FONDEMENTS DE L'APPRENTISSAGE MACHINE (IFT3395)

Professeur: Pascal Vincent

Examen Final

Jeudi 28 juillet 2011

Durée: 2h45

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

Prénom:

Nom:

Code permanent:

Le total de l'examen est sur 100 pts.

Veuillez répondre aux questions dans les zones de blanc laissées à cet effet.

Répondez de manière concise, mais précise.

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: $D_n = \{(x^{(1)}, t^{(1)}), ..., (x^{(n)}, t^{(n)})\}$ dans le cas supervisé (chaque exemple est une paire *observation, cible*) ou bien $D_n = \{x^{(1)}, ..., x^{(n)}\}$ dans le cas non-supervisé, où on n'a pas de notion de cible explicite donc juste un vecteur d'observation. On suppose que chaque observation est constituée de d traits caractéristiques (composantes): $x^{(i)} \in \mathbb{R}^d$: $x^{(i)} = (x_1^{(i)}, ..., x_d^{(i)})$. On vous demande de respecter *scrupuleusement* la notation définie dans cet énoncé.

1 Algorithmes d'apprentissage appropriés pour différents types de problèmes (20 pts)

1.1 Apprentissage supervisé

Indiquez les *différents types de problèmes* d'apprentissage que vous connaissez que l'on considère comme des problèmes d'apprentissage « supervisé ».
Pour *chaque* type de problème: a) expliquez d'abord dans vos mots en quoi il consiste, ce qui le caractérise. **b)** nommez tous les *algorithmes d'apprentissage* que nous avons vus ou que vous connaissez qu'on peut appliquer **directement** pour ce type de problème. Précisez pour chacun de ces algorithmes, un avantage (+) (point fort de l'algorithme) et une limitation (−) (point faible ou inconvénient) et listez ses hyper-paramètres s'il en a (H: ....)

1.2 Apprentissage non supervisé

Indiquez les *différents types de problèmes* d'apprentissage que vous connaissez que l'on considère comme des problèmes d'apprentissage « **non supervisé** ».
Pour *chaque* type de problème: a) expliquez dans vos mots en quoi il consiste. **b)** indiquez à quoi cela peut servir en pratique. **c)** nommez tous les *algorithmes d'apprentissage* que nous avons vus ou que vous connaissez qu'on peut appliquer pour ce type de problème (une liste de noms suffit).

2 Hyper-paramètres, capacité et sélection de modèle (20 pts)

a) Expliquez, dans vos propres mots, le phénomène de sur-apprentissage.

b) Expliquez, dans vos propres mots, le phénomène de sous-apprentissage.

c) Expliquez dans vos propres mots la notion de « capacité » d'un algorithme d'apprentissage (ou modèle), et sa relation avec les phénomènes de sur-apprentissage et sous-apprentissage.

d) BONUS: quel est le lien entre le choix de la capacité d'un algorithme d'apprentissage et le dilemme biais-variance.

e) La plupart des algorithmes d'apprentissage ont des hyper-paramètres permettant plus ou moins de contrôler leur « capacité » 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? élvée? 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.

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

g) 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: $\theta = A(D, \lambda)$, et pour la prédiction du modèle entraîné en un point x: $f_\theta(x)$. 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 $B(D_n, H)$ qui effectuera la sélection de la valeur la plus prometteuse λ^* des hyper-paramètres parmi une liste H de valeurs possibles fournie à l'algorithme ($\lambda \in H$). 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.

3 Modèles graphique probabiliste et mélange de Gaussienne (20 pts)

a) Dessinez le modèle graphique dirigé correspondant à la factorisation suivante de la probabilité jointe entre 5 variables: $P(X_1, X_2, X_3, X_4, X_5) = P\big(X_3|X_5\big)P(X_4)P\big(X_5|X_1, X_4, X_2\big)P(X_2)P\big(X_1|X_2\big)$

b) Écrivez sous forme de pseudo-code, comment on pourrait générer un exemple de cette distribution jointe, en supposant qu'on sait générer de ses facteurs. Par exemple pour signifier qu'on génère une valeur de X_1 selon la loi $P\big(X_1|X_2\big)$ sachant que $X_2 = x_2$ on écrira $x_1 \sim P\big(X_1|X_2 = x_2\big)$.

```
Génére_P():
```

```
return (x_1, x_2, x_3, x_4, x_5)
```

c) Un mélange de Gaussienne peut être représenté sous la forme d'un modèle graphique dirigé avec des variables observées et des variables latentes (cachées). Expliquez ce qu'on entend par variable latente. Dessinez le graphe correspondant, et expliquez à quoi correspond la variable associée à chaque noeud, sa nature ainsi que sa dimensionalité (on suppose qu'on va l'utiliser sur un ensemble de donnée D_n tel qu'indiqué au début de l'examen).

