CSIT5210

Programming Workshop

prepared by Dr. Kevin Wang,
written in Markdown

Prepared by Kevin Wang, kevinw@ust.hk

Today Schedule

- Logistics
- Setting up the environment
- Basic Python for Data Science
- Handling Large Volumn of Data
- Q & A

Browsing the Assignment

About the assignment

Part	Due Date	Submission by	Files to Submit
Part 1	15/11/2019 (Fri) 23:59	email to TA	`ipynb` file.
Part 2	21/11/2019 (Thu) during lecture	Printed hardcopy	Presentation slides (4 slides per page)

Both of them are group work.

Team formation

- Sign up your team formation on the Google Spreadsheet.
- We much prefer a team of 4 and will randomly assign people who has no team.
- Due: 30/9/2019

Submission Method

Complete the Part 1 assignment and submit the `ipynb` by email.

- Remember to **restart kernel** to delete all variables before you submit it!
- Please attached your running result in the notebook.
- If you use any other library, please make sure they can be installed through `pip` or attach your library.
- I will not grade Part 1 - **Task 5 Prediction**. You are not allowed to change your code after 15/11/2019.
- Send me one version per group and write your group name, member name in the email.

Set up the environment

Check list:

- Python 3 **64-bits** (3.6 should be good enough)
- pip (package manager of python)
- web browser

Hardware Requirement:

- Preferable 8GB ram
- several GB storage
- GPU may not be useful at all

Installing Package using pip

Package to install:

- jupyter
- pandas
- numpy
- matplotlib
- sklearn

Type `pip install <PACKAGE_NAME>` in terminal/command prompt, e.g. `c:\>pip install jupyter`

Launching `jupyter notebook`

In terminal/command prompt, type:

```
c:\>jupyter notebook --notebook-dir="d:\CSIT5210\"
```

This will start your notebook at that default directory. You are able to change your directory of course but not switch drive.

Alternatively, using HKUST virtual barn

- `Jupyter`: `start` > `Anancoda 64-bit` > `jupyter notebook`
- `pip` : `start` > `Anancoda 64-bit` > `Anancode Prompt`

config: Xeon 2.6G with 8GB ram, 10GB storage on p drive.

🔍 `HKUST virtual barn`

You need VPN client and VMware Horizon.

Basic Python for Data Science

Useful link:

- numpy tutorial
- pandas reference
- memory management
- sklearn

Vs conventional programming

- We now more work like excel formula setting
- Apply same function in a column to produce another
- Less frequent to use loop and condition
- OOP ? Not for this assignment at least

Some basic

- No bracket, by intendation.
- Can use variable without declaration, but there are different types
- Comment by #
- No compilation needed, just type `alt` (or `ctrl`) + `enter` to run
- Auto/semi-auto memory management

Basic Data Type

memory usage	float	int	uint	datetime	bool	other
1 bytes		int8	uint8		bool	
2 bytes	float16	int16	uint16			
4 bytes	float32	int32	uint32			
8 bytes	float64	int64	uint64	datetime64		
variable						object, string

Some Python fundamentals

http://cs231n.github.io/python-numpy-tutorial/

Step 1, reading data

```
import pandas as pd
df = pd.read_csv('mobikeData.csv', sep=',', index_col= ['orderid'])
```

Reading csv data into `mobikedata` as a pandas DataFrame.

Some more parameters:
`nrow` : fetch first n-rows only. e.g. `nrow = 1000`
`chunksize` : fetch chunk-size rows at each time. e.g. `chunksize=1e6`, each time read 100000 rows.
`parse_dates`: the parse the fields as date. e.g `parse_dates=['starttime']`

Why and what is pandas/DataFrame?

Just because your data is not pure numerical. DataFrame works like a table in database/excel spreadsheet.

Why not just numpy?

numpy works with numbers only. You need to extract your data into number to use numpy.

Basic data manipulation

Let's say your DataFrame is call `df`.

```
df.info()
```

```
df.describe()
```

```
df.head(10)
```

```
df.tail(7)
```

Basic column slicing

```
df['userid'] = df['userid'] + 1
```

```
df.drop(['userid'], axis=1)
del(df['userid'])
```

Applying a formula/function

Using Loop

```
def dist_4(x1,y1,x2,y2):
        return (x1-x2)**2 + (y1-y2)**2

# assume
# df['p'] and df['q'] contain arrays looks like [x, y]
w = []
for key, row in df.iterrows():
        w.append(dist_4(df['p'][0], df['q'][0], \
                        df['p'][1], df['q'][1]))
df['dist'] = w
del(w)
```

Very slow...

