learn-co-curriculum / dsc-distance-metrics-lab Public

View license

0 stars 146 forks

Star Watch

Code Issues Pull requests Actions Projects Security Insights

solution ...

This branch is 9 commits ahead, 11 commits behind master.

sumedh10 update readme ... on Nov 12, 2019 10

View code

README.md

Distance Metrics - Lab

Introduction

In this lab, you'll calculate various distances between multiple points using the distance metrics you learned about!

Objectives

In this lab you will:

- Calculate Manhattan distance between two points
- Calculate Euclidean distance between two points
- Calculate Minkowski distance between two points

Getting Started

You'll start by writing a generalized function to calculate any of the three distance metrics you've learned about. Let's review what you know so far:

> The ***Manhattan distance*** and ***Euclidean distance*** are both special cases of ***Minkowski distance***.

Take a look at the formula for Minkowski distance below:

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

Manhattan distance is a special case where $c = 1$ in the equation above (which means that you can remove the root operation and just keep the summation).

Euclidean distance is a special case where $c = 2$ in the equation above.

Knowing this, you can create a generalized `distance()` function that calculates Minkowski distance, and takes in `c` as a parameter. That way, you can use the same function for every problem, and still calculate Manhattan and Euclidean distance metrics by just passing in the appropriate values for the `c` parameter!

In the cell below:

- Complete the `distance()` function which should implement the Minkowski distance equation above to return the distance, a single number

- This function should take in 4 arguments:

  - `a` : a tuple or array that describes a vector in n-dimensional space
  - `b` : a tuple or array that describes a vector in n-dimensional space (this must be the same length as `a` !)
  - `c` : which tells us the norm to calculate the vector space (if set to `1` , the result will be Manhattan, while `2` will calculate Euclidean distance)
  - `verbose` : set to `True` by default. If true, the function should print out if the distance metric returned is a measurement of Manhattan, Euclidean, or Minkowski distance

- Since euclidean distance is the most common distance metric used, this function should default to using `c=2` if no value is set for `c`

HINT:

1. You can avoid using a `for` loop like we did in the previous lesson by simply converting the tuples to NumPy arrays

2. Use `np.power()` as an easy way to implement both squares and square roots. `np.power(a, 3)` will return the cube of `a`, while `np.power(a, 1/3)` will return the cube root of 3. For more information on this function, refer the NumPy documentation!

```
import numpy as np

def distance(a, b, c=2, verbose=True):
    if len(a) != len(b):
        raise ValueError("Both vectors must be of equal length!")

    if verbose:
        if c == 1:
            print("Calculating Manhattan distance:")
        elif c == 2:
            print("Calculating Euclidean distance:")
        else:
            print(f"Calcuating Minkowski distance (c={c}):")

    return np.power(np.sum(np.power(np.abs(np.array(a) - np.array(b)), c)), 1/c)

test_point_1 = (1, 2)
test_point_2 = (4, 6)
print(distance(test_point_1, test_point_2)) # Expected Output: 5.0
print(distance(test_point_1, test_point_2, c=1)) # Expected Output: 7.0
print(distance(test_point_1, test_point_2, c=3)) # Expected Output: 4.49794144527541
```

```
Calculating Euclidean distance:
5.0
Calculating Manhattan distance:
7.0
Calcuating Minkowski distance (c=3):
4.497941445275415
```

Great job!

Now, use your function to calculate distances between points:

Problem 1

Calculate the ***Euclidean distance*** between the following points in 5-dimensional space:

Point 1: (-2, -3.4, 4, 15, 7)

Point 2: (3, -1.2, -2, -1, 7)

```
print(distance((-2, -3.4, 4, 15, 7), (3, -1.2, -2, -1, 7))) # Expected Output: 17.93
```

```
Calculating Euclidean distance:
17.939899665271266
```

Problem 2

Calculate the ***Manhattan distance*** between the following points in 10-dimensional space:

Point 1: [0, 0, 0, 7, 16, 2, 0, 1, 2, 1]
Point 2: [1, -1, 5, 7, 14, 3, -2, 3, 3, 6]

```
print(distance( [0, 0, 0, 7, 16, 2, 0, 1, 2, 1],  [1, -1, 5, 7, 14, 3, -2, 3, 3, 6],
```

```
Calculating Manhattan distance:
20.0
```

Problem 3

Calculate the ***Minkowski distance*** with a norm of 3.5 between the following points:

Point 1: (-2, 7, 3.4)
Point 2: (3, 4, 1.5)

```
print(distance((-2, 7, 3.4), (3, 4, 1.5), c=3.5)) # Expected Output: 5.2687896591883
```

```
Calcuating Minkowski distance (c=3.5):
5.268789659188307
```

Summary

Great job! Now that you know about the various distance metrics, you can use them to writing a K-Nearest Neighbors classifier from scratch!

Releases

No releases published

Packages

No packages published

Contributors 5

Languages

- Jupyter Notebook 100.0%