Département de Génie Informatique

et Génie Logiciel

INF8225

Laboratoire

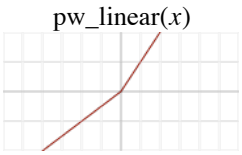
TP 3

Partie I

Lisez le chapitre 10.2 des notes supplémentaires et écrivez quelque pseudocode pour implémenter l'optimisation d'un réseau de neurones avec deux couches cachées utilisant les calculs matriciels et la descente du gradient stochastique par miniensemble.

Partie II

L'objectif de ce travail pratique est d'implémenter un réseau de neurones avec *deux* couches cachées et d'exécuter des expériences comparant avec un modèle possédant une seule couche cachée. Utilisez l'approche de descente de gradient stochastique par miniensemble «mini-batch stochastic gradient descent». Utilisez les fonctions d'activation du type linéaire par morceaux « piecewise linear » pour les couches cachées, et une fonction de perte basée sur la fonction « softmax ».



$$h(x) = \begin{cases} x & \text{if x} \geq 0 \\ ax & \text{if x} < 0 \end{cases} \qquad h'(x) = \begin{cases} 1 & \text{if x} \geq 0 \\ a & \text{if x} < 0 \end{cases}$$

Sélectionnez une valeur pour *a* utilisant l'ensemble de validation, commençant avec *a*=0.1.

Écrivez votre logiciel en MATLAB ou Python. Optimisez les paramètres de vos modèles utilisant un ensemble de données d'apprentissage et utilisez un ensemble de validation pour déterminer quant à arrêter l'optimisation. Montrez quelques courbes d'apprentissage dans votre rapport et donnez les performances sur l'ensemble du test.

Vous devez remettre votre code et un rapport avec vos expériences comparant la performance d'un réseau avec deux couches cachées avec un réseau utilisant une seule couche cachée.

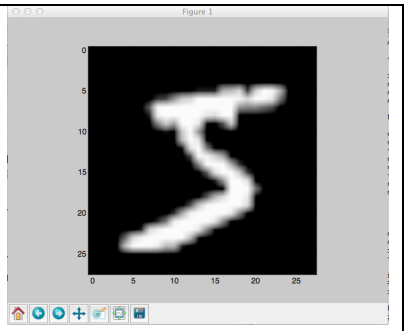
Vous devez exécuter vos expériences utilisant les données de MNIST disponible dans le fichier : http://www.iro.umontreal.ca/~lisa/deep/data/mnist/mnist.pkl.gz
ou sur moodle où il y a aussi un fichier : mnist_all.mat avec une structure de données similaire.

```
  Attr Name        Size                  Bytes  Class
==== ====        ====                  =====  =====
     test_x   50000x784               39200000  uint8
     test_y       1x50000               400000  int64
     train_x  50000x784               39200000  uint8
     train_y      1x50000               400000  int64
     valid_x  10000x784                7840000  uint8
     valid_y      1x10000                 80000  int64
```

Vous pouvez voir un exemplaire avec des commands en Python suivants :

```
import gzip
import pickle
with gzip.open('mnist.pkl.gz', 'rb') as f:
    train_set, valid_set, test_set = pickle.load(f)

train_x, train_y = train_set
import matplotlib.cm as cm
import matplotlib.pyplot as plt
plt.imshow(train_x[0].reshape((28, 28)), cmap = cm.Greys_r)
plt.show()
```



Vous pouvez utiliser l'exemple écrit avec Theano ici comme un guide :
http://deeplearning.net/tutorial/mlp.html

Cependant, vous devez coder les équations pour l'optimisation des modèles utilisant votre pseudocode ci-dessus pour vous guider et les opérations matricielles discutées en classe et dans les notes de cours supplémentaires pour les gradients.

Barème de correction TP 3 (À remettre au début de notre prochain laboratoire)	
Description des requis	**Points alloués**
Partie I, le pseudocode	**5**
Partie II, le code	**15**
Partie II, le rapport avec vos expériences	**10**
Total	**30**