Apprentissage et reconnaissance (GIF-4101 / GIF-7005)
Département de génie électrique et de génie informatique
Automne 2017

UNIVERSITÉ LAVAL

EXAMEN FINAL

Instructions : – Une feuille aide-mémoire recto verso manuscrite est permise ;
– Durée de l'examen : 2 h 50.

Pondération : – Cet examen compte pour 35 % de la note finale.
– La note est saturée à 100 % si le total des points avec bonus excède cette valeur.

Question 1 (24 points sur 100)

Supposons que l'on veut entraîner un autoencodeur à deux couches, la première couche comme encodeur et l'autre comme décodeur, toutes deux entraînées par rétropropagation des erreurs. Les poids de ces deux couches sont distincts pour les besoins de l'entraînement.
Dans ce réseau, une fonction de transfert linéaire est utilisée ($\mathrm{f}_{\text{lin}}(x) = x$). La sortie d'un neurone de la couche d'encodage se modélise comme suit :

$$z_i^t = (\mathbf{w}_i^{\text{enc}})^\top \mathbf{x}^t + w_{i,0}^{\text{enc}}, \quad i = 1, \ldots, K.$$

La sortie de la couche de décodage se modélise comme suit, ce qui correspond à la donnée d'entrée reconstruite :

$$\hat{x}_j^t = (\mathbf{w}_j^{\text{dec}})^\top \mathbf{z}^t + w_{j,0}^{\text{dec}}, \quad j = 1, \ldots, D.$$

Le critère de performance utilisé est l'erreur quadratique moyenne de reconstruction :

$$e_j^t = (x_j^t - \hat{x}_j^t), \qquad E_{\text{rec}}^t = \frac{1}{2}\|\mathbf{x}^t - \hat{\mathbf{x}}^t\|^2 = \frac{1}{2}\sum_{j=1}^{K}(e_j^t)^2, \qquad E_{\text{rec}} = \frac{1}{N}\sum_{t=1}^{N} E_{\text{rec}}^t.$$

Répondez aux questions suivantes sur l'entraînement d'un tel réseau de neurones.

(12) (a) Détaillez les équations pour mettre à jour les poids des neurones de la couche du décodeur, par descente du gradient et en utilisant l'erreur de reconstruction.

Solution: La descente du gradient requiert de calculer les dérivées partielles de la fonction d'erreur, ici l'erreur de reconstruction, selon les paramètres optimisés, ici $\mathbf{w}_j^{\text{dec}}$ et $w_{j,0}^{\text{dec}}$, $j = 1, \ldots, D$. Développons d'abord les dérivées partielles par chaînage des dérivées :

$$\frac{\partial E_{\text{rec}}^t}{\partial w_{j,i}^{\text{dec}}} = \frac{\partial E_{rec}^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial \hat{x}_j^t}\frac{\partial \hat{x}_j^t}{\partial w_{j,i}^{\text{dec}}},$$
$$\frac{\partial E_{\text{rec}}^t}{\partial w_{j,0}^{\text{dec}}} = \frac{\partial E_{rec}^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial \hat{x}_j^t}\frac{\partial \hat{x}_j^t}{\partial w_{j,0}^{\text{dec}}}.$$

Pour l'erreur, les dérivées sont :

$$\frac{\partial E^t_{\text{rec}}}{\partial e^t_j} = \frac{\partial}{\partial e^t_j}\frac{1}{2}\sum_{l=1}^{D}(e^t_l)^2 = e^t_j,$$
$$\frac{\partial e^t_j}{\partial \hat{x}^t_j} = \frac{\partial}{\partial \hat{x}^t_j}(x^t_j - \hat{x}^t_j) = -1.$$

Pour les poids de la couche de décodage, les dérivées sont :

$$\frac{\partial \hat{x}^t_j}{\partial w^{\text{dec}}_{j,i}} = \frac{\partial}{\partial w^{\text{dec}}_{j,i}}\sum_{l=1}^{K} w^{\text{dec}}_{j,l} z^t_l + w^{\text{dec}}_{j,0} = z^t_i,$$
$$\frac{\partial \hat{x}^t_j}{\partial w^{\text{dec}}_{j,0}} = \frac{\partial}{\partial w^{\text{dec}}_{j,0}}\sum_{l=1}^{K} w^{\text{dec}}_{j,l} z^t_l + w^{\text{dec}}_{j,0} = 1.$$

