

SP + ML 2: ANALYZING THE SIGNAL

CSE 599 Prototyping Interactive Systems | Lecture 12 | May 9

Jon Froehlich • Jasper O'Leary (TA)

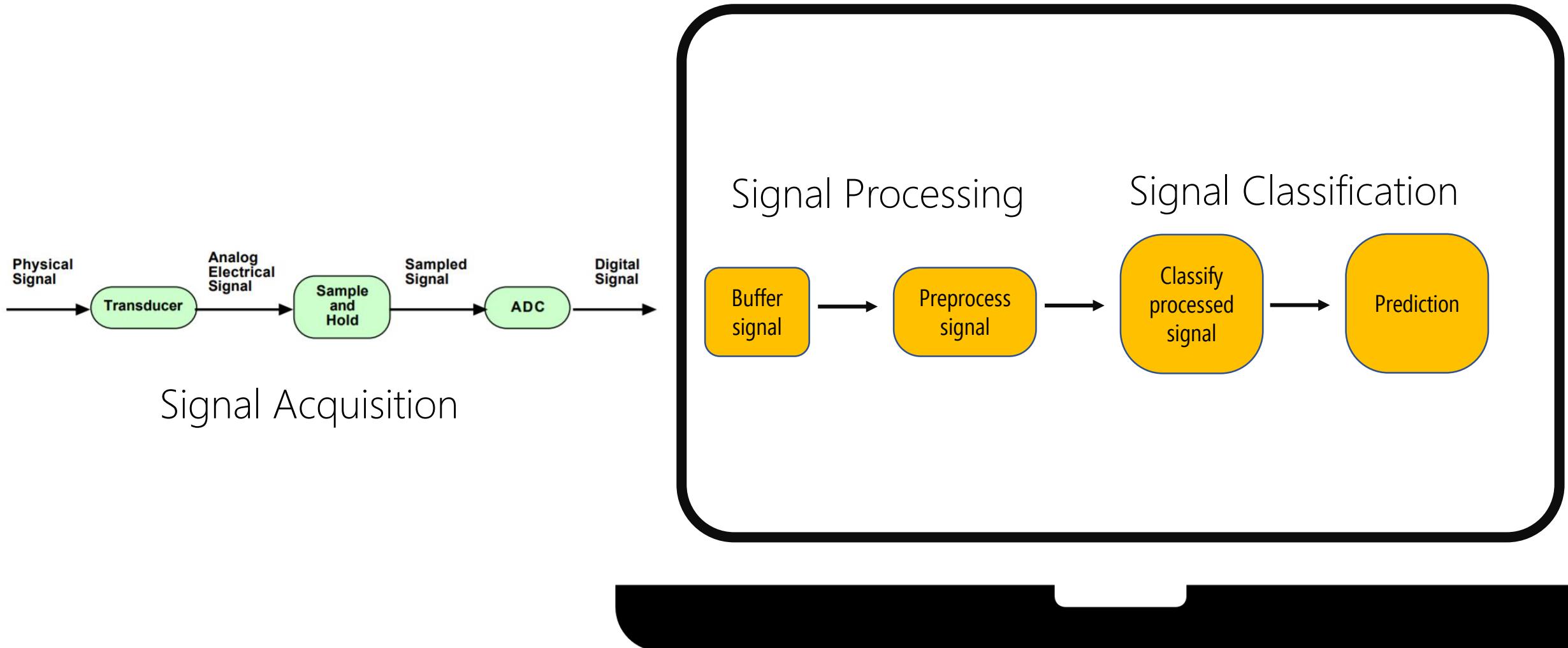
LEARNING GOALS

Basic **signal processing** strategies

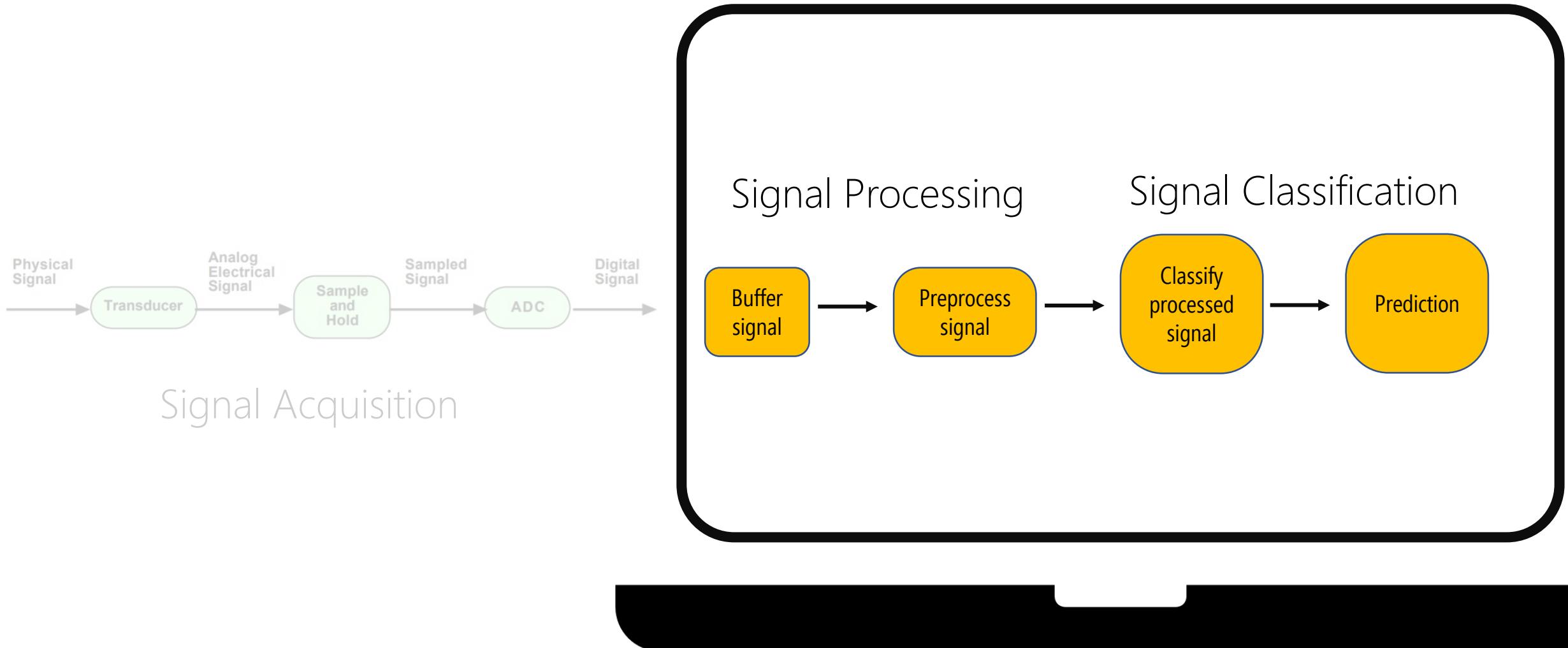
How to **compare two signals** using “shape matching”

General approaches in HCI/Ubicomp for processing & classifying signals

DATA FLOW FROM PHYSICAL SIGNAL TO PREDICTION



DATA FLOW FROM PHYSICAL SIGNAL TO PREDICTION

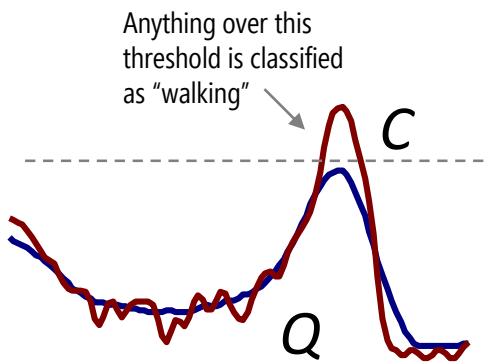


SP + ML IN HCI/UBICOMP

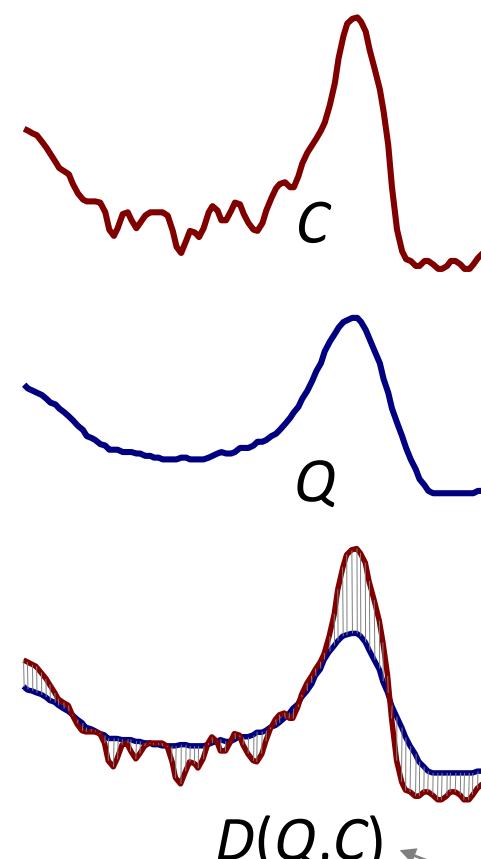
1. **Start by capturing signals under controlled conditions** with "ground truth" (e.g., walk ten steps, wait, walk ten steps, wait). Collect easy, medium, and hard examples.
2. **Use tools to visualize, study, & process the signal** (both in time space and frequency space). Identify patterns. Brainstorm potential approaches for analysis. Calculate descriptive stats (e.g., mean, median) both in time space & freq space.
3. **Search for previous solutions** to comparable signal processing/classification problems (e.g., Google Scholar). What aspects can you re-appropriate for your work?
4. **Generate** and **test approaches offline** using test data. Iterate.
5. When satisfied with offline performance, **adapt approach to perform in real-time**. If real-time performance does not meet expectations, collect larger, more naturalistic test datasets and repeat process.

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

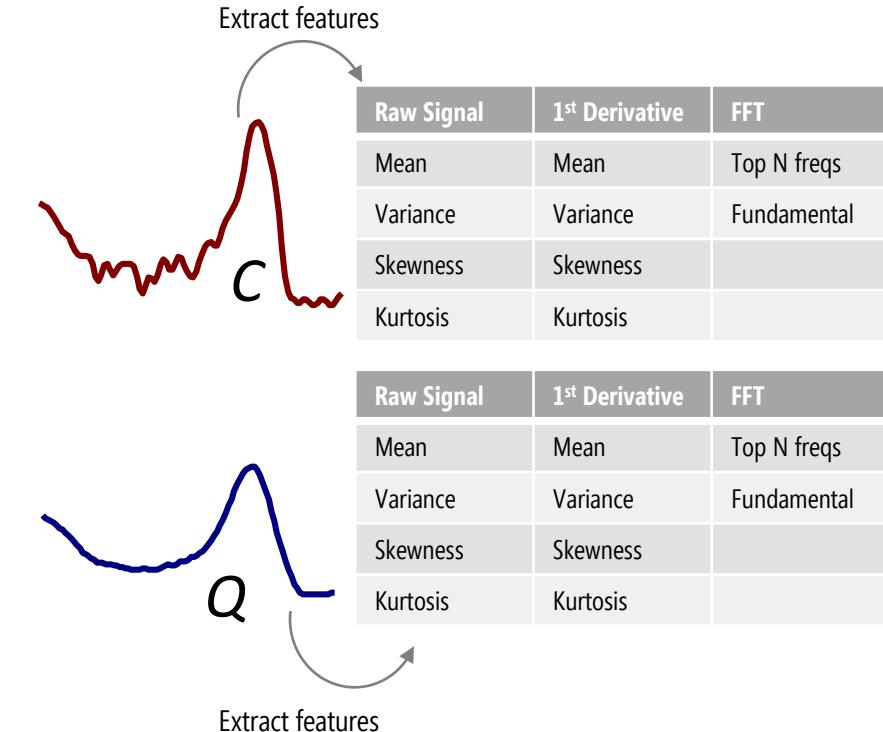
1. Rule-Based



2. Shape-Matching



3. Feature-Based

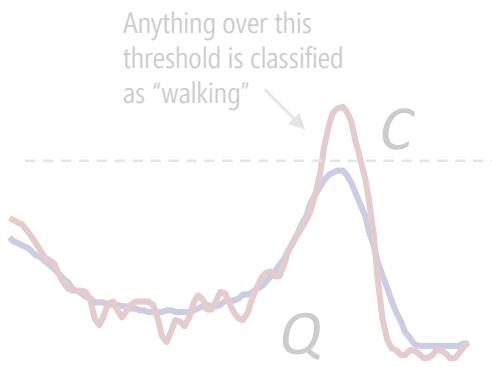


Supervised Learning Model

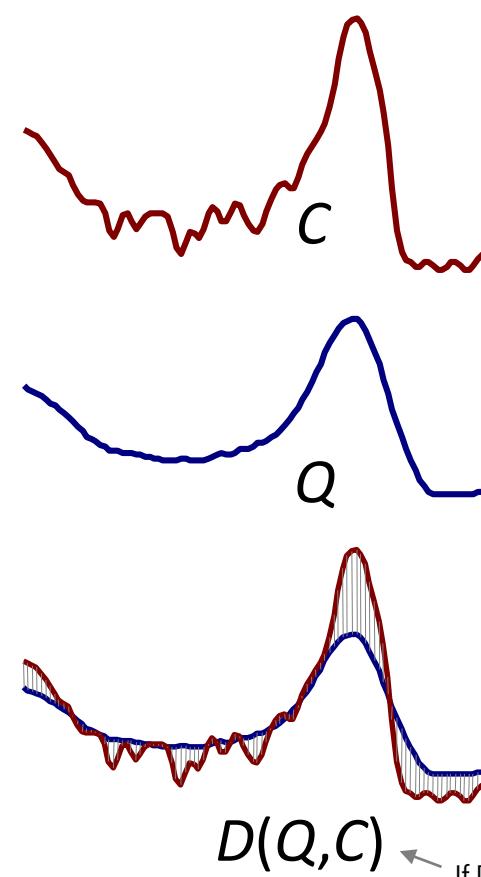
Output

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

1. Rule-Based



2. Shape-Matching



3. Feature-Based



Supervised Learning Model

Output

INTRO TO SHAPE-MATCHING

SIMILARITY METRICS

How do we formally define similarity?



DEFINING DISTANCE MEASURES

Let O_1 and O_2 be two objects from the universe of possible objects.

