Support

Introduction

But du cours

- Dans cette partie
    - Manipulation de données
    - Chargement de données diverses
    - Transformation de données
    - Nettoyage de données
- Fonctionnement de cette partie
    - Un peu de cours pour décrire les besoins/bibliothèques
    - Beaucoup de pratique pour intégrer les concepts en codant/échouant/lisant les documentations
    - **on peut adapter le cours suivant vos besoins/souhaits**
- Dans les autres parties
    - Passage à l'échelle / David Auber
    - Visualisation d'informations / Romain Bourqui
- Évaluation
    - Projet de développement + soutenance
    - Éventuellement en prenant en compte le travail réalisé pendant les séances

Fil conducteur entre les parties

Tentative : analyse de données d'un système de vélo en libre service

- Analyse en python standard avec moi

- Utilisation de bibliothèque de passage à l'échelle (possibilité de distribuer les calculs) avec David Auber
- Visualisation avec Romain Bourqui

Pré-requis

- Programmation Python 3
    - Modules
    - Fonctions
    - Classes
    - Exceptions
    - Lambda expressions
    - Indexation des listes
    - Itération
    - Accès à la documentation
- Installation des dépendances / virtualenv
- Notions de calcul matriciel
- Éventuellement Bash/Makefile
- **Quels rappels doivent être faits ?** https://docs.python.org/3/tutorial/

Écosystème Numpy

Introduction

- Prématuré de parler maintenant de Numpy dans ce cours
    - **mais nécessaire pour le cours de Machine Learning**

Description de la documentation officielle https://numpy.org/ (l'emphase est ajoutée):

> NumPy is the **fundamental package for scientific computing** with Python. It contains among other things:
>
> - a powerful **N-dimensional array** object
> - sophisticated (**broadcasting**) functions
> - tools for integrating C/C++ and Fortran code
> - useful **linear algebra**, Fourier transform, and random number capabilities
>
> Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
>
> NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Il s'agit du standard *de facto* pour la manipulation de tenseurs. Les autres bibliothèques telles que `tensorflow` miment son fonctionnement.

Fonctionnement

- Le tutoriel officiel est suffisement bien écrit pour s'en contenter : https://docs.scipy.org/doc/numpy/user/quickstart.html
- **Parcourons-le ensemble**
- Concepts à retenir :
  - concept d'axes
  - création `np.empty`, `np.arange`
  - indexation
  - broadcasting

Quelques opérations courantes

Center-réduire un jeu de données

```
import numpy as np
```

```
nb_samples = 100
width = 32
height = 32
depth = 3

dataset = np.random.rand( nb_samples, width, height, depth)
assert dataset.shape == (nb_samples, width, height, depth)
print(np.mean(dataset), np.std(dataset))
```

```
0.4990806775831246 0.28865437114379977
```

```
mean_data = np.mean(dataset, axis=0)
assert mean_data.shape == (width, height, depth)
```

```
std_data = np.std(dataset, axis=0)
assert mean_data.shape == (width, height, depth)
```

```
eps = np.finfo(float).eps
standardized = (dataset - mean_data) / (std_data+eps)
assert standardized.shape == (nb_samples, width, height, depth)
print(np.mean(standardized), np.std(standardized))
```

```
0.002395806800272044 1.0145258702976852
```

Séparer les jeux de données

```
import numpy as np

nb_samples = 100
width = 32
height = 32
depth = 3
nb_classes = 5

dataset = np.random.rand( nb_samples, width, height, depth)
ground_truth = np.random.randint(0, nb_classes, nb_samples)
assert len(dataset) == len(ground_truth)
```

```
train_selected = np.random.rand(nb_samples) <= 0.7 # 70%
test_selected = ~train_selected

assert np.sum(train_selected ^ test_selected) == nb_samples
```

```
X_train, y_train = dataset[train_selected],
ground_truth[train_selected]
X_test, y_test = dataset[test_selected],
ground_truth[test_selected]
```

Récupérer la classe résultat

```
import numpy as np

labels = np.array(['voiture', 'chien', 'cheval', 'avion', 'fleur'])
nb_samples = 100
nb_classes = len(labels)
scores = np.random.rand(nb_samples, nb_classes)
counts = np.sum(scores, axis=1)
probas = scores/np.sum(scores, axis=1).reshape( (nb_samples,1) )

assert np.all( (np.sum(probas, axis=1) - 1) < 0.001)
```

```
classes_id = np.argmax(probas, axis=1).astype(int)
samples_labels = labels[classes_id]
```

