Département de Génie Informatique

et Génie Logiciel

INF8225

Laboratoire

TP 2

L'objectif de ce travail pratique est de permettre à l'étudiant de se familiariser avec l'apprentissage automatique via la régression logistique - aussi connue comme une perception d'une seule couche, optimisée avec une fonction de perte donnée par le «soft-max». Nous allons utiliser l'approche de descente du gradient « gradient descent » aussi connue comme l'approche de la plus profonde descente.

L'objectif de la partie I est d'implémenter de comparer a) l'approche de descente du gradient avec b) une variation s'appelle l'algorithme de descente de gradient stochastique par mini-ensemble «mini-batch stochastic gradient descent». Votre but est d'écrire un logiciel en MATLAB ou Python pour optimiser les paramètres d'un modèle étant donné un ensemble de données d'apprentissage, utilisant un ensemble de validation pour déterminer quand arrêter l'optimisation et de montrer la performance sur l'ensemble du test.

Vous devez remettre votre code et un rapport avec vos expériences comparant les deux méthodes d'optimisation différentes. Plus de détails concernant les éléments qui doivent être donnés dans le rapport sont discutés ci-dessous.

Sur moodle, vous trouverez le fichier MATLAB suivant : 20news_w100.mat[1][2]. Ce fichier contient les structures de données suivantes :

```
>> load 20news_w100
>> whos
```

Name	Size	Bytes	Class	Attributes
documents	100x16242	719003	logical	sparse
groupnames	1x4	492	cell	
newsgroups	1x16242	129936	double	
wordlist	1x100	12314	cell	

Dans le répertoire suivant il y a des fichiers X.txt pour chaque variable nommée ci-dessus :
http://www.professeurs.polymtl.ca/christopher.pal/Samsung_Lectures/
ex. documents.txt, groupnames.txt, etc.

[1] Aussi ici: http://www.cs.nyu.edu/~roweis/data/20news_w100.mat
Merci à Sam Roweis pour cet traitement de ces données entre 4 classes en utilisant http://people.csail.mit.edu/jrennie/20Newsgroups/

[2] Noter: comme discuté à http://www.cs.nyu.edu/~roweis/data.html les données sont « A tiny version of the 20newsgroups data, with binary occurrence data for 100 words across 16242 postings. I've also tagged the postings by the highest level domain in the array "newsgroups"».

La régression logistique et apprentissage

Il est possible d'encoder l'information concernant l'étiquetage avec des vecteurs multinomiaux, c.-à-d. un vecteur de zéros avec un seul 1 pour indiquer quand la classe C=*k dans la dimension k*, comme $\mathbf{y} = [0 \quad 1 \quad 0 \quad \dots \quad 0]^T$ pour répresenter la deuxieme classe. Les caractéristiques sont données par des vecteurs $\mathbf{x}_i$ et donc les données consistent de $\mathbf{y}_i$, $\mathbf{x}_i$, i=1…N.

Puis, en définissant les paramètres de notre modèle comme : $\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \dots \quad \mathbf{w}_K]^T$, $\mathbf{b} = [b_1 \quad b_2 \quad \dots \quad b_K]^T$, on peut exprimer notre modèle sous la forme :

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{\exp\left(\mathbf{y}^T \mathbf{W}^T \mathbf{x} + \mathbf{y}^T \mathbf{b}\right)}{\sum_{\mathbf{y}} \exp\left(\mathbf{y}^T \mathbf{W}^T \mathbf{x} + \mathbf{y}^T \mathbf{b}\right)}. \tag{1}$$

Puis, en posant $\mathbf{x} = [\mathbf{x}^T \; 1]^T$, et :

$$\boldsymbol{\theta} = [\mathbf{W} \;\; \mathbf{b}] = \begin{bmatrix} \mathbf{w}_1^T & b_1 \\ \mathbf{w}_2^T & b_2 \\ \vdots & \vdots \\ \mathbf{w}_k^T & b_k \end{bmatrix}, \tag{2}$$

on peut exprimer notre modèle sous une forme très compacte :

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{\exp\left(\mathbf{y}^T \boldsymbol{\theta} \mathbf{x}\right)}{\sum_{\mathbf{y}} \exp\left(\mathbf{y}^T \boldsymbol{\theta} \mathbf{x}\right)} = \frac{1}{Z(\mathbf{x})} \exp\left(\mathbf{y}^T \boldsymbol{\theta} \mathbf{x}\right). \tag{3}$$

Comme nous avons vu en classe, nous pouvons alors dériver le log probabilité de notre modèle et obtenir une forme analytique pour le gradient :