The distance (dissimilarity) is denoted by $D(O_1, O_2)$

Properties of a desirable distance metric:

$$D(A, B) = D(B, A)$$

Symmetry

$$D(A, A) = 0$$

Constancy

$$D(A, B) = 0 \text{ IIf } A = B$$

Positivity

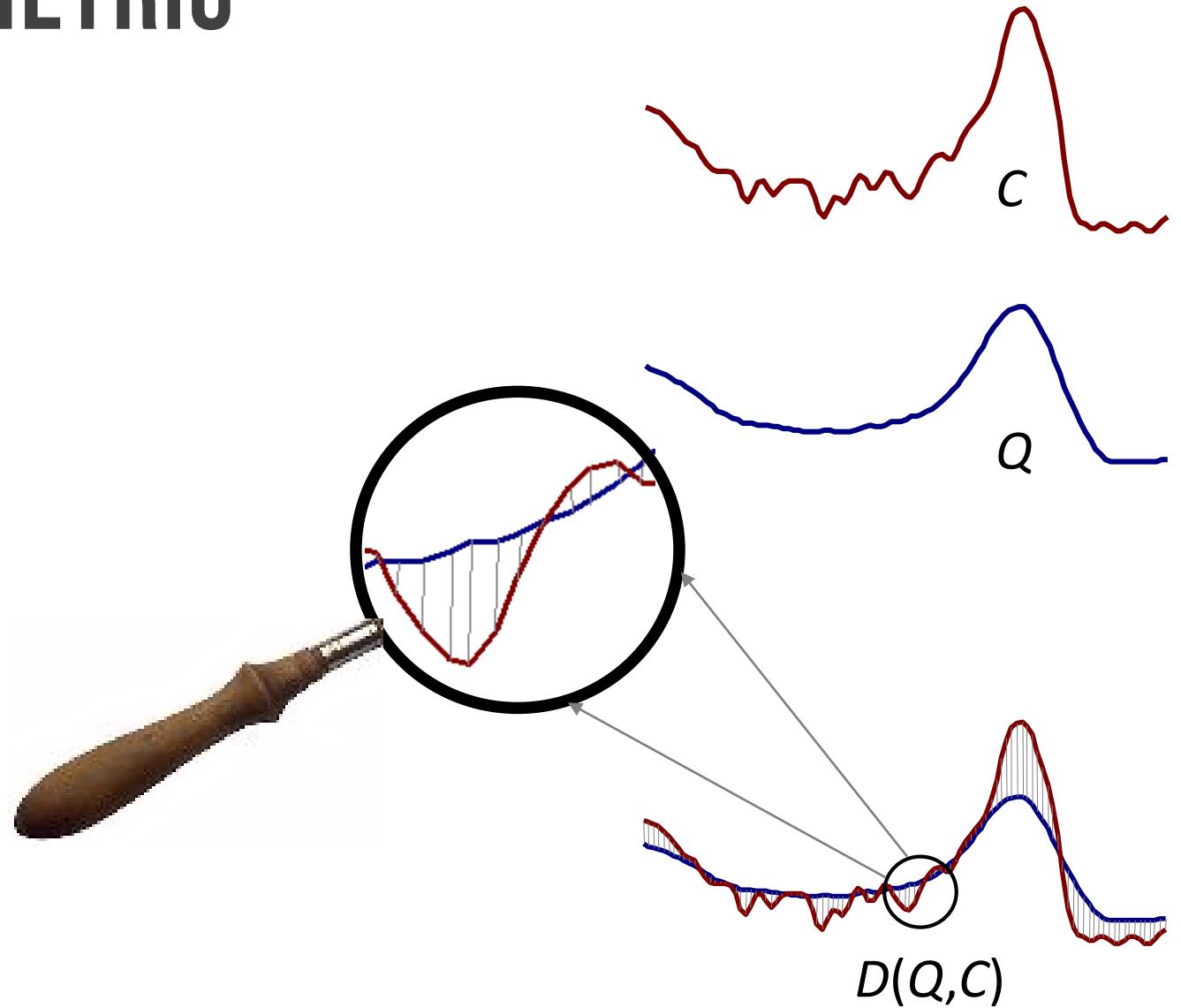
EUCLIDEAN DISTANCE METRIC

Given two time series:

$$Q = q_1 \dots q_n$$

$$C = c_1 \dots c_n$$

$$D(Q, C) \equiv \sqrt{\sum_{i=1}^n (q_i - c_i)^2}$$



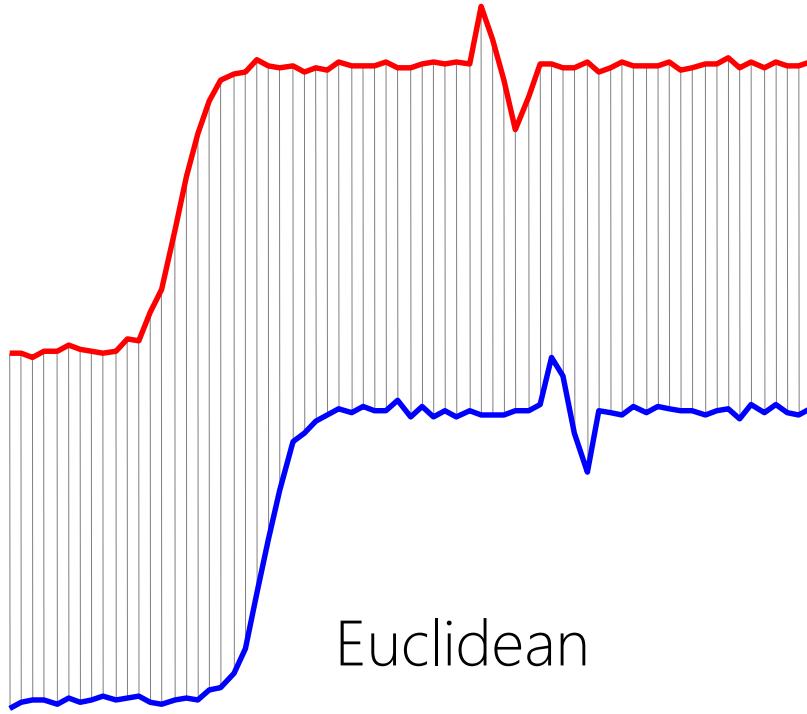
COMMON OPTIMIZATION OF EUCLID DISTANCE

Instead of calculating the raw Euclidean distance, calculate the squared Euclidean distance (to save CPU time)

$$D(Q, C) \equiv \sqrt{\sum_{i=1}^n (q_i - c_i)^2} \quad \rightarrow \quad D_{\text{squared}}(Q, C) \equiv \sum_{i=1}^n (q_i - c_i)^2$$

DYNAMIC TIME WARPING

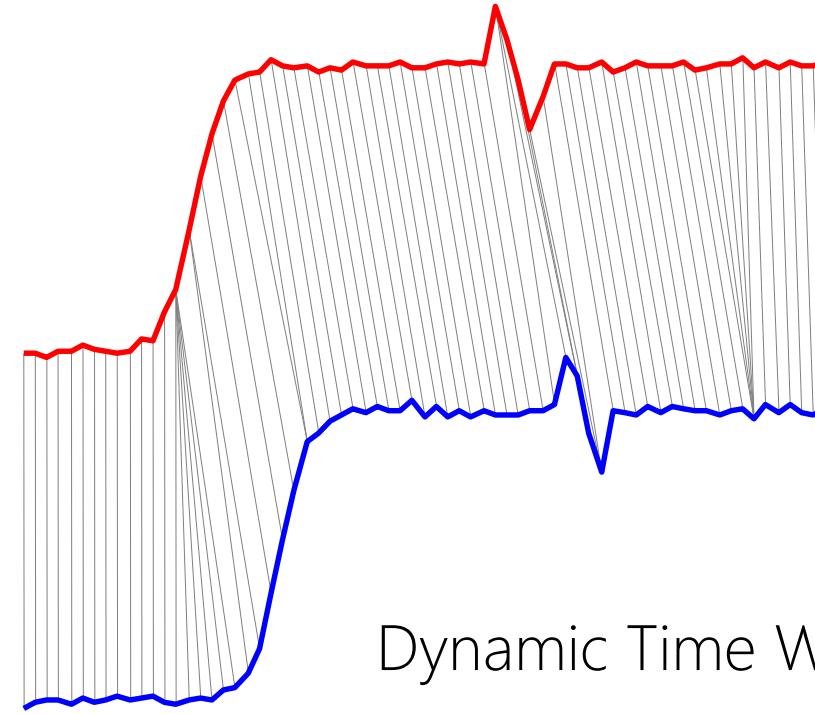
Dynamic Time Warping is a more sophisticated similarity approach; however, it is $O(N^2)$



Euclidean

Fixed Time Axis

Sequences are aligned “one-to-one”



Dynamic Time Warping

“Warped” Time Axis

Non-linear alignments are possible

PREPROCESSING TIME SERIES DATA

Shape-matching algorithms are particularly sensitive to distortions in the data that don't matter but greatly impact performance

Offset translation (e.g., demeaning)

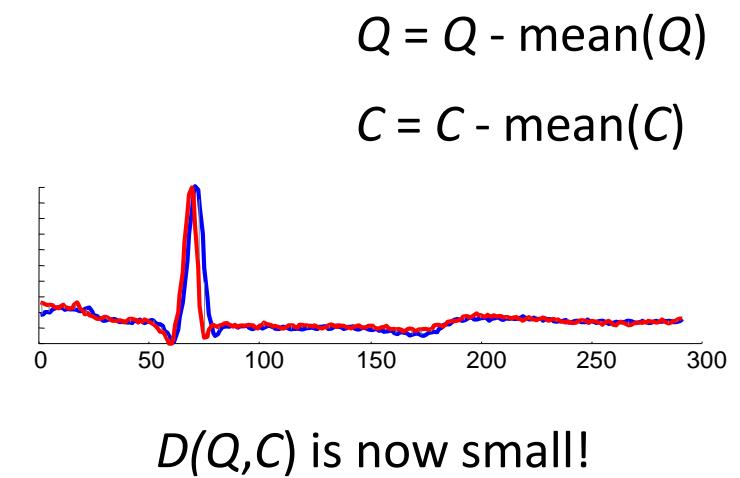
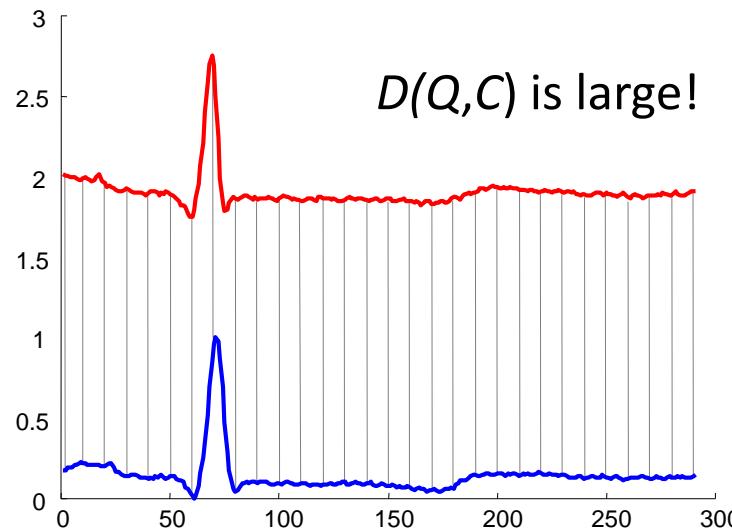
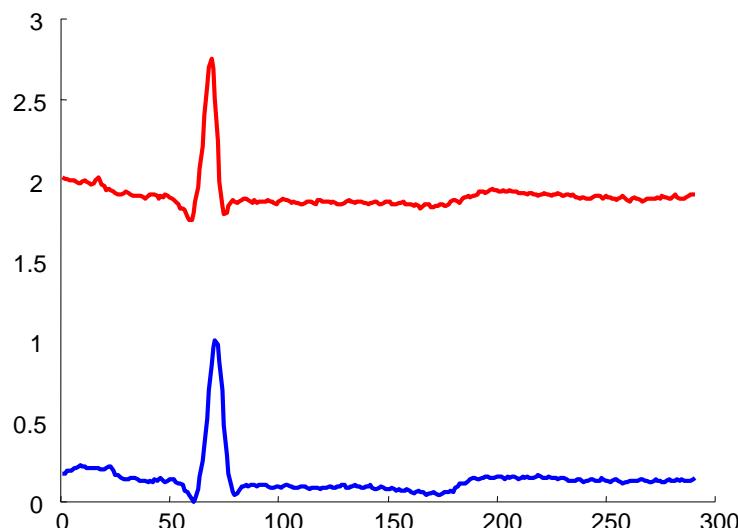
Amplitude scaling

Detrending (e.g., removing linear trend)

Removing noise

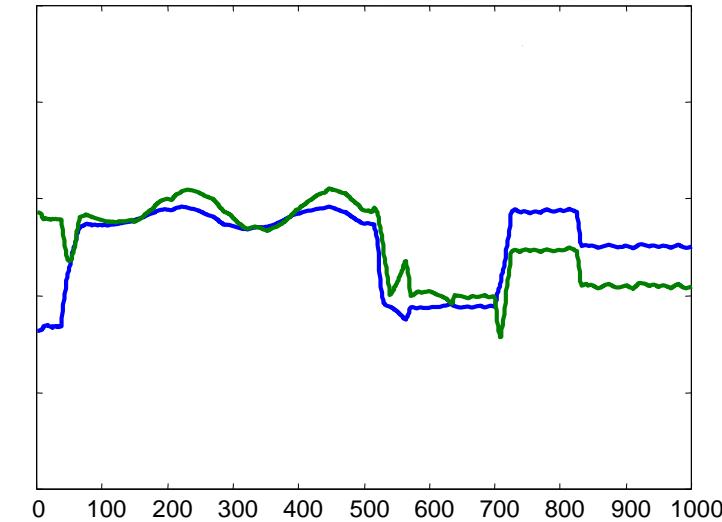
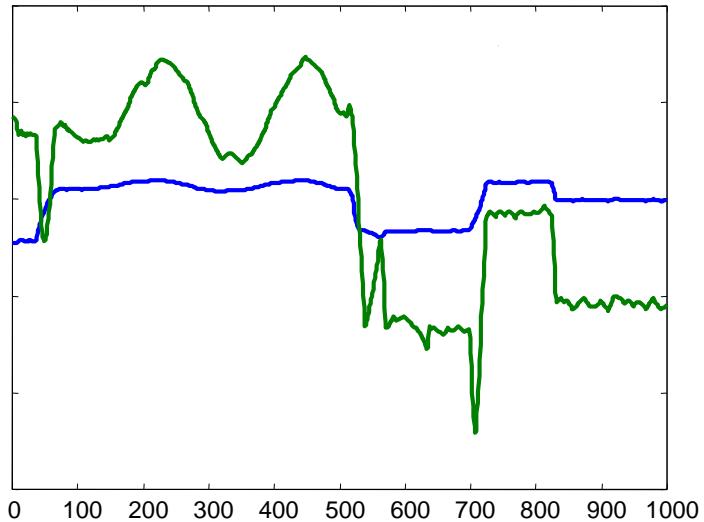
PREPROCESSING TRANSFORMATIONS

