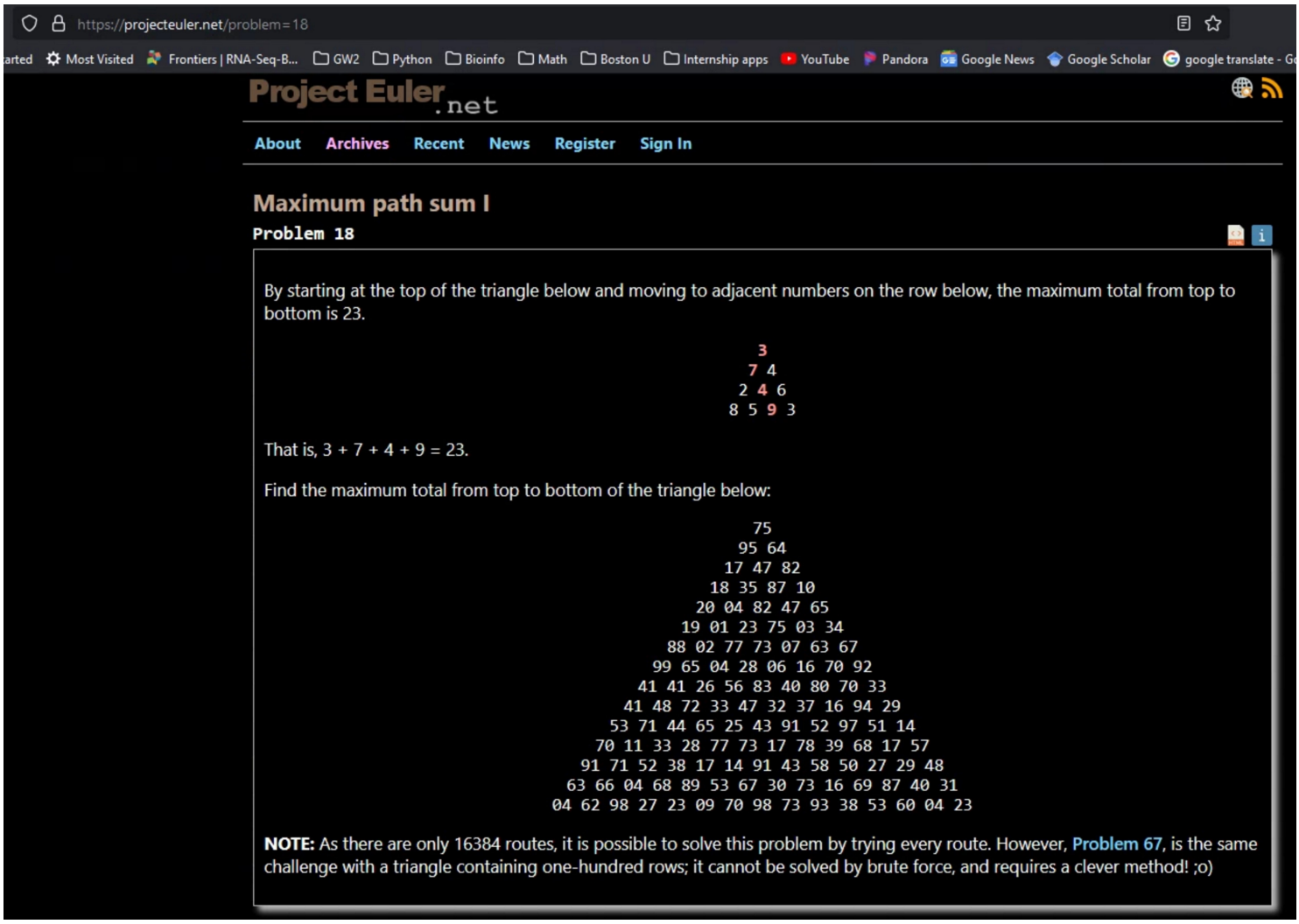
20210915

Wednesday, September 15, 2021 4:30 PM

https://projecteuler.net/archives



Want to find the maximum path sum

Should be able to feed in the top triangle, and get 23

20210915 Clean Code

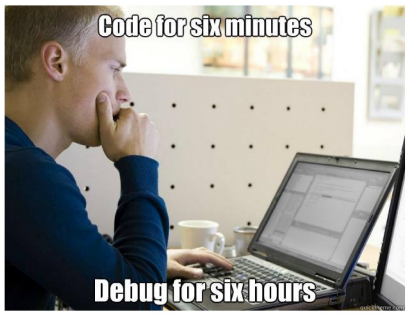
Wednesday, September 15, 2021 3:18 PM

Clean Code

Clean Code

Boston University CS 506 - Lance Galletti

The Problem



The Problem

"Software Systems get replaced not when they wear out but when they crumble under their own weight because they have become too complex"

Why Does this Occur?