En intégrant tout cela, on obtient :

$$\frac{\partial E^t_{\text{rec}}}{\partial w^{\text{dec}}_{j,i}} = -e^t_j\, z^t_i, \qquad \frac{\partial E^t_{\text{rec}}}{\partial w^{\text{dec}}_{j,0}} = -e^t_j.$$

La règle d'apprentissage selon la descente du gradient consiste donc à appliquer itérativement les mises à jour suivantes :

$$\Delta w^{\text{dec}}_{j,i} = -\eta\frac{\partial E_{\text{rec}}}{\partial w^{\text{dec}}_{j,i}} = \frac{\eta}{N}\sum_{t=1}^{N} e^t_j\, z^t_i, \quad i = 1,\ldots,K, \quad j = 1,\ldots,D,$$
$$\Delta w^{\text{dec}}_{j,0} = -\eta\frac{\partial E_{\text{rec}}}{\partial w^{\text{dec}}_{j,0}} = \frac{\eta}{N}\sum_{t=1}^{N} e^t_j, \quad j = 1,\ldots,D.$$

(12) (b) Détaillez maintenant les équations pour mettre à jour les poids des neurones de la couche de l'encodeur, toujours par descente du gradient et en utilisant l'erreur de reconstruction.

Solution: Développons maintenant les dérivées partielles par chaînage des dérivées pour $\mathbf{w}^{\text{enc}}_j$ et $w^{\text{enc}}_{j,0}$, $j = 1,\ldots,K$:

$$\frac{\partial E^t_{\text{rec}}}{\partial w^{\text{enc}}_{j,i}} = \frac{\partial E^t_{rec}}{\partial z^t_j}\frac{\partial z^t_j}{\partial w^{\text{enc}}_{j,i}},$$
$$\frac{\partial E^t_{\text{rec}}}{\partial w^{\text{enc}}_{j,0}} = \frac{\partial E^t_{rec}}{\partial z^t_j}\frac{\partial z^t_j}{\partial w^{\text{enc}}_{j,0}}.$$

Pour l'erreur, les dérivées sont :

$$\frac{\partial E^t_{\text{rec}}}{\partial z^t_j} = \frac{\partial}{\partial z^t_j}\frac{1}{2}\sum_{k=1}^{D}(e^t_k)^2 = \sum_{k=1}^{D} e^t_k \frac{\partial e^t_k}{\partial z^t_j} = \sum_{k=1}^{D} e^t_k \frac{\partial e^t_k}{\partial \hat{x}^t_k}\frac{\partial \hat{x}^t_k}{\partial z^t_j},$$
$$\frac{\partial \hat{x}^t_k}{\partial z^t_j} = \frac{\partial}{\partial z^t_j}\sum_{l=1}^{K} w^{\text{dec}}_{k,l} z^t_l + w^{\text{dec}}_{k,0} = w^{\text{dec}}_{k,j}.$$

Pour les poids de la couche d'encodage :

$$\frac{\partial z^t_j}{\partial w^{\text{enc}}_{j,i}} = \frac{\partial}{\partial w^{\text{enc}}_{j,i}}\sum_{l=1}^{K} w^{\text{enc}}_{j,l} x^t_l + w^{\text{enc}}_{j,0} = x^t_i, \qquad \frac{\partial z^t_j}{\partial w^{\text{enc}}_{j,0}} = \frac{\partial}{\partial w^{\text{enc}}_{j,0}}\sum_{l=1}^{K} w^{\text{enc}}_{j,l} x^t_l + w^{\text{enc}}_{j,0} = 1.$$

En intégrant tout cela, on obtient :

$$\frac{\partial E^t_{\text{rec}}}{\partial w^{\text{enc}}_{j,i}} = -x^t_i \sum_{k=1}^{D} w^{\text{dec}}_{k,j} e^t_k, \qquad \frac{\partial E^t_{\text{rec}}}{\partial w^{\text{enc}}_{j,0}} = -\sum_{k=1}^{D} w^{\text{dec}}_{k,j} e^t_k.$$

