An Illustrative Explanation t Dynamic Time Warpi

+ math behind it

Essi Alizadeh
Oct 11, 2020 7 min read

Getting Started

Dynamic Time Warping (DTW) is a way to compare tv temporal- sequences that do not sync up perfectly. It calculate the optimal matching between two sequenc many domains such as speech recognition, data mini markets, etc. It's commonly used in data mining to me between two time-series.

In this post, we will go over the mathematics behind D illustrative examples are provided to better understan you are not interested in the math behind it, please ju

---

Formulation

X=x[1], x[2], …, x[i], …, x[n]

Y=y[1], y[2], …, y[j], …, y[m]

The sequences X and Y can be arranged to form an n-by-m grid, where each point (i, j) is the alignment between $x[i]$ and $y[j]$.

A warping path W maps the elements of X and Y to m

LATEST

dis_tance be_tween them. W is a sequer

see an example of the warping path later

Recommended Articles
DEEP DIVES
NEWSLETTER
WRITE FOR TDS
Sign in
Submit an Article

Warping Path and DTW distance

The Optimal path to (i_k, j_k) can be computed by:

$$D_{min}(i_k, j_k) = \min_{i_{k-1},\, j_{k-1}} D_{min}(i_{k-1}, j_{k-1}) + d($$

where d is the Euclidean distance. Then, the overall p
calculated as

$$D = \sum_k d(i_k, j_k)$$

Restrictions on the Warping function

The warping path is found using a dynamic programm
align two sequences. Going through all possible paths
"combinatorically explosive" [1]. Therefore, for efficier
important to limit the number of possible warping path
following constraints are outlined:

- **Boundary Condition**: This constraint ensures that the warping path begins with the start points of both signals and terminates with their endpoints.

$$i_1 = 1, i_k = n \quad \text{and} \quad j_1 = 1, j_k = m$$

- **Monotonicity condition**: This constraint preserv[…] points (not going back in time).

$$i_{t-1} \leq i_t \quad \text{and} \quad j_{t-1} \leq j_t$$

- **Continuity (step size) condition**: This constrair transitions to adjacent points in time (not jumping

$$i_t - i_{t-1} \leq 1 \quad \text{and} \quad j_t - j_{t-1} \leq 1$$

In addition to the above three constraints, there are o conditions for an allowable warping path:

- **Warping window condition**: Allowable points ca fall within a given warping window of width ω (a p

$$|i_t - j_t| \leq \omega$$

- **Slope condition**: The warping path can be cons restricting the slope, and consequently avoiding e movements in one direction.

An acceptable warping path has combinations of **che** are:

- Horizontal moves: $(i, j) \rightarrow (i, j+1)$
- Vertical moves: $(i, j) \rightarrow (i+1, j)$
- Diagonal moves: $(i, j) \rightarrow (i+1, j+1)$

Implementation

Let's import all python packages we neec

```
import pandas as pd
import numpy as np

# Plotting Packages
import matplotlib.pyplot as plt
import seaborn as sbn

# Configuring Matplotlib
import matplotlib as mpl
mpl.rcParams['figure.dpi'] = 300
savefig_options = dict(format="png", dpi=300, bbox_inc

# Computation packages
from scipy.spatial.distance import euclidean
from fastdtw import fastdtw
```

Let's define a method to compute the accumulated cc warp path. The cost matrix uses the Euclidean distan distance between every two points. The methods to c Euclidean distance matrix and accumulated cost matı below:

```
def compute_euclidean_distance_matrix(x, y) -> np.array:
```

```
    """Calculate distance matrix
    This method calcualtes the pairwise Euclidean distance between two sequences.
    The sequences can have different lengths.
    """
    dist = np.zeros((len(y), len(x)))
    for i in range(len(y)):
        for j in range(len(x)):
            dist[i,j] = (x[j]-y[i])**2
    return dist
```

dtw_compute_euclidean_distance_matrix.py hosted with ❤ by Gi


```
def compute_accumulated_cost_matrix(x, y) -> np.
    """Compute accumulated cost matrix for warp path using Euclic
    """
    distances = compute_euclidean_distance_matrix(x, y)

    # Initialization
    cost = np.zeros((len(y), len(x)))
    cost[0,0] = distances[0,0]

    for i in range(1, len(y)):
        cost[i, 0] = distances[i, 0] + cost[i-1, 0]

    for j in range(1, len(x)):
        cost[0, j] = distances[0, j] + cost[0, j-1]

    # Accumulated warp path cost
    for i in range(1, len(y)):
        for j in range(1, len(x)):
            cost[i, j] = min(
                cost[i-1, j],    # insertion
                cost[i, j-1],    # deletion
                cost[i-1, j-1]   # match
            ) + distances[i, j]

    return cost
```

