\mathbf{x})$$

Dans le cas présent, les résultats seraient :

$$\begin{aligned}
\bar{\mathrm{h}}(\mathbf{x}^1) &= (w_{SVM}(\mathbf{x}^1)\mathrm{h}_{SVM}(\mathbf{x}^1)) + (w_{PMC}(\mathbf{x}^1)\mathrm{h}_{PMC}(\mathbf{x}^1)) + (w_{Ada}(\mathbf{x}^1)\mathrm{h}_{Ada}(\mathbf{x}^1)) \\
&= (0.8 \times -1) + (0.5 \times -1) + (0.6 \times -1) = -0.8 - 0.5 - 0.6 = -1.9 \\
&\Rightarrow \mathbf{x}^5 \in C_1 \\
\bar{\mathrm{h}}(\mathbf{x}^2) &= (w_{SVM}(\mathbf{x}^2)\mathrm{h}_{SVM}(\mathbf{x}^2)) + (w_{PMC}(\mathbf{x}^2)\mathrm{h}_{PMC}(\mathbf{x}^2)) + (w_{Ada}(\mathbf{x}^2)\mathrm{h}_{Ada}(\mathbf{x}^2)) \\
&= (0.9 \times -1) + (0.5 \times -1) + (0.7 \times -1) = -0.9 - 0.5 - 0.7 = -2.1 \\
&\Rightarrow \mathbf{x}^5 \in C_1 \\
\bar{\mathrm{h}}(\mathbf{x}^3) &= (w_{SVM}(\mathbf{x}^3)\mathrm{h}_{SVM}(\mathbf{x}^3)) + (w_{PMC}(\mathbf{x}^3)\mathrm{h}_{PMC}(\mathbf{x}^3)) + (w_{Ada}(\mathbf{x}^3)\mathrm{h}_{Ada}(\mathbf{x}^3)) \\
&= (0.5 \times -1) + (0.3 \times 1) + (0.65 \times -1) = -0.5 + 0.3 - 0.65 = 0.85 \\
&\Rightarrow \mathbf{x}^5 \in C_2 \\
\bar{\mathrm{h}}(\mathbf{x}^4) &= (w_{SVM}(\mathbf{x}^4)\mathrm{h}_{SVM}(\mathbf{x}^4)) + (w_{PMC}(\mathbf{x}^4)\mathrm{h}_{PMC}(\mathbf{x}^4)) + (w_{Ada}(\mathbf{x}^4)\mathrm{h}_{Ada}(\mathbf{x}^4)) \\
&= (0.65 \times -1) + (0.3 \times -1) + (0.6 \times 1) = -0.65 - 0.3 + 0.6 = -0.35 \\
&\Rightarrow \mathbf{x}^5 \in C_1 \\
\bar{\mathrm{h}}(\mathbf{x}^5) &= (w_{SVM}(\mathbf{x}^5)\mathrm{h}_{SVM}(\mathbf{x}^5)) + (w_{PMC}(\mathbf{x}^5)\mathrm{h}_{PMC}(\mathbf{x}^5)) + (w_{Ada}(\mathbf{x}^5)\mathrm{h}_{Ada}(\mathbf{x}^5)) \\
&= (0.7 \times -1) + (0.4 \times 1) + (0.7 \times -1) = -0.7 + 0.4 - 0.7 = -1 \\
&\Rightarrow \mathbf{x}^5 \in C_1 \\
\bar{\mathrm{h}}(\mathbf{x}^6) &= (w_{SVM}(\mathbf{x}^6)\mathrm{h}_{SVM}(\mathbf{x}^6)) + (w_{PMC}(\mathbf{x}^6)\mathrm{h}_{PMC}(\mathbf{x}^6)) + (w_{Ada}(\mathbf{x}^6)\mathrm{h}_{Ada}(\mathbf{x}^6)) \\
&= (0.65 \times 1) + (0.45 \times 1) + (0.7 \times 1) = 0.65 + 0.45 + 0.7 = 1.8 \\
&\Rightarrow \mathbf{x}^5 \in C_2
\end{aligned}$$

La mixture d'experts fait donc deux erreurs ($\mathbf{x}^4$ et $\mathbf{x}^5$), pour un taux d'erreurs de classement de $0.333$.

