Prénom et nom de l'étudiant: _____

Code permanent: _____

IFT 3395 ou 6390?

FACULTE DES ARTS ET DES SCIENCES
DEPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPERATIONNELLE

TITRE DU COURS: **Fondements de l'apprentissage machine**
SIGLE DU COURS : **IFT3395/6390 A10**

NOM DU PROFESSEUR: Yoshua Bengio

DATE DE L'EXAMEN FINAL A10: 16 décembre 2010 HEURE: 9h30 - 12h30
SALLE: AA-1177

DIRECTIVES PÉDAGOGIQUES: - Documentation permise.
- Répondre directement sur le questionnaire. Vous pouvez utiliser l'arrière des pages aussi si vous en avez besoin.
- Soyez brefs et précis dans vos réponses.
- **Si vous manquez de temps: l'important est de montrer que vous avez compris le problème, plutôt que les détails de la réponse.**
- Échanger des informations lors d'un examen (ou autres formes de tricherie) est du **plagiat**, qui est passible de sanctions allant jusqu'à l'exclusion du programme.
- Suggestion: commencez par tout lire, et faire les questions les plus faciles.

---

1. (9 points) Nommez au moins 3 hyper-paramètres des réseaux de neurones multi-couches (points **boni** si vous en trouvez 4).

2. (18 points) Pour chacun d'entre eux, expliquez leur effet présumé sur l'erreur d'apprentissage (tends à l'augmenter, à la diminuer, ou on attend une augmentation suivie d'une diminution, ou bien une diminution suivie d'une augmentation). Points **boni** si vous faites la même chose pour un 4ème hyper-paramètre.

3. (8 points) Pourquoi est-ce qu'initialiser les paramètres d'un réseau de neurones tous à 0 est déconseillé?

4. (8 points) Comment initialiser les paramètres d'un réseau de neurones? quels est l'objectif que l'on cherche à atteindre?

5. (16 points) Considérez un réseau de neurones qui prédit non seulement l'espérance conditionnelle $\mu(x) = E[Y|x] = b + w'h(x)$ (avec $h(x) = \tanh(Vx)$) mais aussi la variance conditionnelle $\sigma^2(x) = Var[Y|x] = \log(1 + e^{c+u'h(x)})$. Le critère d'entraînement naturel pour un tel modèle est la log-vraisemblance conditionnelle de $Y = y$ étant donné x, où l'on suppose que $P(Y|x)$ est une normale d'espérance $\mu(x)$ et de variance $\sigma^2(x)$. Écrivez d'abord la fonction de perte L (comme une fonction des paramètres $\theta = (w, u, V, b, c)$). Montrez ensuite comment calculer le gradient dans ce réseau, en vous arrêtant au calcul de $\frac{\partial L}{\partial b}$ et $\frac{\partial L}{\partial c}$ (puisqu'à partir de là c'est comme pour un réseau ordinaire).

6. (9 points) Quel est l'avantage computationnel de l'arrêt prématuré (early stopping) par rapport au contrôle de capacité obtenu avec d'autres hyper-paramètres?

7. (8 points) Pourquoi est-ce que les arbres de décision ont un avantage computationel au moment de faire une prédiction, par rapport aux autres algorithmes d'apprentissage vu dans le cours?

8. (14 points) Les classifieurs ou régression utilisant le truc du noyau sont avantageux d'une part par rapport à leur contrepartie linéaire et d'autre part par rapport aux réseaux de neurones. Quelles sont les avantages invoqués dans chacune de ces deux comparaisons? Pour illustrer votre argument, faites un dessin montrant schématiquement une différence qualitative entre la fonction de perte (en fonction des paramètres) dans un cas vs l'autre, en expliquant pourquoi cette différence peut être pertinente.

9. (10 points) Écrivez un pseudo-code générique qui permet de choisir des valeurs pour les hyper-paramètres λ d'un algorithme d'apprentissage A. Celui-ci est spécifié par une méthode `A.train(data,`λ`)` (avec effet secondaire) et une méthode `A.test(data)` (qui retourne l'erreur mesurée sur l'ensemble `data` donné en argument). L'usager spécifie un ensemble S de valeurs possibles pour λ.

10. (15 points **boni**) Écrivez sous forme de pseudo-code commenté et de haut niveau l'algorithme de descente de gradient stochastique pour le modèle de la question 5. Les hyper-paramètres sont fixes. Divisez votre code en terme des fonctions suivantes (qui peuvent s'appeler les unes les autres):

    - `initialize()` ; initialise les paramètres
    - `train(data)` ; entraîne à partir des données
    - `update(`x, y`)` ; met à jour les paramètres pour l'exemple (x, y)
    - `gradient(`x, y`)` ; calcule le gradient de la fonction de perte pour (x, y)