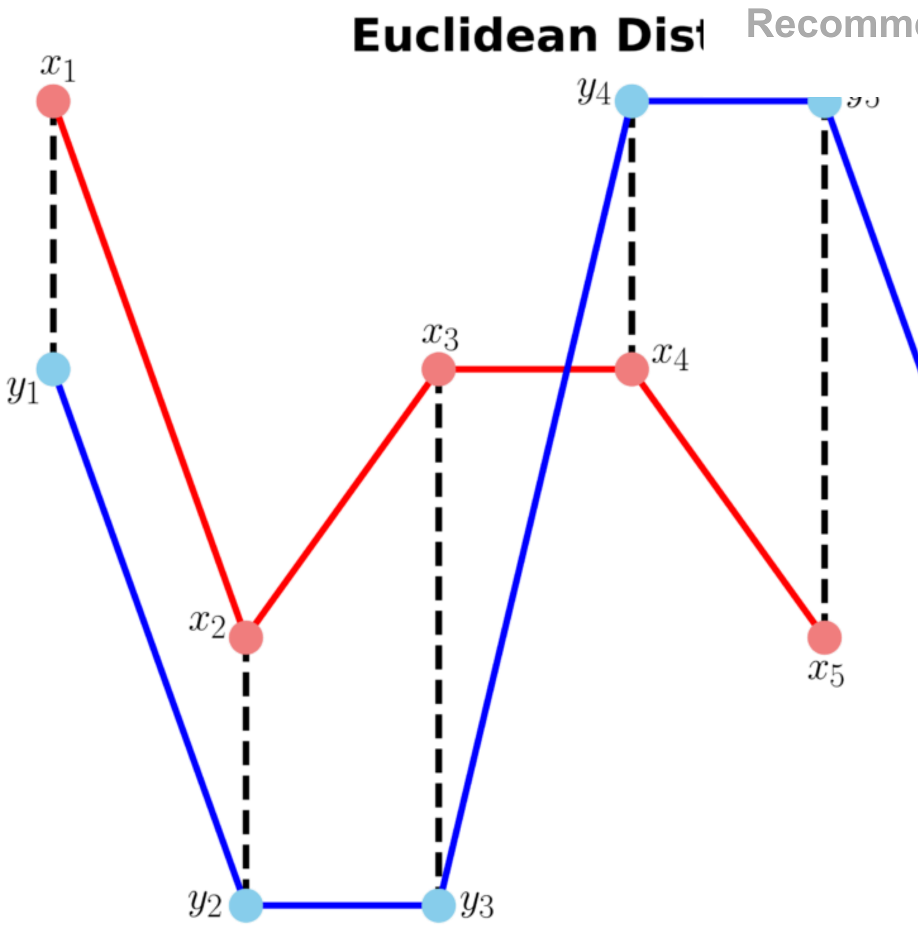
dtw_compute_accumulated_cost_matrix.py hosted with ❤ by GitHub

Example 1

In this example, we have two sequences *x* and *y* with different lengths.

```
# Create two sequences
x = [3, 1, 2, 2, 1]
y = [2, 0, 0, 3, 3, 1, 0]
```

We cannot calculate the Euclidean distance between *x* and *y* since they don't have equal lengths.



Example 1: Euclidean distance between *x* and y (is it possible? 🤔) (In

Compute DTW distance and warp pat

Many Python packages calculate the DTW by just prc
sequences and the type of distance (usually Euclidea

popular Python implementation of DTW that is FastDTW which is an approximate DTW algorithm with lower time and memory complexities [2].

