IFT2015: Travail pratique 1

24 septembre 2018

Un tableau multi-dimensionnel classique requiert $\Theta(n^d)$ (i.e. $O(n^d)$ et $\Omega(n^d)$) de mémoire où d est le nombre de dimensions. Pour certaines applications, comme le codage de graphe et le *word embedding*, le nombre d'entrées utiles est souvent très petit par rapport à l'espace total nécessaire.

Pour palier à ce problème, il existe une variété de codages épars. Dans ce devoir, nous allons explorer le codage de rangées compressées mieux connu sous le nom de codage de Yale.

Par exemple, la matrice suivante:

```
[[0, 1, 0],
 [2, 0, 0],
 [0, 4, 3]]
```

est encodée comme suit:

```
n      = 3
m      = 3
nnz    = 4
rowptr = [0, 1, 2, 4]
colind = [1, 0, 1, 2]
data   = [1, 2, 4, 3]
```

`n`, `m` et `nnz` correspondent respectivement aux nombres de lignes, de colonnes et d'entrées non-nulles dans la matrice.

Le tableau `rowptr`, de taille `n + 1`, indique les intervalles alloués aux entrées non-nulles et à leurs colonnes de chaque ligne. En particulier, l'intervalle `[rowptr[i], rowptr[i+1][` contient les indices de `colind` et `data` où les colonnes et les valeurs non-nulles de la rangée `i` sont stockées.

Les tableaux `colind` et `data` sont de tailles `nnz`.

En assumant que j est le k-ième (en commençant par zéro) élément non-nul de la ligne i, l'élément correspondant aux indices (i, j) peut être récupéré de la manière suivante:

```
assert colind[rowptr[i] + k] == j
assert rowptr[i] + k < rowptr[i + 1]
data[rowptr[i] + k]
```

Instructions

Le travail peut être fait en équipe de deux (2). Assurez-vous d'inscrire vos noms et matricules. Vous devez également développer **VOTRE PROPRE CODE**; tout plagiat détecté entraînera automatiquement un échec.

Le code soumis devra être de bonne qualité, lisible et bien documenté. Jusqu'à dix (10) points pourront être déduits de la note finale. Vous êtes encouragé à consulter PEP8 qui est un guide de style pour Python à la page suivante: https://www.python.org/dev/peps/pep-0008/

Une pénalité de vingt (20) points sera appliquée pour chaque début de période de 24 heures suivant la date limite de remise.

1) Implémenter le codage épars Yale pour matrice. Votre implémentation doit être capable de prendre un itérable de triplets `(i, j, v)` et un tuple de dimensions comme argument du constructeur et permettre d'inspecter les tableaux `rowptr`, `colind` et `data`. Elle doit également fournir une méthode `todense` qui retourne une version dense. /20

   Discuter des détails importants de votre implémentation dans le rapport. /5

   Quelle est la complexité de votre approche pour initialiser les tableaux? /5

```python
class SparseMatrix:
    def __init__(self, fromiter, shape):
        n, m = shape
        self.n = n
        self.m = m
        self.nnz = 0     # TODO: nombre de valeurs non-nulles
        self.rowptr = [] # liste de taille n + 1 des intervalles des colonnes
        self.colind = [] # liste de taille nnz des indices des valeurs non-nulles
        self.data = []   # liste de taille nnz des valeurs non-nulles

    def __getitem__(self, k):
        i, j = k
        # TODO: retourner la valeur correspondant à l'indice (i, j)

    def todense(self):
        # TODO: encoder la matrice en format dense
        pass

mat = SparseMatrix(fromiter=[(0, 1, 1), (1, 0, 2), (2, 1, 4), (2, 2, 3)], shape=(3, 3))
```

2) Télécharger l'ensemble de données MNIST qui contient des bitmaps de chiffres manuscripts depuis StudiUM. À l'aide de votre implémentation, encoder la première image dans une matrice éparse et décoder la à l'aide de `todense`. Les deux images doivent être identiques: comparez-les pixel par pixel. /15

   Vous pouvez accéder aux images à l'aide de NumPy et les inspecter avec Matplotlib de la façon suivante:

```python
import numpy as np
import matplotlib.pyplot as plt

mnist_dataset = np.memmap('train-images-idx3-ubyte', offset=16, shape=(60000, 28, 28))
first_image = mnist_dataset[0].tolist() # first_image est de taille (28, 28)
plt.imshow(first_image, cmap='gray_r')
plt.show()
```

   Note: Vous devez convertir le bitmap en une liste de coordonnées pour le fournir au constructeur de `SparseMatrix`.

