FONDEMENTS DE L'APPRENTISSAGE MACHINE (IFT3395/6390)

Professeur: Pascal Vincent

Examen Final

Jeudi 17 décembre 2009

Durée: 2h00

Aucune documentation n'est permise

Prénom:

Nom:

Code permanent:

IFT 3395 ou 6390?

Le total de l'examen est sur 100 pts.
Veuillez répondre aux questions dans les zones de blanc laissées à cet effet.

Notations

Les notations suivantes sont définies pour tout l'examen, là où elles ont un sens:
On suppose qu'on dispose d'un ensemble de données de n exemples: $D_n = \{z^{(1)}, ..., z^{(n)}\}$. Dans le cas supervisé chaque exemple $z^{(i)}$ est constitué d'une paire *observation, cible:* $z^{(i)} = (x^{(i)}, t^{(i)})$, 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 (composantes): $x^{(i)} \in \mathbb{R}^d$: $x^{(i)} = (x_1^{(i)}, ..., x_d^{(i)})$

1 Apprentissage et optimisation (20 pts)

En apprentissage statistique, l'apprentissage va souvent consister en l'utilisation une technique de type descente de gradient pour trouver un minimum d'une fonction.

1.1 A quoi correspond cette fonction dont on cherche le minimum: c'est une fonction qui **calcule quoi en fonction de quoi?** Expliquez-le précisément en Français, dans vos propres mots, sans équation mathématique.

1.2 Écrivez sous forme d'un bref pseudo-code *commenté* de haut niveau, l'algorithme de descente de gradient (batch). Définissez/expliquez toute notation et variable que vous utilisez.

1.3 **Nommez** un algorithme d'apprentissage que vous connaissez où la fonction à optimiser est convexe, et où les techniques de type descente de gradient sont donc garanties de converger vers le minimum global. Puis **illustrez schématiquement** (graphiquement) une telle fonction, en précisant les labels de vos axes, et en indiquant la position du minimum trouvé.

1.4 **Nommez** un algorithme d'apprentissage pour lequel la fonction optimisée n'est pas convexe, et où on risque donc de se trouver coincé dans un minimum local. Puis **illustrez schématiquement** le problème des minimas locaux (précisez les labels de vos axes).

1.5 Est-ce nécessairement problématique si on n'arrive pas à trouver le minimum global de la fonction qu'on optimise? Pourquoi? (Indication: pensez, étant donné un problème d'apprentissage, à ce qui nous intéresse vraiment...)

2 Paramètres, hyper-paramètres, et sélection de modèle (20 pts)

2.1 D'une manière générale, expliquez la différence entre les **paramètres** et les **hyper-paramètres** d'un algorithme.

2.2 Comment procède-t-on le plus souvent pour trouver les **paramètres**? Selon quel critère?

2.3 Pourquoi ne choisit-on pas les **hyper-paramètres** selon ce même critère (celui utilisé pour les paramètres)?

2.4 Comment peut-on procéder pour trouver les **hyper-paramètres**? Selon quel critère?

2.5 Soit A la procédure qui permet d'apprendre les paramètres (et qui les retourne). Écrivez sous forme de pseudo-code, une procédure B qui va permettre de choisir une bonne valeur pour les hyper-paramètres (et qui la retourne).

2.6 Les hyper-paramètres contrôlent souvent la « **capacité** » d'un algorithme d'apprentissage. Expliquez dans vos propres mots votre compréhension de cette notion de « **capacité** ».

2.7 Numérotez en ordre de **capacité** croissante (de la plus faible à la plus forte), les classifieurs binaires suivants. Vous pouvez supposer par ex. que $d = 10$ et $n = 1000$. Indiquez à côté de chaque modèle son numéro, en partant de 1 pour celui qui a la plus faible capacité, jusqu'à 7 pour celui qui a la plus forte.

a) k-PPV (k-NN) avec $k = 1$

b) réseau de neurones avec un petit nombre d'unités cachées (sans autre forme de régularisation)

c) réseau de neurones avec un grand nombre d'unités cachées (sans autre forme de régularisation)

d) k-PPV avec $k = n$

e) classifieur linéaire de type perceptron

f) régression logistique régularisée avec λ assez grand

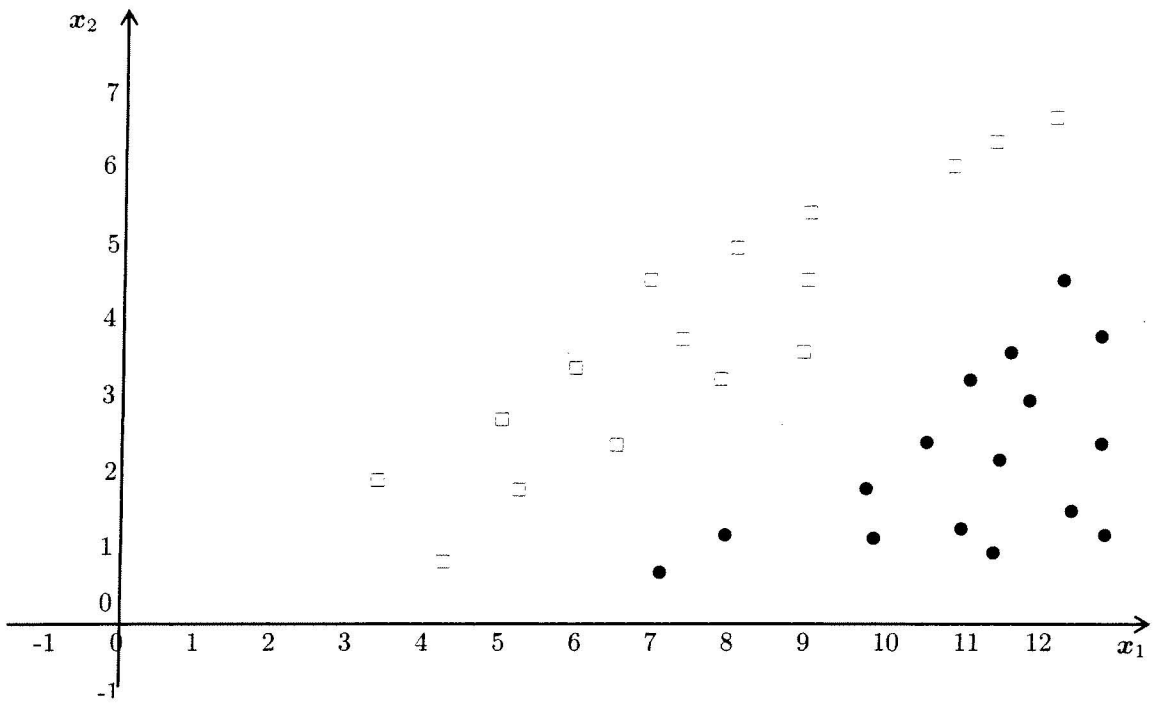
g) perceptron à noyau polynômial de degré 5.