La règle d'apprentissage selon la descente du gradient consiste donc à appliquer itérativement les mises à jour suivantes :

$$\Delta w^{\text{enc}}_{j,i} = -\eta\frac{\partial E_{\text{rec}}}{\partial w^{\text{enc}}_{j,i}} = \frac{\eta}{N}\sum_{t=1}^{N} x^t_i \sum_{k=1}^{D} w^{\text{dec}}_{k,j} e^t_k, \quad i = 1,\ldots,D, \quad j = 1,\ldots,K,$$
$$\Delta w^{\text{enc}}_{j,0} = -\eta\frac{\partial E_{\text{rec}}}{\partial w^{\text{enc}}_{j,0}} = \frac{\eta}{N}\sum_{t=1}^{N} \sum_{k=1}^{D} w^{\text{dec}}_{k,j} e^t_k, \quad j = 1,\ldots,K.$$

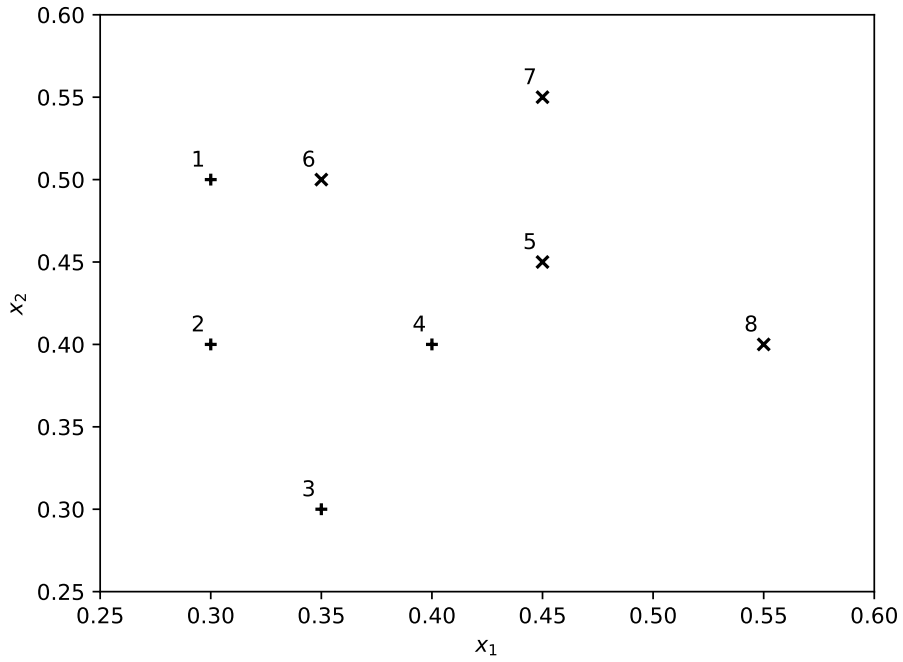
Question 2 (22 points sur 100)

Soit le jeu de données suivant, en deux dimensions :

$$\mathbf{x}^1 = [0{,}3\ 0{,}5]^\top, \quad \mathbf{x}^2 = [0{,}3\ 0{,}4]^\top, \quad \mathbf{x}^3 = [0{,}35\ 0{,}3]^\top, \quad \mathbf{x}^4 = [0{,}4\ 0{,}4]^\top,$$
$$\mathbf{x}^5 = [0{,}45\ 0{,}45]^\top, \quad \mathbf{x}^6 = [0{,}35\ 0{,}5]^\top, \quad \mathbf{x}^7 = [0{,}45\ 0{,}55]^\top, \quad \mathbf{x}^8 = [0{,}55\ 0{,}4]^\top.$$

Les étiquettes de ces données sont $r^1 = r^2 = r^3 = r^4 = -1$ et $r^5 = r^6 = r^7 = r^8 = 1$.

Le graphique ici bas présente le tracé de ces données.



Nous obtenons le résultat suivant en effectuant l'entraînement d'un SVM linéaire à **marge douce** avec ces données, en utilisant comme valeur de paramètre de régularisation $C = 200$:

$$\alpha^1 = 180, \quad \alpha^2 = 0, \quad \alpha^3 = 0, \quad \alpha^4 = 200, \quad \alpha^5 = 180, \quad \alpha^6 = 200, \quad \alpha^7 = 0, \quad \alpha^8 = 0,$$
$$w_0 = -11{,}6.$$

(5) (a) Calculez les valeurs du vecteur $\mathbf{w}$ de l'hyperplan séparateur de ce classifieur.

