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

Examen Final

Automne 2015

Professeur : Pascal Vincent

Jeudi 10 décembre 2015
Durée : 2h45

Documentation permise : 4 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 $x \in \mathbb{R}^d$. Bon examen !**

1 Concepts graphiques (20 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).

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

2 Capacité et courbes d'apprentissage (30 pts)

1. Expliquez les différences entre les paramètres et les hyper-paramètres d'un algorithme d'apprentissage

2. La plupart des algorithmes d'apprentissage possèdent un ou plusieurs paramètres permettant de contrôler leur "capacité". Expliquez dans vos propres mots cette notion de capacité.

3. Nommez un algorithme et un de ses hyper-paramètre, qui si on augmente sa valeur, cela va **augmenter** la "capacité" effective de l'algorithme.

4. Tracez sur un même graphique, l'allure typique des courbes d'apprentissage d'entrainement (en trait plein) et de validation (en pointillé) qu'on obtiendrait en faisant varier cet hyper-paramètre (à mettre sur l'axe des abscisses).

   (a) Indiquez clairement les quantités sur vos axes

   (b) Mettez-en évidence, sur l'axe des abscisses, la valeur optimale de l'hyper-paramètre

   (c) Sur l'axe des abscisses, identifiez clairement les zones correspondant à du SUR-APPRENTISSAGE et les zones correspondant à du SOUS-APPRENTISSAGE.

   (d) Sur l'axe des abscisses, identifiez clairement quelle zone correspond à une CAPACITÉ TROP FAIBLE et quelle zone correspond à une CAPACITÉ TROP FORTE

5. Nommez un algorithme et un de ses hyper-paramètre, qui si on augmente sa valeur, cela va **diminuer** la "capacité" effective de l'algorithme.

6. Tracez un graphique de courbes d'apprentissage (mêmes instructions que précédemment) pour cet autre algorithme et hyper-paramètre.

7. Soient les ensembles d'entrainement D_{train} et de validation D_{valid} (contenant respectivement $|D_{train}|$ et $|D_{valid}|$ exemples, écrivez le pseudo-code qui permettrait de tracer ces courbes d'apprentissage (d'entrainement et de validation) en fonction d'un hyper-paramètre que nous nommerons λ. Assurez-vous de clairement distinguer les exemples d'entrainement des exemples de validation. Définissez toutes vos notations (notamment pour la procédure d'entrainement).

3 Réseau de neurones et rétro-propagation du gradient (30 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. $c \leftarrow c - \epsilon \frac{\partial L}{\partial c}$
5. $U \leftarrow U - \epsilon \frac{\partial L}{\partial U}$
6. $\frac{\partial L}{\partial c} = \frac{\partial L}{\partial r}$
7. Calculer $\frac{\partial L}{\partial r}$ (selon la formule répondue à la question 6).
8. $\frac{\partial L}{\partial U} = \frac{\partial L}{\partial a} x^T$
9. $a = Ux + b$
10. $\frac{\partial L}{\partial a} = \frac{\partial L}{\partial h} \odot s'(a)$ (où s' dénote la dérivée de la fonction s et $\odot$ le produit terme à terme)
11. $V \leftarrow V - \epsilon \frac{\partial L}{\partial V}$
12. $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}$
13. $h = s(a)$
14. $L = \|r - x\|^2$
15. $\frac{\partial L}{\partial h} = V^T \frac{\partial L}{\partial r}$

4 Nombres de paramètres (20 pts)

Les algorithmes d'apprentissage, apprennent (calculent, trouvent, optimisent ou mémorisent) durant une phase d'*entraînement*, un certain nombre de *paramètres*, qui seront par la suite utilisés pour faire une prédiction sur de nouveaux points de test. Ces paramètres sont constitués d'un certain nombre de *scalaires* (nombres réels). Ex : si les paramètres sont une matrice 20×30 et un vecteur de taille 10, alors le nombre total de paramètres scalaires appris est 610. Pour chacun des cas et algorithmes ci-dessous écrivez **à gauche de son numéro** le **nombre total de paramètres scalaires** appris ou mémorisés (nécessaires au moment d'effectuer une prédiction sur un nouveau point de test). On exclut de ce compte les hyper-paramètres.

On considérera les dimensions du problème suivantes : un ensemble d'entrainement de **800** exemples dont les entrées sont de **dimension 10** (10 traits caractéristiques, $x \in \mathbb{R}^{10}$), sauf dans les cas où il est spécifié différemment.

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.

1. Problème de régression simple ; algorithme de régression linéaire (affine)
2. Problème de classification binaire (2 classes) ; algorithme du perceptron
3. Problème de classification binaire (2 classes) ; algorithme de régression logistique
4. Problème de classification binaire (2 classes) ; algorithme de SVM linéaire
5. Problème de classification binaire (2 classes) ; algorithme de SVM à noyau (non linéaire)
6. Problème de régression avec $x \in \mathbb{R}$ (1 dimension) ; algorithme de régression polynomiale de degré 2
7. Problème de régression avec $x \in \mathbb{R}^2$ (2 dimension) ; algorithme de régression polynomiale de degré 2
8. Problème de régression avec avec $x \in \mathbb{R}^4$. Histogramme avec chaque dimension subdivisée en 10 intervalles (spécifiés à l'avance par des hyper-paramètres).
9. Problème d'estimation de densité (avec $x \in \mathbb{R}^{10}$). Une *Gaussienne isotropique*
10. Problème d'estimation de densité. Une *Gaussienne* avec covariance *diagonale*
11. Problème d'estimation de densité. Une *Gaussienne* avec covariance pleine
12. Problème de classification multiclasse (3 classes) ; classifieur de Bayes utilisant des *Gaussiennes diagonales* (sans autre contrainte ou partage de paramètres. Attention, n'oubliez rien !)
13. Problème d'estimation de densité ; estimateur de Parzen avec noyau Gaussien isotropique de variance fixée à l'avance, entraîné sur un ensemble comportant 800 exemples (pensez à ce qu'il est nécessaire de conserver en mémoire afin d'effectuer une prédiction sur un point de test)
14. Problème de classification multiclasses (3 classes) ; classifieur de fenêtres de Parzen avec noyau Gaussien isotropique de variance fixée à l'avance.
15. Problème de classification multiclasse (3 classes). Arbre de décision binaire classique ayant 5 noeuds de décision et 6 feuilles (qui se contentent d'émettre la classe majoritaire dans leur région).
16. L'algorithme des k-moyenne avec k=3
17. Une analyse en composante principale (PCA) produisant les 5 premières composantes principales.
18. Problème de classification binaire (2 classes) : Adaboost utilisant des Perceptrons, au bout de 5 itérations.
19. Problème de classification multiclass (3 classes) avec $x \in \mathbb{R}^{10}$; réseau de neurone de type MLP (perceptron multicouche) à une chouche cachée de 5 neurones.
20. Problème d'estimation de densité. Mélange (mixture) de 3 Gaussiennes diagonales.