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

Professeur : Pascal Vincent

Mercredi 7 décembre 2011
Durée : 2h00

Documentation permise : 2 feuilles recto/verso pour votre résumé de cours.

Prénom :
Nom :
Code permanent :
IFT3395 ou IFT6390 :
Programme d'études (et laboratoire s'il y a lieu) :

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 !**

1 Tâches de l'apprentissage (25 pts)

Dans chacun des cas suivants, indiquez à quelle tâche (type de problème standard en apprentissage) cela correspond, et *nommez* tous les algorithmes d'apprentissage spécifiques que vous connaissez qui peuvent s'y appliquer.

1. Vous voulez identifier des groupes "naturels", c.a.d. des sous-ensembles d'exemples similaires, parmi un ensemble de données.

2. Chaque exemple de l'ensemble de données dont vous disposez comporte déjà une variable qui indique son appartenance à un groupe spécifique (parmi k groupes). Mais pour les futurs exemples, cette information ne sera pas disponible et vous voulez pouvoir la prédire.

3. Vous disposez d'un ensemble de 500 exemples comportant chacun 10 traits caractéristiques réels, et vous voulez pouvoir visualiser cet ensemble comme un nuage de points en 3 dimension qui le représente au mieux (c.a.d. avec un minimum de perte d'information).

4. Vous voulez pouvoir prédire la valeur espérée d'une caractéristique correspondant à un nombre réel, sachant la valeur des autres caractéristiques d'un exemple.

2 Sur-apprentissage, sous-apprentissage, capacité et sélection de modèle (25 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) : +2 pour une bonne réponse, -2 pour une mauvaise (le minimum pour l'exercice est 0, 25, le maximum 25, 25).

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 élevée, 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) a 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.

3 Arbres de décision (25 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 :

4 Réseau de neurones et rétro-propagation du gradient (25 pts)

Un certain réseau de neurones du type *auto-encodeur* qui reçoit une entrée $x \in \mathbb{R}^d$ (vecteur colonne), a une couche cachée de taille k, et calcule une sortie donnée par la formule suivante :

$$r(x) = Vs(Ux + b)) + c$$

où U et V sont des matrices de poids, b et c sont des vecteurs de biais, et s est une fonction scalaire appliquée élément par élément au niveau de la couche cachée (par ex. s pourrait être une sigmoïde ou une tanh, ou encore autre chose).

La sortie $r(x)$ est appelée une "reconstruction" de l'entrée x et a la même dimension que x.

Un tel réseau est traditionnellement entraîné à bien reconstruire son entrée, en minimisant une perte telle que l'erreur quadratique entre l'entrée et sa reconstruction : on minimise $L(r, x) = \|r - x\|^2$.

Remarquez qu'il n'y a pas de "cible" autre que l'entrée, il s'agit donc d'une approche d'apprentissage non supervisé.

1. À quoi peut servir un tel réseau ?

2. Indiquez l'ensemble θ des paramètres de ce réseau et les *dimensions* de chacun.

3. Écrivez la formule du risque empirique qu'on va chercher à minimiser lors de l'entraînement d'un tel réseau de neurones sur un ensemble de d'entraînement $D_n = \{x^{(1)}, \ldots, x^{(n)}\}$.

4. Écrivez la formule de haut-niveau de mise à jour des paramètres θ d'un tel réseau par descente de gradient (batch) sur ce risque empirique (avec un pas de gradient ϵ).

5. Écrivez la formule de haut niveau de mise à jour des paramètres θ d'un tel réseau par descente de gradient stochastique (avec un pas de gradient ϵ)

6. Exprimez le calcul de $\frac{\partial L(r,x)}{\partial r}$ en fonction de x et r.

7. Considérons la descente de gradient stochastique, où on va mettre à jour es paramètres après avoir vu un exemple. On utilise la technique efficace de *rétro-propagation du gradient* pour le calcul des gradients pour cet exemple. On décompose les calculs effectués en trois phases consécutives : a) une phase de propagation avant (**forward_prop**) qui calcule la reconstruction en fonction de l'entrée et la perte (erreur de reconstruction), b) une phase de propagation arrière (**back_prop**) qui fait le calcul des gradients à proprement dits, et c) une phase de mise à jour des paramètres (**param_update**) à l'aide des gradients qu'on vient de calculer. On a ci-dessous mélangé l'ordre de toutes les opérations de ces différentes phases. Indiquez dans le bon ordre, parmi la liste ci-dessous, les opérations pour chacune des phases. Ex : forward_prop : 14, 3, 10, 7, 2, 1 (évidemment ce n'est pas la bonne réponse !).

   (a) Opérations de la phase **forward_prop** :

   (b) Opérations de la phase **back_prop** :

   (c) Opérations de la phase **param_update** :

Liste d'opérations mélangées [1] :

1. $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial r} h^T$
2. $b \leftarrow b - \epsilon \frac{\partial L}{\partial b}$
3. $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}$
4. $\frac{\partial L}{\partial c} = \frac{\partial L}{\partial r}$
5. Calculer $\frac{\partial L}{\partial r}$ (selon la formule répondue à la question 6).
6. $\frac{\partial L}{\partial U} = \frac{\partial L}{\partial a} x^T$
7. $\frac{\partial L}{\partial a} = \frac{\partial L}{\partial h} \odot s'(a)$ (où s' dénote la dérivée de la fonction s et $\odot$ le produit terme à terme)
8. $c \leftarrow c - \epsilon \frac{\partial L}{\partial c}$
9. $\frac{\partial L}{\partial h} = V^T \frac{\partial L}{\partial r}$
10. $r = Vh + c$
11. $V \leftarrow V - \epsilon \frac{\partial L}{\partial V}$
12. $U \leftarrow U - \epsilon \frac{\partial L}{\partial U}$
13. $h = s(a)$
14. $L = \|r(x) - x\|^2$
15. $a = Ux + b$

1. Note : on a ici adopté la convention que la dévirée partielle d'un scalaire par rapport à un vecteur ou une matrice a les mêmes dimensions que ce vecteur ou cette matrice.