- english words + grammar != poetry
- Need analysis of the code itself - not just algorithms

Gems of Clean Code

- Structure
  - Before writing code ask “How will someone use this (or part of this) code?”. Minimize side effects
  - Do one thing
- Method
  - Top Down Approach
  - Bottom up Approach
  - Solve the problem first - then improve / refine
- General
  - Boring code is good code: keep it simple
  - Late Binding: start vague and refine (don’t commit to specificity)
  - Many functions with small bodies > one function with large body
  - Check soundness by reading your code before testing

Clean Code

'Clean code' by Robert C. Martin

Github Gist Summary of the Book:
https://gist.github.com/wojteklu/73c6914cc446146b8b533c0988cf8d29

A Look into Functional Programming

"Functional Programs are mathematical expressions that are evaluated and reasoned about much like ordinary mathematical functions. As a result, these expressions are simple to analyze and compose for large-scale programs"

- Combinators on Lists
- An example coding exercise

Combinators on Lists

xs = [1, 4, 9, 16, 25]

- map(xs, lam(x) => f(x)) = [f(1), f(4), f(9), f(16), f(25)]

  Ex:

  map(xs, lam(x) => x % 2) = [1, 0, 1, 0, 1]
- Foldleft - idea of consuming the list via an accumulator

  Ex:

  foldleft(xs, init, lam(x, acc) => f(x, acc))
  - acc = init
  - acc = f(1, acc)
  - acc = f(4, acc)
  - acc = f(9, acc)
  - acc = f(16, acc)
  - acc = f(25, acc)
  - Return acc

foldleft(xs, 0, lam(x, acc) => x + acc)

1. Acc = 0
2. Acc = 0 + 1 = 1
3. Acc = 1 + 4 = 5
4. Acc = 5 + 9 = 14
5. Acc = 14 + 16 = 30
6. Acc = 30 + 25 = 55
7. Return 55

Combinators on Lists

xs = [1, 4, 9, 16, 25] ys = [1, 2, 3, 4, 5]

- map2(xs, ys, lam(x, y) => f(x, y)) = [f(1, 1), f(4, 2), f(9, 3), f(16, 4), f(25, 5)]

  Ex:

  map2(xs, ys, lam(x, y) => x * y) = [1, 8, 27, 64, 125]
- Foldleft2 - consume both lists (like foldleft on zip)

  Ex: foldleft2(xs, ys, 1, lam(x, y, acc) => acc * (x / y))

  1. acc = 1
  2. acc = 1 * (1 / 1) = 1
  3. acc = 1 * (4 / 2) = 2
  4. acc = 2 * (9 / 3) = 6
  5. acc = 6 * (16 / 4) = 24
  6. acc = 24 * (25 / 5) = 120
  7. Return 120

Example: Max Path Sum

Starting at the top of the triangle and moving down to adjacent numbers below: Find the path from the root to a leaf with the maximum sum.

```
    3
   7 4
  2 4 6
 8 5 9 3
```

Example: Max Path Sum

Starting at the top of the triangle and moving down to adjacent numbers below: Find the path from the root to a leaf with the maximum sum.

```
    3
   7 4
  2 4 6
 8 5 9 3
```

Example: Max Path Sum

Starting at the top of the triangle and moving down to adjacent numbers below: Find the path from the root to a leaf with the maximum sum.

```
    3
   7 4
  2 4 6
 8 5 9 3
```

Example: Max Path Sum

Starting at the top of the triangle and moving down to adjacent numbers below: Find the path from the root to a leaf with the maximum sum.

3
7 4
2 4 6
8 5 9 3

Example: Max Path Sum

Starting at the top of the triangle and moving down to adjacent numbers below: Find the path from the root to a leaf with the maximum sum.