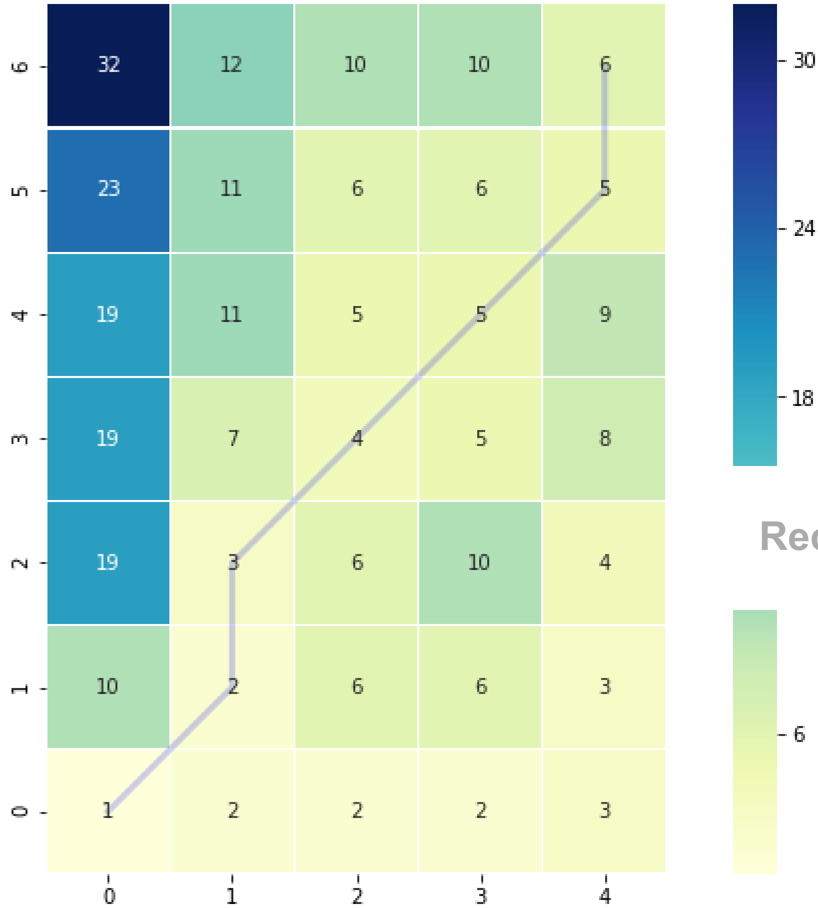
```
dtw_distance, warp_path = fastdtw(x, y, dist=euclidean)
```

Note that we are using SciPy's distance function *Euclidean* that we imported earlier. For a better understanding of the wa compute the accumulated cost matrix an grid. The following code will plot a heatm matrix.


```
cost_matrix = compute_accumulated_cost_matrix(x, y)
```

```
fig, ax = plt.subplots(figsize=(12, 8))
ax = sbn.heatmap(cost_matrix, annot=True, square=True, linewidths
ax.invert_yaxis()

# Get the warp path in x and y directions
path_x = [p[0] for p in warp_path]
path_y = [p[1] for p in warp_path]

# Align the path from the center of each cell
path_xx = [x+0.5 for x in path_x]
path_yy = [y+0.5 for y in path_y]

ax.plot(path_xx, path_yy, color='blue', linewidth=3, alpha=0.2)

fig.savefig("ex1_heatmap.png", **savefig_options)
```

dtw_warp_path_heatmap.py hosted with ❤ by GitHub



Example 1: Accumulated cost matrix and warping path (Image b

The color bar shows the cost of each point in the grid
the warp path (blue line) is going through the lowest c
Let's see the DTW distance and the warping path by
variables.

```
>>> DTW distance:  6.0
>>> Warp path: [(0, 0), (1, 1), (1, 2), (2, 3), (3, 4)
```

The warping path starts at point (0, 0) and ends at (4,
Let's also calculate the accumulated cost most using
defined earlier and compare the values with the heatr

```
cost_matrix = compute_accumulated_cost_matrix(x, y)
print(np.flipud(cost_matrix)) # Flipping the cost matr
```