OFFSET TRANSLATION



PREPROCESSING TRANSFORMATIONS

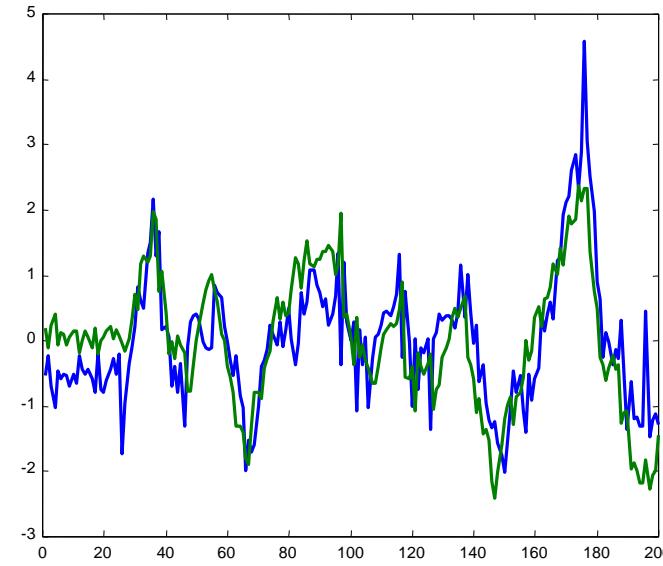
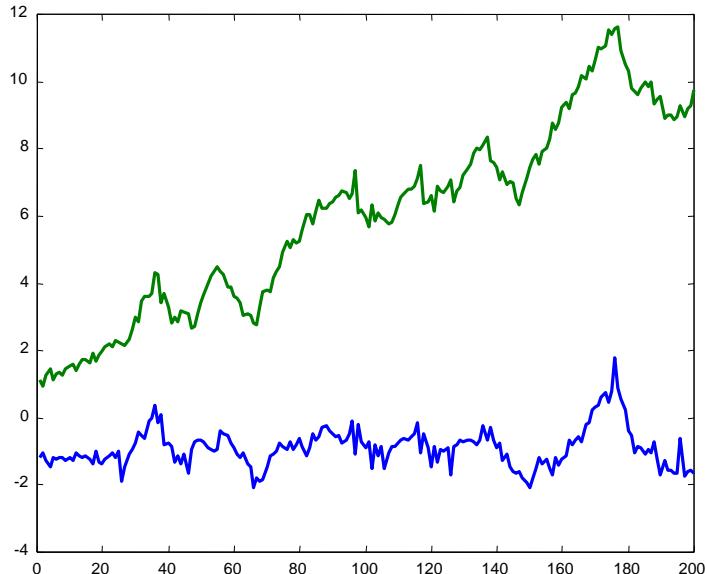
AMPLITUDE SCALING



$$Q = (Q - \text{mean}(Q)) / \text{std}(Q)$$

$$C = (C - \text{mean}(C)) / \text{std}(C)$$

PREPROCESSING TRANSFORMATIONS DETRENDING

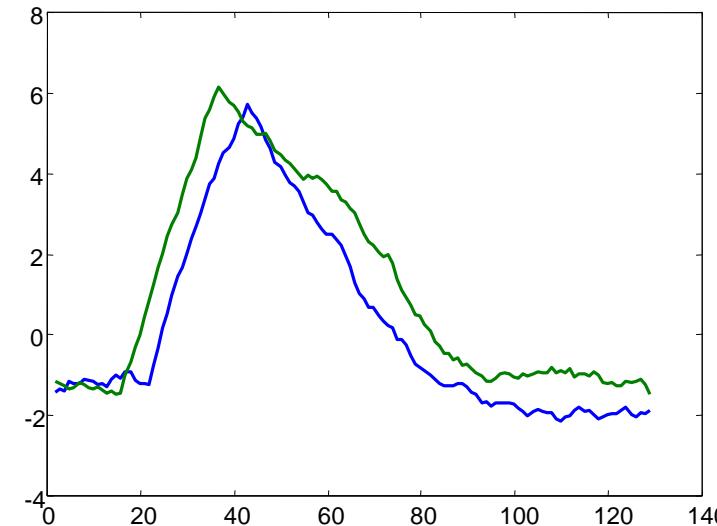
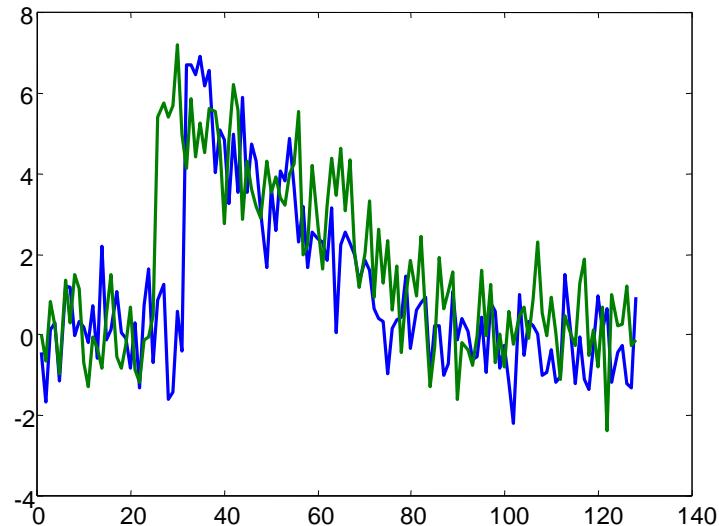


Find best fit line to time series,
then subtract that line from the
signal

$$Q = \text{detrend}(Q)$$
$$C = \text{detrend}(C)$$

PREPROCESSING TRANSFORMATIONS

SMOOTHING



$$Q = \text{smooth}(Q)$$

$$C = \text{smooth}(C)$$



hydroSense

P160-100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

100

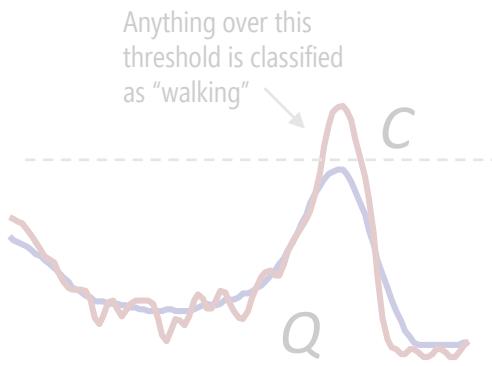
100

100

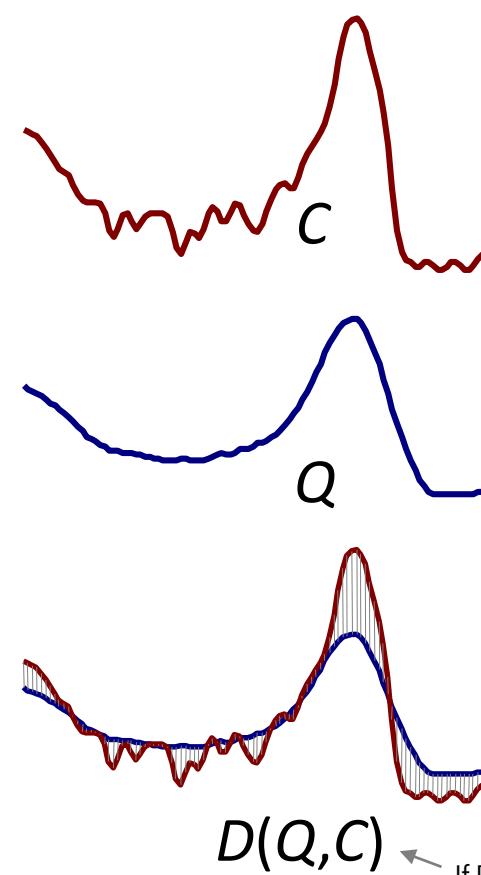
100

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

1. Rule-Based



2. Shape-Matching



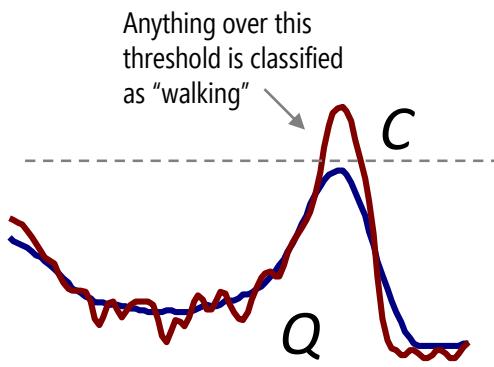
3. Feature-Based



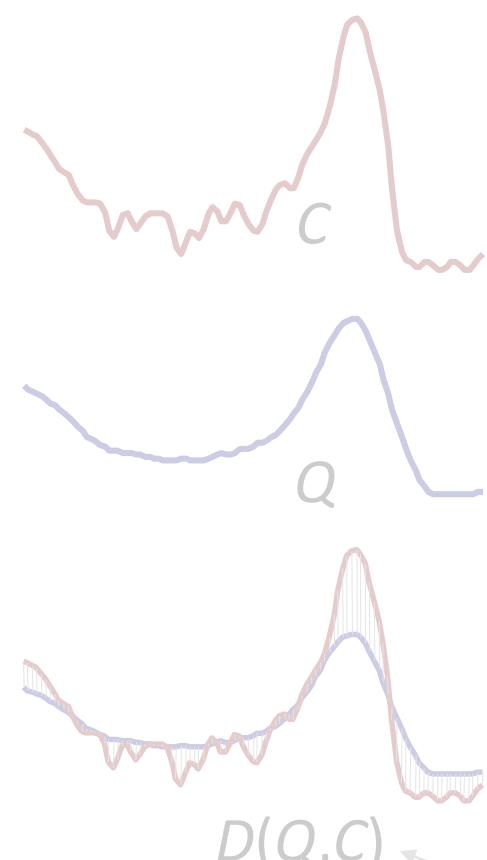
If $D(Q, C) < \text{threshold}$, we consider them the same signal

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

1. Rule-Based



2. Shape-Matching



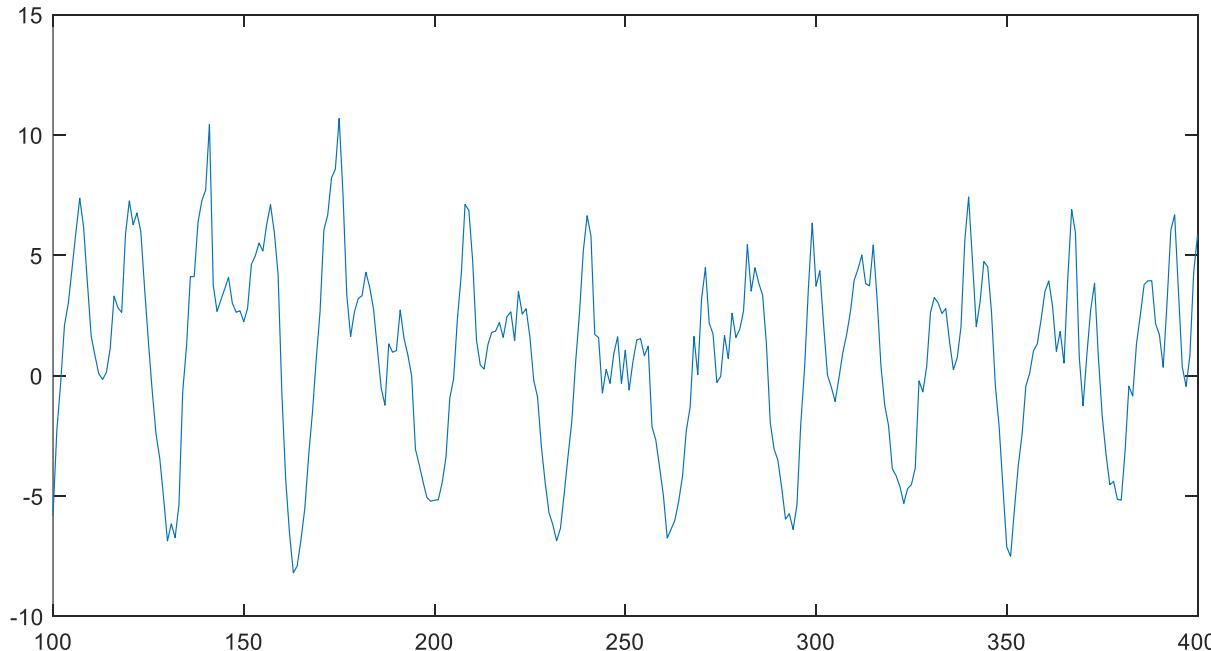
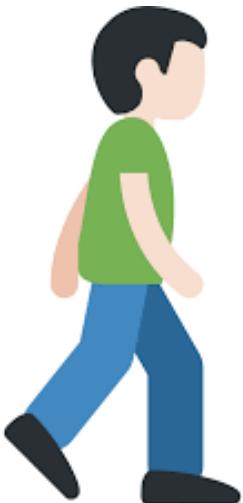
3. Feature-Based



Supervised Learning Model

Output

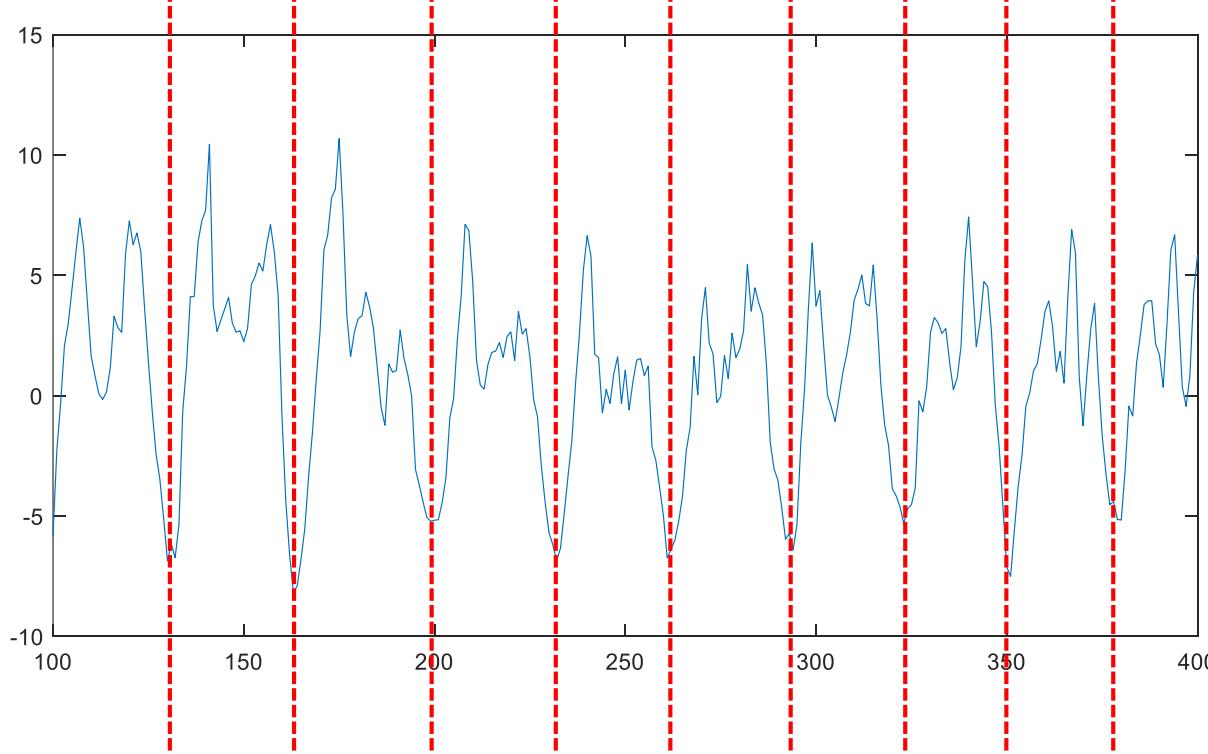
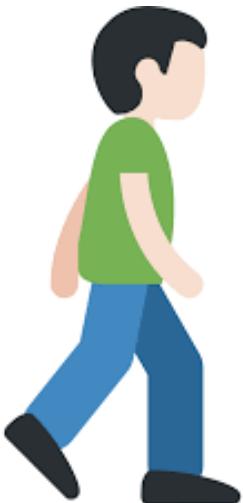
STEP TRACKER ACCELERATION SIGNAL



Think, Pair, Share

Take out a piece of paper and analyze this signal. Brainstorm and write down patterns that you observe. Then share them with a partner and discuss.