3
7 4
2 4 6
8 5 9 3

We are going to fold the triangle

Example: Max Path Sum

Starting at the top of the triangle and moving down to adjacent numbers below: Find the path from the root to a leaf with the maximum sum.

3
7 4
2 4 6
8 5 9 3

In the above example the max path sum in 3 + 7 + 4 + 9 = 23

Example: Max Path Sum

Starting at the top of the triangle and moving down to adjacent numbers below: Find the path from the root to a leaf with the maximum sum.

```
   3
  7 4
 2 4 6
8 5 9 3
```

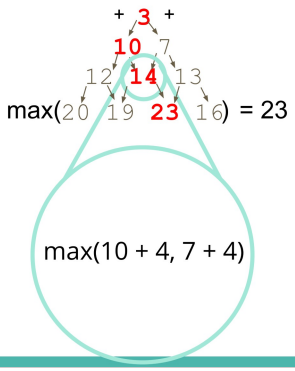
In the above example the max path sum in 3 + 7 + 4 + 9 = 23

Brute Force: Find all paths and get the max.

Source: https://projecteuler.net/problem=18

Example: Max Path Sum

Our Algorithm



Code Using Combinators

Triangle = list of lists. Can we use foldleft?

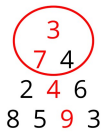
max(**foldleft**(triangle, [], **lam**(xs, acc) **=>** **myfold**(xs, acc)))

```
   3
  7 4
 2 4 6
8 5 9 3
```

Code Using Combinators

Triangle = list of lists. Can we use foldleft?

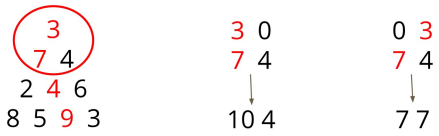
max(**foldleft**(triangle, [], **lam**(xs, acc) **=> myfold**(xs, acc)))



Code Using Combinators

Triangle = list of lists. Can we use foldleft?

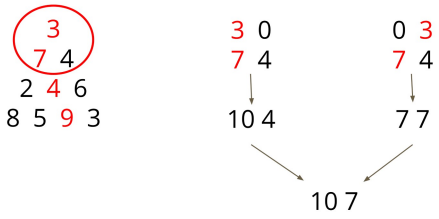
max(**foldleft**(triangle, [], **lam**(xs, acc) **=> myfold**(xs, acc)))



Code Using Combinators

Triangle = list of lists. Can we use foldleft?

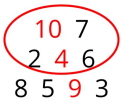
max(**foldleft**(triangle, [], **lam**(xs, acc) **=> myfold**(xs, acc)))



Code Using Combinators

Triangle = list of lists. Can we use foldleft?

max(**foldleft**(triangle, [], **lam**(xs, acc) **=> myfold**(xs, acc)))

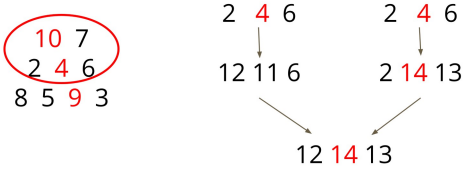


Always take the maximum of the maximum to get to that particular leaf node of intermediary in the triangle

Code Using Combinators

Triangle = list of lists. Can we use foldleft?

max(**foldleft**(triangle, [], **lam**(xs, acc) **=> myfold**(xs, acc)))



Add 0 to right of prev line to get left sum, and 0 to left of prev lin to get right sum

(To handle lists of different lengths…)

Code Using Combinators

Triangle = list of lists. Can we use foldleft?

max(**foldleft**(triangle, [], **lam**(xs, acc) **=> myfold**(xs, acc)))

myfold(xs, acc) :
    option1 = **map2**([0, acc…], xs, **lam**(x, y) **=>** x + y)
    option2 = **map2**([acc…, 0], xs, **lam**(x, y) **=>** x + y)
    return map2(option1, option2, **lam**(x, y) **=> if** x > y **then** x **else** y)

Python Examples

myfolder
```
├── main.py
└── data.txt
```

main.py
```
nums = []
with open("../data.txt", "w+") as f:
  lines = f.readlines()
  for line in lines:
    nums += [int(x) for x in line.split(",")]

print(sum(nums))
```

data.txt
```
0,1,2,3,4
```

What is the output of this code?
a. Prints “10”
b. Raises an Error
c. Prints “0”

