IFT 3395/6390 Hiver 2008
Fondements de l'apprentissage machine
Professeur: Pascal Vincent

Note: N'attendez pas la date de remise du devoir 2 pour commencer à travailler sur le devoir 3 (qui en est la suite)!

Pour remettre tous vos fichiers (dont un rapport.pdf) connectez vous sur la machine remise (`ssh remise`) puis utilisez la commande `remise`.

- Si vous êtes en IFT3395 faites par ex.:
  `remise ift3395 tp2 rapport.pdf code.r`
- Si vous êtes en IFT6390 faites par ex.:
  `remise ift6390 tp2 rapport.pdf code.r`

Devoir 2

à remettre au plus tard le lundi 31 mars 2008 à minuit.

1 Séparabilité linéaire

On suppose qu'on dispose d'un sensemble de données $D_n = \{(x_1, y_1), ..., (x_n, y_n)\}$ avec $x_i \in \mathbb{R}^d$ et $y_i \in \{-1, +1\}$ indiquant la classe.

a) Écrivez la forme d'une *fonction discriminante linéaire* g pour ce problème en précisant ses paramètres, ainsi que leur nature et leur dimension. A combien de paramètres scalaires cela correspond-t il?

b) Que veut-on dire lorsque l'on dit que les points de D_n sont linéairement séparables? Expliquez-le d'abord en Français dans vos propres mots, puis sous forme d'une expression mathématique.

c) Écrivez l'expression du risque empirique (associé à la fonction discriminante g) qui comptabiliserait le taux d'erreur de classification.

d) On définit la distance entre un point x et un ensemble de points comme la distance entre x et le point le plus proche de l'ensemble. On cherche la distance entre un point x et la frontière de décision induite par la fonction discriminante g. Exprimez le calcul de cette distance comme un problème d'optimisation (minimisation avec contrainte) et résolvez-le.

e) On suppose que D_n comporte au moins un point de chaque classe, que tous les x_i sont distincts, et qu'ils sont répartis sans former aucun "alignement" particulier (aucun $n' \leqslant n$ points ne tombent sur un sous-espace linéaire de dimension $n' - 2$). Montrez que si $n \leqslant d + 1$, les points de D_n sont toujours linéairement séparables, avec n'importe quelle affectation des étiquettes binaires (les y_i).

2 Apprentissage des paramètres d'un réseau de neurones

On suppose qu'on dispose d'un ensemble de données $D_n = \{(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_n}, y_n)\}$ avec $x_i \in \mathbb{R}^d$ et $y_i \in \{1, ..., m\}$ indiquant la classe parmi m classes.

Soit un réseau de neurones de type *Perceptron multicouche* avec une seule couche cachée (donc 3 couches en tout si on compte la couche d'entrée et la couche de sortie). La couche cachée est constituée de d_h neurones complètement connectés à la couche d'entrée. Ils ont une non-linéarité de type tangente hyperbolique (tanh). La couche de sortie est constituée de m neurones, complètement connectés à la couche cachée. Ils ont une non-linéarité de type softmax. La sortie du $j^{\text{ème}}$ neurone de la couche de sortie donnera un score pour la classe j interprété comme la probabilité que l'entrée x soit de cette classe j.

Il vous est fortement conseillé de dessiner le réseau de neurones au fur et à mesure afin que vous puissiez mieux suivre les étapes (mais pas besoin de nous fournir un dessin!)

a) Soit $\boldsymbol{W}^{(1)}$ la matrice $d_h \times d$ de poids et soit $\boldsymbol{b}^{(1)}$ le vecteur de biais caractérisant des connexions synaptiques allant de la couche d'entrée à la couche cachée. Indiquez la dimension de $\boldsymbol{b}^{(1)}$. Donnez la formule de calcul du vecteur d'activations (i.e. avant non-linéarité) des neurones de la couche cachée $\boldsymbol{h}^a$ à partir d'une observation d'entrée $\boldsymbol{x}$, d'abord sous la forme d'une expression de calcul matriciel, puis détaillez le calcul d'un éléments $\boldsymbol{h}^a_j$. Exprimez le vecteur des sorties des neurones de la couche cachée $\boldsymbol{h}^s$ en fonction de $\boldsymbol{h}^a$.

b) Soit $\boldsymbol{W}^{(2)}$ la matrice de poids et soit $\boldsymbol{b}^{(2)}$ le vecteur de biais caractérisant les connexions synaptiques allant de la couche cachée à la couche de sortie. Indiquez les dimensions de $\boldsymbol{W}^{(2)}$ et $\boldsymbol{b}^{(2)}$. Donnez la formule de calcul du vecteur d'activations des neurones de la couche de sortie $\boldsymbol{o}^a$ à partir de leurs entrées $\boldsymbol{h}^s$ sous la forme d'une expression de calcul matriciel, puis détaillez le calcul de $\boldsymbol{o}^a_k$.

