Fondements de l'Apprentissage Machine (IFT 3395/6390)
Hiver 2012 – Examen Final

Professeur : Pascal Vincent

Jeudi 10 janvier 2013
Durée : 2h45

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. **Dans tout l'examen on suppose que les entrées $x \in \mathbb{R}^d$, et que l'ensemble d'entraînement D_n contient n exemples. Bon examen !**

1 Hyper-paramètres, capacité et sélection de modèle (29 pts)

1. Expliquez, dans vos propres mots, le phénomène de **sur-apprentissage**.

2. Expliquez, dans vos propres mots, le phénomène de **sous-apprentissage**.

3. Expliquez dans vos propres mots la notion de **"capacité"** d'un algorithme d'apprentissage.

4. Entourez ce qui *augmente* le risque de **sur-apprentissage** :
   (a) avoir un plus grand nombre d'exemples d'entraînement
   (b) avoir un plus petit nombre d'exemples d'entraînement
   (c) augmenter la capacité de l'algorithme (ou choisir un algo avec une plus grande capacité)
   (d) diminuer la capacité de l'algorithme (ou choisir un algorithme avec une plus petite capacité)

5. Entourez ce qui *augmente* le risque de **sous-apprentissage** :
   (a) avoir un plus grand nombre d'exemples d'entraînement
   (b) avoir un plus petit nombre d'exemples d'entraînement
   (c) augmenter la capacité de l'algorithme (ou choisir un algo avec une plus grande capacité)
   (d) diminuer la capacité de l'algorithme (ou choisir un algorithme avec une plus petite capacité)
6. Quel est le lien entre le choix de la capacité d'un algorithme d'apprentissage et le dilemme biais-variance ?

7. La plupart des algorithmes d'apprentissage ont des hyper-paramètres permettant plus ou moins de contrôler leur ń capacité ż effective. Que se passe-t-il si on choisit la valeur de ces hyper-paramètres qui conduit à l'erreur d'apprentissage la plus faible sur l'ensemble d'entraînement ? Vers quel choix de capacité (faible ? élevée ? optimale ?) cela va-t-il nous mener ? Donnez un exemple de ce phénomène : que se passerait avec les hyper-paramètres spécifiques d'un algorithme de votre choix.

8. Nommez les hyper-paramètres typiquement utilisés pour un réseau de neurones à une couche cachée (avec régularisation de type "weight decay"). Indiquez à côté de chacun si vous pensez qu'augmenter la valeur de cet hyper-paramètre augmente la capacité effective du réseau (écrivez $\uparrow$), la diminue (écrivez $\downarrow$). Si vous pensez qu'il n'a pas un effet clair et précis sur la capacité écrivez $-$.

9. Expliquez, brièvement de manière informelle, la procédure qu'on peut utiliser pour sélectionner la valeur la plus prometteuse des hyper-paramètres.

10. Plus 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 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.

2 Concepts graphiques (14 pts)

On apprend les paramètres θ d'une fonction de classification binaire$f_{\lambda,\theta}(x)$ qui a aussi des hyper-paramètres λ. Cette fonction $f_{\lambda,\theta}(x)$ donne **une estimation de la probabilité que x soit de la classe 1** (par opposition à la classe 0).

Soit le risque empirique $\hat{R}$ associé à une telle fonction sur un ensemble D de paires d'exemples (x,t) avec $t \in \{0,1\}$ qui indique la classe :

$$\hat{R}(f_{\lambda,\theta}, D) = \sum_{x,t \in D} L(f_{\lambda,\theta}(x), t)$$

Soit les **"concepts graphiques"** suivants :

1. Frontière de décision
2. région de décision de la classe 0
3. région de décision de la classe 1
4. ensemble des points mal classifiés par $f_{\lambda,\theta}$
5. ensemble des points bien classifiés par $f_{\lambda,\theta}$
6. paysage de coût ou d'erreur
7. courbe d'apprentissage

Considérez les **équations** suivantes, et écrivez **à gauche** chaque équation, le numéro du "concept graphique" auquel elle correspond (si il y en a un qui correspond parmi la liste ci dessus).