Python Examples

```
class Bank:
  def __init__(self, balance):
    self.balance = balance

  def is_overdrawn(self):
    return self.balance < 0

myBank = Bank(100)
if myBank.is_overdrawn :
  print("OVERDRAWN")
else:
  print("ALL GOOD")
```

What is the output of this code?
a. Prints “OVERDRAWN”
b. Prints “ALL GOOD”
c. Raises an Error

Python Examples

```
for i in range(4):
    print(i)
  i = 10
```

What is the output of this code?
a. Prints “0”
b. Prints “0 1 2 3”
c. Raises an Error

```
some_string = "what"
some_dict = {}
for i, some_dict[i] in enumerate(some_string):
   i = 10

print(some_dict)
```

What is the output of this code?
a. Prints “{}”
b. Prints “{0: 'w', 1: 'h', 2: 'a', 3: 't'}”
c. Raises an Error

https://book.pythontips.com/en/latest/enumerate.html

Python Examples

```
row = [""] * 3 # row i['', '', '']
board = [row] * 3
print(board) # [['', '', ''], ['', '', ''], ['', '', '']]
print(board[0]) # ['', '', '']
print(board[0][0]) # ''
board[0][0] = "X"
print(board)
```

What is the output of this code?

a. Prints "[['X', '', ''], ['', '', ''], ['', '', '']]"
b. Prints "[['', '', ''], ['', '', ''], ['', '', '']]"
c. Prints "[['X', '', ''], ['X', '', ''], ['X', '', '']]"

Python Examples

```
funcs = []
results = []
for x in range(3):
    def some_func():
        return x
    funcs.append(some_func)
    results.append(some_func())  # note the function call here

funcs_results = [func() for func in funcs]
print(results) # [0,1,2]
print(funcs_results)
```

What is the output of this code?

a. Prints "[0,1,2]"
b. Prints "[2,2,2]"
c. Prints "[]"
d. Raises an Error

Python Examples

You can find more such Python examples here [1]

[1] https://github.com/satwikkansal/wtfpython

He'll upload code
You can still upload your notes

Go through the Python examples and see what you expect it to produce, and see what is different, and try to understand why

wtfpython is a great repo for understanding these kind of things

Try to implement this workflow when you can

Tendency to make things perfect from the getgo, but the priority is to get something done

```
git checkout -b triangle
git log
mkdir 03-triangle
cd


# Missed some
# Can use VS code, pycharm, spyder, whatever
# Create a triangle.txt. file that is the triangle from the exercise
# Create a triangle.py file that contains code
```

File - C:\Users\Wolfs\PycharmProjects\cs506_20210915\triangle.py

```
# Basically wrote the highest level function first, and worked backwards
# To convince you that it will work, now read from bottom up

def read_triangle(path_to_file):
    # Input in a string representing the file location
    # returns a list of list of ints
    f = open(path_to_file, "r")
    lines = f.readlines()
    triangle = []
    for line in lines:
        # Split and make into int with list comprehension
        triangle.append([int(x) for x in line.split(sep=' ')])
    # Close file since we didn't use with open
    f.close()
    return triangle


def add_levels(x, y):
    if len(x) != len(y):
        raise(ValueError("lengths not equal"))
    res = []
    for i in range(len(x)):
        res.append(x[i] + y[i])
    return res


def elementwise_max(x, y):
    # If you wanted to track path, then this function needs to be modified
    if len(x) != len(y):
        raise(ValueError("lengths not equal"))
    res = []
    for i in range(len(x)):
        res.append(max(x[i], y[i]))
    return res