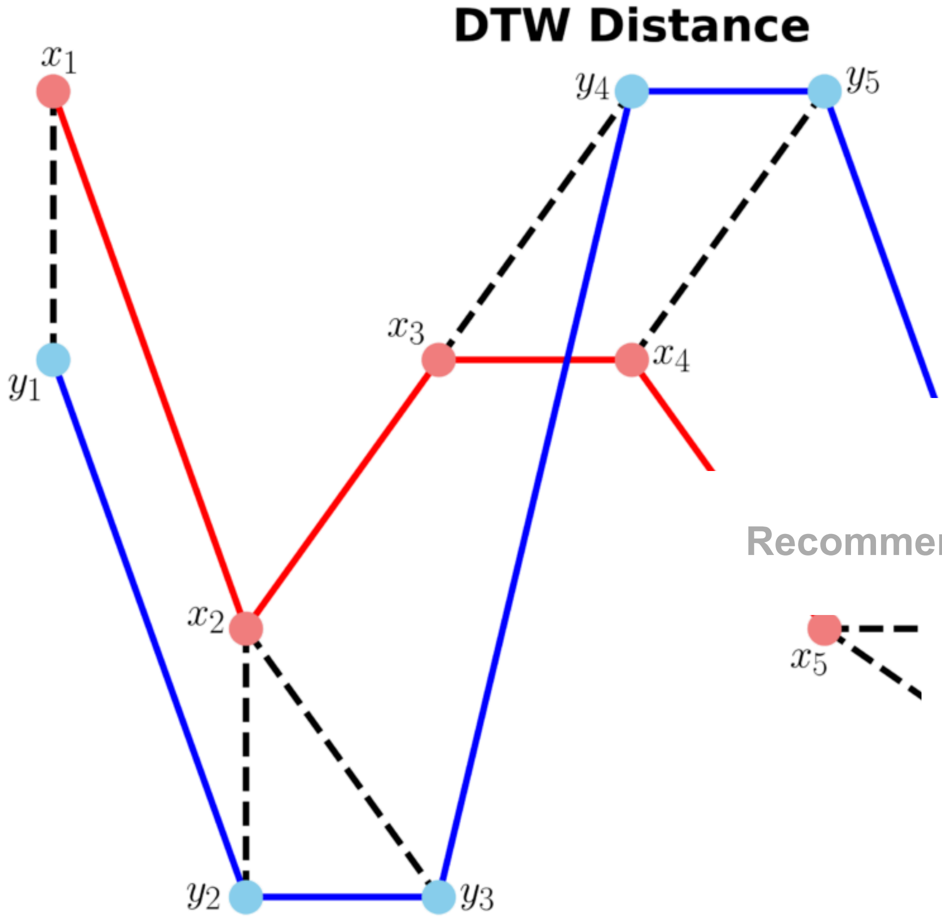
```
>>> [[32. 12. 10. 10.  6.]
     [23. 11.  6.  6.  5.]
     [19. 11.  5.  5.  9.]
     [19.  7.  4.  5.  8.]
     [19.  3.  6. 10.  4.]
     [10.  2.  6.  6.  3.]
     [ 1.  2.  2.  2.  3.]]
```


The cost matrix is printed above has simi

Now let's plot the two sequences and connect the ma code to plot the DTW distance between *x* and *y* is giv


```
fig, ax = plt.subplots(figsize=(14, 10))

# Remove the border and axes ticks
fig.patch.set_visible(False)
ax.axis('off')

for [map_x, map_y] in warp_path:
    ax.plot([map_x, map_y], [x[map_x], y[map_y]], '--k', linewidt

ax.plot(x, '-ro', label='x', linewidth=4, markersize=20, markerfa
ax.plot(y, '-bo', label='y', linewidth=4, markersize=20, markerfa
ax.set_title("DTW Distance", fontsize=28, fontweight="bold")

fig.savefig("ex1_dtw_distance.png", **savefig_options)
```

dtw_ex1_dtw_plot.py hosted with ❤ by GitHub



Example 1: DTW distance between *x* and y (Image by Auth

Example 2

In this example, we will use two sinusoidal signals an be matched by calculating the DTW distance betweer

```
time1 = np.linspace(start=0, stop=1, num=50)
time2 = time1[0:40]

x1 = 3 * np.sin(np.pi * time1) + 1.5 * np.sin(4*np.pi * time1)
x2 = 3 * np.sin(np.pi * time2 + 0.5) + 1.5 * np.sin(4*np.pi * time
```