c) La sortie des neurones de sortie est donnée par

$$\begin{aligned} \boldsymbol{o}^s &= \text{softmax}(\boldsymbol{o}^a) \\ \boldsymbol{o}^s_k &= \frac{\exp(\boldsymbol{o}^a_k)}{\sum_{k'=1}^{m} \exp(\boldsymbol{o}^a_{k'})} \end{aligned}$$

Montrez que les $\boldsymbol{o}^s_k$ sont positifs et somment à 1. Pourquoi est-ce important?

d) Le réseau de neurones calcule donc, pour un vecteur d'entrée $\boldsymbol{x}$, un vecteur de scores (probabilités) $\boldsymbol{o}^s(\boldsymbol{x})$. La probabilité, calculée par le réseau de neurones, qu'une observation $\boldsymbol{x}$ soit de la classe y est donc donnée par la $y^{i\text{ème}}$ sortie $\boldsymbol{o}^s_y(\boldsymbol{x})$. Ceci suggère d'utiliser la fonction de perte:

$$L(\boldsymbol{x}, y) = \text{-log } \boldsymbol{o}^s_y(\boldsymbol{x})$$

L'entraînement du réseau de neurones va consister à trouver les paramètres du réseau qui minimisent le risque empirique $\hat{R}$ correspondant à cette fonction de perte. Formulez $\hat{R}$. Indiquez précisément de quoi est constitué l'ensemble θ des paramètres du réseau. Indiquez à combien de paramètres scalaires n_θ cela correspond. Formulez le problème d'optimisation qui correspond à l'entraînement du réseau permettant de trouver une valeur optimale des paramètres.

e) Pour trouver la solution à ce problème d'optimisation, on va utiliser une technique de descente de gradient. Exprimez sous forme d'un bref pseudo-code la technique de descente de gradient (batch) pour ce problème.

f) Notez que le calcul du vecteur de gradient du risque empirique $\hat{R}$ par rapport à l'ensemble des paramètres θ peut s'exprimer comme

$$\begin{pmatrix} \frac{\partial \hat{R}}{\partial \theta_1} \\ \vdots \\ \frac{\partial \hat{R}}{\partial \theta_{n_\theta}} \end{pmatrix} = \frac{1}{n} \sum_{i=1}^{n} \begin{pmatrix} \frac{\partial L(\boldsymbol{x}_i, y_i)}{\partial \theta_1} \\ \vdots \\ \frac{\partial L(\boldsymbol{x}_i, y_i)}{\partial \theta_{n_\theta}} \end{pmatrix}$$

Il suffit donc de savoir calculer le gradient du coût $C = L(\boldsymbol{x}, y)$ encouru pour un exemple $(\boldsymbol{x}, y)$, que l'on définit comme:

$$\frac{\partial C}{\partial \theta} = \begin{pmatrix} \frac{\partial C}{\partial \theta_1} \\ \vdots \\ \frac{\partial C}{\partial \theta_{n_\theta}} \end{pmatrix} = \begin{pmatrix} \frac{\partial L(\boldsymbol{x}, y)}{\partial \theta_1} \\ \vdots \\ \frac{\partial L(\boldsymbol{x}, y)}{\partial \theta_{n_\theta}} \end{pmatrix}$$

Pour cela on va appliquer la technique de rétropropagation du gradient, en partant du coût C, et en remontant de proche en proche vers la sortie $\boldsymbol{o}$ puis vers la couche cachée $\boldsymbol{h}$ et enfin vers l'entrée $\boldsymbol{x}$.

Commencez donc par calculer le vecteur de dérivées partielles du coût C par rapport aux sorties des neurones de la couche de sortie:

$$\frac{\partial C}{\partial \boldsymbol{o}^s} = \begin{pmatrix} \frac{\partial C}{\partial \boldsymbol{o}^s_1} \\ \vdots \\ \frac{\partial C}{\partial \boldsymbol{o}^s_m} \end{pmatrix}$$

Dans ce qui suit, quand on demande de "calculer" des gradients ou dérivées paritielles, il s'agit simplement d'exprimer leur calcul en fonction d'éléments déjà calculés aux questions précédentes (ne substituez pas les expressions de dérivées partielles déjà calculées lors des questions d'avant!)

g) Calculez les dérivées partielles par rapport aux activations des neurones de sortie (aussi appelées *sensibilité* des neurones de sortie):

$$\frac{\partial C}{\partial \boldsymbol{o}^a} = \begin{pmatrix} \frac{\partial C}{\partial \boldsymbol{o}^a_1} \\ \vdots \\ \frac{\partial C}{\partial \boldsymbol{o}^a_m} \end{pmatrix}$$