```

File - C:\Users\Wolfs\PycharmProjects\cs506_20210915\triangle.py

```
37 def fold_level(cur, nxt):
38     #
39     # Need to know what is the left and the right
40     # Add 0 to right of prev level to get the left
41     # Add 0 to left of prev level to get the right
42     # Try to have descriptive names that are not
   misleading
43     # Add 0 to left side:
44     right_options = add_levels([0] + cur, nxt)
45     # Add to right side:
46     left_options = add_levels(cur + [0], nxt)
47     max_options = elementwise_max(right_options,
   left_options)
48     return max_options
49
50
51 # Write this next function with a top down approach
52 def fold(triangle):
53     # Returns list of max path sums
54     # note: does not work when triangle length is 1 (
   can enhance later for edge case)
55     current_level = triangle[0]
56     for level in triangle[1:]:
57         current_level = fold_level(current_level,
   level)
58     return current_level
59
60
61 # or "./triangle.txt"
62 # print(read_triangle("triangle.txt"))
63 print(max(fold(read_triangle("triangle.txt"))))
64 # Result: 23, it works
65 # Big triangle: 1074
66 print(max(fold(read_triangle("triangle2.txt"))))
```



```
23
1074

Process finished with exit code 0
```

20210915 Introduction

Wednesday, September 15, 2021 3:19 PM

Introduction

Introduction

Boston University CS 506 - Lance Galletti

Now starting the actual data science

Data Representation

How we represent data is linked to what information we are able to retrieve from it.

Data Representation - Records

m-dimensional points / vectors

Example: (name, age, balance) -> (“John”, 20, 100)

Typically stored as a tuple representing a vector

Data Representation - Records

m-dimensional points / vectors

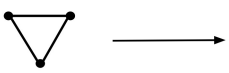
Example: (name, age, balance) -> (“John”, 20, 100)

Now you can do anything in the realm of linear algebra on these two vectors

Data Representation - Graphs

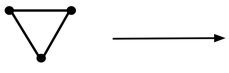
Nodes connected by edges

Example:



Adjacency Matrix

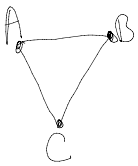
$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

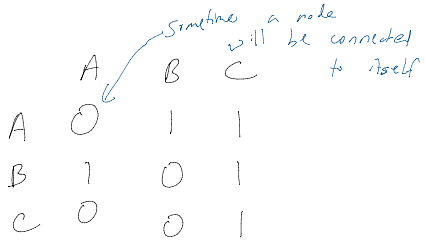


Adjacency List

1 : {2, 3}
2 : {1, 3}
3 : {1, 2}

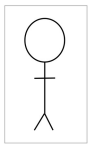
- We'll see more of this at the end of the semester
- Downward facing triangle
  - One way to represent this is through an adjacency matrix



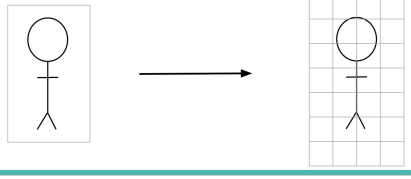


Adjacency list may be more space efficient

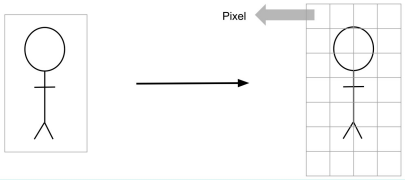
Data Representation - Images



Data Representation - Images

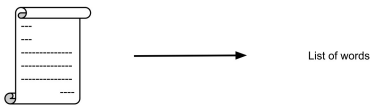


Data Representation - Images



- Matrix or list of pixels

Data Representation - Text



- text is usually a list of words

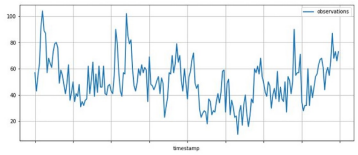
Data Representation - Strings

DNA seq (A T G C C G T A …) -> list of characters

- DNA sequence

Data Representation - Time Series

List of data at specific intervals of time



- Can be any of the data we saw before
  - Video, images
  - or simple values like temp as a function of time
    - Metal plate example Lec0 was a time series

Types of Learning

- Supervised Learning
- Unsupervised Learning

- We will tackle two different types of learning
  - Unsupervised first and is most of the course
  - Supervised learning
    - What people think of for machine learning?

