Vendredi 27 avril 2007

IFT3390/6390 H2007
Fondements de l'apprentissage machine

Professeur: Pascal Vincent

Examen Final

Durée: 2h45

Prénom:
Nom:
Code permanent:
IFT 3390 ou 6390?

Veuillez répondre directement sur la feuille, dans l'espace prévu à cet effet.

Notations

Les notations suivantes sont définies pour tout l'examen:
On suppose qu'on dispose d'un ensemble de données de n exemples: $D = \{z_1, ..., z_n\}$. Dans le cas supervisé (classification, régression), chaque exemple z_i est constitué d'une paire *observation, cible:* $z_i = (x_i, y_i)$, alors que dans le cas non-supervisé, on n'a pas de notion de cible explicite donc juste un vecteur d'observation: $z_i = x_i$. On suppose que chaque observation est constituée de d traits caractéristiques: $x_i \in \mathbb{R}^d$.

1 Compréhension de concepts clés (18 pts)

Répondez brièvement aux questions suivantes (3 à 5 lignes dans l'espace prévu). Pour cette question, **répondez en Français, sans utiliser de symboles mathématiques.** Le but est de montrer que vous avez compris le concept, pas de recopier machinalement des formules ou des symboles des notes de cours... **Soyez bref, précis et concis.**

a) Qu'est-ce qu'on appelle *fonction de décision* pour un classifieur? Que reçoit-elle en entrée, et que calcule-t-elle? (c'est une fonction de quoi vers quoi?)

b) Qu'est-ce qu'on appelle *fonction de coût* (ou de *perte*) pour un algorithme d'apprentissage supervisé? Que reçoit-t-elle en entrée et que calcule-t-elle? (c'est une fonction de quoi vers quoi?)

c) Qu'est-ce qu'on appelle *risque empirique*, et en quoi diffère-t-il du *risque espéré* (ou erreur de généralisaiton)?

d) Quelle différence y a-t-il entre les *paramètres* d'un modèle et ses *hyper-paramètres*? Donnez un exemple de paramètre et d'hyper-paramètre pour un réseau de neurones. Comment apprend-t-on les *paramètres* d'un modèle? Comment peut-on trouver une bonne valeur pour les *hyper-paramètres*?

e) Qu'est-ce qu'on appelle *courbes d'apprentissage*, pour un algorithme d'apprentissage (ex: réseau de neurones). Quel genre de quantité met-on sur les axes d'une telle courbe en abscisse et en ordonnée?

f) Qu'est-ce qu'on appelle les *régions de décision*, d'un classifieur? Il s'agit de régions de quel espace? Quelle propriété vérifie une telle région?

g) Qu'est-ce qu'on appelle *frontière ou surface de décision* pour un classifieur binaire?

h) Qu'est-ce qu'on appelle *surface d'erreur* (*error surface* ou encore *paysage d'erreur, error landscape*) pour un algorithme d'apprentissage supervisé? Elle représente quelle quantité en fonction de quelle(s) quantité(s)?

i) Quel rapport y a-t-il entre l'apprentissage des *paramètres* d'un modèle, et le concept de *surface d'erreur*?

2 Régression constante pondérée (10 pts)

On a vu qu'il pouvait être utile de pouvoir entraîner un algorithme sur un ensemble de données *pondérées*, c.a.d. un ensemble pour lequel on a, associé à chaque observation x_i, un poids w_i. Par exemple AdaBoost nécessite la possibilité d'entraîner l'algorithme utilisé comme classifieur faible sur un ensemble de données pondérées.

Dans cet exercice on va considérer le modèle de *régression* le plus simple qui soit: le modèle *constant*. C'est un modèle un peu idiot qui pour tout x prédit $f(x) = C$. Le modèle a donc un seul "paramètre" qu'on peut apprendre: C. Pour rendre la paramétrisation explicite, on pourra écrire $f = f_C$

a) Rappelez la fonction de perte $L((x, y), f)$ habituellement utilisée pour les problèmes de régression.

b) Le paramètre C optimal pour l'ensemble de données pondéré, est celui qui minimise le *risque empirique pondéré*:

$$\sum_{i=1}^{n} w_i \, L((x_i, y_i), f_C)$$

En utilisant ce principe de minimisation du risque empirique pondéré, calculez cette valeur optimale de C.