4. Soit l'algorithme d'apprentissage AdaBoost, présenté dans le pseudo-code suivant.

> 1. Initialiser les probabilités de chaque données, $p_1^t = 1/N,\ t = 1, \ldots, N$
> 2. Pour chaque classifieur de base $j = 1, \ldots, L$ :
>  2.1. Échantillonner jeu $\mathcal{X}_j$ à partir de $\mathcal{X}$ selon probabilités $p_j^t$
>  2.2. Entraîner classifieur $\mathrm{h}_j$ avec jeu $\mathcal{X}_j$
>  2.3. Calculer l'erreur du classifieur, $\epsilon_j = \sum_t p_j^t \ell_{0-1}(r^t, \mathrm{h}_j(\mathbf{x}^t))$
>  2.4. *Étape mystère*
>  2.5. Calculer $\beta_j = \frac{\epsilon_j}{1-\epsilon_j}$
>  2.6. Calculer les nouvelles probabilités $p_{j+1}^t$
>
> $$p_{j+1}^t = \frac{q_j^t}{\sum_s q_j^s}, \quad q_j^t = \begin{cases} \beta_j p_j^t & \text{si } \mathrm{h}_j(\mathbf{x}^t) = r^t \\ p_j^t & \text{autrement} \end{cases}, \quad t = 1, \ldots, N$$

(a) (4pts) Donnez l'énoncé de l'étape 2.4 (étape mystère) de l'algorithme AdaBoost et expliquez pourquoi cette étape est nécessaire.

**Solution :** L'énoncé de l'étape 2.4 est :

Si erreur $\epsilon_j > 0.5$, alors $L = j - 1$ et arrêter l'algorithme

Cette vérification du taux d'erreur est nécessaire pour s'assurer que la condition d'une taux d'erreur inférieur ou égale à $50\%$ des *weak classifiers* soit respectée. Cette condition nous assure que l'algorithme ne va pas diverger par l'ajout de classifieurs performants moins bien que des classifieurs aléatoires.

(b) (3pts) Expliquez de quelle façon les probabilités $p_j^t$ varient dans le temps avec AdaBoost et quelle est l'influence de ces probabilités sur la génération des classifieurs formant l'ensemble.

**Solution :** Les probabilités $p_j^t$ tendent à augmenter à chaque itération pour les données qui sont considérées comme étant difficiles (données mal classées par les classifieurs) et à baisser pour les données plus faciles (bien classées par les classifieurs générés par AdaBoost).

Les probabilités $p_j^t$ servent à former les jeux d'entraînement de classifieurs entraînés à chaque itération, de sorte que les données ayant des probabilités associées élevées ont de plus forte chance de faire parti (en plusieurs copies) du jeu d'entraînement. De cette façon, l'entraînement des classifieurs formés à chaque itération est biaisé pour apprendre à bien classer les données difficiles.

(c) (3pts) Expliquez le lien qu'il y a entre la maximisation des marges géométriques faite avec les SVM et la maximisation des marges de classement effectuée par AdaBoost.

**Solution :** Avec les SVM, on tend à maximiser la distance entre l'hyper-plan séparateur et les données près de cet hyper-plan, en maximisant la distance géométrique qu'il y a entre ces données et l'hyper-plan (marge géométrique). Donc, l'apprentissage est fait en se basant sur les données les plus près de l'hyper-plan séparateur, donc d'un point de vue les données les plus ambigüe ou difficile.

Avec AdaBoost, l'apprentissage des classifieurs se fait en insistant sur les données difficiles, un peu comme des vecteurs de support. Les données faciles ont une probabilité $p_j$ associée, qui tend vers zéro, similairement au $\alpha^t$ des données n'étant pas des vecteurs de support avec les SVM. De plus, l'apprentissage avec AdaBoost tends à séparer les données afin de prendre une décision selon l'équation suivante.

$$\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} \log\left(\frac{1}{\beta_j}\right) \mathrm{h}_j(\mathbf{x})$$

De sorte que lorsque le nombre de classifieurs générés avec AdaBoost est très élevé ($L \to \infty$), on s'attend à ce que $\bar{\mathrm{h}}(\mathbf{x}) \to -\infty$ pour des données de la première classe et $\bar{\mathrm{h}}(\mathbf{x}) \to \infty$ pour la deuxième. On fait donc une maximisation de la marge de classement $\bar{\mathrm{h}}(\mathbf{x} \in C^2) - \bar{\mathrm{h}}(\mathbf{x} \in C^1)$ pour les données des deux classes.