Supervised Learning

cricket chirps / min	temperature
10	40
5	37
17	53
55	103
40	78

- Total nonsense data

Supervised Learning

cricket chirps / min	temperature
10	40
5	37
17	53
55	103
40	78

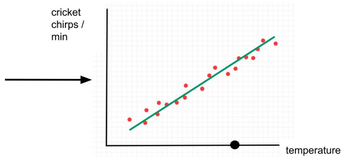
Supervised Learning

cricket chirps / min	temperature
10	40
5	37
17	53
55	103
40	78

- Want to create some sort of model (green line)
- For a temperature, what is your expected number of cricket chirps

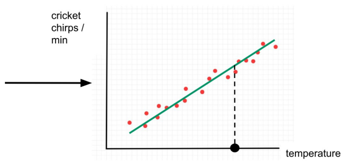
Supervised Learning

cricket chirps / min	temperature
10	40
5	37
17	53
55	103
40	78



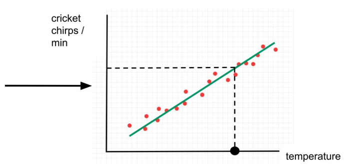
Supervised Learning

cricket chirps / min	temperature
10	40
5	37
17	53
55	103
40	78



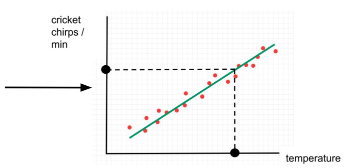
Supervised Learning

cricket chirps / min	temperature
10	40
5	37
17	53
55	103
40	78



Supervised Learning

cricket chirps / min	temperature
10	40
5	37
17	53
55	103
40	78



This type of supervised learning is referred to as regression

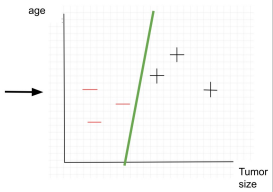
- Will see regression again in this course

Supervised Learning

age	tumor size	malignant
20	12	0
22	15	1
47	20	1
59	2	1

Supervised Learning

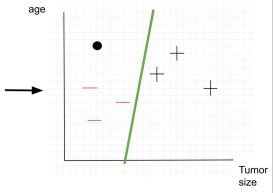
age	tumor size	malignant
20	12	0
22	15	1
47	20	1
59	2	1



- Again simplified unreal dataset

Supervised Learning

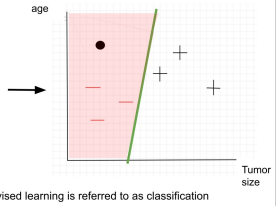
age	tumor size	malignant
20	12	0
22	15	1
47	20	1
59	2	1



- If we have a new point, do we have a malignant or a benign tumor

Supervised Learning

age	tumor size	malignant
20	12	0
22	15	1
47	20	1
59	2	1



This type of supervised learning is referred to as classification

- Want to be able to classify new datapoints

Unsupervised Learning

Goal: Find interesting structure in the data

- Unsupervised
  - Less interested in the specific label, but is interested in the inherent structure of the data
    - Like clusters

Unsupervised Learning

Goal: Find interesting structure in the data

This type of unsupervised learning is referred to as clustering

- Clustering
  - Assign each point to a given cluster
  - Will cover this a lot

Unsupervised Learning

Dataset: Collection of Articles

Question: Are these articles covering the same topics?

- Will collection of articles or books
  - High level concepts: Space, …
  - Similarity of concepts shows up in the structure of the article
  - No one told us what topic the article was, we are grouping articles into topics

20210915 Distance & Similarity

Wednesday, September 15, 2021 3:18 PM

Distance & Similarity

Distance & Similarity

Boston University CS 506 - Lance Galletti

- Starting with some unsupervised tools

Data

$$\underbrace{\begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{im} \\ \vdots & & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nm} \end{pmatrix}}_{\textbf{m features}} \left.\vphantom{\begin{pmatrix} a \\ \vdots \\ a \end{pmatrix}}\right\} \textbf{n data points}$$

- m*n attributes?

Feature Space

From our data we can generate a **feature space** of all possible values for the set of features in our data.

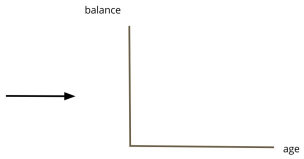
name	age	balance
Jane	25	150
John	30	100

- Name, age, balance
  - Name shouldn't matter in a data science model
    - Maybe it would in some models
  - Focus on the numerical features

Feature Space

From our data we can generate a **feature space** of all possible values for the set of features in our data.

name	age	balance
Jane	25	150
John	30	100

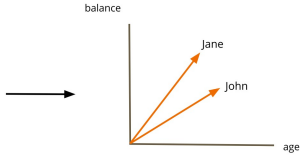


- What are all the possible values age can have
- What are all the possible values balance can have

Feature Space

From our data we can generate a **feature space** of all possible values for the set of features in our data.

name	age	balance
Jane	25	150
John	30	100



Our feature space is the Euclidean plane

- Plot our vectors in this feature space

Distance

In order to uncover interesting structure from our data, we need a way to **compare** data points.