3 Réseau de neurones régularisé avec données pondérées (15 pts)

Soit un réseau de neurones de type perceptron multi-couche. Le réseau accepte une entrée de dimension d ($x \in \mathbb{R}^d$), a une couche cachée comportant n_h neurones (avec des fonctions d'activation tanh), et un unique neurone de sortie.

a) Énumérez l'ensemble θ des paramètres du réseau. Quelle est la dimension de chacun des paramètres? A combien de nombre réels (scalaires) cela correspond-t-il?

b) Notons $a_\theta(x)$ la fonction calculant l'*activation* (donc avant l'éventuelle fonction d'activation ou non-linéarité) du neurone de sortie en fonction d'une observation d'entrée x. Écrivez $a_\theta(x)$ sous forme d'une expression matricielle d'une ligne.

c) Notons $f_\theta(x)$ la sortie du neurone de sortie (après une éventuelle non-linéarité), c.a.d. la prédiction du réseau de neurones. Indiquez quelle fonction d'activation on devrait utiliser et exprimez $f_\theta(x)$ en fonction de $a_\theta(x)$ dans les deux cas suivants:

   i. On veut prédire une valeur réelle ($y \in \mathbb{R}$)

   ii. On veut prédire la probabilité d'appartenance à la classe 1 (pour un problème de classification binaire où les classes ont les numéros 0 et 1)

d) Indiquez la fonction de coût (aussi appelée fonction de perte) L qu'on devrait utiliser pour chacun des cas i. et ii. ci-dessus.

e) Formulez le problème d'optimisation qui permettra de trouver une valeur optimale des paramètres en fonction de l'ensemble d'apprentissage D. (minimiser le risque empirique)

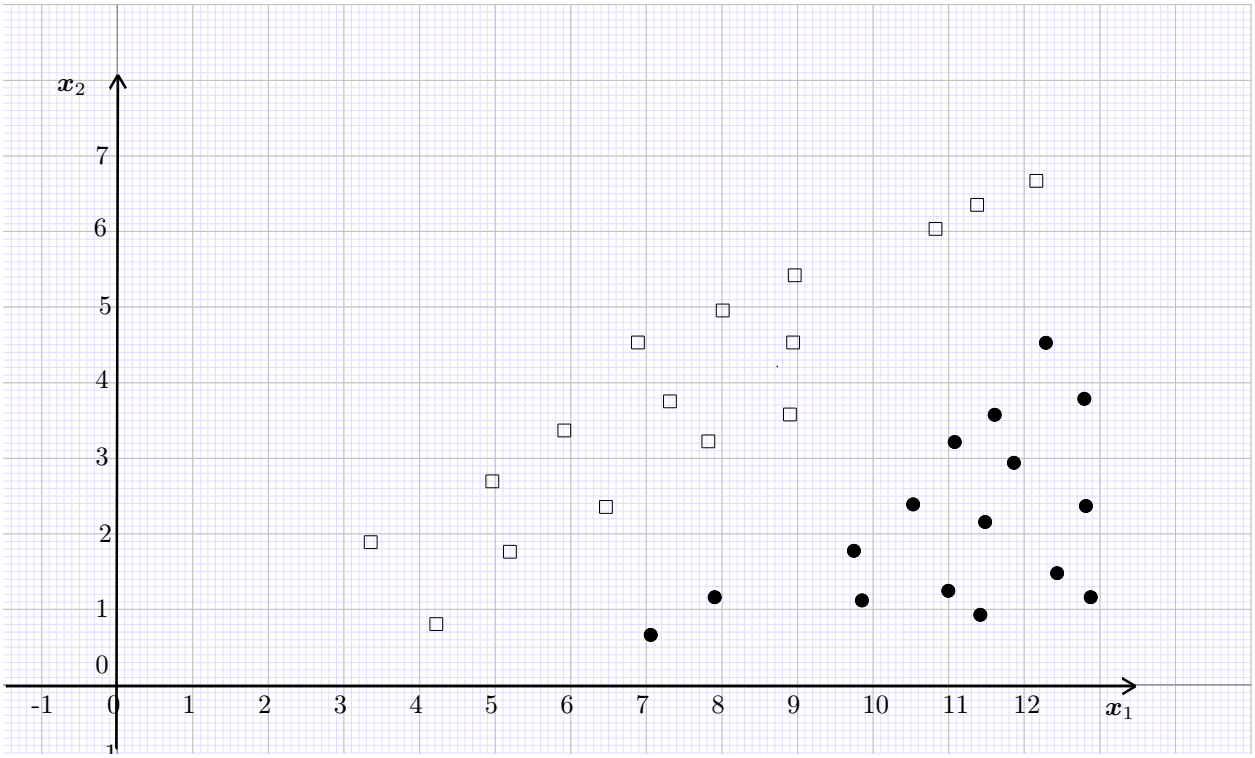
f) On va étendre la formulation que vous avez donné en e) en définissant un risque modifié $\tilde{R}$ qui d'une part permet de prendre en compte une pondération w_i pour les exemples d'apprentissage de la même manière qu'à la question 2 b), et d'autre part ajoute un terme de régularisation de type "weight-decay" de la forme $+\lambda \|\theta\|^2$, pour pénaliser des valeurs élevées des paramètres. ($\|\theta\|^2$ doit être compris comme la somme des carrés des valeurs de tous les éléments des paramètre). Exprimez $\tilde{R}$ et le problème d'optimisation ainsi étendu.