5. Répondez aussi brièvement et clairement que possible aux questions suivantes.

(a) (2pts) Expliquez pourquoi l'entraînement d'un discriminant linéaire par descente du gradient en utilisant le critère d'erreur du perceptron est peu recommandé comparativement à d'autres approches existantes.

**Solution :** L'entraînement d'un discriminant linéaire selon le critère du perceptron est peu recommandé car la mesure de l'erreur est nulle pour tout placement séparant parfaitement un certain jeu de données, de sorte qu'à partir du moment où l'algorithme a atteint une solution satisfaisante, aucune amélioration est possible. De cette façon, si l'algorithme converge vers un placement d'hyper-plan risqué, c'est-à-dire près de certaines données du jeu d'entraînement, l'algorithme risque de faire des erreurs de classement. De plus, pour des données non-linéairement séparable, l'algorithme ne converge généralement pas et ne sera donc pas d'une grande utilité.

(b) (2pts) Expliquez brièvement les approches *un contre tous* et l'approche *séparation par paires* pour l'apprentissage de données comportant plusieurs classes, lorsqu'on veut utiliser des classifieurs à deux classes tels les discriminants linéaires, SVM ou AdaBoost.

**Solution :** Avec l'approche *un contre tous*, on génère $K$ problèmes de classement, où $K$ est le nombre de classes du problème initial, en entraînant les classifieurs à deux classes dans chaque problème, afin de discriminer les données d'une classe des données des autres classes, avec une classe d'intérêt différent pour les $K$ problèmes. Si aucuns classifieur ou plus d'un classifieur signale qu'une donnée appartient à la classe qui lui est associé, il y a ambiguité ou rejet dans le classement. Autrement, une donnée évaluée est assignée au classifieur ayant indiqué qu'elle lui est associée.

Avec l'approche de *séparation par paires*, on défini $K(K-1)/2$ problèmes de classement, où on traite toutes les combinaisons distinctes de deux classe comportant. Pour chaque combinaison de deux classe, on entraîne un classifieur. On détermine le classement selon les réponses obtenues, en présumant que pour une certaine donnée, tous les classifieurs ayant été entraînés sur la classe correspondant à la donnée vont correctement la classer.

(c) (2pts) Dans l'utilisation de codes à correction d'erreur pour faire du classement avec des ensembles, expliquez brièvement pourquoi on veut maximiser la distance de Hamming (nombre de valeurs différentes) entre les codes de décision de chaque classe.

**Solution :** La moitié de la distance de Hamming minimale moins un ($\lfloor (d-1)/2 \rfloor$) entre deux codes de décision nous donne le nombre d'erreurs qui peut être fait dans la décision avant que la décision d'ensemble soit erronée. Donc, en maximizant la distance de Hamming, on augmente la robustesse de l'ensemble par rapport aux erreurs faites par les classifieurs le composant.

(d) (2pts) Développez l'équation démontrant que la variance globale d'un ensemble est inférieure à la variance des classifieurs individuels la composant, lorsqu'on fait l'hypothèse que les décisions des classifieurs individuels sont statistiquement indépendantes.

**Solution :**

$$\mathrm{Var}(\bar{\mathrm{h}}) = \mathrm{Var}\left(\sum_{j=1}^{L} \frac{1}{L}\mathrm{h}_j\right) = \frac{1}{L^2} L \mathrm{Var}(\mathrm{h}_j) = \frac{1}{L}\mathrm{Var}(\mathrm{h}_j)$$

(e) (2pts) Lorsque le nombre de données d'entraînement est très grand, $N \to \infty$, on peut dire que les étiquettes de classe des plus proches voisins à une donnée $\mathbf{x}$ ont une probabilité $P(C_i|\mathbf{x})$ d'appartenir à la classe $C_i$, où $P(C_i|\mathbf{x})$ est la probabilité *a posteriori* de la véritable densité de probabilité des données. Donnez la probabilité de classer une donnée $\mathbf{x}$ comme appartenant à la classe $C_i$ en fonction de $P(C_i|\mathbf{x})$ lorsqu'on utilise l'algorithme des $k$-plus proches voisins, en utilisant $k = 3$ voisins et que $N \to \infty$.

**Solution :** Il y a quatre possibilités d'échantillonner les trois plus proches voisins d'une donnée d'une donnée $\mathbf{x}$ comme appartenant à la classe $C_i$.

— Probabilité $P(C_i|\mathbf{x}) \times P(C_i|\mathbf{x}) \times P(C_i|\mathbf{x}) = P(C_i|\mathbf{x})^3$ de tirer trois voisins de la classe $C_i$.