A **dissimilarity function** is a function that takes two objects (data points) and returns a **large value** if these objects are **dissimilar**.

A special type of dissimilarity function is a **distance** function

- Key point for unsupervised learning:
  - Really fundamental thing in the dataset
  - Are these points similar, dissimilar, etc
  - Want to be able to compare the datapoints in some way
  - Don't need a distance function, but we could make a dissimilarity function
  - Distance is a more mathematical function

Distance

d is a distance function if and only if:

- d(i, j) = 0 if and only if i = j
- d(i, j) = d(j, i)
- d(i, j) ≤ d(i, k) + d(k, j)

We don't **need** a distance function to compare data points, but why would we prefer using a distance function?

- What is a distance function?
  - e.g. Square root of the sum of squares
- For a distance function to be defined, it needs 3 properties
- if and only if (iff) statement required in both directions
- Why would we prefer a distance function?
  - Always positive
  - Easy to understand and graph
  - Is intuitive

Minkowski Distance

For **x**, **y** points in **d**-dimensional real space

I.e. $\mathbf{x} = [x_1, \dots, x_d]$ and $\mathbf{y} = [y_1, \dots, y_d]$

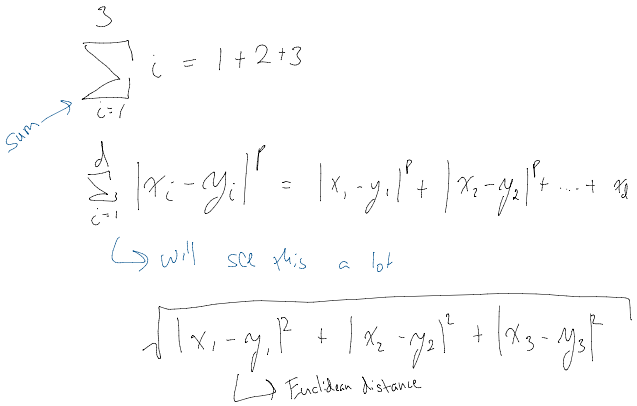
p ≥ 1

$$L_p(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

When **p** = 2 -> Euclidean Distance

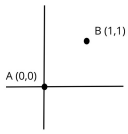
When **p** = 1 -> Manhattan Distance

- WARNING: Math (lol)
- Hopefully you've taken linear algebra...
- d-dimensional points have d features/components
- Lingoski? function
  - Should have seen this before, review if needed
  - Take the p-th root



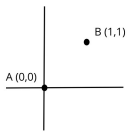
Example

d = 2



Example

d = 2

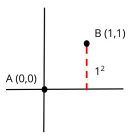


p = 2 $L_p(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$

- Standard Euclidean distance you should already know

Example

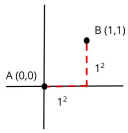
d = 2



p = 2 $L_p(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$

Example

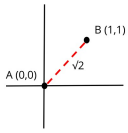
d = 2



p = 2 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

Example

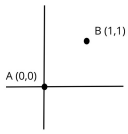
d = 2



p = 2 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

Example

d = 2

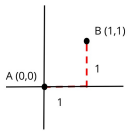


p = 1 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

- if p=1
- This is Manhatten distance
- Can only move along x and y and not directly

Example

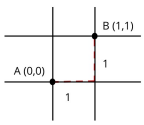
d = 2



p = 1 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

Example

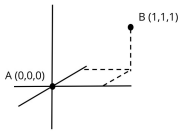
d = 2



p = 1 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

Example

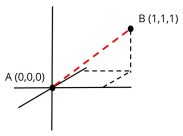
d = 3



p = 2 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

Example

d = 3

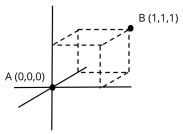


p = 2 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

- Euclidean: can go directly

Example

d = 3

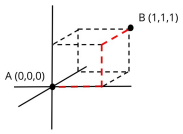


p = 1 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

- Manhatten distance
  - Move along the 3D grid edges
  - Multiple ways to get to B, but all length 3

Example

d = 3



p = 1 $L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

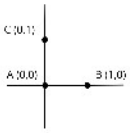
Minkowski Distance

Is L_p a distance function when **0 < p < 1** ?