Solution: Les valeurs du vecteur $\mathbf{w}$ sont calculées selon l'équation suivante :

$$\mathbf{w} = \sum_t \alpha^t \, r^t \, \mathbf{x}^t.$$

Dans le cas présent, les valeurs du vecteur sont $\mathbf{w} = [17 \; 11]^\top$.

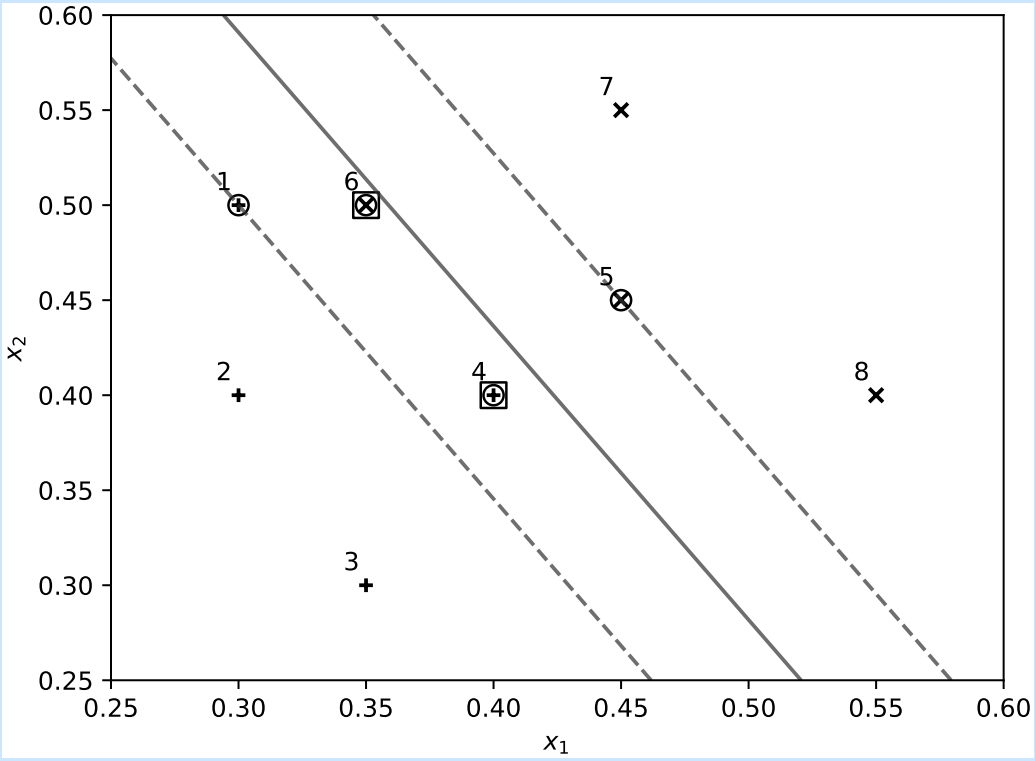
(12) (b) Déterminez les données qui sont des vecteurs de support ainsi que les données qui sont dans la marge ou mal classées. Tracez ensuite un graphique représentant toutes les données du jeu, en encerclant les vecteurs de support et en encadrant les données dans la marge ou mal classées. Tracez également la droite représentant l'hyperplan séparateur ainsi que deux droites pointillées représentant les limites de la marge.

N'utilisez pas le graphique du préambule de l'énoncé de la question pour donner votre réponse, tracez vous-même un nouveau graphique dans votre cahier de réponse.

Solution: Les données $\mathbf{x}^1$, $\mathbf{x}^4$, $\mathbf{x}^5$ et $\mathbf{x}^6$ représentent les vecteurs de support du classifieur, comme leur α^t respectif est non nul.

Les données dans la marge ou mal classées ont une valeur de α^t correspondant au paramètre de régularisation C. Donc, les données $\mathbf{x}^4$ et $\mathbf{x}^6$ sont dans la marge ou mal classées, avec $\alpha^4 = \alpha^6 = C = 200$.

Le graphique demandé correspond à ce qui suit.



(5) (c) Supposons maintenant que l'on veut traiter une donnée $\mathbf{x} = [0{,}37\ 0{,}45]^\top$ avec ce SVM. Calculez la valeur $\mathrm{h}(\mathbf{x})$ correspondante (valeur réelle avant seuillage de la sortie).