— Probabilité $P(C_i|\mathbf{x}) \times P(C_i|\mathbf{x}) \times (1 - P(C_i|\mathbf{x})) = P(C_i|\mathbf{x})^2(1 - P(C_i|\mathbf{x}))$ de tirer les deux premiers voisins de la classe $C_i$ et le troisième d'une autre classe.

— Probabilité $P(C_i|\mathbf{x}) \times (1 - P(C_i|\mathbf{x})) \times P(C_i|\mathbf{x}) = P(C_i|\mathbf{x})^2(1 - P(C_i|\mathbf{x}))$ de tirer le premier et le troisième voisin de la classe $C_i$ et le deuxième d'une autre classe.

— Probabilité $(1 - P(C_i|\mathbf{x})) \times P(C_i|\mathbf{x}) \times P(C_i|\mathbf{x}) = P(C_i|\mathbf{x})^2(1 - P(C_i|\mathbf{x}))$ de tirer les deux derniers voisins de la classe $C_i$ et le premier d'une autre classe.

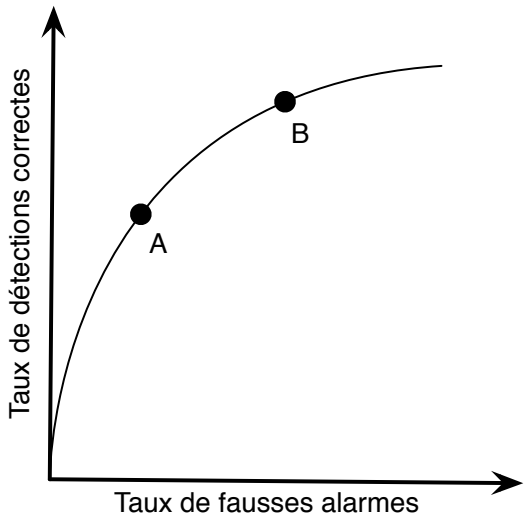
En conséquence, la probabilité de classer la donnée $\mathbf{x}$ comme étant de la classe $C_i$ est de :

$$P(C_i|\mathbf{x})^3 + 3P(C_i|\mathbf{x})^2(1 - P(C_i|\mathbf{x})) = 3P(C_i|\mathbf{x})^2 - 2P(C_i|\mathbf{x})^3$$

(f) (2pts) Combien d'entraînements distincts de classifieur sont fait lorsque l'on veut en évaluer les performances avec la validation croisée $5 \times 2$ ?

**Solution :** Dix entraînements de classifieur sont nécessaires, car on va faire deux entraînements sur les deux partitions générés pour chacune des cinq répétitions.

(g) (2pts) Soit la courbe ROC (*receiver operator characteristics*) présentée ci-bas, décrivant les performance pour la détection d'intrusions de différentes paramétrisation de classifieurs à deux classes.




Expliquez brièvement l'effet occasionné sur les performances du système (intrusions manquées, fausses détections) par le passage de la paramétrisation correspondant point opératoire A sur la courbe ROC à la paramétrisation correspondant au point opératoire B.

**Solution :** En passant du point opératoire A au point opératoire B, on devrait observer une augmentation du nombre de faux positifs (données détectées à tort comme étant des intrusions) et une diminution du nombre de faux négatifs (nombre d'intrusions non détectées).

(h) (2pts) Le test statistique de l'analyse de variance (ANOVA) peut être utilisé pour déterminer si plusieurs algorithmes de classement ont des performances statistiquement similaires ou non. Ce test fait deux estimations distinctes de $\sigma$, soit la variance sur les taux de classement obtenue sur tous les plis et tous les algorithmes testés. Indiquez en quoi diffèrent conceptuellement ces deux estimateurs de $\sigma$ utilisés pour l'ANOVA.

**Solution :** Le premier estimateur de $\sigma$ est calculé en supposant que l'hypothèse $H_0$ est vraie, alors que le deuxième estimateur ne fait pas cette hypothèse. L'hypothèse $H_0$ toutes les moyennes $u_j$ des taux de classement sur tous les plis de chacun des algorithmes de classement testés sont égaux.

$$H_0 : u_1 = u_2 = \cdots = u_L$$

(i) (2pts) Voici l'équation pour mettre à jour les poids $\mathbf{w}$ de discriminants logistiques.

$$\Delta w_j = \eta \sum_{t=1}^{N} (r^t - y^t) x_j^t$$

Indiquez si cette algorithme correspond à un apprentissage de type *batch* ou de type en-ligne, en justifiant brièvement votre réponse.