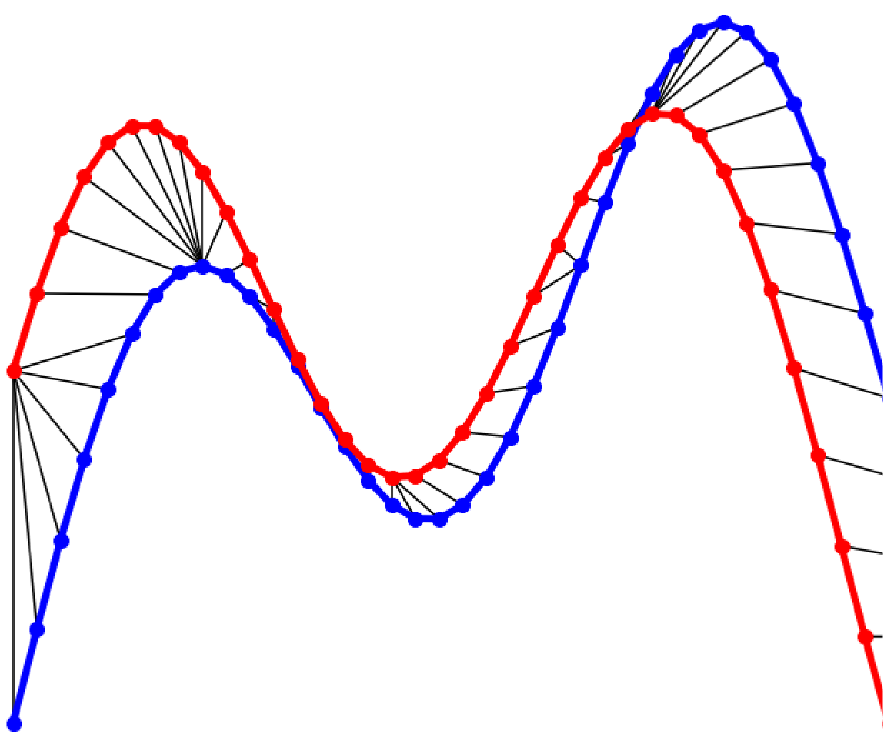
dtw_ex2.py hosted with ❤ by GitHub

Just like Example 1, let's calculate the DTW distance for *x1* and *x2* signals using FastDTW package.

```
distance, warp_path = fastdtw(x1, x2, dist=euclidean)
```

```
fig, ax = plt.subplots(figsize=(16, 12))

# Remove the border and axes ticks
fig.patch.set_visible(False)
ax.axis('off')

for [map_x, map_y] in warp_path:
    ax.plot([map_x, map_y], [x1[map_x], x2[map_y]], '-k')

ax.plot(x1, color='blue', marker='o', markersize
ax.plot(x2, color='red', marker='o', markersize=
ax.tick_params(axis="both", which="major", labelsize=18)

fig.savefig("ex2_dtw_distance.png", **savefig_options)
```

dtw_ex2_plot.py hosted with ❤ by GitHub



Example 2: DTW distance between *x1* and x2 (Image by Au

As can be seen in above figure, the DTW distance between the two signals is particularly powerful when the signals have similar patterns. The extrema (maximum and minimum points) between the two signals are correctly mapped. Moreover, unlike Euclidean distance, we may see many-to-one mapping when DTW distance is used, particularly if the two signals have different lengths.

You may spot an issue with dynamic time warping fro
Can you guess what it is?

The issue is around the head and tail of time-series tl
match. This is because the DTW algorithm cannot aff
invariance for at the endpoints. In short, the effect of t
difference at the sequence endpoints will tend to cont
disproportionately to the estimated similarity[3].

Conclusion

DTW is an algorithm to find an optimal alignment betv
sequences and a useful distance metric to have in ou
technique is useful when we are working with two nor
particularly if one sequence is a non-linear stretched/s
the other. The warping path is a combination of "ches
starts from the head of two sequences and ends with

You can find the Jupyter notebook for this blog post h
reading!

Thanks for reading! 📚

I'm a senior data scientist 📊 and engineer, writing about statistics, machine learning, Python, and more.

> 🌱 I also curate a weekly newsletter called **AI Sprout** where I provide hands-on reviews and analysis of the latest AI tools and innovations. Subscribe to explore emerging AI with me!

- *Follow me on Medium* 👋 *to get my latest post*
- *Subscribe to my mailing list* ✉️ *for u*
- _Let's connect on LinkedIn and Twitt _

Join Medium with my referral link – Esmaeil Ali

References

[1] Donald J. Berndt and James Clifford, Using Dynar Find Patterns in Time Series, 3rd International Confe Knowledge Discovery and Data Mining

[2] Salvador, S. and P. Chan, FastDTW: Toward accu warping in linear time and space __ (2007), Intelligen

[3] Diego Furtado Silva, *et al.*, On the effect of endpoi time warping (2016), SIGKDD Workshop on Mining a Time Series