STEP TRACKER ACCELERATION SIGNAL

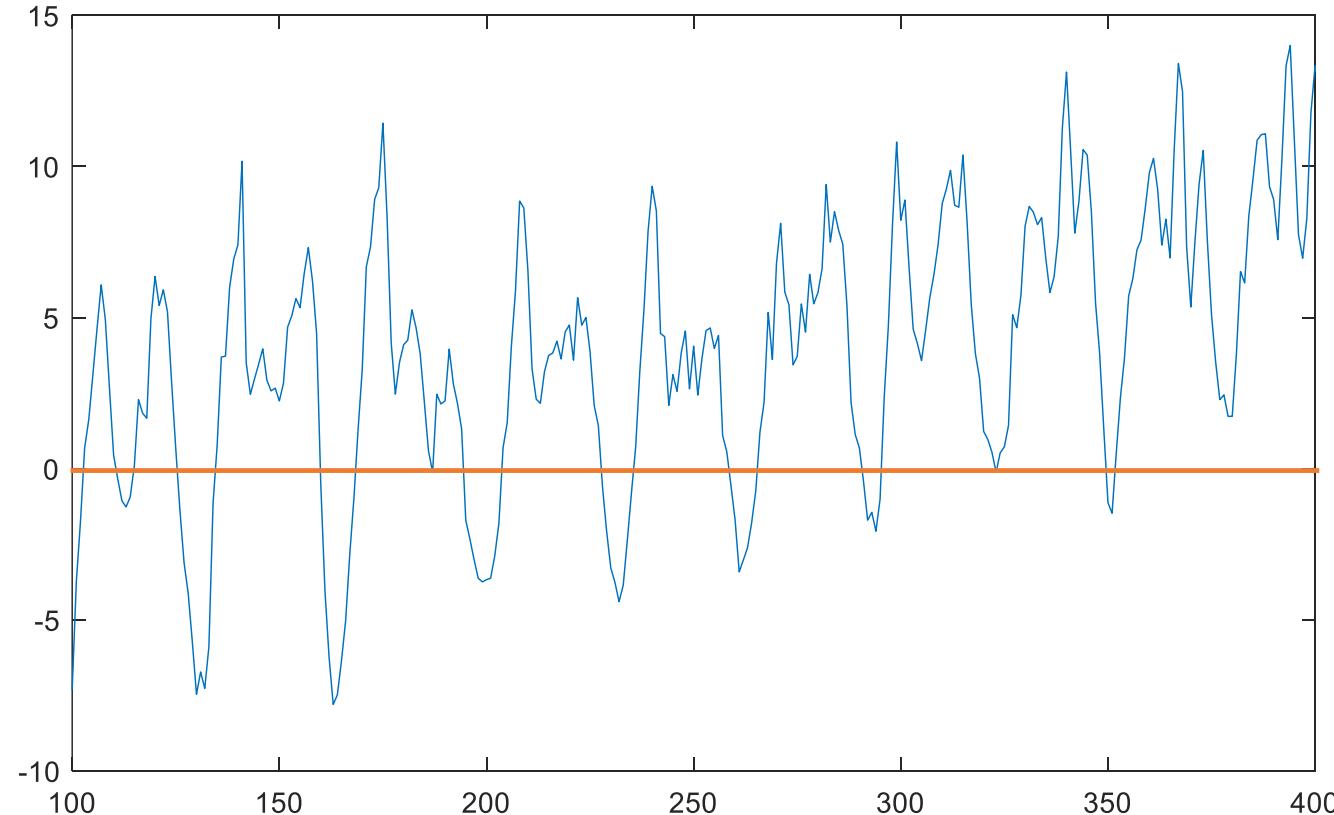
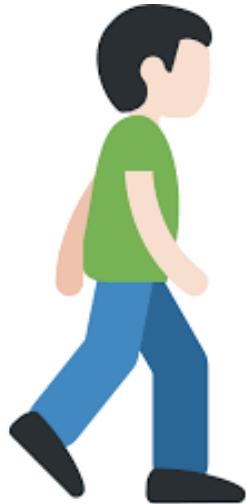


Signal repetition. Each cycle corresponds to a step

Think, Pair, Share

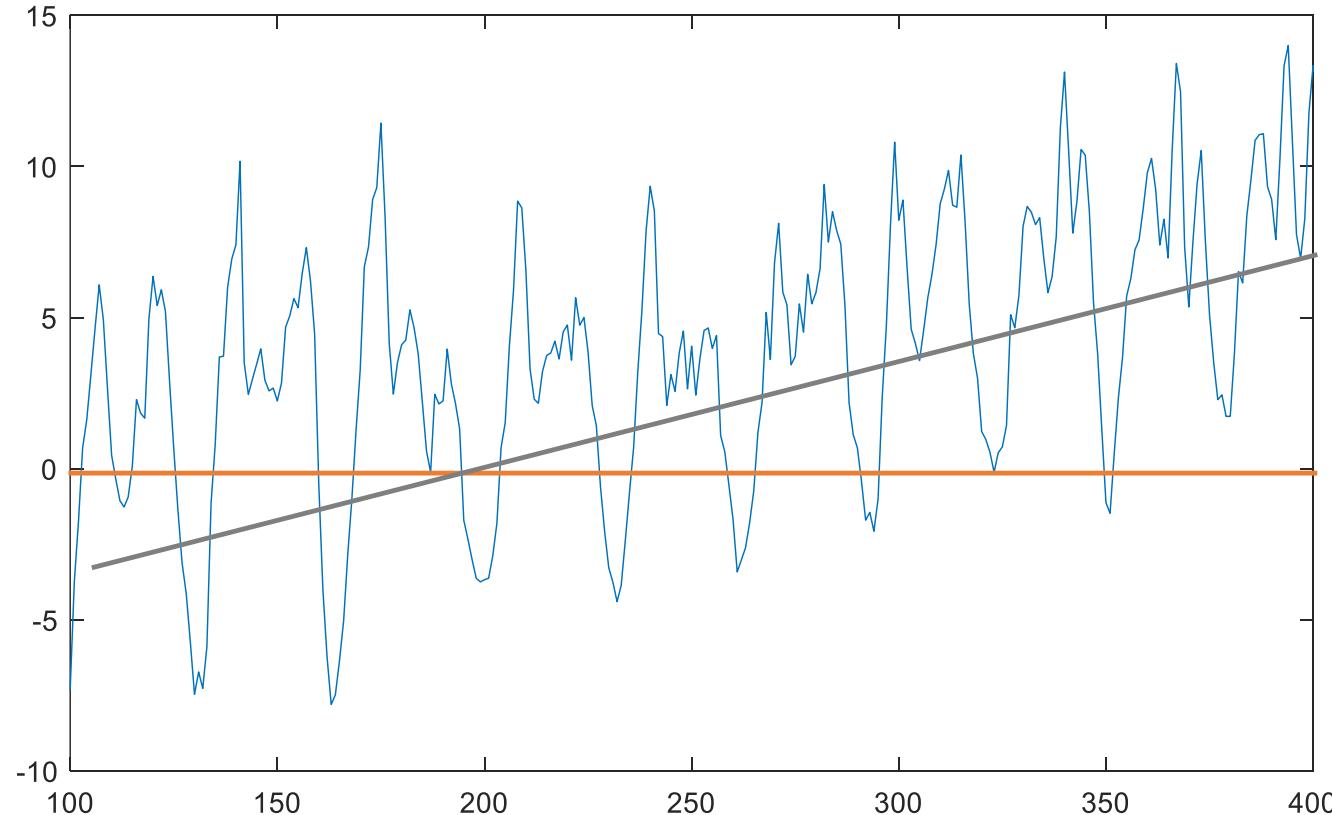
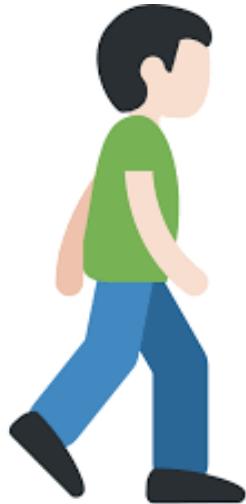
Take out a piece of paper and analyze this signal. Brainstorm and write down patterns that you observe. Then share them with a partner and discuss.

ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?



What's a potential problem with this?
Drift. The signal trends upwards.

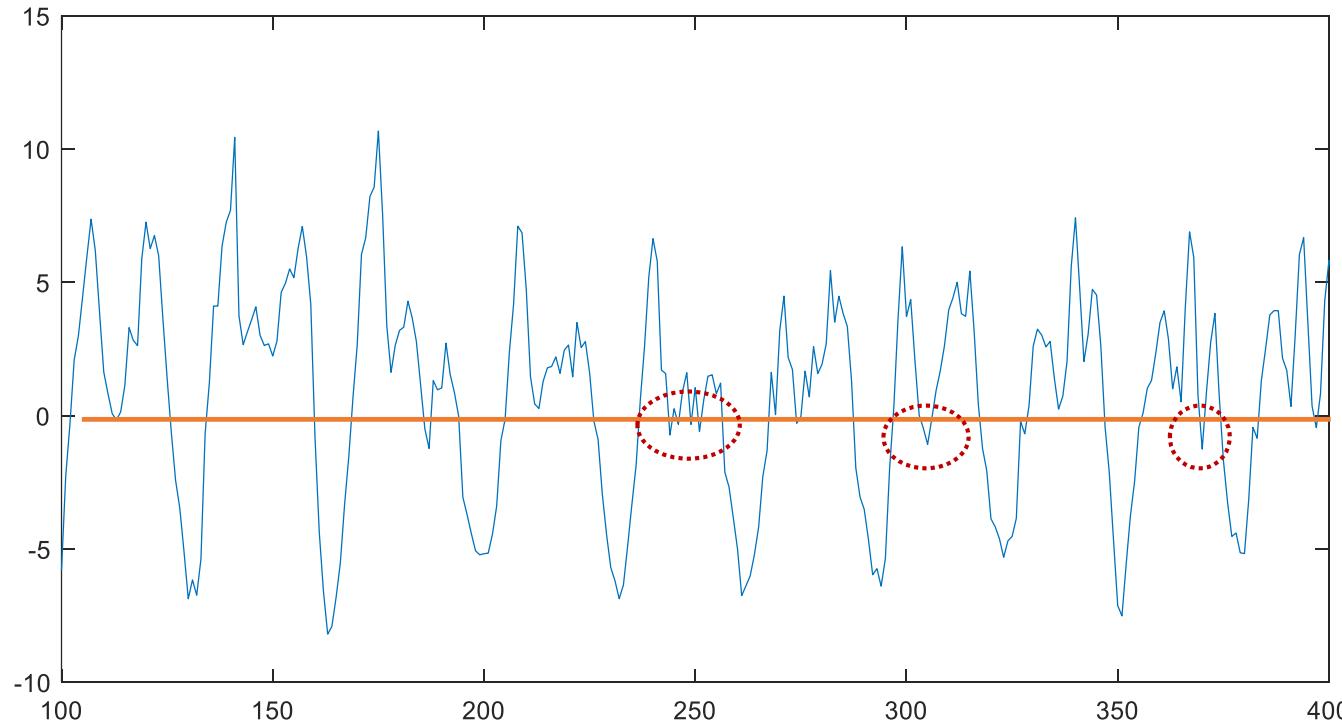
ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?



What's a potential problem with this?
Drift. The signal trends upwards.

Solution
Detrend the data. Subtract the mean or a best-fit line (like a regression line) from the data.

ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?

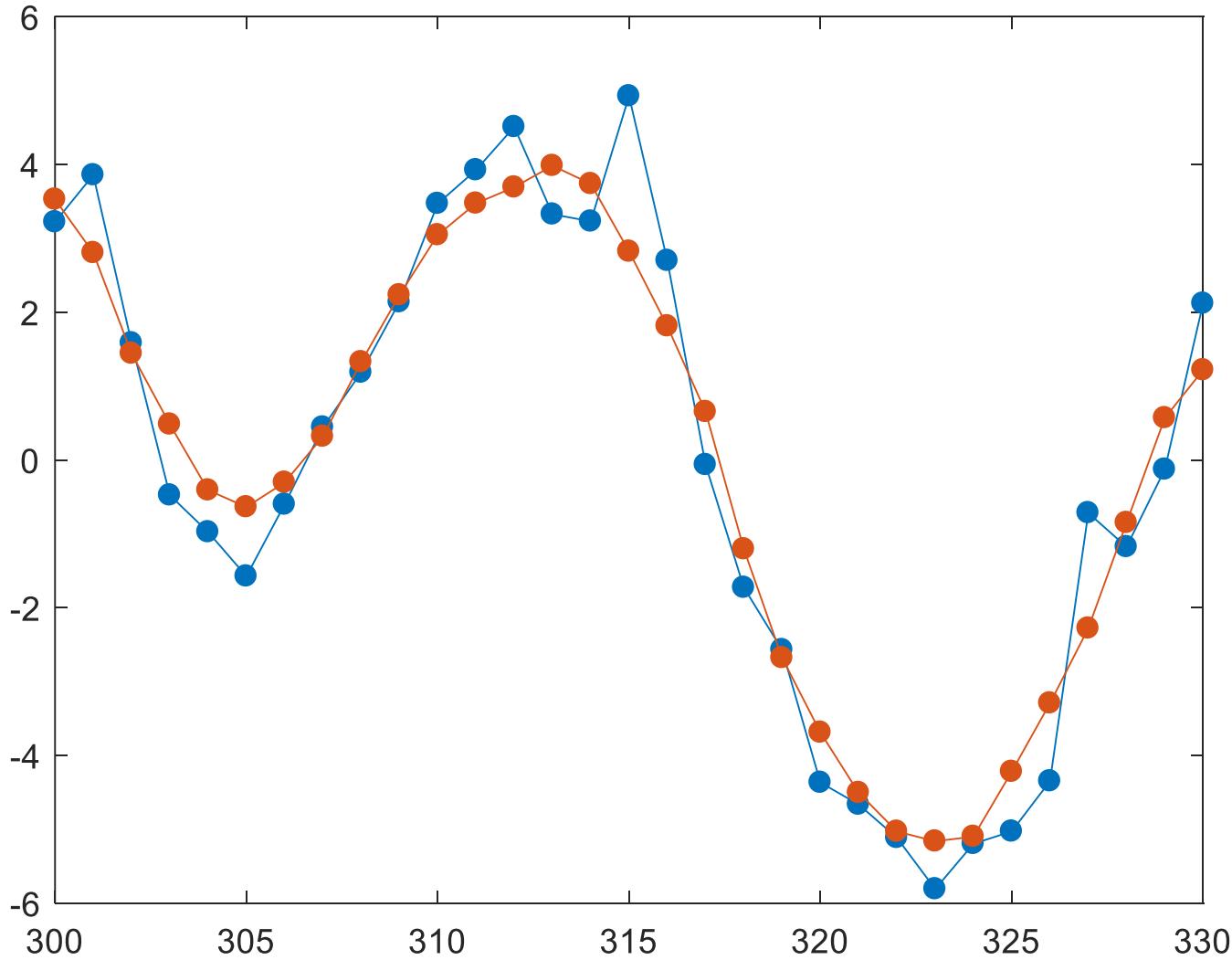


OK, this is better, but...
There are false positives.