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}} \log \prod_i p(\tilde{\mathbf{y}}_i \mid \tilde{\mathbf{x}}_i; \boldsymbol{\theta}) &= \sum_{i=1}^{N} \left[\frac{\partial}{\partial \boldsymbol{\theta}} (\tilde{\mathbf{y}}_i{}^T \boldsymbol{\theta} \tilde{\mathbf{x}}_i) - \frac{\partial}{\partial \boldsymbol{\theta}} \log Z(\tilde{\mathbf{x}}_i) \right] \\ &= \sum_{i=1}^{N} \tilde{\mathbf{y}}_i \tilde{\mathbf{x}}_i^T - \sum_{i=1}^{N} \sum_{\mathbf{y}} P(\mathbf{y} \mid \tilde{\mathbf{x}}_i) \mathbf{y} \tilde{\mathbf{x}}_i^T \\ &= \sum_{i=1}^{N} \tilde{\mathbf{y}}_i \tilde{\mathbf{x}}_i^T - \sum_{i=1}^{N} \mathrm{E}[\mathbf{y} \tilde{\mathbf{x}}_i^T]_{P(\mathbf{y} \mid \tilde{\mathbf{x}}_i)}, \end{aligned} \tag{4}$$

Ainsi, on peut effectuer l'apprentissage des paramètres avec le modèle par l'approche de la *descente* du gradient et le *négative* de (4) avec le pseudo-code suivant :

$\boldsymbol{\theta} = \boldsymbol{\theta}_o$ // initialize parameters
while converged == FALSE
$\mathbf{G} = \frac{\partial}{\partial \boldsymbol{\theta}} \left[-\sum_{i=1}^{N} \log p(\tilde{\mathbf{y}}_i \mid \tilde{\mathbf{x}}_i; \boldsymbol{\theta}) \right]$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \mathbf{G}$

avec un taux d'apprentissage donné par η .

Description des tâches

Créez le code nécessaire pour suivre la pente du gradient. Mettez à jour les paramètres de notre modèle avec les deux approches : a) la descente par « batch » et b) la descente par « mini-batch ». Voir les notes de cours supplémentaires disponibles sur moodle pour plus de détails concernant la descente du gradient par « mini-batch ». Utilisez 20 «mini-batches». Explorez quelques taux d'apprentissage et quelques autres grandeurs pour les mini-batch et discutez de leur impact.

Il y a un squelette du code de Matlab ci-dessous. Il illustre quelques manipulations de base pour la partie a). Essayez d'utiliser Python pour vos expériences, cependant Matlab est aussi acceptable. En tout cas, il faut remettre les fichiers avec le code pour exécuter vos expériences.

```
load 20news_w100
n = 4;
m = size(newsgroups,2);
o = ones(1,m);
i = 1:m;
j = newsgroups;
Y = sparse(i,j,o,m,n);

Theta = rand(4,101)-.5;
X = documents;
X = [X ; ones(1,16242)];
taux_dapprentissage =0.0005;

[XA, XV, XT] = create_train_valid_test_splits(X);

while ~converged

      % Calculer le log vraisemblance conditionnelle chaque itération
      .
      .
      .
      % Calculer la précision sur l'ensemble d'apprentissage et
      % l'ensemble de validation après chaque itération
      .
      .
      .
      % Utilisez l'ensemble de validation pour votre critère d'arrêt
      % Calculer la mise à jour pour les paramètres

end


% Calculer la précision sur l'ensemble du test
```

Préparez un rapport sur votre code et vos expériences

Exécutez des expériences avec trois différents ensembles, un ensemble d'apprentissages avec 70% des exemples (choisis au hasard) et un ensemble de validation avec 15% et un ensemble de test avec 15%, pour vos expériences. Comparez la performance sur l'ensemble d'apprentissages avec l'ensemble de validation après chaque itération d'apprentissage dans une figure. Créer un rapport avec vos expériences incluant les figures avec les courbes d'apprentissage.

Montrez une figure avec les log vraisemblances et un autre avec le taux d'exemples correct (la précision). Pour la courbe pour l'ensemble d'apprentissages, montrez les résultats de chaque approche (a et b) après chaque itération ou époque (passe sur l'ensemble d'apprentissages au complet). Sur le même graphique, aussi montrez la performance sur l'ensemble de validation après chaque époque pour l'approche a), mais après la mise à jour faite avec chaque «mini-batch» pour l'approche b).

Barème de correction TP 2 (Date limite : le 20 février 12h30)	
Description des requis	**Points alloués**
Le rapport avec les expériences et le code de Matlab.	**20**
Total	**20**