Lecture 13 - Data Structures (Part 3)

DSE 511

Drew Schmidt
2022-10-11

Announcements

- Nothing unresolved from last time
- Homework not yet graded
- Questions?

Content

- Dataframes
- Complexity

Dataframes

Dataframes

- 2-dimensional / tabular data format.
- Columns are variables ("features"), rows are observations ("samples").
- Like a matrix, but types can vary across columns.
- **The** fundamental object for statistical learning.
- Implementations vary pretty significantly
  - R
  - Python (Pandas)
  - Arrow

Dataframes in R

- List of vectors
  - Each has same length
  - The "columns" of the table
  - Values are the "rows"
  - Types may differ
- May or may not have name attributes

```r
head(df)
```

```
##   V1 V2 V3
## 1  1 11 21
## 2  2 12 22
## 3  3 13 23
## 4  4 14 24
## 5  5 15 25
## 6  6 16 26
```

```r
is.list(df)
```

```
## [1] TRUE
```

Dataframes in R

```
set.seed(1234)
x = 1:3
y = rnorm(3)
l = list(x=x, y=y)
as.data.frame(l)
```

```
##   x          y
## 1 1 -1.2070657
## 2 2  0.2774292
## 3 3  1.0844412
```

```
z = 1:2
l = list(z=z, y=y)
as.data.frame(l)
```

```
## Error in (function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE, : arguments im
```

Dataframes in Python

> Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). … Can be thought of as a dict-like container for Series objects.
>
> Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).

The Pandas help https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DataFrame.html

Dataframes in Python

```python
import pandas as pd
x = range(1, 4)
y = [1.2, 2.3, 3.4]
l = [x, y]
l
```

```
[range(1, 4), [1.2, 2.3, 3.4]]
```

```python
pd.DataFrame(l)
```

	0	1	2
0	1.0	2.0	3.0
1	1.2	2.3	3.4

```python
l = list(zip(x, y))
l
```

```
[(1, 1.2), (2, 2.3), (3, 3.4)]
```

```python
pd.DataFrame(list(zip(x, y)))
```

	0	1
0	1	1.2
1	2	2.3
2	3	3.4

Transferring Data Between R and Python

In Memory

- Recommended Solutions
  - rpy2 (Python calling R)
  - reticulate (R calling Python)
- Possible but complicated
  - Sockets (e.g. ZeroMQ)
  - Web APIs
  - ...

On Disk

- CSV
- Database
- JSON
- Arrow
- HDF5 (better for matrices than dataframes)
- ...

JSON Example

```
df = as.data.frame(list(x = 1:3, y = letters[1:3]))
df
```

```
##   x y
## 1 1 a
## 2 2 b
## 3 3 c
```

JSON Example

```r
path = "/tmp/df.json"
jsonlite::write_json(df, path = path, auto_unbox = TRUE, pretty = TRUE)
cat(paste(readLines(path), collapse = "\n"))
```

```
## [
##   {
##     "x": 1,
##     "y": "a"
##   },
##   {
##     "x": 2,
##     "y": "b"
##   },
##   {
##     "x": 3,
##     "y": "c"
##   }
## ]
```

JSON Example

```python
import pandas as pd
df = pd.read_json('/tmp/df.json')
df
```

	x	y
0	1	a
1	2	b
2	3	c

Programming with Dataframes

- Very language dependent.
- Generally speaking
  - Easy things work well in R and Pandas
  - R has *significantly* more options for complex operations

Programming with Dataframes

R

```r
sum(df$x)
```

```
## [1] 6
```

```r
suppressMessages(library(dplyr))
df %>% select(x) %>% sum()
```

```
## [1] 6
```

Python

```python
df['x'].sum()
```

```
6
```

Complexity

Complexity

- "Big O" - literally upper case O
- Mathematical notation
- Describes *asymptotic behavior* of a function
    - Sometimes called *asymptotic complexity*
    - Other models of "complexity" exist
- Used in CS to understand runtime behavior as data grows