Notez que comme C ne dépend des activations $\boldsymbol{o}^a$ qu'indirectement par l'intermédiaire des sorties $\boldsymbol{o}^s$ il nous faut utiliser la règle de dérivation en chaîne:

$$\frac{\partial C}{\partial \boldsymbol{o}^a_k} = \sum_{k'=1}^{m} \frac{\partial C}{\partial \boldsymbol{o}^s_{k'}} \frac{\partial \boldsymbol{o}^s_{k'}}{\partial \boldsymbol{o}^a_k}$$

h) Calculez les gradients par rapport aux paramètres $\boldsymbol{W}^{(2)}$ et $\boldsymbol{b}^{(2)}$ de la couche de sortie. Comme C ne dépend des $\boldsymbol{W}^{(2)}_{kj}$ et $\boldsymbol{b}^{(2)}_k$qu'au travers de $\boldsymbol{o}^a_k$ la règle de dérivation en chaîne nous donne:

$$\frac{\partial C}{\partial \boldsymbol{W}^{(2)}_{kj}} = \frac{\partial C}{\partial \boldsymbol{o}^a_k} \frac{\partial \boldsymbol{o}^a_k}{\partial \boldsymbol{W}^{(2)}_{kj}}$$

et

$$\frac{\partial C}{\partial \boldsymbol{b}^{(2)}_k} = \frac{\partial C}{\partial \boldsymbol{o}^a_k} \frac{\partial \boldsymbol{o}^a_k}{\partial \boldsymbol{b}^{(2)}_k}$$

i) Calculez les dérivées partielles du coût C par rapport aux sorties des neurones de la couche cachée. Comme C dépend d'un neurone caché $\boldsymbol{h}^s_j$ au travers des activations de tous les neurones de sortie $\boldsymbol{o}^a$ reliés à ce neurone caché, la règle de dérivation en chaîne nous donne:

$$\frac{\partial C}{\partial \boldsymbol{h}^s_j} = \sum_{k=1}^{m} \frac{\partial C}{\partial \boldsymbol{o}^a_k} \frac{\partial \boldsymbol{o}^a_k}{\partial \boldsymbol{h}^s_j}$$

j) Calculez les dérivées partielles par rapport aux activations des neurones de la couche cachée. Comme C ne dépend de l'activation $\boldsymbol{h}^a_j$ d'un neurone de la couche cachée qu'au travers de la sortie $\boldsymbol{h}^s_j$ de ce neurone, la règle de dérivation en chaîne donne:

$$\frac{\partial C}{\partial \boldsymbol{h}^a_j} = \frac{\partial C}{\partial \boldsymbol{h}^a_j} \frac{\partial \boldsymbol{h}^s_j}{\partial \boldsymbol{h}^a_j}$$

Notez que $\boldsymbol{h}^s_j = \tanh(\boldsymbol{h}^a_j)$, où la tangente hyperbolique s'applique élément par élément. La formule de la tangente hyperbolique est: $\tanh(z) = \frac{\sinh z}{\cosh z} = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z}-1}{e^{2z}+1}$. Comme étape intermédiaire, commencez par démontrer que $\frac{\partial \tanh(z)}{\partial z} = \tanh'(z) = 1 - \tanh^2(z)$.

k) Calculez les gradients par rapport aux paramètres $\boldsymbol{W}^{(1)}$ et $\boldsymbol{b}^{(1)}$ de la couche cachée.

l) Calculez les dérivées partielles du coût C par rapport au vecteur d'entrée $\boldsymbol{x}$.

m) Nous allons maintenant considérer un risque empirique **régularisé**: $\tilde{R} = \hat{R} + \lambda\,\mathcal{L}(\theta)$, où θ est le vecteur de tous les paramètres du réseau et $\mathcal{L}(\theta)$ calcule une pénalité scalaire en fonction des paramètres θ, plus ou moins importante selon une préférence à priori qu'on a sur les valeurs de θ. λ est un hyper-paramètre (scalaire, positif ou nul) qui contrôle le compromis entre trouver des valeurs des paramètres qui minimisent le risque empirique ou qui minimisent cette pénalité. On va considérer ici une régularization de type "weight decay" quadratique qui pénalise la norme carrée (norme L_2) des poids (mais pas des biais):

$$\begin{aligned}\mathcal{L}(\theta) &= \|\boldsymbol{W}^{(1)}\|^2 + \|\boldsymbol{W}^{(2)}\|^2 \\ &= \sum_{i,j}\left(\boldsymbol{W}^{(1)}_{ij}\right)^2 + \sum_{i',j'}\left(\boldsymbol{W}^{(2)}_{i'j'}\right)^2\end{aligned}$$

On veut en fait minimiser le risque régularisé $\tilde{R}$ plutôt que $\hat{R}$. Que devient le gradient par rapport aux différents paramètres?