Solution: On calcule la valeur de $\mathrm{h}(\mathbf{x})$ selon l'équation suivante :

$$\mathrm{h}(\mathbf{x}) = \sum_t \alpha^t r^t (\mathbf{x}^t)^\top \mathbf{x} + w_0.$$

Dans le cas présent, avec $\mathbf{x} = [0{,}37\ 0{,}45]^\top$, la sortie correspondante du classifieur est $\mathrm{h}(\mathbf{x}) = -0{,}36$. Donc, la donnée est assignée à la classe des données négatives ($r = -1$).

Question 3 (15 points sur 100)

Une matrice de décision $\mathbf{W}$, de taille $K \times L$, permet de combiner les décisions d'un ensemble de L classifieurs à deux classes, pour faire du classement de données à K classes. L'équation de décision basée sur cette matrice est la suivante :

$$\bar{\mathrm{h}}_i(\mathbf{x}) = \sum_{j=1}^{L} w_{i,j} \mathrm{h}_{j,i}(\mathbf{x}),$$

où :

— $\mathrm{h}_{j,i}(\mathbf{x})$ est le j-ème classifieur de base de l'ensemble ;

— $w_{i,j}$ est l'élément à la position (i,j) dans la matrice de décision $\mathbf{W}$;
— $\bar{\mathrm{h}}_i(\mathbf{x})$ est la décision combinée de l'ensemble pour la classe C_i.

(5) (a) Supposons que l'on veut résoudre un problème à $K = 4$ classes à l'aide d'un ensemble de classifieurs à deux classes combinés selon la méthode *un contre tous* (en anglais, *one against all*). Donnez le nombre de classifieurs à deux classes à utiliser ainsi que la matrice de décision $\mathbf{W}$ correspondant à cette configuration.

Solution: Avec un ensemble de classifieurs de type *un contre tous*, le nombre de classifieurs à utiliser correspond au nombre de classes traitées, soit $L = K = 4$ classifieurs. La matrice de décision à utiliser pour cette configuration est la suivante :

$$\mathbf{W} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}.$$

(5) (b) Supposons maintenant que l'on veut résoudre ce problème à $K = 4$ classes toujours à l'aide d'un ensemble de classifieurs à deux classes, mais cette fois en combinant les classifieurs selon la méthode de *séparation par paires* (en anglais, *pairwise separation*). Donnez le nombre de classifieurs à deux classes à utiliser ainsi que la matrice de décision $\mathbf{W}$ correspondant à cette configuration.

Solution: Le nombre de classifieurs à utiliser correspond au nombre de combinaisons de classes possibles, en tenant compte de la symétrie entre les classifieurs (discriminer C_i de C_j correspondant à discriminer C_j de C_i à un signe près), soit $L = K(K-1)/2 = 6$ classifieurs. La matrice de décision correspondante est la suivante :

$$\mathbf{W} = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}.$$

(5) (c) Finalement, supposons que l'on veut résoudre ce problème à $K = 4$ classes d'un ensemble redondant de $L = 9$ classifieurs, avec une matrice de décision basée sur un code à correction d'erreur (en anglais, *error code output correction*). Donnez la matrice de décision $\mathbf{W}$ correspondant à cette configuration. Déterminez également le nombre d'erreurs de classement des classifieurs de base que cette configuration de système peut tolérer sans se tromper.

Solution: Une configuration possible de matrice de décision à code de correction d'erreur est la suivante :

$$\mathbf{W} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 \end{bmatrix}.$$

La distance de Hamming minimale entre chaque combinaison de lignes est de 5. Ceci indique donc que cette configuration de système peut tolérer deux erreurs de classement ($\lfloor (5-1)/2 \rfloor = 2$) par les classifieurs de base sans prendre une mauvaise décision de classement.

Question 4 (39 points sur 100)

Répondez aussi brièvement et clairement que possible aux questions suivantes.

(3) (a) Expliquez pourquoi un SVM classique ne peut traiter que deux classes.

Solution: Un SVM est un discriminant où l'assignation à une classe se fait selon le signe de la fonction $\mathrm{h}(\mathbf{x})$. Gérer plusieurs classes implique d'avoir plusieurs SVM, par exemple en utilisant une approche de type une contre tous, K SVM seront nécessaires pour gérer K classes, soit un SVM associé à chaque classe.