Solution
Smooth the signal. The simplest is a moving average filter (which also serves as a cheap low-pass filter, that is it removes high frequency noise in your signal)

STEP TRACKER EXERCISE

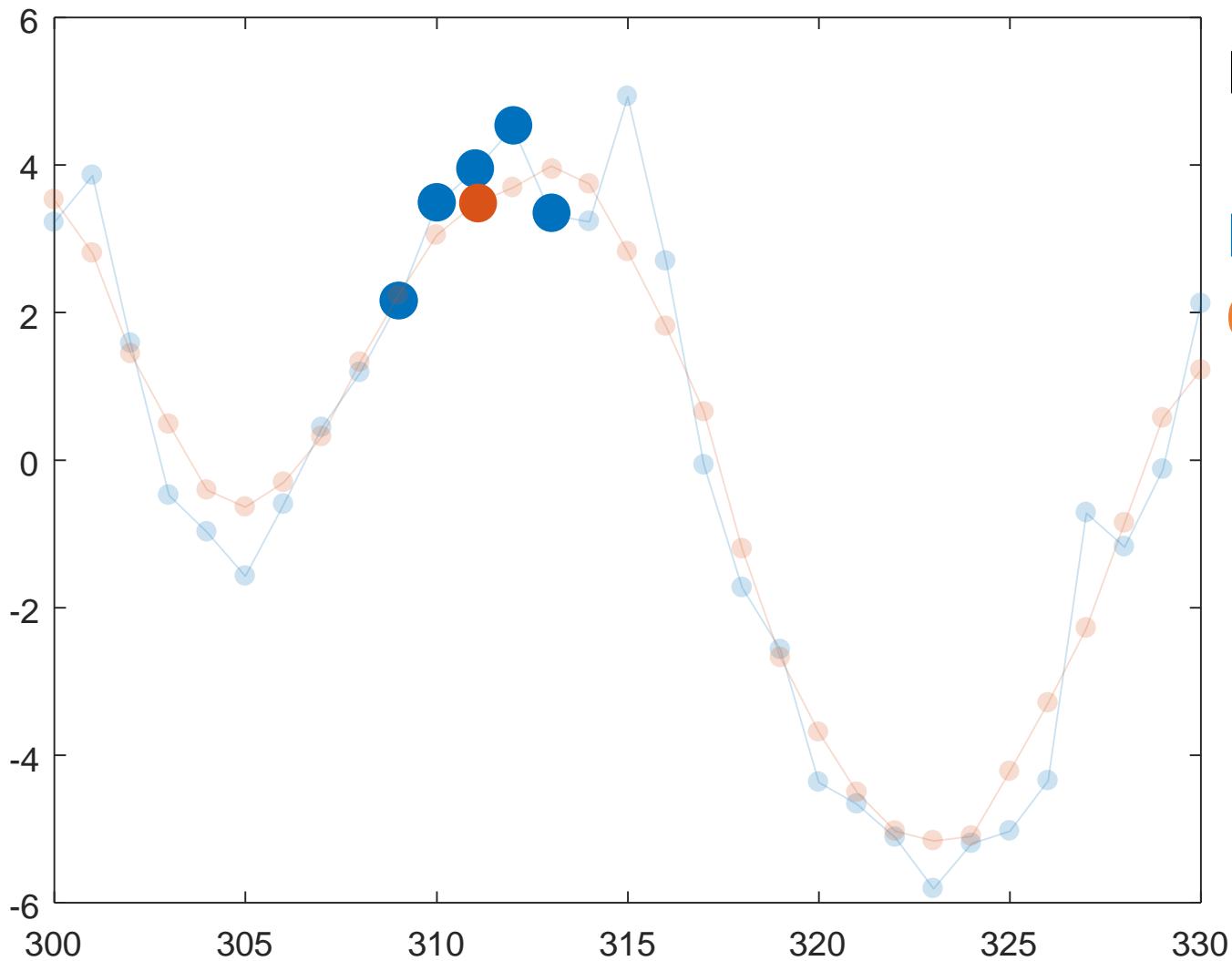
MOVING AVERAGE FILTER



Noisy Signal
Filtered Signal

STEP TRACKER EXERCISE

MOVING AVERAGE FILTER

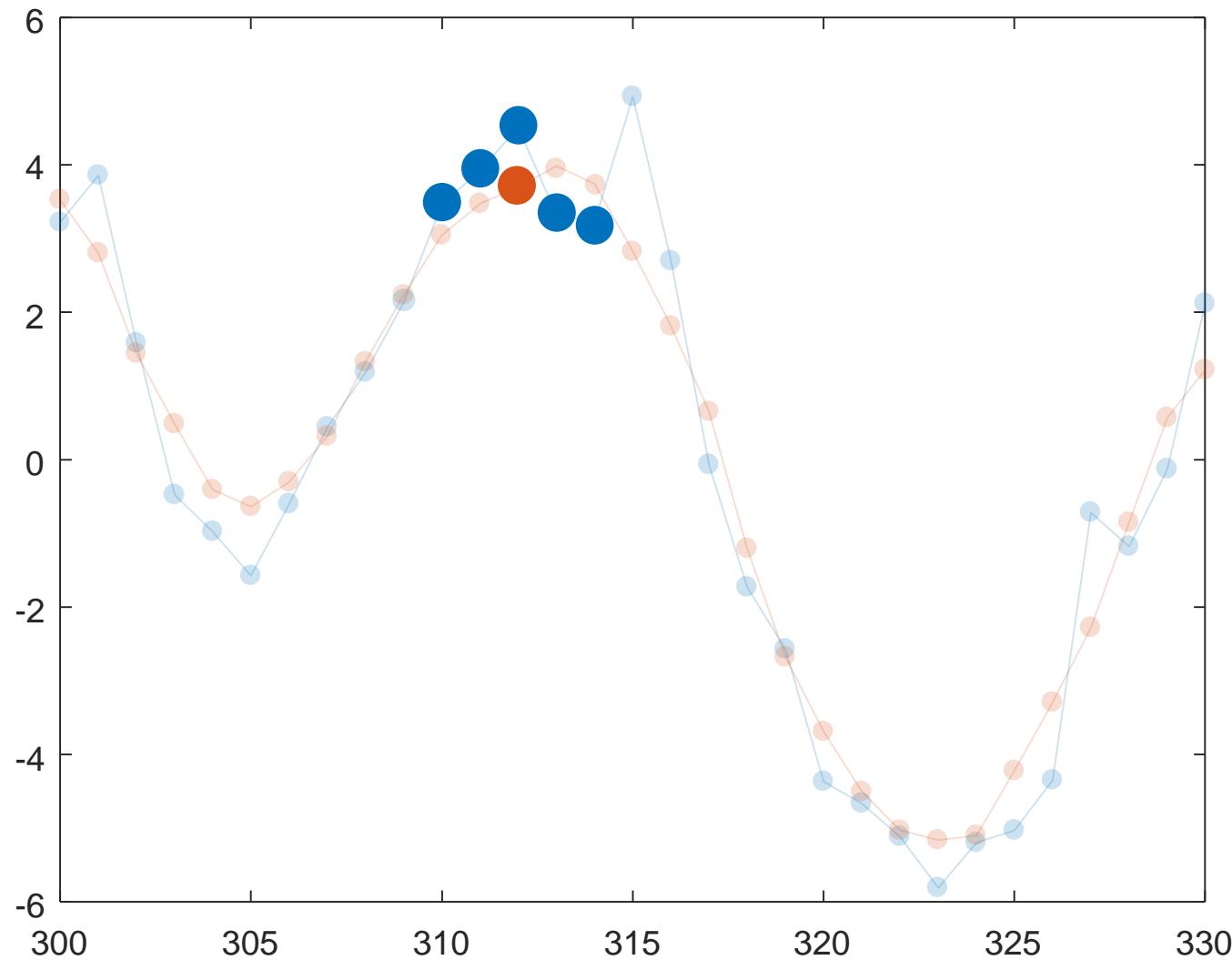


Filter Length [N] = 5

Input = [2.4,4,4.5,5,3.5]

Output = $(2.4+4+4.5+5+3.5)/5 = 3.9$

MOVING AVERAGE FILTER



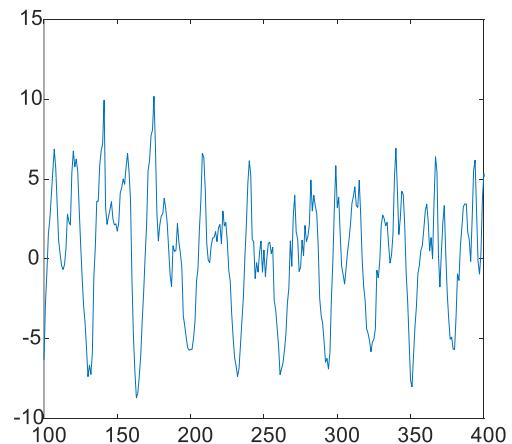
Filter Length [N] = 5

Input = [4,4.2,5,3.5,3.2]