g) Pour résoudre ce problème d'optimisation, on peut utiliser une technique de descente de gradient. Exprimez le gradient $\frac{\partial \tilde{R}}{\partial \theta}$ qu'on va utiliser pour effectuer cette optimisation en fonction du gradient du coût sur des exemples individuels de D.

h) Au regard de ce que vous avez indiqué ci-dessus, comment l'utilisation d'une pondération différente pour chaque exemple affecte-t-il le calcul du gradient? (répondez en maximum 3 lignes)

4 Arbres de décision (15 pts)

Pour un problème de classification binaire (2 classes) en 2 dimension, on donne l'ensemble de données d'entraînement suivant:



On considère un classifieur de type arbre de décision *binaire* classique où chaque noeud n'effectue que la comparaison d'une seule variable d'observation à un seuil choisi (ex. CART). L'arbre est entraîné avec le critère suivant: on choisit des subdivisions qui minimisent la proportion d'erreurs de classification, et ceci jusqu'à obtenir 0 erreurs sur l'ensemble d'entraînement.

a) Sur le graphique ci-dessus, tracez une à une, en les numérotant, les subdivisions de l'espace que réaliserait un tel classifieur. Hachurez la *région de décision* correspondant à la classe des ronds noirs.

b) Dessinez ci-dessous l'*arbre de décision binaire* correspondant, en indiquant sur chaque arc la condition qui doit être vérifiée pour suivre cet arc. Indiquez à l'intérieur de chaque noeud, sous forme d'une paire fractions, la proportion d'exemples de chacune des deux classes qu'on a au niveau de ce noeud. La première fraction de chaque paire devra correspondre à la proportion des carrés, et la deuxième à la proportion des ronds noirs.

c) Exprimez ci-dessous, sous forme d'une règle logique (avec des ET et des OU), la règle représentée par cet arbre qui permet de décider si un point de test $x = (x_1, x_2)$ est de la classe des ronds noirs.

d) Calculez et indiquez ci-dessous, pour le noeuds racine et ses deux noeuds fils, les critère d'impureté suivants (vous pouvez laisser les résultats sous forme de fractions):

	Impureté de *mauvaise classification*	Impureté de *Gini*
noeud racine		
fils gauche		
fils droit		

e) Tracez approximativement en pointillé la frontière de décision qu'on obtiendrait avec un classifieur linéaire de type perceptron, et exprimez ci-dessous la *forme* de la règle de décision permettant de décider de la classe d'un point de test x. (on ne demande pas de calculer la valeur numérique des paramètres de ce classifieur, juste de donner la forme de la règle de décision.)

f) Indiquez au moins un avantage des arbres de décision classiques par rapport à un classifieur linéaire de type perceptron, et un avantage d'un classifieur linéaire par rapport à un arbre de décision classique.

5 Astuce du noyau pour classifieur de plus proche moyenne (15 pts)

On suppose qu'on a affaire à un problème de classification avec m classes: $y_i \in \{1, ..., m\}$

Dans cette question, on s'intéresse à l'algorithme de classification de plus proche moyenne (aussi appelé classifieur à centroide) qui apprend pour chaque classe un unique point prototype: le *centroide* des points de la classe, c.a.d. le point "moyenne" des points de la classe dans l'ensemble d'apprentissage. La décision pour un point de test consiste alors simplement à répondre la classe dont le centroide est le plus proche en distance Euclidienne.

Pour simplifier la formulation de l'algorithme, on suppose qu'on a déjà divisé l'ensemble de données D en m sous-ensembles d'observations: D_1, ..., D_m contenant chacun toutes les observations d'une unique classe: $D_k = \{x_i \in D | y_i = k\}$, et on note n_k le nombre d'exemples de la classe k, $k \in \{1, ..., m\}$.

