Fondements de l'Apprentissage Machine (IFT 3395/6390)

Examen Final

Automne 2013

Professeur : Pascal Vincent

Jeudi 12 décembre 2013
Durée : 2h30

- Seule documentation permise : 4 feuilles recto/verso (format letter 8" 1/2 x 11") pour votre résumé de cours.
- L'utilisation d'appareils électroniques n'est pas autorisée durant l'examen (à l'exception d'une montre pour connaître l'heure).
- Le total de l'examen est sur 100pts. Veuillez répondre aux questions directement dans les zones de blanc laissées à cet effet. Répondez de manière concise, mais précise. **Bon examen !**

(4 pts)
Prénom :
Nom :
Code permanent :
IFT3395 ou IFT6390 :

Programme d'études :
Laboratoire d'attache (s'il y a lieu) :

Notation

Pour toutes les questions, on suppose qu'on travaille avec un ensemble de données de départ comportant n exemples noté $D_n = \{z^{(1)}, \ldots, z^{(n)}\}$ avec, dans les cas supervisés, $z^{(k)} = (x^{(k)}, t^{(k)})$ où $x^{(k)} \in \mathbb{R}^d$ est l'entrée et $t^{(k)}$ est la cible correspondante. On vous demande de respecter scrupuleusement les notations de cet énoncé (c.a.d. ne vous contentez-pas de retranscrire des formules telles quelles mais adaptez les aux notations de l'énoncé si besoin).

1 Exercice de classification (10 pts)

On a affaire à un problème de classification à 4 classes. L'ensemble de données D_n contient $n = 1000$ points, dont 400 sont de la classe 1, 400 sont de la classe 2, 100 sont de la classe 3, et 100 sont de la classe 4. On suppose qu'on a créé 4 estimateurs de densité $\hat{f}_1, \hat{f}_2, \hat{f}_3, \hat{f}_4$, et entraîné chacun uniquement sur les points d'une classe ($\hat{f}_1$ a été entraîné sur les points de la classe 1, $\widehat{f_2}$ sur ceux de la classe 2, etc...).

Pour un nouveau point de test x que l'on désire classifier, on obtient en appliquant ces 4 estimateurs de densité à ce point :

$$\begin{aligned} \hat{f}_1(x) &= 0.5 \\ \hat{f}_2(x) &= 1.0 \\ \hat{f}_3(x) &= 2.5 \\ \hat{f}_4(x) &= 1.5 \end{aligned}$$

1. Expliquez brièvement comment vous vous y prendriez pour calculer le vecteur des probabilités d'appartenance aux classes pour ce point x : $(P(t = 1|x), P(t = 2|x), P(t = 3|x), P(t = 4|x))$. Calculez ce vecteur.

2. Quelle classe d'appartenance décidera-t-on pour ce point x ?

3. Comment s'appelle cette technique ou ce genre de classifieur ?

2 Arbres de décision (14 pts)

Un arbre de décision a été entraîné sur un problème de classification à 2 classes (classe A et classe B) sur des données en deux dimensions $x = (x_1, x_2)$. La règle de décision correspondant à l'arbre ainsi appris est :

Classe A si ($x_2 \leq 3$ **ET** $x_1 > 2$ **ET** $x_2 > 1$) **OU** ($x_2 > 3$ **ET** $x_1 \leq 0$)
Classe B sinon

1. Dessinez l'arbre de décision correspondant à cette règle. Chaque noeud interne (ainsi que la racine) devra contenir un test du type $x_i \leq t$ (**obligatoirement avec des** $\leq$**)** que vous indiquerez explicitement dans le noeud. Par convention, l'arc menant au fils gauche est suivi si le résultat du test est vrai, celui menant au fils droit si le test est faux. Indiquez vrai ou faux sur chaque arc pour que ce soit clair. Les feuilles de l'arbre devront indiquer la décision de classe A ou B.

2. Sur un graphique 2D avec des axes, clairement gradués entre -5 et 5, hachurez la région de décision de la classe **A** induite par ce classifieur.