- At beginning we said p>=1
  - Would it still be a distance function?
    - Can you find a counter example?

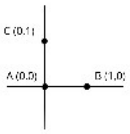
Minkowski Distance

Is L_p a distance function when $0 < p < 1$?



Minkowski Distance

Is L_p a distance function when $0 < p < 1$?



D(B,A) = D(A, C) = 1

D(B, C) = $2^{1/p}$

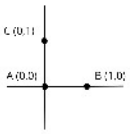
$D(B,C) = (1+1)^{1/p}$

if $p < 1$

then $1/p > 1$

Minkowski Distance

Is L_p a distance function when $0 < p < 1$?



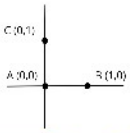
D(B,A) + D(A, C) = 2

D(B, C) = $2^{1/p}$

But... if **p < 1** then **1/p > 1**

Minkowski Distance

Is L_p a distance function when $0 < p < 1$?



D(B,A) + D(A, C) = 2

D(B, C) = $2^{1/p}$

So **D(B, C) > D(B, A) + D(A, C)** which violates the triangle inequality

Cosine Similarity

A **similarity** function is a function that takes two objects (data points) and returns a **large value** if these objects are **similar**.

$$s(x, y) = \cos(\theta)$$

where θ is the angle between **x** and **y**

Monday: spark pitches are remote, don't come to class
Have office hours after this

Cosine Similarity

To get a corresponding **dissimilarity** function, we can usually try

d(x, y) = 1 / s(x, y)

or

d(x, y) = k - s(x, y) for some **k**

Here, we can use

d(x, y) = 1 - s(x, y)

Cosine Similarity

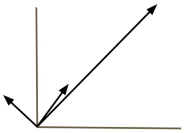
When should you use **cosine (dis)similarity** over **euclidean distance**?

When **direction** matters more than **magnitude**

Cosine Similarity

When should you use **cosine (dis)similarity** over **euclidean distance**?

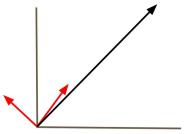
When **direction** matters more than **magnitude**



Cosine Similarity

When should you use **cosine (dis)similarity** over **euclidean distance**?

When **direction** matters more than **magnitude**

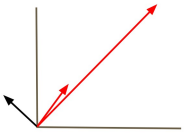


Close under
Euclidean distance

Cosine Similarity

When should you use **cosine (dis)similarity** over **euclidean distance**?

When **direction** matters more than **magnitude**



Close under Cosine
Similarity

Jaccard Similarity

How similar are the following documents?

	w_1	w_2	...	w_d
x	1	0	...	1
y	1	1	...	0

Jaccard Similarity

One way is to use the Manhattan distance which will return the size of the set difference

	w_1	w_2	...	w_d
x	1	0	...	1
y	1	1	...	0

$$L_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$

Jaccard Similarity

One way is to use the Manhattan distance which will return the size of the set difference

	w_1	w_2	...	w_d
x	1	0	...	1
y	1	1	...	0

$$L_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$

Will only be 1 when $x_i \neq y_i$

Jaccard Similarity

But how can we distinguish between these two cases?

	w_1	w_2	...	w_{d-1}	w_d
x	1	1	1	0	1
y	1	1	1	1	0

Only differ on the last two words

	w_1	w_2
x	0	1
y	1	0

Completely different

Jaccard Similarity

But how can we distinguish between these two cases?

	w_1	w_2	...	w_{d-1}	w_d
x	1	1	1	0	1
y	1	1	1	1	0

Only differ on the last two words

	w_1	w_2
x	0	1
y	1	0

Completely different

Both have Manhattan distance of 2

Jaccard Similarity

We need to account for the size of the intersection!

$$JSim(x, y) = \frac{|x \cap y|}{|x \cup y|}$$

$$JDist(x, y) = 1 - \frac{|x \cap y|}{|x \cup y|}$$

Implement these distance functions in the CS506 python package