Réorganiser les axes d'un tenseur

(opération courante en fonction de la bibliothèque d'apprentissage utilisée)

```
import numpy as np

nb_samples = 100
width = 8
height = 4
depth = 7

data = np.random.rand(width, height, depth, nb_samples)
print(data.shape)
print(data[..., 0, 0].shape)
print(data[..., 0, 0])
```

```
(8, 4, 7, 100)
(8, 4)
[[0.07106781 0.58885087 0.76641997 0.78761934]
 [0.60092936 0.79137134 0.72444527 0.47539522]
 [0.36762757 0.30983784 0.80277362 0.51663467]
 [0.80566992 0.50464139 0.35376165 0.35756682]
 [0.08135704 0.06651439 0.46824767 0.58285456]
 [0.4080921  0.76362656 0.25193058 0.43386794]
 [0.19673697 0.45208775 0.52252395 0.84595186]
 [0.38941082 0.54510322 0.43295482 0.48717848]]
```

```
data2 = np.moveaxis(data, -1, 0)
print(data2.shape)
print(data2[0,...,0].shape)
print(data2[0,...,0])
```

```
(100, 8, 4, 7)
(8, 4)
[[0.07106781 0.58885087 0.76641997 0.78761934]
 [0.60092936 0.79137134 0.72444527 0.47539522]
 [0.36762757 0.30983784 0.80277362 0.51663467]
 [0.80566992 0.50464139 0.35376165 0.35756682]
 [0.08135704 0.06651439 0.46824767 0.58285456]
 [0.4080921  0.76362656 0.25193058 0.43386794]
 [0.19673697 0.45208775 0.52252395 0.84595186]
 [0.38941082 0.54510322 0.43295482 0.48717848]]
```

Exercices

- Voir support exercices

Ressources supplémentaires

- https://www.machinelearningplus.com/python/numpy-tutorial-part1-array-python-examples/
- https://github.com/rougier/numpy-100/blob/master/100_Numpy_exercises.md

Écosystème Pandas

Introduction

- Prématuré de parler de Pandas https://pandas.pydata.org/pandas-docs/stable/index.html
    - mais nécessaire pour le cours de Machine Learning
- Repose intensivement sur
    - Numpy (2d array)
    - Matplotlib (voir dans quelques séances)
- Mais apporte :
    - une meilleure expérience utilisateur avec le nommage des colonnes
    - une simplification de l'affichage des données

Description de ses auteurs :

> pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

Fonctionnement de Pandas

- Tutorial officiel également suffisant : https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html
- **Parcourons le ensemble**
- Concepts à retenir : objets Series et DataFrame, création depuis dictionnaires ou listes, sélection de lignes/colonnes, opérations diverses, fusion, jointures, groupes

- Utilisation plus avancée ici :https://pandas.pydata.org/pandas-docs/stable/user_guide/cookbook.html#cookbook

Exercices

- Voir support exercices

Autres bibliothèques importantes

scikit-learn

- Algorithmes d'apprentissage automatique (ça n'est pas ce qui nous intéresse)
- Ensemble de classes/utilitaires pour l'apprentissage automatique (c'est ce qui nous intéresse)
    - validation croisée, optimisation des paramètres, calcul de métriques de performance
- https://scikit-learn.org/stable/index.html

HDF5

- Stockage de tableaux `numpy` qui ne tiennent pas en mémoire dans un dictionnaire hierarchique sur disque
- http://docs.h5py.org/en/stable/

joblib

- Parallelisation de calculs
- Gestion de cache
- https://joblib.readthedocs.io/en/latest/

dask

- Parallelisation et distribution des calculs dans des Dataframe

- https://dask.org/

NetworkX

- Gestion de graphes
- https://networkx.github.io/documentation/stable/

IPython

- Session Python intéractive dans un environnement WEB qui facilite le prototypage
- https://ipython.org/

nltk

- Traitement de la langue
- https://www.nltk.org/

Généralités sur les données - ETL

Qu'est-ce que la donnée

- information élémentaire pour créer une cascade d'information
- plus de données = informations fiables et de qualité
- plus proche de la données = moins de distortions

Et le big data ?

- **Volume**
- **Variété**
- **Vélocité**
- Véracité

- Valeur
- Viabilité
- Validité

Sources de données

- Pistages des sites web/téléphones portables/logiciels/objets connectés/...
- Bases de données publiques
- Bases de données privées

Collecte des données

- Quelle est leur disponibilité ?
    - Téléchargement en live ? batch ?
    - À quel cout ?
    - En quelle quantité ?
- Comment les récupérer ?
- Y a-t-il des problèmes juridiques à utiliser ces données ?

Hétérogénéité des données

- Type de données
    - tabulaire, multi-dimensionnelles, relations
- Format des données
- Fréquence de rafraîchissement
- Durée de vie

Format de fichiers

Format de fichiers standards

- `CSV` (Comma-separated values)

```
MENU;POPUP;VALUE;ONCLICK
File;0;New;CreateNewDoc()
File;0;New;OpenDoc()
File;0;New;CloseDoc()
```

- `XML` (Extensible Markup Language)

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

- `JSON` (JavaScript Object Notation)

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

- `HTML`

```
<div id="file"><span>File</span>
  <ul>
    <li><a onclick="CreateNewDoc()">New</a></li>
    <li><a onclick="OpenDoc()">Open</a></li>
    <li><a onclick="CloseDoc()">Close</a></li>
```

```
  </ul>
</div>
```

- Tableur, traitement de texte, pdf
- **Avantages/inconvénients ?**

Outils pour visualiser ces fichiers

- `cat` / `less` / `head` / `tail`
- `zcat` / `zless`
- Éditeur de texte `vim`, `emacs`, `code`
- Suite bureautique
- Navigateur web
- ...