---

Useful Links

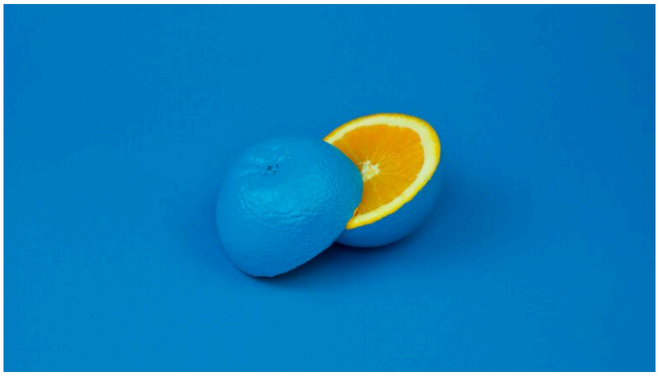
[1] https://nipunbatra.github.io/blog/ml/2014/05/01/dtw

[2] https://databricks.com/blog/2019/04/30/understanding-dynamic-time-warping.html

WRITTEN BY

Essi Alizadeh

Related Articles



DATA SCIENCE

Hands-on Time Series Anomaly Detection using Autoencoders, with Python

Here's how to use Autoencoders to detect signals with anomalies in a few lines of…

Piero Paialunga

August 21, 2024 12 min read

DATA SCIENCE

Back To Basic Regression ar

An illustrated guide learning concepts

Shreya Rao

February 3, 2023 6

LATEST

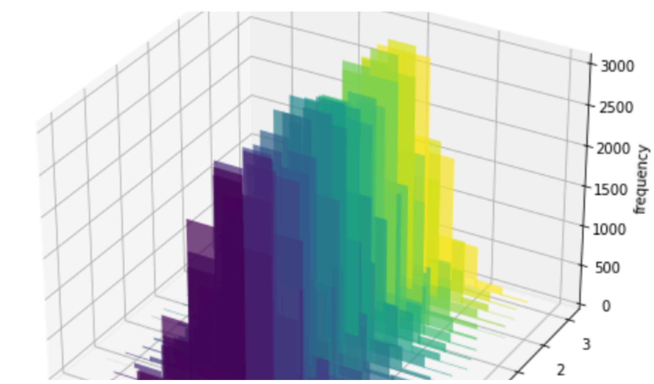
Recommended Articles

NEWSLETTER

WRITE FOR TDS

Sign in

Submit an Article



DATA SCIENCE

Must-Know in Statistics: The Bivariate Normal Projection Explained

Derivation and practical examples of this powerful concept

Luigi Battistoni

August 14, 2024 7 min read

DATA SCIENCE

Back to Basic: Logistic Regre

An illustrated guide know about Logisti

Shreya Rao

March 2, 2023 8 m

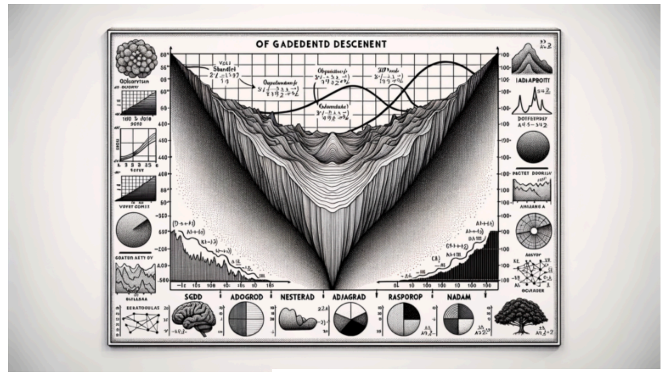
DATA SCIENCE

Squashing the Average: A Dive into Penalized Quantile Regression for Python

How to build penalized quantile regression models (with code!)

Álvaro Méndez Civieta

August 16, 2024 5 min read



DATA SCIENCE

Th
Op
an

This is a bit differe

Peng Qian

August 17, 2024 9

DATA SCIENCE

How Do Computers Actually Compute?

A Budding Data Scientist's Introduction to Computer Hardware

W. Caden Hamrick

June 10, 2024 10 min read

Subscribe to Our Newsletter

WRITE FOR TDS · ABOUT · ADVERTISE · PRIVACY POLICY · TERMS OF USE

COOKIES SETTINGS

LATEST

Recommended Articles

DEEP DIVES

NEWSLETTER

WRITE FOR TDS

Sign in

Submit an Article