On peut alors résumer l'algorithme de la manière suivante:

Apprentissage:
On apprend les vecteurs moyenne pour $k \in \{1, ..., m\}$

$$\mu_k = \frac{1}{n_k} \sum_{x \in D_k} x$$

Classification d'un point de test:
Pour un point de test x, on calcule la distance Euclidienne entre x et μ_k pour chacune des classes k et on décide la classe dont le centroide μ_k est le plus proche:

$$\begin{aligned} f(x) &= \text{ArgMin}_k \ \text{DistanceEuclidienne}(x, \mu_k) \\ &= \text{ArgMin}_k \ \text{DistanceEuclidienne}(x, \mu_k)^2 \\ &= \text{ArgMin}_k \ \|x - \mu_k\|^2 \end{aligned}$$

Remarquez que dans le cas de deux classes cela donne lieu à une frontière de décision linéaire. (Elle est constituée des points à égal distance de μ_1 et μ_2: c'est la bissectrice du segment $[\mu_1, \mu_2]$).

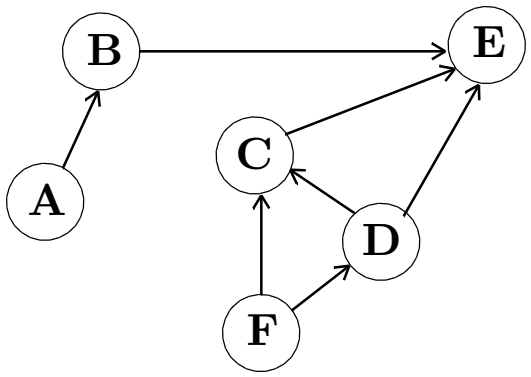
Si on veut pouvoir obtenir une frontière de décision plus souple, on pourrait vouloir utiliser l'astuce du noyau pour définir une version non-linéaire *"kernelisée"* de cet algorithme. C'est ce qu'on vous demande de faire.

a) Appliquez l'astuce du noyau, pour dériver une version "kernelisée" de l'algorithme de plus proche moyenne, puis écrivez l'algorithme sous forme d'un pseudo-code.

b) A quel autre algorithme classique ce classifieur de plus proche moyenne kernelisé ressemble-t-il?

6 Modèle graphique probabiliste (10 pts)

Soit le modèle graphique probabiliste suivant (réseau Bayesien).
On rappelle que chaque noeud du graphe représente une variable aléatoire.



a) Écrivez la probabilité jointe de tous les noeuds du graphe sous la forme du produit de probabilités (conditionnelles) qu'implique ce modèle graphique.

b) On suppose qu'on sait effectuer des tirages aléatoires (générer des valeurs) selon les lois $P(F)$ et $P(A)$ et selon *chacune* des probabilités conditionnelles que vous avez indiquées dans votre décomposition ci-dessus. Écrivez sous forme d'un bref algorithme (pseudo-code), comment on pourrait générer une matrice de donnée $\boldsymbol{X}$ comportant n rangées de 6 éléments, où chaque rangée correspondrait à un vecteur d'observations $\boldsymbol{X}_i = (a_i, b_i, c_i, d_i, e_i, f_i)$ distribué selon la loi $P(A, B, C, D, E, F)$ ci-dessus.

7 Choix d'un algorithme d'apprentissage (4 pts)

Nous avons vu un grand nombre d'algorithmes d'apprentissage. Lorsqu'on a affaire à un problème particulier, et qu'on a isolé un ensemble d'algorithmes qui pourraient potentiellement s'appliquer à ce problème (ex: si on a un problème de classification en haute dimension, on a à notre disposition plusieurs algorithmes de classification possibles). Indiquez comment vous procéderiez pour décider quel est le meilleur algorithme pour votre problème.

8 Bagging (9 pts)

Expliquez brièvement le principe du bagging appliqué à un ensemble de données D, avec un algorithme d'apprentissage $\mathcal{A}$. On suppose que $\mathcal{A}$ retourne, pour tout ensemble de données D', une fonction de prédiction $f_{D'} = \mathcal{A}(D')$. (ex: si $\mathcal{A}$ est l'algorithme du perceptron, alors $\mathcal{A}(D')$ retournerait une fonction de décision linéaire dont les paramètres sont ajustés à l'ensemble D').

Sous forme d'un pseudo-code de haut niveau, décrivez a) la procédure d'entraînement, et b) comment prendre une décision pour un point de test x.

9 Clustering (4 pts)

a) Expliquez dans vos propres mots en quoi consiste le type de problème d'apprentissage que l'on nomme *"clustering"*

b) Précisez en quoi cette tâche est semblable et en quoi elle est différente d'une tâche de *classification*.