(3) (b) Expliquez la relation entre la fonction sigmoïde et les probabilités de classement de modèles paramétriques basés sur une loi normale.

Solution: Si la matrice de covariance est partagée entre les différentes densités de chaque classe et que les probabilités *a priori* sont égales, la probabilité *a posteriori* d'une classe ($P(C_i|\mathbf{x})$) correspond à une fonction sigmoïde.

(3) (c) Expliquez en quoi consiste le « truc du noyau » avec les SVM et autres méthodes à noyau.

Solution: Le truc du noyau permet de traiter des fonctions noyau générant des espaces à haute dimensionnalité (possiblement infinie), sans travailler directement dans ces espaces.

(3) (d) Expliquez les principales similarités et différences entre une matrice de covariance et une matrice de Gram.

Solution:

Similarités matrices carrées et symétriques, rapportant des mesures entre deux éléments.

Différences matrice de Gram représente les similarités entre chaque donnée (taille $N \times N$) alors que la matrice de covariance représente les covariances entre les variables (taille $D \times D$).

(3) (e) Expliquez pourquoi une fonction seuil ne peut pas être utilisée comme fonction de transfert de neurones dans un perceptron multicouche entraîné par rétropropagation des erreurs.

Solution: La rétropropagation des erreurs exige d'évaluer les dérivées partielles de différents éléments du réseau. Le calcul des dérivées partielles n'est pas possible avec une fonction seuil.

(3) (f) Expliquez l'intérêt d'effectuer de la compositionnalité de modèles dans des approches telles que l'apprentissage profond.

Solution: Comme le langage, il faut composer des éléments simples pour définir un langage donnant un sens à des notions complexes, non précisées dans les éléments de base. De plus, exploiter la compositionnalité permet un gain exponentiel en puissance de représentation, ce qui est utile pour décrire le monde efficacement.

(3) (g) Expliquez en quoi le préentraînement non supervisé de réseau de neurones profonds était utile et nécessaire, avant l'émergence des techniques modernes d'apprentissage profond.

Solution: Le préentraînement non supervisé permettait d'initialiser les réseaux de neurones profonds dans des régions de l'espace des paramètres pertinents et dans des bassins d'attractions d'optimum intéressants, permettant de générer des solutions de bonne performance. Sans préentraînement, on observe une grande variabilité dans les solutions obtenues.

(3) (h) Présentez un avantage important observé avec l'apprentissage multitâches avec des réseaux de neurones profonds, outre le fait que cela permet d'apprendre des modèles capables de faire plusieurs tâches simultanément.

Solution: L'apprentissage multitâche permet de mieux apprendre des représentations, souvent capable de mieux généraliser que des représentations apprises pour une seule des tâches à effectuer.

(3) (i) Présentez les avantages principaux associés à l'utilisation de graphes computationnels dans des outils d'apprentissage profonds modernes tels que TensorFlow.

Solution: Les graphes computationnels permettent de calculer automatiquement les gradients analytiques de réseaux de neurones quelconques. De plus, leurs utilisation rend également possible d'optimiser les traitements sur l'architecture visée, comme les GPU.

(3) (j) Expliquez pourquoi un réseau de neurones profond nécessite de très grands jeux de données.

Solution: Un réseau de neurones profond comporte un très grand nombre de poids qui doivent être estimés à partir d'un nombre conséquent d'exemples. En effet, un réseau de neurones est une technique à biais faible et variance élevée, qui exige beaucoup de données pour atténuer cette variance globale.

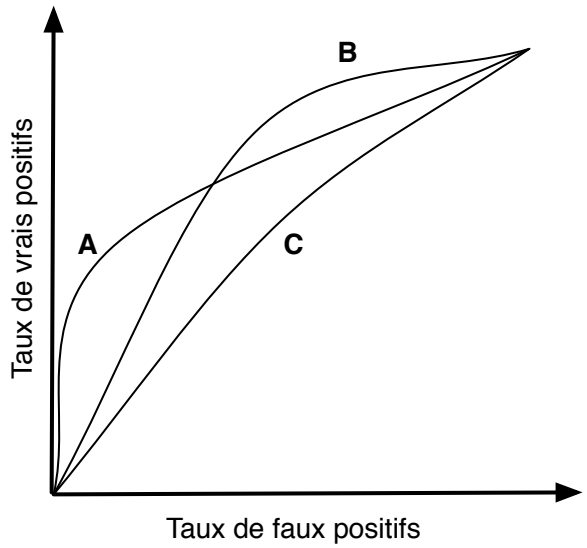
(3) (k) Expliquez pourquoi dit-on que les méthodes de *Bagging* permettent de générer de la diversité passivement, alors que les différentes approches de *Boosting* le font activement.