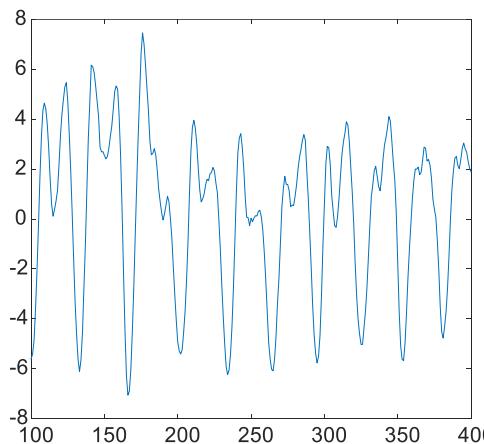
Output = $(4+4.2+5+3.5+3.2)/5 = 4.0$

STEP TRACKER EXERCISE

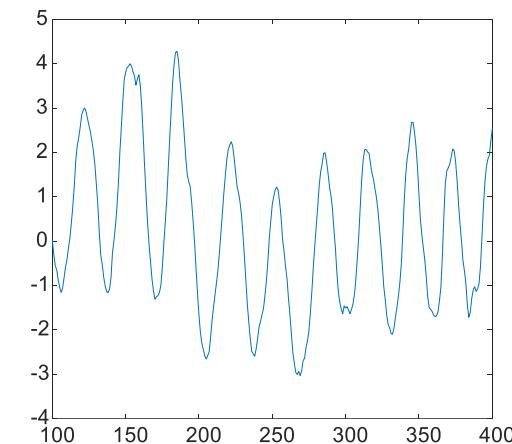
MOVING AVERAGE FILTER: DIFFERENT WINDOW SIZES



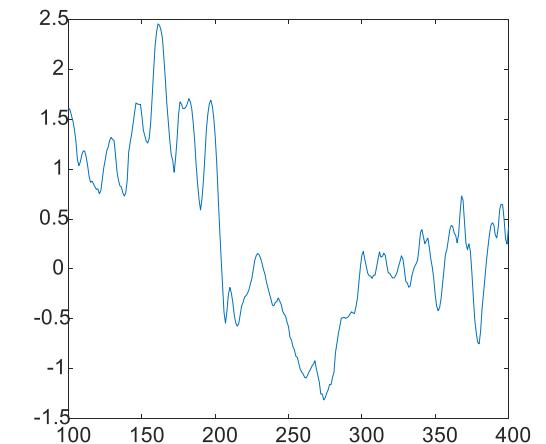
$N = 1$



$N = 5$

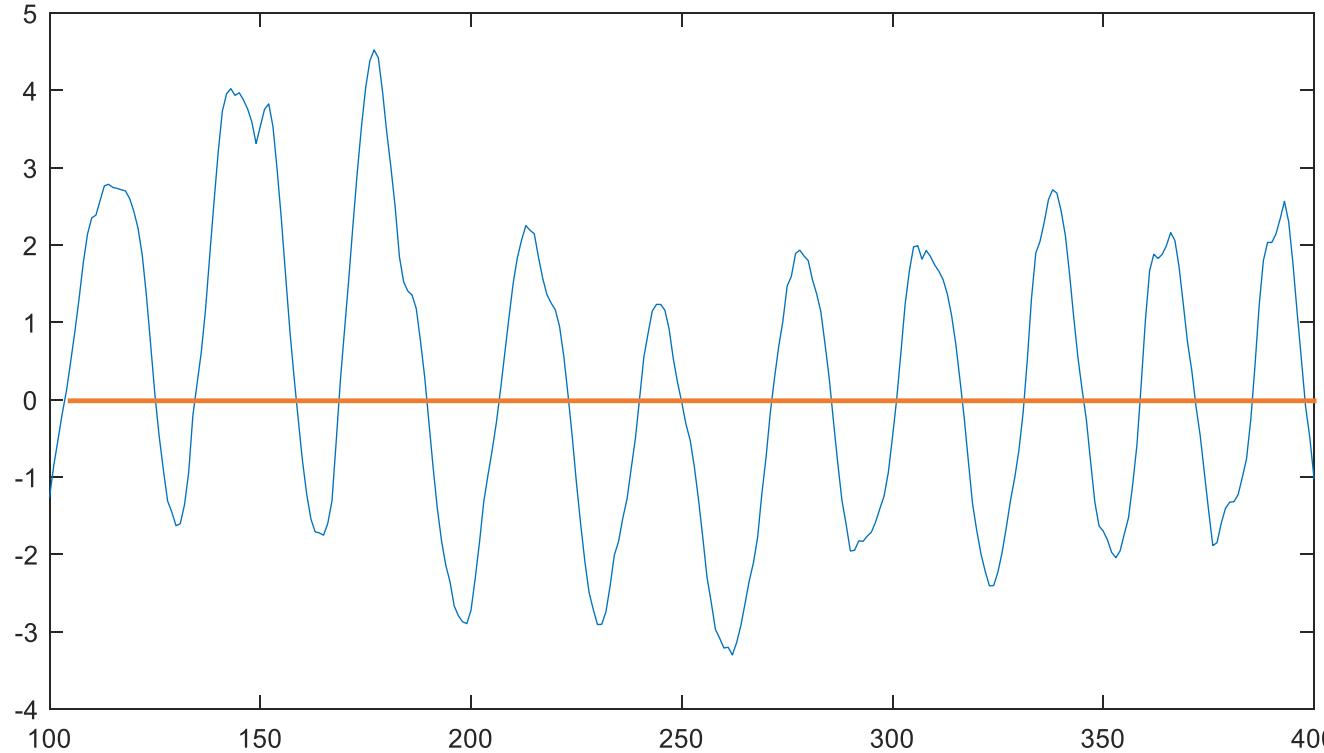


$N = 15$



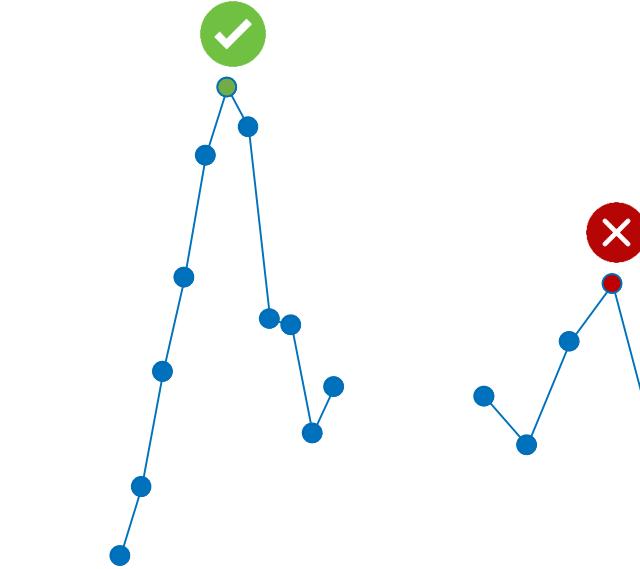
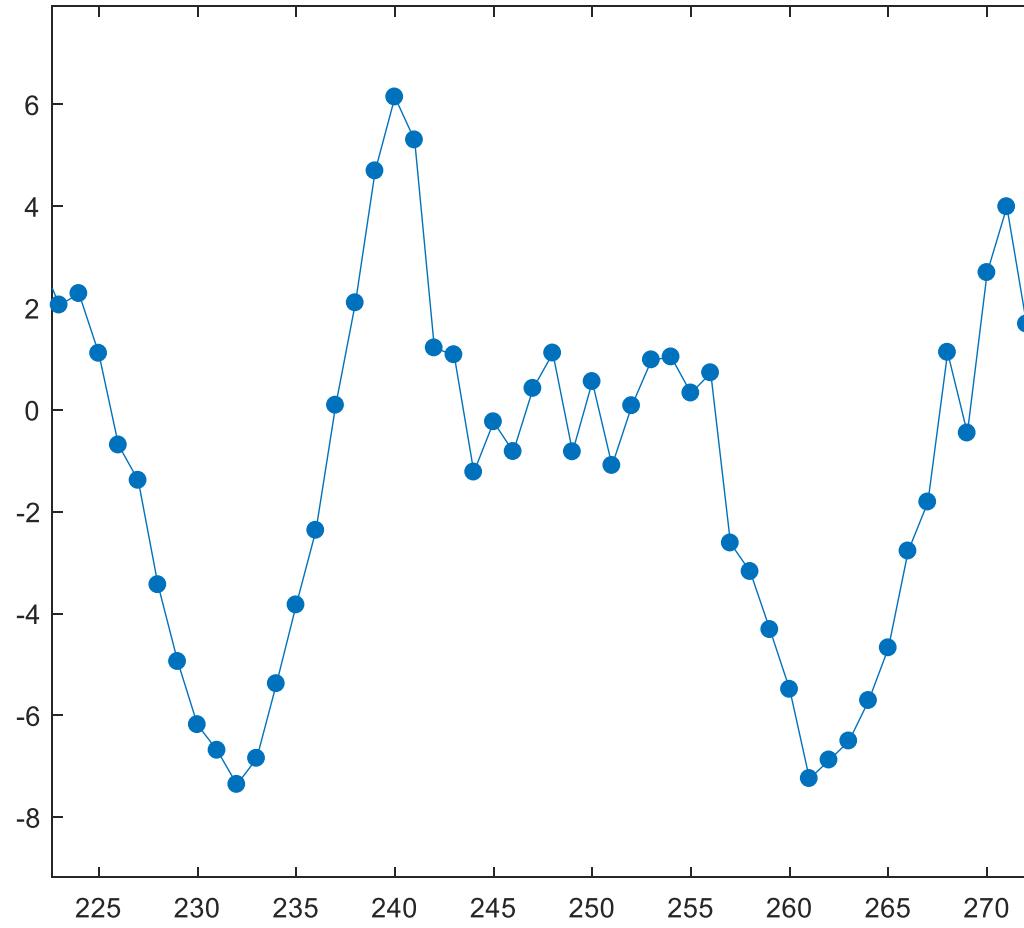
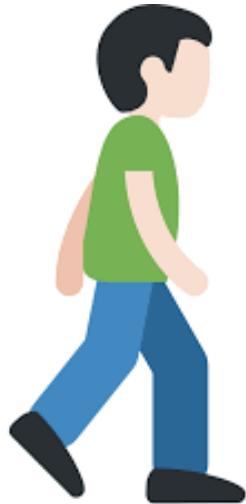
$N = 30$

SMOOTHED SIGNAL: HOW TO DETECT STEPS?



Smoothed signal.

ONE WAY TO COUNT STEPS: DETECT PEAKS?

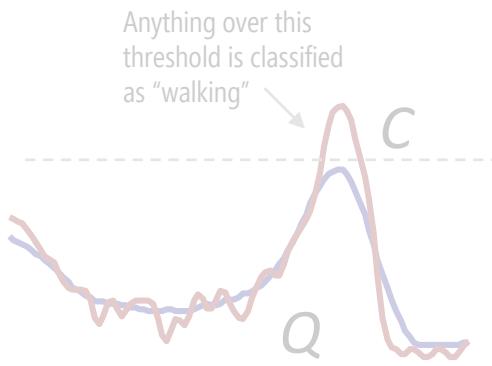


Final Think, Pair, Share

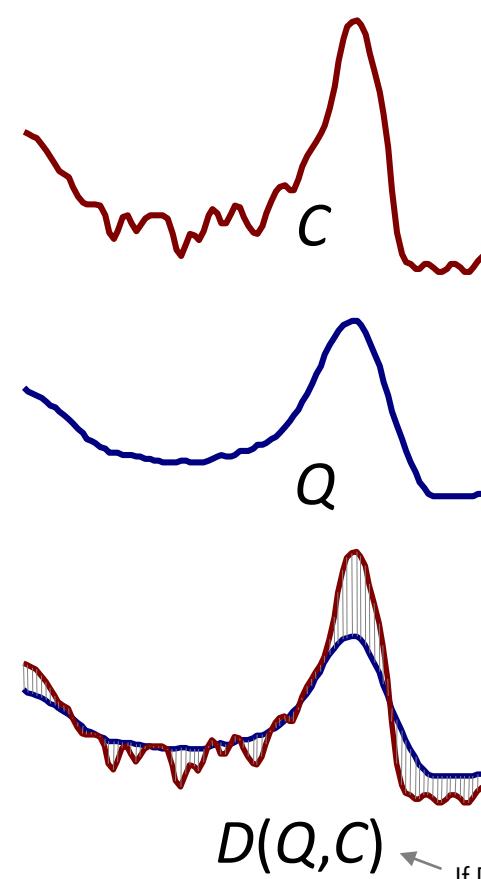
Take out a piece of paper and try to come up with a peak finding algorithm. I'll announce when it's time to share and discuss with a partner.

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

1. Rule-Based



2. Shape-Matching



3. Feature-Based



If $D(Q, C) < \text{threshold}$, we consider them the same signal

Spring 2019

Home

Announcements

Assignments

Discussions

Grades

People

Pages

Files

Syllabus

Outcomes

Quizzes

Modules

Conferences

Collaborations

Chat

Attendance

UW Libraries

Add 4.0 Grade Scale

Panopto

Recordings

A3: Signal Processing + Machine Learning 1: Shape-Matching Gesture Recognizer

Published**Edit**

:

Related Items

SpeedGrader™

Overview

Imagine working for a new hardware startup designing new input controllers. You've been asked to prototype a new, custom input device with gesture recognition (using an accelerometer)--for example, to use the input controller as a paddle in tennis, as a "ball" in bowling, or to recognize the "overhand throwing motion" in baseball. In this assignment, you will build your own 3D gesture recognizer to automatically recognize these gestures.

While in the "real world" you would ultimately need to create a real-time gesture recognizer, for this assignment you will make an *offline* version in [Jupyter Notebook](#) (we strongly recommend the Anaconda Distribution). Specifically, in this assignment you will build a *shape-matching* (or *template-matching*) recognizer such as via a Euclidean distance metric or Dynamic Time Warping. In A4, you will build a *feature-based* (or *model-based*) recognizer using a [support-vector machine \(SVM\)](#) (recommended) or an alternative supervised learning approach of your choosing (e.g., an HMM).

Within Jupyter Notebook, we will use Python 3 and these amazing libraries [numpy](#), [scipy](#), [matplotlib](#), and [scikit-learn](#). Numpy and scipy provide numeric array handling and signal processing, matplotlib provides visualization, and scikit-learn is the de facto machine learning library in Python. You are welcome to use other libraries as well (e.g., [this DTW library](#)).

For your deliverables, you will turn in your Jupyter Notebook, your recorded gestures, and a slide deck report on your algorithmic approaches and performance results.

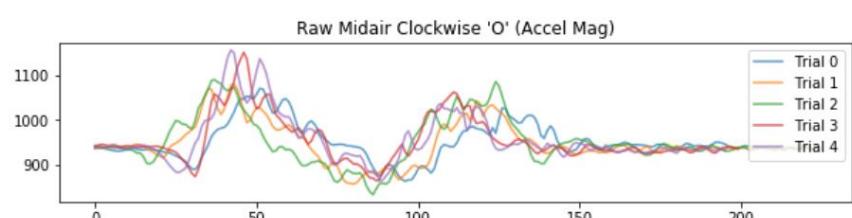
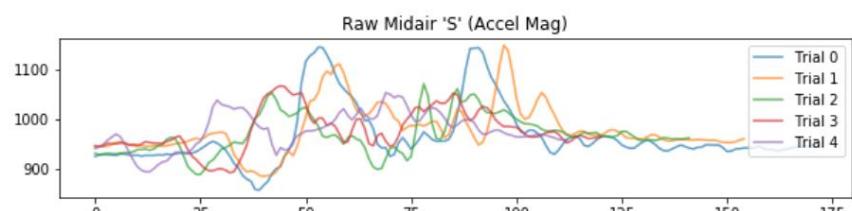
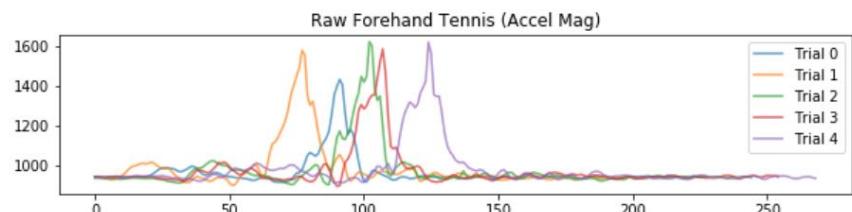
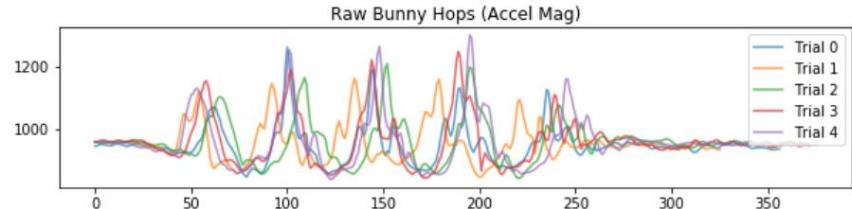
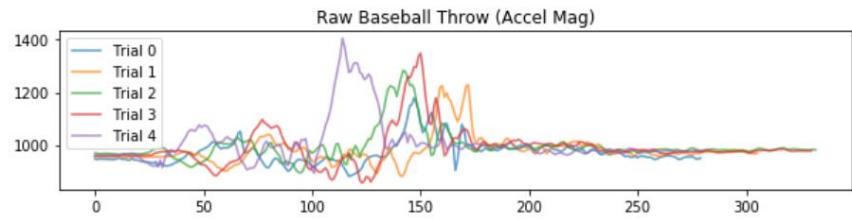
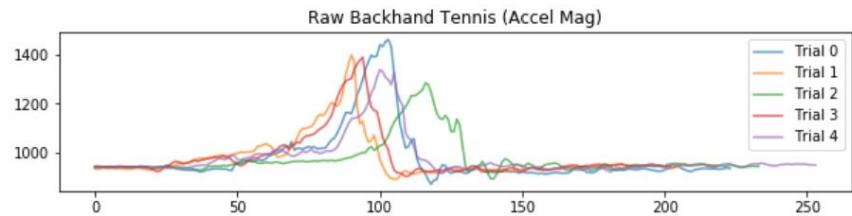
Things You Need