Note: Pour les perceptrons, on suppose qu'on utilise une version modifiée de l'algorithme qui n'a pas le problème de boucler à l'infini si il n'est pas capable de séparer les données: il va s'arrêter en retournant la meilleure solution trouvée.

2.8 Pour un problème particulier, comment pourra-t-on choisir le meilleur algorithme d'apprentissage parmi plusieurs candidats?

3 Arbres de décision (20 pts)

Soit l'ensemble de données d'entraînement suivant (problème de classification binaire en 2 dimension) :



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* de 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) 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.)

e) 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.

f) Quel est le *principal* point faible des algorithmes de type arbre de décision? Connaissez-vous une technique qui, utilisée en conjonction avec les arbres de décision, permet de contrebalancer ou mitiger ce point faible?

4 Apprentisage non-supervisé (20 pts)

4.1 Qu'est-ce qui distingue l'apprentissage non-supervisé de l'apprentissage supervisé?

4.2 Indiquez **les principales tâches** qu'on retrouve en apprentissage **non-supervisé**. Pour chacune, expliquez le **but recherché**, dites **à quoi cela peut servir**, et **nommez 2 algorithmes** d'apprentissage typiques utilisés pour cette tâche.

4.3 Écrivez sous forme de pseudo-code l'algorithme de k-moyennes (k-means)

4.4 Un mélange de Gaussienne peut être représenté sous la forme d'un modèle graphique avec des variables observées et des variables latentes (cachées). Expliquez ce qu'on entend par là: que représentent les variables latentes dans ce cas-ci (on suppose qu'on a un ensemble de donnée D_n tel qu'indiqué au début de l'examen).

4.5 Dessinez le modèle graphique dirigé (réseau Bayesien) correspondant à la factorisation de probabilité suivante:

$P(X) = P(X_3)\, P(X_2|X_3, X_1)\, p(X_1|X_4)\, p(X_5)\, p(X_4|X_3, X_5)$
où $X = (X_1, X_2, X_3, X_4, X_5)$

5 Réseaux de neurones (20 pts)

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 $y = f_\theta(x)$.

5.1 Contrôle de capacité d'un réseau de neurones

Dans un modèle de réseau de neurones 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* du modèle (et le risque de sur-apprentissage) ou au contraire *diminuer la capacité* (et augmenter le risque de sous-apprentissage).

5.2 Calcul du gradient par différence finie

a) Expliquez brièvement dans vos mots la technique de calcul du calcul du gradient par différences finies.

b) En quoi cette technique est-elle utile (pourquoi s'en sert-on typiquement)?

c) Quel est l'intérêt d'implémenter des méthodes de calcul du gradient par rétropropagation (backprop) puisqu'on peut calculer le gradient par différences finies en utilisant juste des propagations avant (forward-prop)?

d) Ecrivez un pseudo-code de haut niveau pour calculer le gradient par différences finies.

5.3 Réseaux de neurones de type Radial Basis Function

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 $h_k = \text{sigmoid}(< w_k, x > + b_k)$, où $< w_k, x >$ 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 $(h_1, ..., h_m) = h$ est typiquement suivie d'une couche de sortie linéaire avec des poids $(a_1, ..., a_m) = a$ pour donner une sortie $y = f_\theta(x) = < a, h > = \sum_{k=1}^{m} a_k h_k$.

5.3.1 Quel est l'ensemble θ des paramètres (excluant les hyper-paramètres) d'un tel réseau RBF?

$$\theta = \{ \qquad\qquad \}$$

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

5.3.2 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 $D_n = \{(x^{(1)}, t^{(1)}), ..., (x^{(n)}, t^{(n)})\}$. Exprimez ce problème de minimisation.

5.3.3 On s'intéresse au gradient, c.a.d la dérivée partielle du coût L par rapport aux paramètres, **on suppose qu'on a déjà calculé** $\frac{\partial L}{\partial y}$, et on va rétropropager le gradient. Exprimez et calculez (en fonction de a_k, h_k, et $\frac{\partial L}{\partial y}$):

$$\frac{\partial L}{\partial a_k} =$$

$$\frac{\partial L}{\partial h_k} =$$

puis, en fonction (entre autres) de $\frac{\partial L}{\partial h_k}$

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

$$\frac{\partial L}{\partial w_{kj}} =$$