3. Les arbres de décision sont-ils des classifieurs linéaires ? (oui/non)
4. Nommez *deux* avantages des arbres de décision :

5. Nommez le principal inconvénient des arbres de décision :
6. Nommez une technique souvent utilisée avec les arbres de décision pour pallier à cet inconvénient :

3 Sur-apprentissage, sous-apprentissage, capacité (14 pts)

On rappelle que la "capacité" d'un algorithme d'apprentissage correspond, informellement, à la taille, la "richesse" ou la "complexité" de l'ensemble de fonctions considérées parmi lesquelles il trouve sa fonction de prédiction.

Répondez VRAI ou FAUX à la gauche de chaque ligne (ou bien abstenez vous) : +1 pour une bonne réponse, -1 pour une mauvaise (le minimum pour l'exercice est 0/14, le maximum 14/14).

1. Le sur-apprentissage se traduit par un taux d'erreur très faible sur l'ensemble de validation.
2. Le sous-apprentissage se traduit par un taux d'erreur trop élevé, à la fois sur l'ensemble d'entraînement et sur l'ensemble de validation.
3. Lorsqu'on a le choix entre plusieurs algorithmes d'apprentissage, on devrait choisir celui qui parvient le mieux à apprendre les exemples sur lesquels il est entraîné.
4. Plus on a d'exemples pour l'entraînement, plus il y a de risque de sur-apprentissage.
5. Un classifieur de fenêtres de Parzen à noyau Gaussien avec une largeur de fenêtre σ trop élevée mène à du sur-apprentissage.
6. Plus un algorithme d'apprentissage a une capacité élevée, meilleure sera sa prédiction pour de nouveaux exemples de test.
7. Plus un algorithme d'apprentissage a une capacité élevée, moins il fera d'erreurs sur un ensemble d'entraînement compliqué.
8. Plus la capacité de l'algorithme est faible, plus on risque un sur-apprentissage.
9. L'algorithme des *Machines à Vecteurs de Support linéaire* a une capacité plus faible que le *1-plus proche voisin*.
10. La capacité d'un algorithme d'apprentissage peut généralement se contrôler à travers la valeur de ses *hyper-paramètres*.
11. Un algorithme d'apprentissage avec une plus forte capacité (qu'un autre) tendra à avoir un plus grand biais et une plus petite variance.
12. Si on choisissait la valeur des *hyper-paramètres* qui produit l'erreur la plus petite sur l'ensemble d'entraînement (sur lequel on apprend les *paramètres*) cela mènerait toujours à choisir la valeur des hyper-paramètres donnant la capacité la plus élevée possible. C'est pour cette raison qu'on utilise un ensemble de validation séparé.
13. L'astuce du noyau permet d'augmenter significativement la capacité de classifieurs ou régresseurs linéaires.
14. Une SVM linéaire a une capacité plus faible qu'une SVM avec un noyau du type produit scalaire usuel.

4 Sélection de modèle et d'hyper-paramètres (16 pts)

1. Expliquez, selon votre compréhension, ce qui distingue les "hyper-paramètres" des "paramètres" d'un algorithme d'apprentissage.

2. On vous demande d'écrire la procédure de haut niveau qu'on va typiquement utiliser pour sélectionner la valeur la plus prometteuse des hyper-paramètres. On suppose qu'on dispose d'un très grand nombre n d'exemples dans un ensemble de données D_n. Formellement, soit $A(D, \lambda)$ la procédure d'entraînement d'un algorithme d'apprentissage qui, sur un ensemble d'entraînement D et avec des valeurs d'hyper-paramètres λ, apprend la valeur optimale des paramètres θ d'une fonction de prédiction f_θ. La procédure A, qu'on suppose déjà correctement écrite, et à laquelle vous pouvez faire appel, retourne la valeur des paramètres appris, de sorte que l'on peut écrire pour l'entraînement : $\theta = A(D, \lambda)$. Une fonction de prédiction avec des paramètres θ va donner, en un point x, une prédiction $f_\theta(x)$. Soit $\hat{R}(f_\theta, D)$ l'erreur moyenne commise par cette fonction de prédiction f_θ sur un ensemble d'exemples D. Écrivez le pseudo-code détaillé de la procédure $B(D_n, \mathcal{H})$ qui effectuera la sélection de la valeur la plus prometteuse λ^* des hyper-paramètres parmi une liste $\mathcal{H}$ de valeurs possibles fournie à l'algorithme ($\lambda \in \mathcal{H}$). Cette procédure devra naturellement faire appel à A et $\hat{R}$ (sur des sous-ensembles de données appropriés) et devra retourner *les valeurs optimales* λ^* et θ^* retenues pour les paramètres et hyper-paramètres.