Solution: Avec le *Bagging*, la diversité provient de l'utilisation de classifieurs instables, entraînés sur différents jeux formés de données tirées aléatoirement du jeu d'origine. La méthode est passive, car cette diversité n'est pas guidée dans une direction ou une autre. Avec le *Boosting*, le choix des données d'entraînement est guidé par des probabilités ajustées selon l'historique actuel de données et leur classement par les modèles produits. Ce choix est donc fait activement, étant guidé par la performance des modèles à correctement traiter la donnée jusqu'à présent.

(3) (l) Expliquez pourquoi favorise-t-on des approches méthodologiques de type validation croisée à K-plis avec de petits jeux de données, alors qu'un partitionnement en jeux d'entraînement et de test semble suffisant avec de plus gros jeux.

Solution: Avec de petits jeux de données, chaque instance compte, autant en entraînement, où la performance est souvent proportionnelle à la taille de la partition d'entraînement, qu'en test, où de petits jeux de données génère de mauvais estimateurs (biais d'estimateurs avec peu d'échantillons). Lorsque le jeu de données est plus vaste, cet effet est moins perceptible, mais les ressources d'entraînement sont souvent plus limitées. Il faut alors éviter de faire trop d'entraînement coûteux en temps et en ressources.

(3) (m) Soit la courbe ROC suivante, présentant les performances de trois classifieurs (A, B et C) opérant selon deux classes (données positives et négatives).



Expliquez en termes clairs et généraux quels classifieurs seraient les plus intéressants à utiliser selon les circonstances rencontrées.

Solution: Le classifieur A offre un meilleur taux de reconnaissance lorsque l'on veut conserver le taux de fausses alarmes bas, quitte à manquer des données positives. À l'opposé, le classifieur B offre les meilleures performances lorsque l'on veut avoir un taux de décisions correctes (taux de vrais positifs) élevé, quitte à ce qu'il y ait régulièrement des fausses alarmes. Le classifieur C offre des performances inférieures en toute circonstance comparativement aux classifieurs A et B, et n'est donc pas intéressant à utiliser dans le cas présent.

Question 5 (10 points bonus)

Supposons que l'on doit choisir une méthode d'ajustement d'hyperparamètres parmi plusieurs techniques, pour un modèle d'apprentissage particulier (par exemple, paramètres C et σ d'un SVM à noyau gaussien). La comparaison expérimentale se fait sur une centaine de jeux de données standard (provenant de UCI et statlog). Présentez clairement et précisément une méthodologie expérimentale permettant de comparer les méthodes d'ajustement d'hyperparamètres et de déterminer celles offrant les meilleures performances. Prenez soin de proposer une méthode qui atténue autant que possible l'effet des facteurs de nuisance.

Solution: Différentes solutions sont possibles. Une approche possible consiste à partitionner chaque jeu de données en trois partitions, en faisant une dizaine de répétitions :

1. Pour chaque jeu de données, $i = 1, \ldots, L$:
   (a) Pour chaque répétition, $j = 1, \ldots, 10$:
      i. Partitionner aléatoirement le jeu en trois partitions : entraînement, validation et test.
      ii. Pour chaque technique d'optimisation d'hyperparamètres, $k = 1, \ldots, K$:
         A. Faire une recherche d'hyperparamètres avec la méthode k, en entraînant le modèle sur la partition d'entraînement du jeu de données i et sélectionnant les hyperparamètres selon la partition de validation.
         B. Réentraîner le modèle avec les meilleurs hyperparamètres trouvés à l'étape précédente sur la jonction de la partition d'entraînement et de validation et rapporter les performances sur la partition de test.
   (b) Rapporter la performance moyenne de chaque technique d'optimisation d'hyperparamètres sur les partitions de test pour toutes les répétitions et calculer le rang de chaque méthode sur ces performances moyennes pour chaque jeu de données.
2. Calculer le rang moyen de chaque technique d'optimisation d'hyperparamètres et faire des tests statistiques non paramétriques sur ceux-ci pour déterminer la meilleure technique.