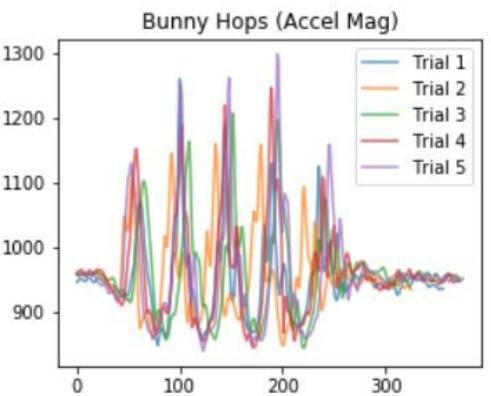
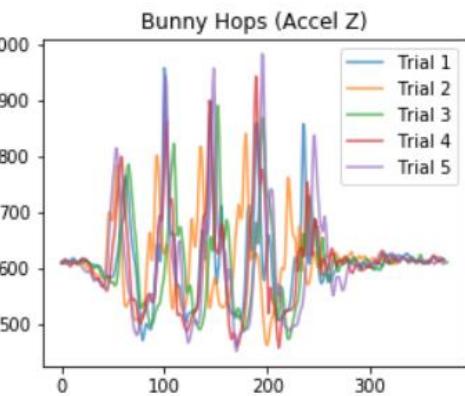
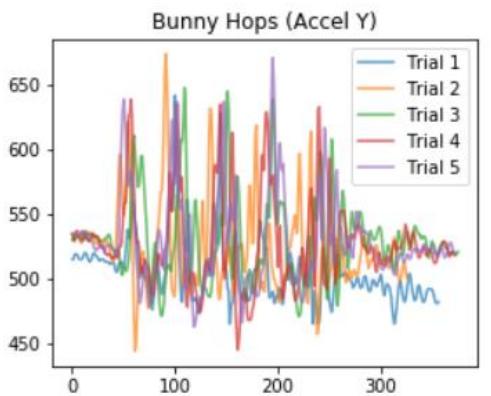
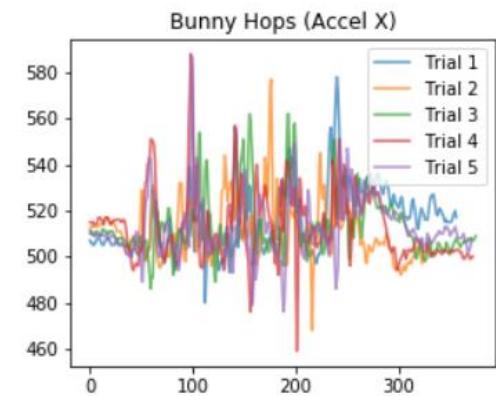
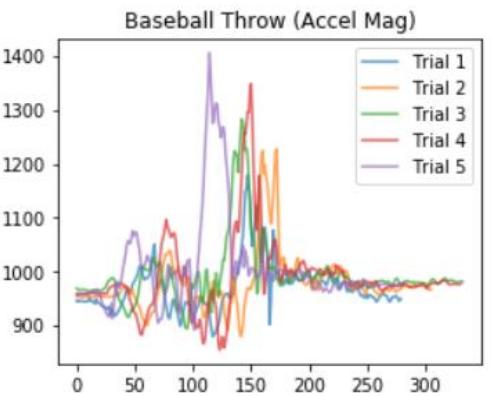
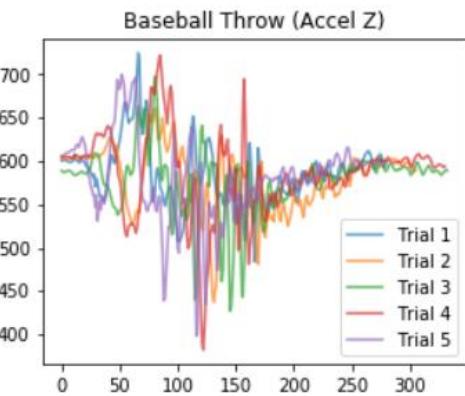
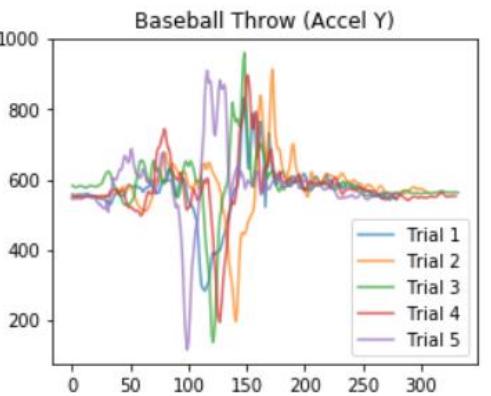
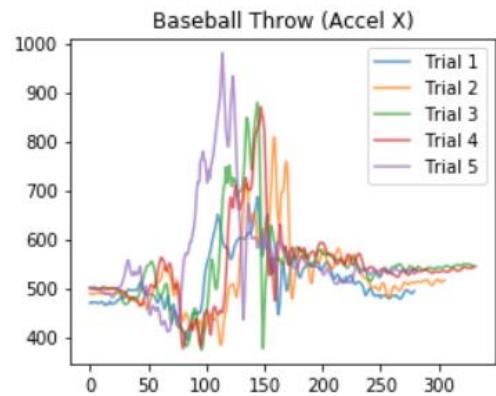
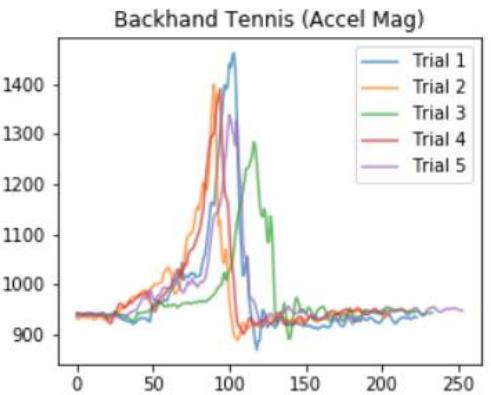
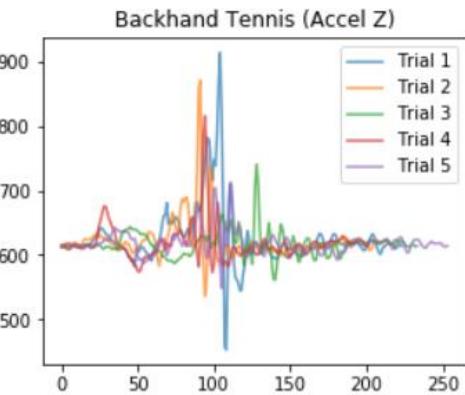
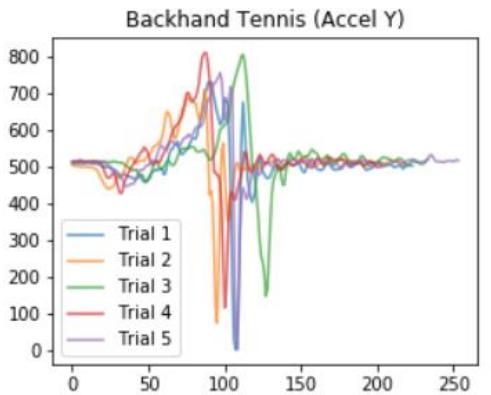
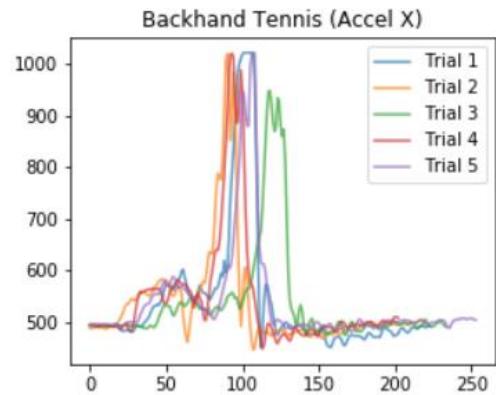
- Download and install [Jupyter Notebook](#) for offline data analysis, signal processing, and machine learning. We've created an initial skeleton to parse the data, [which is available here](#).

A3: SHAPE-BASED GESTURE RECOGNITION

GENERAL APPROACH

1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms

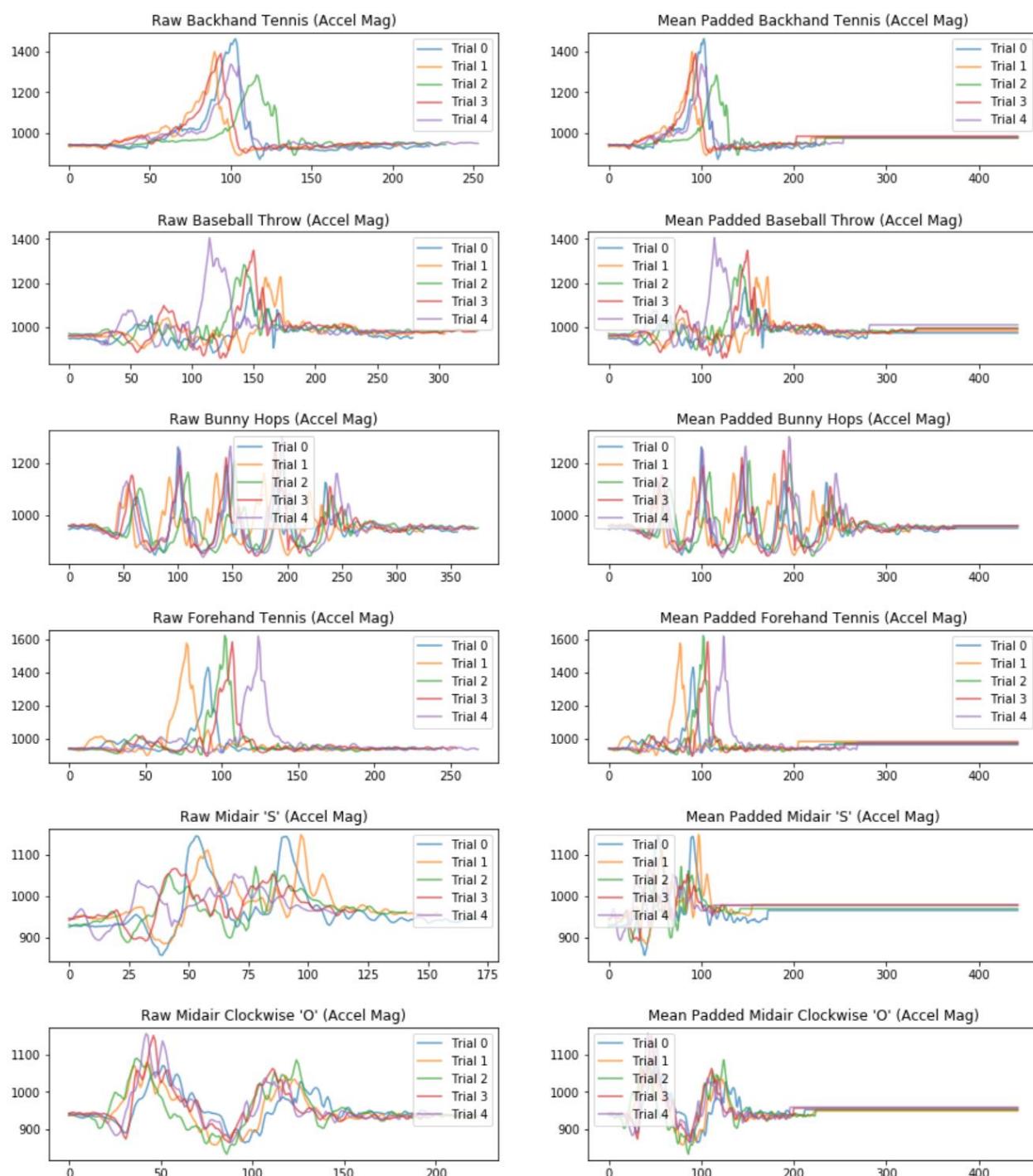




A3: SHAPE-BASED GESTURE RECOGNITION

GENERAL APPROACH

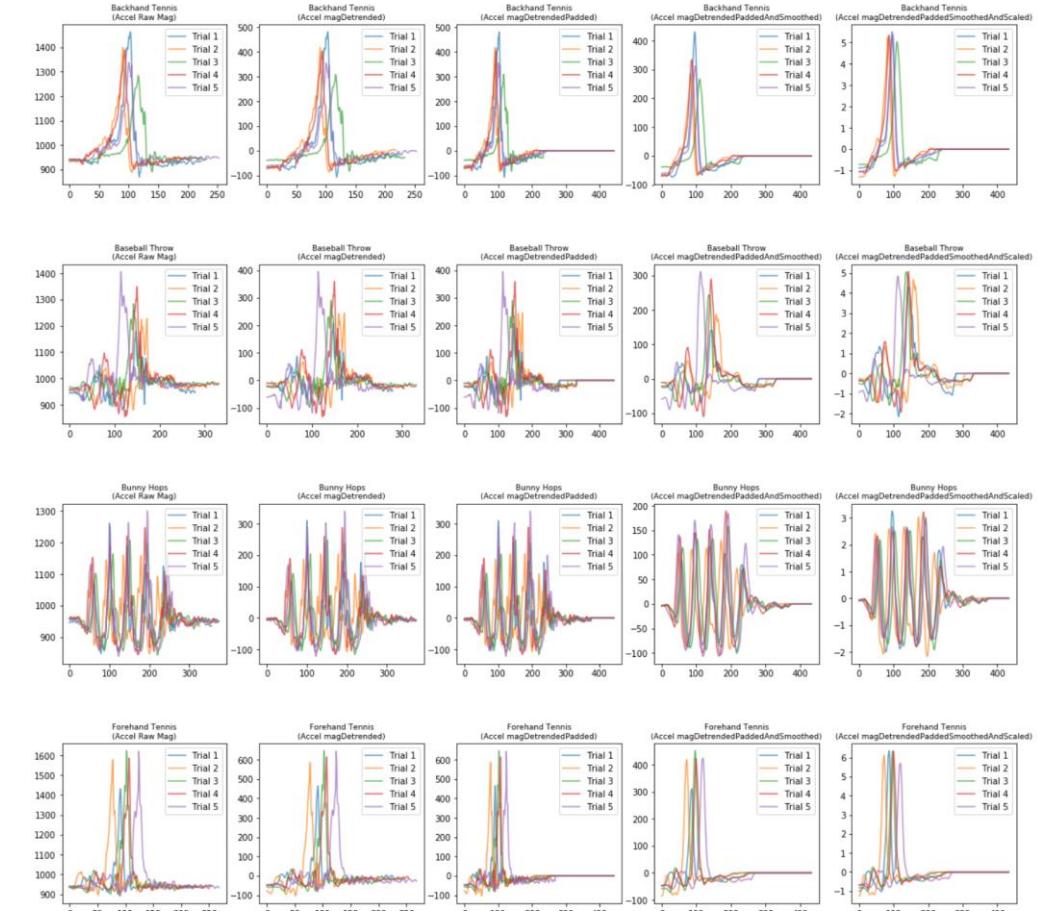
1. **Visualize the data.** Look for patterns.
Brainstorm how to leverage those patterns in your algorithms
2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.