   FONCTION $B(D_n, \mathcal{H})$:

5 Modèles graphique probabiliste et mélange de Gaussienne (18 pts)

1. Dessinez le modèle graphique dirigé correspondant à la factorisation suivante de la probabilité jointe entre 5 variables : $P(X_1, X_2, X_3, X_4, X_5) = P(X_3|X_5)P(X_4)P(X_5|X_1, X_4, X_2)P(X_2)P(X_1|X_2)$
2. Écrivez sous forme de pseudo-code, comment on pourrait générer un exemple de cette distribution jointe, en supposant qu'on sait générer de ses facteurs. Par exemple pour signifier qu'on génère une valeur de X_1 selon la loi $P(X_1|X_2)$ sachant que $X_2 = x_2$ on écrira $x_1 \sim P(X_1|X_2 = x_2)$.

   Procédure Génère_de_P :

   return ( )
3. Les modèles graphiques probabilistes sont représentés avec des noeuds et des arcs, de même que les réseaux de neurones feed-forward (perceptorn multi-couche ou similaire). Mais dans ces deux cas, les noeuds représentent quelquechose de nature fondamentalement différente. Quelle est cette différence ?
4. Qu'est-ce qu'on nomme "variable latente" dans un modèle graphique ?
5. Un mélange de Gaussienne peut être représenté sous la forme d'un modèle graphique dirigé avec des variables observées et des variables latentes. Dessinez le graphe correspondant, et expliquez à quoi correspond la variable associée à chaque noeud, sa nature ainsi que sa dimensionalité (on suppose qu'on va l'utiliser sur un ensemble de donnée D_n tel qu'indiqué au début de l'examen).
6. A quoi sert un tel mélange de Gaussienne : pour quel(s) type(s) de problèmes d'apprentissage peut-on s'en servir ?
7. A quel(s) autre(s) algorithmes d'apprentissage s'apparente le mélange de Gaussiennes ?

8. Expliquez en quoi consiste le problème de l'inférence dans un modèle graphique en général, et dans le mélange de Gaussienne en particulier.

9. Nommez deux approches algorithmiques différentes permettant d'apprendre les paramètres d'un mélange de Gaussienne.

6 Réseaux de neurones (28 pts en tout)

On considère un réseau de neurones, paramétré par un ensemble de paramètres θ, comme une fonction $f_\theta(x)$. Pour une entrée $x \in \mathbb{R}^d$, il produit une sortie $y = f_\theta(x)$.

6.1 Contrôle de capacité d'un réseau de neurones (5 pts)

Dans un modèle de réseau de neurones de type MLP (perceptron multicouche) on dispose de plusieurs leviers (hyper-paramètres) et techniques pour contrôler la capacité du modèle et ainsi gérer le risque de sur-apprentissage. Quels sont ces leviers et techniques ? Nommez-les, expliquez précisément ce qu'ils représentent et, pour chacun, indiquez le sens de l'effet du levier, c.a.d. si le fait de l'augmenter va augmenter la capacité ($\uparrow$) du modèle ou au contraire diminuer la capacité ($\downarrow$) du modèle.

6.2 Réseaux récurrents (5 pts)

1. Expliquez selon votre compréhension ce qu'est un réseau récurrent et en quoi il diffère d'un réseau de type MLP ordinaire.