3) Montrer que les accès des éléments peuvent se faire en $O(\log m)$. /15

4) Généraliser le codage Yale à un tenseur de rang 3 (i.e. un tableau de 3 dimensions) et en fournir une implémentation. L'interface est similaire à celle de `SparseMatrix` à la différence que le constructeur acceptera un itérable de quadruplets d'éléments `(i,j,k,v)` et un triplet de dimensions. Les indices /20

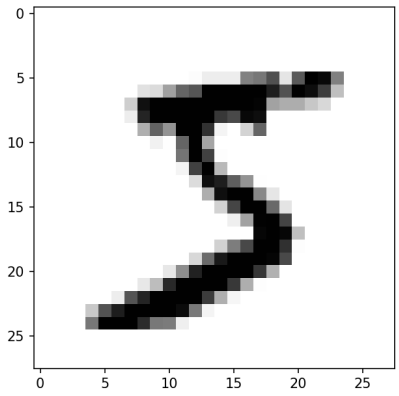

Figure 1: Example d'image de l'ensemble de données MNIST.

seront désormais des triplet `(i,j,k)` et `todense` retournera un tableau de trois dimensions. Comparer toutes les images en vous assurant qu'elles sont identiques à l'aide de boucles imbriqués.

Existe-il une façon plus compacte d'encoder toutes ces images à l'aide du codage Yale? /5

5) À l'aide de votre implémentation à la question 3, encoder efficacement toutes les images dans un seul tenseur épars. Comparer l'espace occupé par ce tenseur avec celui occupé par la version dense. /13

   Est-ce que l'encodage Yale est toujours plus compact? /2

6) **Bonus:** Généraliser le codage Yale à un tenseur de rang arbitraire. Il n'est pas nécessaire de fournir une implémentation, mais seulement de montrer comment l'indice du tableau `data` se calcule. /10

Remettre un rapport d'au plus deux (2) pages décrivant vos implémentations et élaborant les questions théoriques en format PDF ainsi que tout votre code avant le **19 octobre 2018** sur StudiUM. Compresser le tout dans une archive ZIP qui devra contenir les fichiers suivants:

- sparse.py
- main.py
- rapport.pdf

En particulier, le module `sparse.py` doit fournir les classes `SparseMatrix` et `SparseTensor` respectant les interfaces décrites plus haut. Tout l'aspect fonctionnel du code sera évalué automatiquement.

Le fichier `main.py` doit importer vos implémentations et réaliser les différentes parties du TP. Assurez-vous de bien identifier les sections correspondants aux questions. Il est également possible de fournir un notebook Jupyter.

Envoyez vos questions par courriel à guillaume.poirier-morency@umontreal.ca.

Amusez-vous bien!

Pour vous aider, voici l'encodage de la première image:

```
n = 28
m = 28
nnz = 166
rowptr = [0,   0,   0,   0,   0,   0,   12,  28,  44,  55,  64, 69, 73, 77, 83,
          89, 95, 100, 104, 111, 119, 128, 138, 148, 158, 166, 166, 166, 166]

colind = [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 8,  9,  10, 11, 12,
          13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 7,  8,  9,  10, 11, 12,
          13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 7,  8,  9,  10, 11, 12, 13,
          14, 15, 16, 17, 8,  9,  10, 11, 12, 13, 14, 16, 17, 9,  10, 11, 12,
          13, 11, 12, 13, 14, 11, 12, 13, 14, 12, 13, 14, 15, 16, 17, 13, 14,
          15, 16, 17, 18, 14, 15, 16, 17, 18, 19, 15, 16, 17, 18, 19, 17, 18,
          19, 20, 14, 15, 16, 17, 18, 19, 20, 12, 13, 14, 15, 16, 17, 18, 19,
          10, 11, 12, 13, 14, 15, 16, 17, 18, 8,  9,  10, 11, 12, 13, 14, 15,
          16, 17, 6,  7,  8,  9,  10, 11, 12, 13, 14, 15, 4,  5,  6,  7,  8,
           9, 10, 11, 12, 13, 4,  5,  6,  7,  8,  9,  10, 11]


data = [3,   18,  18,  18,  126, 136, 175, 26,  166, 255, 247, 127, 30,  36,  94,
        154, 170, 253, 253, 253, 253, 253, 225, 172, 253, 242, 195, 64,  49,
        238, 253, 253, 253, 253, 253, 253, 253, 253, 251, 93,  82,  82,  56,
        39,  18,  219, 253, 253, 253, 253, 253, 198, 182, 247, 241, 80,  156,
        107, 253, 253, 205, 11,  43,  154, 14,  1,   154, 253, 90,  139, 253, 190,
        2,   11,  190, 253, 70,  35,  241, 225, 160, 108, 1,   81,  240, 253,
        253, 119, 25,  45,  186, 253, 253, 150, 27,  16,  93,  252, 253, 187,
        249, 253, 249, 64,  46,  130, 183, 253, 253, 207, 2,   39,  148, 229, 253,
        253, 253, 250, 182, 24,  114, 221, 253, 253, 253, 253, 201, 78,  23,
        66,  213, 253, 253, 253, 253, 198, 81,  2,   18,  171, 219, 253, 253,
        253, 253, 195, 80,  9,   55,  172, 226, 253, 253, 253, 253, 244, 133,
        11,  136, 253, 253, 253, 212, 135, 132, 16]
```