$$\begin{aligned}
O(\theta) &= \hat{R}(f_{\lambda,\theta}, D_n) \\
\mathcal{A} &= \{x \in \mathbb{R}^d | f_{\lambda,\theta}(x) < 0.5\} \\
\hat{R}^*(\lambda) &= \min_\theta \hat{R}(f_{\lambda,\theta}, D_n) \\
\mathcal{B} &= \{(x,t) \in D_n | \, (f_{\lambda,\theta}(x) - 0.5)\,(t - 0.5) > 0\} \\
\\
\mathcal{C} &= \{x \in \mathbb{R}^d | f_{\lambda,\theta}(x) = 1 - f_{\lambda,\theta}(x)\} \\
\frac{\partial \hat{R}}{\partial \theta} &= \sum_{(x,t) \in D_n} \frac{\partial}{\partial \theta} L(f_{\lambda,\theta}(x), t) \\
\mathcal{E} &= \{x \in \mathbb{R}^d | f_{\lambda,\theta}(x) = 0.5\}
\end{aligned}$$

3 Questions variées (29 pts)

1. Quel est l'intérêt d'utiliser l'astuce du noyau pour les SVMs (et d'ailleurs pour de nombreux autres algorithmes) ?

2. Nommez **tous** les algorithmes d'apprentissage que vous connaissez qui, appliqués à un problème de classification binaire, apprennent un **classifieur linéaire.**

3. Nommez **tous** les algorithmes d'apprentissage que vous connaissez qui, sur un problème de classification binaire, permettent d'apprendre des **classifieurs non-linéaires.** Pour chacun précisez, si vous les connaissez, ses principaux hyper-paramètres de contrôle de capacité.

4. Quel algorithme d'apprentissage permet d'apprendre un classifieur de forte capacité en apprenant et combinant des "classifieurs faibles" ?

5. Nommez deux avantages des arbres de décisions ?

6. Dans quel cas dit-on qu'un noeud d'un arbre de décision est "pur" ? Soyez précis dans votre réponse.

7. Comment peut-on remédier au problème d'instabilité des arbres de décision ?

8. En quoi une tâche de "partitionnement" (clustering) est-elle semblable à une tâche de classification ? En quoi est-elle différente ?

9. Quels sont les paramètres appris par l'algorithme des k-moyennes ? Précisez également leur dimension.

10. Une fois l'apprentissage terminé comment l'algorithme des k-moyennes décide-t-il à quel cluster affecter un point x.

11. Écrivez sous forme d'un pseudo-code de haut niveau l'algorithme d'apprentissage des k-moyennes (k-means).

12. L'algorithme des k-moyennes est très relié à un modèle de densité de probabilité spécifique. **Lequel ?**

    Qu'est-ce qui dans ce modèle joue le rôle des k "moyennes" ?

    Le numéro de cluster attribué à un point de test x dans les k-moyennes, correspond à quoi dans ce modèle probabiliste ?

13. Nommez un algorithme de réduction de dimensionalité, et indiquez à quoi il peut servir.

14. Quelle similarité et quelle différence voyez-vous entre les "neurones cachés" d'un réseau de neurone de type perceptron multi-couche, et les "variables latentes" d'un modèle probabiliste ?

4 Réseau de neurones et rétro-propagation du gradient (28 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. $r = Vh + c$
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. $a = Ux + b$
8. $\frac{\partial L}{\partial a} = \frac{\partial L}{\partial h} \odot s'(a)$ (où s' dénote la dérivée de la fonction s et $\odot$ le produit terme à terme)
9. $\frac{\partial L}{\partial h} = V^T \frac{\partial L}{\partial r}$
10. $V \leftarrow V - \epsilon \frac{\partial L}{\partial V}$
11. $U \leftarrow U - \epsilon \frac{\partial L}{\partial U}$
12. $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}$
13. $h = s(a)$
14. $L = \|r - x\|^2$
15. $c \leftarrow c - \epsilon \frac{\partial L}{\partial c}$

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.