2. Sur quel type de données les réseaux récurrents sont-ils particulièrement appropriés ?

3. Dans un réseau récurrent, l'état caché à un temps t est calculé comme une fonction directe de quoi ?

4. Nommez une difficulté particulière à l'entrainement d'un réseau de neurone récurrent par descente de gradient.

6.3 Réseaux de neurones de type Radial Basis Function (18 pts)

Pour les réseaux de type Perceptron Multicouche (MLP) vus en cours, un neurone N_k de la première couche cachée reçoit une entrée x et a un vecteur de poids synaptiques $w_k \in \mathbb{R}^d$ et un biais $b_k \in \mathbb{R}$. Il calcule sa sortie h_k avec la formule $h_k = \text{sigmoid}(\langle w_k, x\rangle + b_k)$, où $\langle w_k, x\rangle$ dénote le produit scalaire usuel.

On s'intéresse pour cette partie à un type de réseaux de neurones différent, nommé RBF (Radial Basis Function). Ces réseaux à une couche cachée sont très similaires aux MLP. La différence est qu'un neurone RBF N_k de la couche cachée, ayant un vecteur de poids w_k calcule sa sortie h_k ainsi :

$$\begin{aligned} h_k &= \exp(-\beta\|x - w_k\|^2) \\ &= \exp\left(-\beta \sum_{j=1}^{d} (x_j - w_{kj})^2\right) \end{aligned}$$

où exp désigne l'exponentielle et β est un hyper-paramètre (le même pour tous les neurones de la première couche cachée. Remarquez aussi qu'il n'y a *pas de biais*. Une unique couche cachée de m neurones RBF ayant des sorties $(h_1, \ldots, h_m) = h$ est typiquement suivie d'une couche de sortie linéaire avec des poids $(a_1, \ldots, a_m) = a$ pour donner une sortie $y = f_\theta(x) = \langle a, h\rangle = \sum_{k=1}^{m} a_k h_k$.

1. Quel est l'ensemble θ des paramètres (excluant les hyper-paramètres) d'un tel réseau RBF ?
   $\theta = \{$

   A combien de nombres réels ajustables (combien de scalaires) cela correspond-t-il ?

2. Le coût pour un exemple x pour lequel le réseau prédit $f_\theta(x)$ alors que la vraie cible est t est donné par une fonction de coût différentiable $L(f_\theta(x), t)$. On cherche les valeurs des paramètres qui vont minimiser le coût empirique moyen sur un ensemble d'apprentissage $D_n = \{(x^{(1)}, t^{(1)}), \ldots, (x^{(n)}, t^{(n)})\}$. Exprimez ce problème de minimisation.

3. On suppose qu'on a organisé θ comme un vecteur contenant tous les paramètres, et qu'on dispose d'une fonction permettant de calculer efficacement $\frac{\partial L}{\partial \theta}$. Écrivez la procédure de décente de gradient stochastique qui permettrait d'optimiser les paramètres :

4. On veut utiliser ce réseau pour prédire une cible t réelle. Il sagit d'un problème de régression. On notera $y = f_\theta(x)$ la prédiction du réseau. Écrivez la fonction de perte plus spécifique qu'on va généralement utiliser pour un tel problème :

   $L(y, t) =$

5. Pour pouvoir effectuer la descente de gradient, cela suppose de savoir calculer le gradient c.a.d la dérivée partielle du coût L par rapport aux paramètres θ. On va pour cela utiliser la rétropropagation des gradients.
   Commencez par exprimer
   $$\frac{\partial L}{\partial y} =$$

6. Exprimez et calculez (en fonction de a_k, h_k, et $\frac{\partial L}{\partial y}$ sans les substituer par leurs expressions complètes) :
   $$\frac{\partial L}{\partial a_k} =$$
   $$\frac{\partial L}{\partial h_k} =$$

7. Puis exprimez et calculer le gradient par rapport aux paramètres w (en fonction entre autres de $\frac{\partial L}{\partial h_k}$ sans le substituer par son expression complète) :
   $$\frac{\partial L}{\partial w_{kj}} =$$

   Rappel de la formule pour dériver une exponentielle : $\exp(u)' = u' \exp(u)$, *ou encore :* $\frac{\partial \exp(u)}{\partial \theta} = \frac{\partial u}{\partial \theta} \exp(u)$

8. Écrivez, en fonction des quantités calculées plus haut (mais là encore, sans substituer leur expressions que vous avez déjà données), de manière plus détaillée, la mise à jour spécifique des paramètres du réseau lors d'un pas de gradient stochastique (avec un learning rate η)