**Solution :** L'apprentissage est de type *batch* car les poids sont mis à jour en tenant compte de la correction occasionnée par tout le jeu d'entraînement, et non seulement une données.

(j) (2pts) L'algorithme des $K$-means en-ligne vise à minimiser l'erreur de reconstruction de $K$ groupes. Cette erreur est présentée dans l'équation suivante.

$$E(\{\mathbf{m}_i\}_{i=1}^K|\mathbf{x}^t) = \frac{1}{2}\sum_{i=1}^{K} b_i^t \|\mathbf{x}^t - \mathbf{m}_i\|^2$$

Développez l'équation de $\Delta m_{i,j}$ permettant de mettre à jour les valeurs des positions des centres $\mathbf{m}_i$ de chacun des $K$ groupes par descente du gradient.

**Solution :** La dérivée de l'erreur de reconstruction selon $m_{i,j}$ est :

$$\frac{\partial E^t}{\partial m_{i,j}} = \frac{\partial \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{D} b_i^t (x_j^t - m_{i,j})^2}{\partial m_{i,j}} = -b_i^t(x_j^t - m_{i,j})$$

En conséquence, on obtient l'équation de $\Delta m_{i,j}$ suivante.

$$\Delta m_{i,j} = -\eta \frac{\partial E^t}{\partial m_{i,j}} = \eta b_i^t (x_j^t - m_{i,j})$$

(k) (2pts) Dans un processus général de traitement de séquences avec états discrets, on exprime les probabilités de transition de l'état actuel vers l'état $S_i$ au temps $t$ par la suivante.

$$P(s_{t+1} = S_j | s_t = S_i, s_{t-1} = S_k, \ldots, s_1 = S_l)$$

Donnez l'expression simplifiée correspondant à cette probabilité de transition pour un processus de Markov discret.

**Solution :** Dans un processus de Markov, on ne tient compte que de l'état $S_i$ actuel, de sorte que la probabilité de transition est donnée par l'équation suivante.

$$P(s_{t+1} = S_j | s_t = S_i, s_{t-1} = S_k, \ldots, s_1 = S_l) = P(s_{t+1} = S_j | s_t = S_i)$$

(l) (2pts) Expliquez en quoi consiste précisemment le problème du décodage avec les modèles de Markov cachés.

**Solution :** Problème du décodage : étant donné un modéle $\lambda$ du modèle de Markov caché et une séquence d'observations $O$, quelle est la séquence d'état $S = \{s_1, s_2, \ldots, s_T\}$ qui a vraisemblablement générée $O$ ?

(m) (2pts) L'algorithme Baum-Welch permet de faire un apprentissage du modèle $\lambda$ d'un modèle de Markov caché à partir de plusieurs séquences d'observation. Il a été mentionné que l'algorithme Baum-Welch est une forme spécifique de l'algorithme EM. De ce point de vue, indiquez à quoi correspondent l'étape E et l'étape M avec l'algorithme Baum-Welch.

**Solution :**

— Étape E : estimer les probabilités d'être dans un certain état ($\gamma_t(i)$) et de faire une certaine transition ($\xi_t(i,j)$) à partir d'un modèle ($\lambda$).

— Étape M : estimer la paramétrisation du modèle ($\lambda$) à partir des probabilités d'être dans un état ($\gamma_t(i)$) et de prendre une transition ($\xi_t(i,j)$).

(n) (2pts) Expliquez brièvement ce que représente la probabilité $\alpha_t(i)$ dans la procédure avant, utilisée pour évaluer la probabilité $P(O|\lambda)$, avec un modèle de Markov caché défini par le modèle $\lambda$, d'avoir une séquence d'observations $O = \{o_1, o_2, \ldots, o_T\}$.

**Solution :** La variable $\alpha_t(i)$ correspond à la probabilité d'observer la séquence partielle $\{o_1, o_2, \ldots, o_t\}$ jusqu'au temps $t \leq T$.

(o) (2pts) Expliquez la tâche générale (haut niveau) que l'on veut effectuer avec l'apprentissage par renforcement.

**Solution :** Avec l'apprentissage par renforcement, on veut inférer une stratégie pour contrôler un agent opérant dans un environnement, afin de sélectionner les actions permettant de maximiser une forme de récompense à long terme.

---

*JOYEUSES FETES*

18 décembre 2009
CG