d) A quoi sert un tel mélange de Gaussienne: pour quel(s) type(s) de problèmes d'apprentissage peut-on s'en servir?

e) A quel(s) autre(s) algorithmes d'apprentissage s'apparente le mélange de Gaussiennes?

f) Expliquez en quoi consiste le problème de l'inférence dans un modèle graphique en général, et dans le mélange de Gaussienne en particulier.

g) Nommez deux approches algorithmiques différentes permettant d'apprendre les paramètres d'un mélange de Gaussienne.

4 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 $y = f_\theta(x)$.

4.1 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* ($\uparrow$) du modèle (et le risque de sur-apprentissage) ou au contraire *diminuer la capacité* ($\downarrow$) du modèle (et augmenter le risque de sous-apprentissage).

4.2 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)?

4.3 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:

$$\begin{aligned} h_k &= \exp\left(-\beta \|x - w_k\|^2\right) \\ &= \exp\left(-\beta \sum_{j=1}^{d} (x_j - w_{kj})^2\right) \end{aligned}$$

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 $y = f_\theta(x) = < a, h > = \sum_{k=1}^{m} a_k h_k$.

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

$\theta = \{$ $\}$

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

4.3.2 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.

4.3.3 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:* $\frac{\partial \exp(u)}{\partial \theta} = \frac{\partial u}{\partial \theta} \exp(u)$

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

5 Perceptron et astuce du noyau (20 pts)

On se limite ici à des problèmes de classification binaire avec la cible $t \in \{-1, +1\}$.

a) Donnez la forme de la fonction de décision (classification) qu'on obtient avec un algorithme de type perceptron ordinaire (sans noyau).

$f(x) =$

Précisez-en les paramètres et leur dimension.

Quelle est la complexité algorithmique de la décision sur un point de test?

b) Quelle est la limitation de l'algorithme du perceptron original que permet de dépasser le perceptron à noyau?

c) Donnez la forme de la fonction de décision qu'on obtient, pour un poblème de classification binaire, avec un algorithme à noyau du type Perceptron à noyau (ou SVM machine à vecteur de support à noyau) pour un noyau K.

$f(x) =$

Précisez-en les paramètres et leur nombre et dimension.

Quelle est la complexité algorithmique de la décision sur un point de test (en supposant que la complexité du calcul de $K(a, b)$ est $O(d)$)?

d) On rappelle qu'un noyau $K(a, b)$ peut être vu comme une astuce pour calculer efficacement un produit scalaire $< \varphi(a), \varphi(b) >$ quand on ne peut se permettre de calculer explicitement les $\varphi(a)$. Exprimez les **paramètres** dans votre formulation classique a) en fonction des **paramètres** de la formulation à noyau c) (et de la fonction φ).

e) On rappelle dans ses grandes lignes l'algorithme du perceptron « apprentissage en ligne » ordinaire:

```
Initialiser les paramètres (vecteur de poids et biais) à 0.
Boucler sur les exemples (x, t) de l'ensemble d'apprentissage:
  Classifier x avec les paramètres actuels
    Si bien classifié, ne rien faire,
    Si mal classifié, ajouter tx au vecteur de poids et t au biais
Jusqu'à ce que tous les points soient bien classifiés (ou qu'on ait atteint un
nombre maximal d'itérations)
```

Précisez-le sous la forme d'un pseudo-code plus détaillé, en utilisant la paramétrisation que vous avez donné dans votre formulation a)

f) Donnez un pseudo-code détaillé correspondant à la version à noyau de cet algorithme, permettant d'apprendre les paramètres de votre formulation à noyau c).