Formats propriétaires ou non standards

- Format dépendant du fournisseur de la source de données
- Code de lecteur nécessairement spécifique et difficilement réutilisable
- Les spécifications doivent être suffisement précises pour pouvoir écrire le parseur
- Il faut espérer que les spécifications ne changent pas au cours du temps
- Données géographiques (e.g. `geojson`)

Métadonnées

Plusieurs formats de fichiers sont également accompagnés de méta-données.

- `EXIF` ou `XMPP` pour les photos
- `ID3` pour le MP3
- droits d'accès dans les systèmes de fichiers
- ...

Encodage des fichiers

- Plusieurs encodages existent
    - latin1
    - ascii
    - utf8
    - ...
- Décoder correctement les caractères nécessite de les connaître
- Python est très pointilleux sur cet aspect

Exercices

- Voir support exercices

Traitements standards sur les données

Extraction depuis des chaines de caractères

- Chaîne vers type primitif (`int` , `float`) en prenant en compte la localisation
- Chaîne vers date
- Attention au format anglo-saxon
- Attention aux incohérences dans les données
- Découpage de chaines
- Expressions régulières

Extraction depuis des images et vidéos

- Histogramme de couleurs https://fr.wikipedia.org/wiki/Histogramme_(imagerie_num%C3%A9rique)
- Filtres de Gabor https://fr.wikipedia.org/wiki/Filtre_de_Gabor
- Descripteurs locaux (SURF, SIFT, ...)

- Sacs de mots visuels
- Extraction de caractéristique par réseaux de neurones profonds

Transformations

Tendences centrales

- Moyenne `np.mean` / Sensbiles aux valeurs extrêmes
- Médiane `np.median`
- Quantile `np.quantile`
- Mode `statistics.mode`

Dispersion

- Envergure max(x) - min(x)
- Variance (carré de la déviation à la moyenne)
- Écart type (racine carrée de la variance)
- Différence entre les valeurs des 75iem et 25ieme centiles / Moins sensible aux valeurs extrêmes

Corrélation

- Covariance / mesure de l'écart à la moyenne de deux variables simultanément `np.cov`
- Correlation / plus facile à interprêter `np.corrcoef`

Probabilités

- E et F indépendants : P(E,F) = P(E)P(F)
- E dépend de F : P(E|F) = P(E,F)/P(F)
- histogrammes

Transformations numériques

- Changement d'échelle
    - normalisation min-max : \frac{x-min}{max-min}

        - standardisation : \frac{x-mean}{std}
    - Réduction de dimensionalité
        - Analyse en Composante Principale
    - Discrétisation
    - Binarisation

Transformation de données quantitative

- Conversion en donnée numérique
- One Hot Encoding

Transformations textuelles

- Segmentation des mots/phrases
- Normalisation / forme canonique
- Filtrage

Transformation de dates

- Éclatement d'une date en plusieurs sous-informations
    - année, mois, heure, ...
    - jour de la semaine, jour férié

Transformation d'images et vidéos

- Égalisation de l'histogramme de couleurs
- Standardisation des images

Ensemble de valeurs vers objet plus complexe

- Extraction de différentes informations pour générer une donnée composite (e.g. localisation spation temporelle, détection de la langue, ...)
- Génération de tenseur à partir de données diverses (e.g., geojson)

Aggrégation de données

Réduction de la quantité de données :

- Calcul des tendences centrales, dispértions, corrélation
- Fusion d'informations hétérogènes

Nettoyage des données

Constat

- Les données disponibles sont rarement propres (fausses, manquantes, biaisées, aléatoire)
- Les outils d'analyse nécessitent des données propres
- Il faut donc les nettoyer après chargement ou écrire des chargeurs robustes

On estime que 70% du travail consiste à nettoyer les données

Données fausses

- Souvent seule l'expértise du domaine permet d'identifier et corriger de tels cas
- statistique
- recherche de valeurs abérentes
- Il faut les éliminer ou trouver d'autres sources de données

Données manquantes

-Causes : - Trous dans les données temporelles - Valeurs non renseignées - ...

Données biaisées

- Très problématique dans le cas de l'intelligence artificielle
- Il faut chercher d'autres sources de données

Stratégies de remplacement

- Ne rien faire et favoriser des algorithmes robustes

- Valeur numérique : médianne/moyenne/précédente
- Valeur quantitative : plus courante/plus significative/précédente

Enrichissement de données

Il est pertinent d'enrichir les données de première main à l'aide d'autres données

- autres données internes
- partenariat
- achat
- dépôt de données libre https://www.etalab.gouv.fr/plateformes

Outils ETL

- https://airflow.apache.org/
- https://github.com/spotify/luigi