Using apply()

```
def discount(x):
        return x * 0.8

df['sq'] = df['price'].apply(discount)
```

Much faster.

Another function example

```
# assume pt is an array of [x1, y1, x2, y2]
def dist_arr(pt):
        return (pt[0] - pt[2]) ** 2 + (pt[1] - pt[3]) ** 2

df['pt'].apply(dist_arr)
```

Another example

```
# assume you store all points deparately into four variables:
#  x1, x2, y1, y2
def dist_row(row):
        return (row['x1'] - row['x2']) ** 2 +  \
               (row['y1'] - row['y2']) ** 2

df['dist']=df.apply(dist_row, axis=1)
```

When function has more than one parameters..

```
# Rank which number is larger
def best(t1, t2, t3):
    if t1 > t2 and t1 > t3:
        return 1
    elif t2 > t3:
        return 2
    else:
        return 3
df['best'] = np.vectorize(df)(df['A'], df['B'], df['C'])
```

df['best']	df['A']	df['B']	df['C']
1	100	50	60
2	4	5	1
1	3	0	0
3	3	4	50

Some other useful DataFrame APIs

- `sort_value()` : sort by a particular column
- `min()`, `max()`, `count()`, `average()`: basic statistic
- `append`: append row
- `head()`, `tail()`: view a first/last 5 rows
- `merge`: merge two DataFrames using database like join function
- `groupby()`: grouping some data according to their indexes (like database)
- `between_time`: select values between particular time

Handle Large Volume of Data

ref: https://www.dataquest.io/blog/pandas-big-data/

Technique #1 - Batch Processing

```
size = 1e6
dflist = pd.read_csv('train.csv', sep=',', chunksize = size)
w = []
for df in dflist:
    df['result'] = df['data'].apply(expensive_function)
    w.append(df)

#merge the data
newdf=pd.concat(w, sort=False)
del(w)
```

Technique #2 - Change Type

```
df['dist'] = df['dist'].asType('float16')
df['areacode'] = df['areacode'].asType('uint8')
```

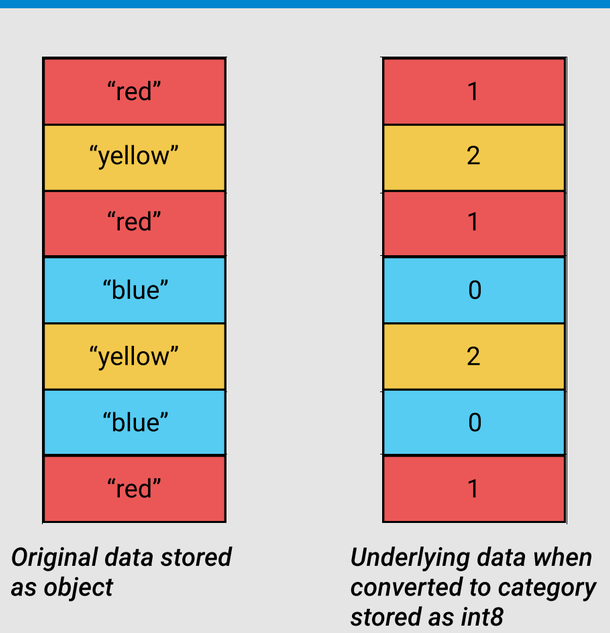
Downcast your data may lose your precision, if you are affordable.

Sometimes your data does not need that much space to store. e.g.:
`areacode`, `class_id`, `jersey_number`

Technique #3- Catagorize Data

```
df.describe()
```

It shows you how many unique value in your column. If they are very few, you might want to catagorize them



Original data stored as object

Underlying data when converted to category stored as int8

```
dow = gl_obj.day_of_week
print(dow.head())

dow_cat = dow.astype('category')
print(dow_cat.head())
```

```
0    Thu
1    Fri
2    Sat
3    Mon
4    Tue
Name: day_of_week, dtype: object
0    Thu
1    Fri
2    Sat
3    Mon
4    Tue
Name: day_of_week, dtype: category
Categories (7, object): [Fri, Mon, Sat, Sun, Thu, Tue, Wed]
```

Technique #4 Trim Data

```
df.drop(['useless data1', 'useless data2'])
del(df['useless data1'])
del(df['useless data2'])
```

Python will auto recycle variable *if it is no longer referenced*. You need to manually recycling it if you are not writing a function!

Work with an example

1. Load first 1000 rows of data
2. Find some statistics
3. Decode the geohash

Q & A

email me: kevinw@ust.hk

references:

- http://cs231n.github.io/python-numpy-tutorial/
- https://www.dataquest.io/blog/pandas-big-data/