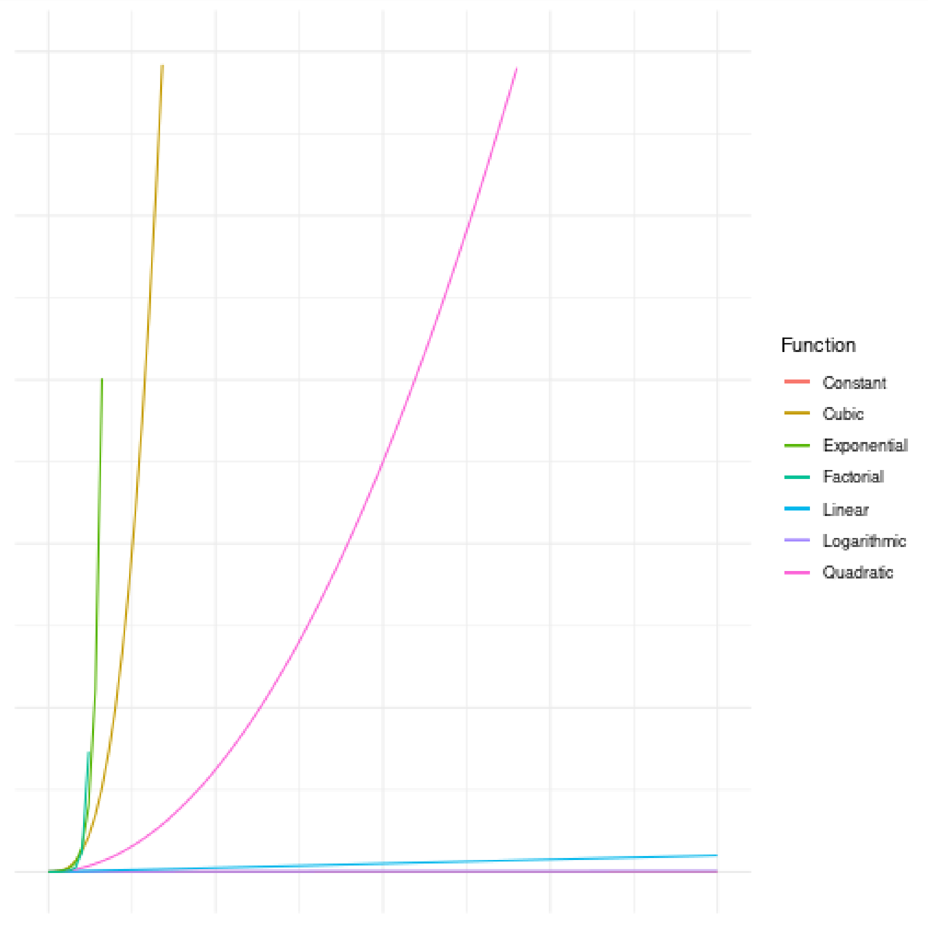
Big O

- Describes an upper bound for a function
- Formally: $f(x) = O(g(x))|_{x \to \infty} \iff \exists (M, x_0) \in \mathbb{R}^2 : |f(x)| \leq M \cdot g(x) \forall x \geq x_0$
- Said another way: $f(x) = O(g(x))|_{x \to \infty} \iff \limsup_{x \to \infty} \frac{|f(x)|}{g(x)} < \infty$
- In CS, we usually talk about $f(n)$
- See also: "little o"

https://en.wikipedia.org/wiki/Big_O_notation

Common Functions

- Constant - $O(1)$
- Logarithmic - $O(\log(n))$
- Linear - $O(n)$
- Quadratic - $O(n^2)$
- Polynomial - $O(n^a)$ for some $a > 0$
- Exponential - $O(\exp(n))$
- Factorial - $O(n!)$



Calculation Example

Fact: $3n^2 + 2n + 1 = O(n^2)$

Proof: Notice that $3n^2 + 2n + 1 \leq 3n^2 + n^2 = 4n^2$ whenever $n^2 \geq 2n + 1$. By inspection, $n \geq \sqrt{2n+1}$ whenever $n \geq 3$, since $\sqrt{7} \approx 2.65$. So let $M = 4$ and $x_0 = 3$ QED.

Big O Properties

- You can ignore constant factors: $\forall c > 0, f(n) = cg(n) \implies O(f(n)) = O(g(n))$
- O of a sum is O of the "most expensive" summand
  - Corollary: $a_m n^n + a_{n-1} n^{m-1} \cdots + a_1 n^1 + a_0 = O(n^m)$
- It is multiplicative: $f_1 = O(g_1)$ and $f_2 = O(g_2)$ then $f_1 \cdot f_2 = O(g_1 \cdot g_2)$
- All logarithms grow at the same rate (change of basis + constant factor)
- Constant < Logarithmic < Polynomial < Exponential < Factorial

Some Common Algorithms - Searches and Sorts

Sorts

- Bubblesort $O(n^2)$
- Quicksort $O(n \cdot \log n)$

Search

- Binary search $O(\log n)$
- Linear search $O(n)$
- Hash table lookup $O(1)$

Some Common Algorithms - Graphs

- BFS $O(V + E)$
- SSSP (Dijkstra) $O((V + E) \log V)$
- ASSP (Floyd-Warshall) $O(V^3)$

Some Common Algorithms - Data Science

- Square matrix multiplication (gemm) FLOPs: $2n^3$
- Cholesky FLOPs: $\frac{1}{3}n^3$
- PCA FLOPs: $6mn^2 + 20n^3 + 2mn^2 + 2mn + n$

Complexity Analysis

- Useful, but not an exact science
- Careful not to deceive yourself
- It's about *asymptotics*

Problems with Asymptotic Complexity

> Asymptotics will eventually win out, *as long as everything else stays fixed*. But that's the precise problem. Everything else doesn't stay fixed. Well before your $n \log n$ algorithm beats the n^2 algorithm, we run out of memory, or local cache, or something else, and the computational model changes on us.

Suresh Venkatasubramanian

Wrapup

Homework

- Coming "soon" (tm)
- What to ask?
  - Some basic R/Python tasks
  - Implement some basic thing using classes (R or Python)
  - A few questions about "Big O"

Wrapup

- From a user point of view, R and Pandas dataframes are basically the same.
- In memory, they are *very* different.
- "Big O" can be a useful analysis, but there are some subtleties.
- Next time: Introduction to the Shell

Questions?