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!

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 tp3 rapport.pdf code.r`
- Si vous êtes en IFT6390 faites par ex.:
  `remise ift6390 tp3 rapport.pdf code.r`

Devoir 3

à remettre au plus tard le lundi 14 avril 2008 à minuit.

1 Implémentation d'un réseau de neurones

On vous demande d'implémenter un réseau de neurones avec le calcul du gradient pas à pas tel que vous l'avez dérivé dans le devoir 2 (donc incluant le weight decay). Vous ne pouvez pas utiliser une impémentation existante de réseau de neurones, mais devez suivre la structure de la derivation du devoir 2 (avec des noms de variables s'en inspirant, etc.). Vous pouvez l'implementer dans un des langages suivants: R, C, C++, Java, Python, Matlab, ou tout autre langage pour lequel vous aurez au préalable obtenu l'approbation des démos. Notez qu'avec R vous avez déjà toute une structure d'algorithme d'apprentissage, de calcul et d'affichage de régions de décision 2D dont vous pouvez vous inspirer et peut-être (partiellement) réutiliser, même si vous implémentez votre réseau dans un autre langage...

Détails utiles sur l'implémentation:

- **Initialisation des paramètres.** Comme vous le savez, il est nécessaire d'initialiser aléatoirement les paramètres du réseau (dans le but d'éviter les symétries et la saturation des neurones et idéalement pour se situer au point d'inflexion de la non-linéarité de façon à avoir un comportement non-linéaire). Nous vous proposons d'initialier les *poids* d'une couche en les tirant d'une uniforme sur $\left[\frac{-1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$, où n est le nombre d'entrées de la couche. Les *biais* peuvent quant à eux être initialisés à 0. Justifiez votre choix de toute autre initialisation.
- **Taille des lots**. On demande que votre descente de gradient opère sur des mini-lots (minibatch, par opposition à batch) de taille ajustable par un hyper-paramètre. Dans le cas de mini-lots, on ne manipule pas un unique vecteur d'entrée, mais plutôt un lot de vecteurs d'entrée, groupés dans une matrice (qui donneront de même une matrice au niveau de la couche cachée, et de la sortie). Dans le cas d'une taille de lot de un, on obtient l'équi8valent d'un gradient stochastique. Étant donné que R est efficace sur les opérations matricielles, il est possible de faire les calculs efficacement pour un mini-lot au complet. Ceci affecte grandement le temps d'exécution.
- **fprop** et **bprop**. On vous suggère d'écrire des méthodes fprop et bprop. fprop fait la propagation "avant" c.a.d. le calcul étape par étape depuis l'entrée, jusqu'à la sortie et au coût, des activations de chaque couche. bprop se sert des activations calculées par le précédent appel à fprop et effectue la rétropropagation du gradient depuis le coût jusqu'à l'entrée en suivant les étapes que vous avez normalement dérivées dans le devoir 2.

- **Verification du gradient par différence finie.** On peut estimer le gradient numériquement par différences finies. Vous devez implémenter cette estimation de façon à vérifier votre calcul du gradient (c'est un outil de développement). Pour ce faire, calculez d'abord la valeur de la perte pour la valeur courante des paramètres (sur un exemple, ou un lot). Ensuite, pour chaque paramètre θ_k (un scalaire), modifiez ce paramètre par une petite valeur ($10^{-6} < \varepsilon < 10^{-4}$) et recalculez la perte (même exemple ou lot) puis ramenez le paramètre à sa valeur de départ . La dérivée partielle (le gradient) par rapport à chaque paramètre est alors estimé en divisant la variation de la perte par ε. Le ratio de votre gradient calculé par rétropropagationet du gradient estimé par différence finie devrait se situer entre 0.99 et 1.01.

Manipulations:

1. Verification du gradient : produire un exemple d'affichage de verification du gradient par difference finie pour votre réseau (pour un petit réseau, par ex. $d = 2$ et $d_h = 2$).
2. Entrainer votre réseau de neurones sur les données du problème des deux-lunes (disponible sur la page des démos). Afficher les régions de décision pour differentes valeurs d'hyper-parametres (weight decay, nombre d'unites cachées, arrêt prématuré) de façon à illustrer leur effet sur le contrôle de capacité.
3. Entrainer votre réseau sur les données de MNIST (disponible sur la page des démos) et produire les courbes d'entrainement, de validation et de test. Faites la sélection des paramètres de façon à obtenir de bons résultats sur l'ensemble de validation (moins de 5%). Points boni pour une erreur de test inférieure à 2%.