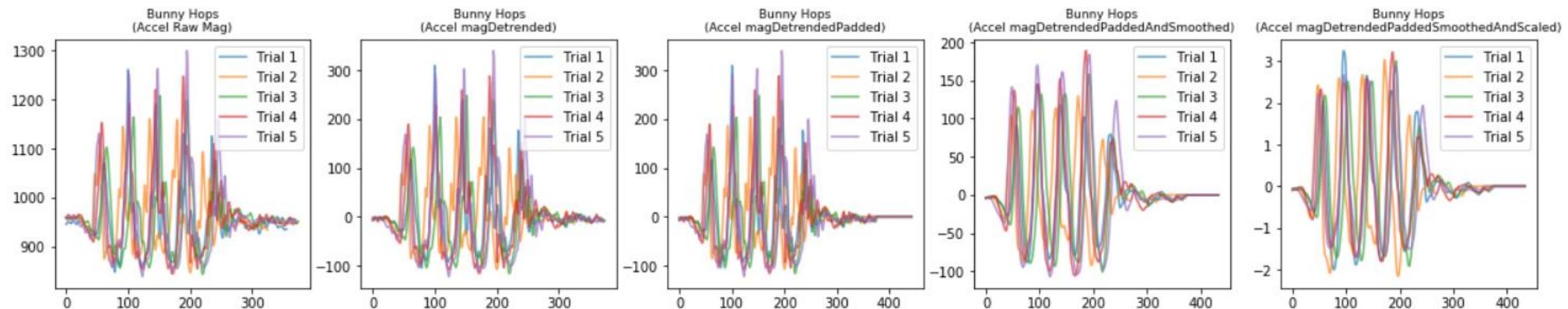
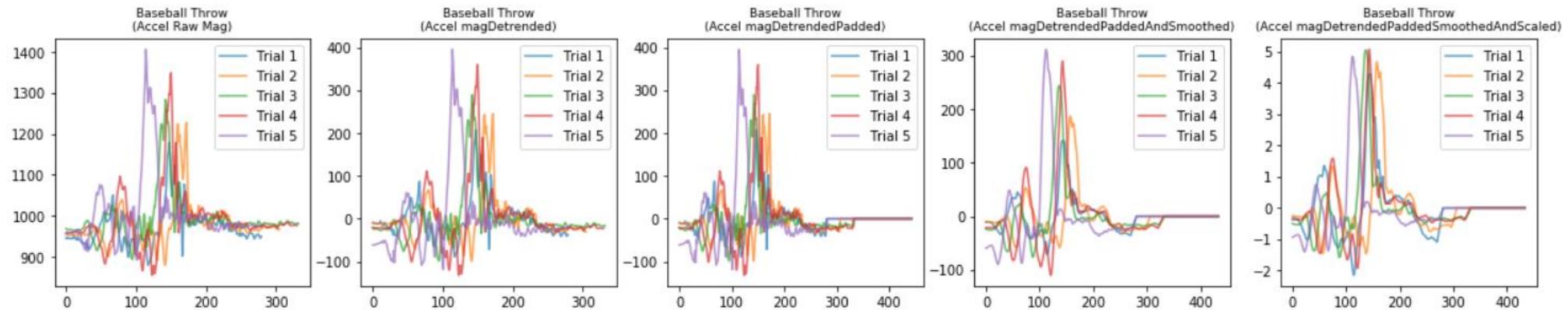
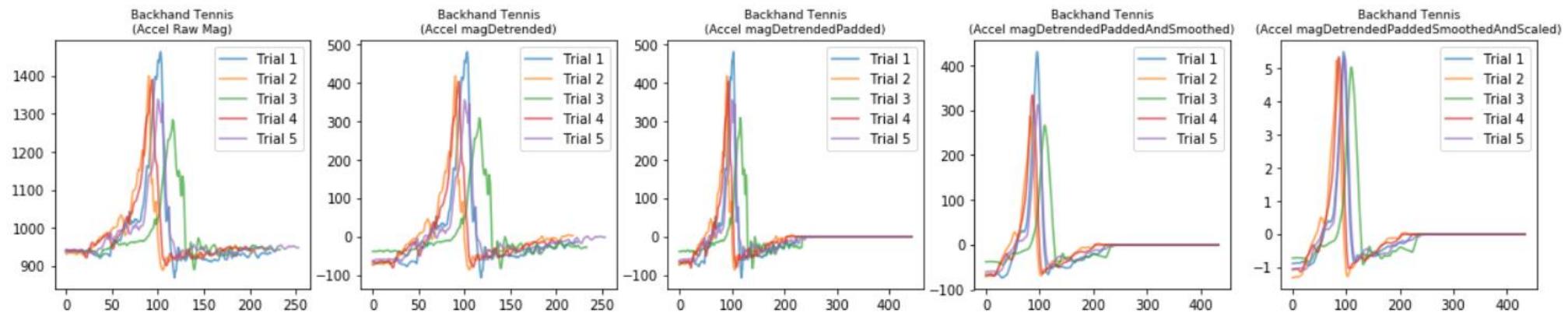


A3: SHAPE-BASED GESTURE RECOGNITION

GENERAL APPROACH

1. **Visualize the data.** Look for patterns.
Brainstorm how to leverage those patterns in your algorithms
2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.
3. **Play around with various transforms of the data.** Smoothed. Detrended. Scaled?

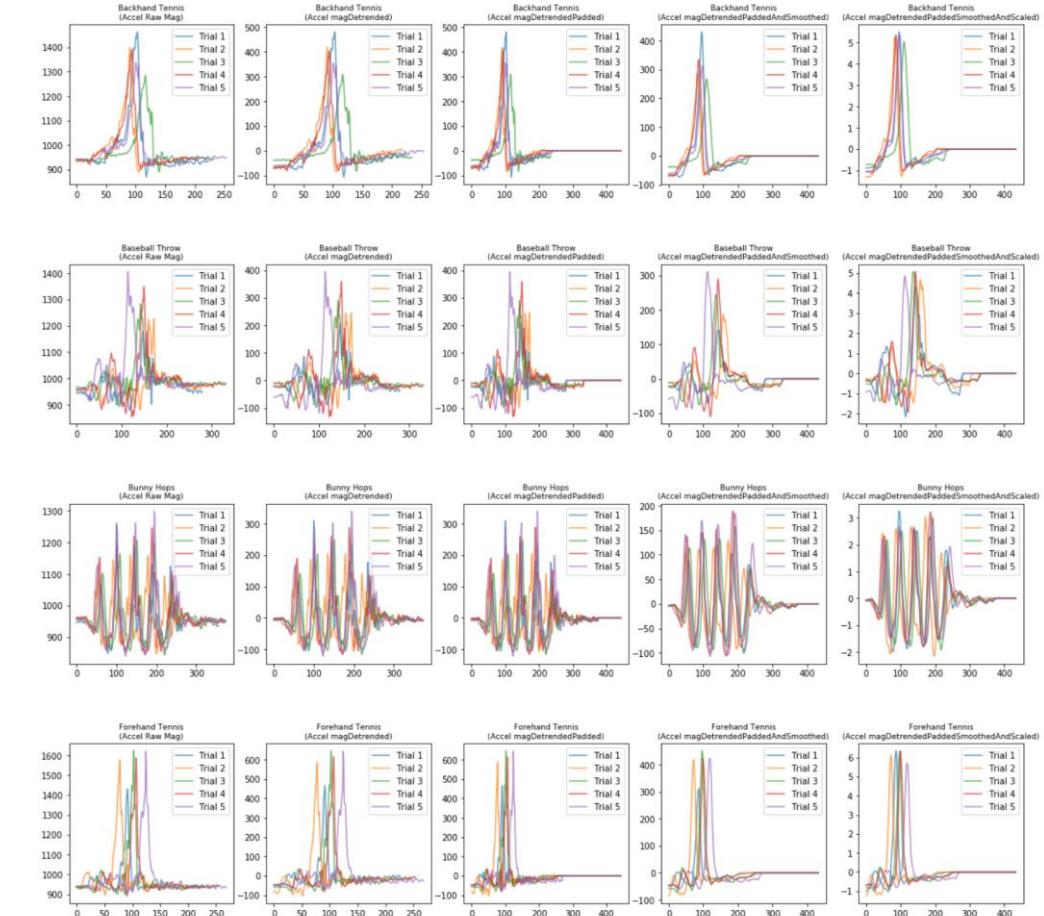




A3: SHAPE-BASED GESTURE RECOGNITION

GENERAL APPROACH

1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms
2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.
3. **Play around with various transforms of the data.** Smoothed. Detrended. Scaled?
4. **Determine strategy/algorithm for comparing two signals.** Euclidean distance? Dynamic time warping?

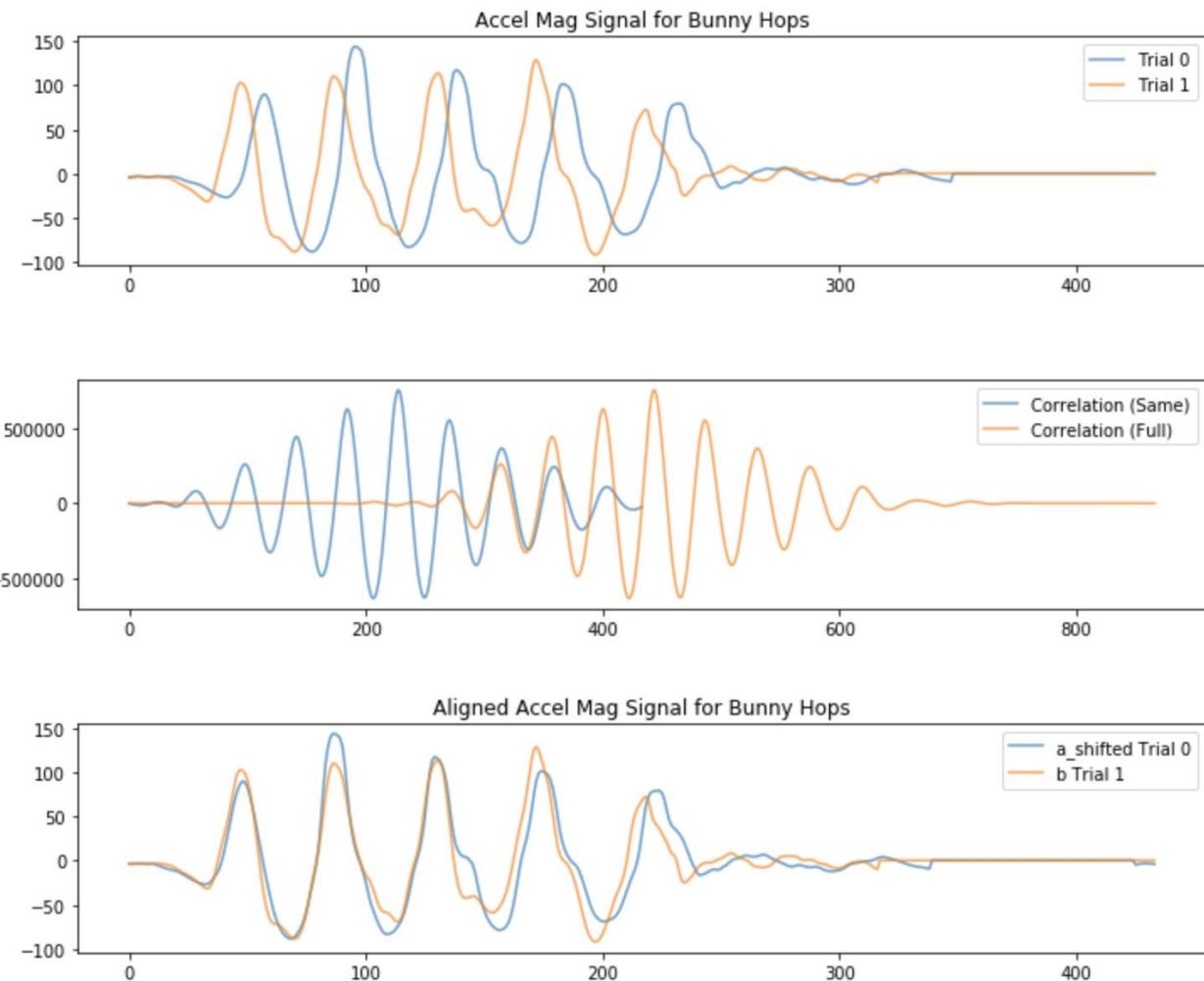


A3: SHAPE-BASED GESTURE RECOGNITION

HINT: EXPLORE CROSS CORRELATION TO ALIGN TWO SIGNALS BEFORE COMPARISON

See:

<https://www.mathworks.com/help/signal/examples/measuring-signal-similarities.html>



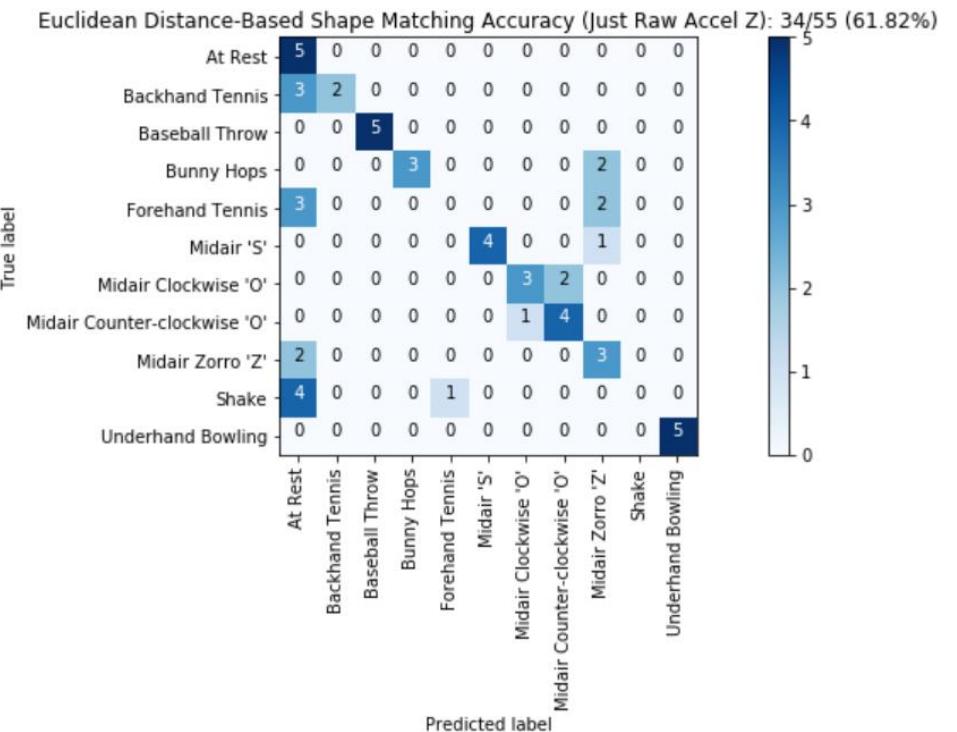
GENERAL APPROACH

1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms
2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.
3. **Play around with various transforms of the data.** Smoothed. Detrended. Scaled?
4. **Determine strategy/algorithm for comparing two signals.** Euclidean distance? Dynamic time warping?
5. **Setup experiment infrastructure to help you evaluate and compare performance.** In this case, we are using k-fold cross validation.

A3: SHAPE-BASED GESTURE RECOGNITION

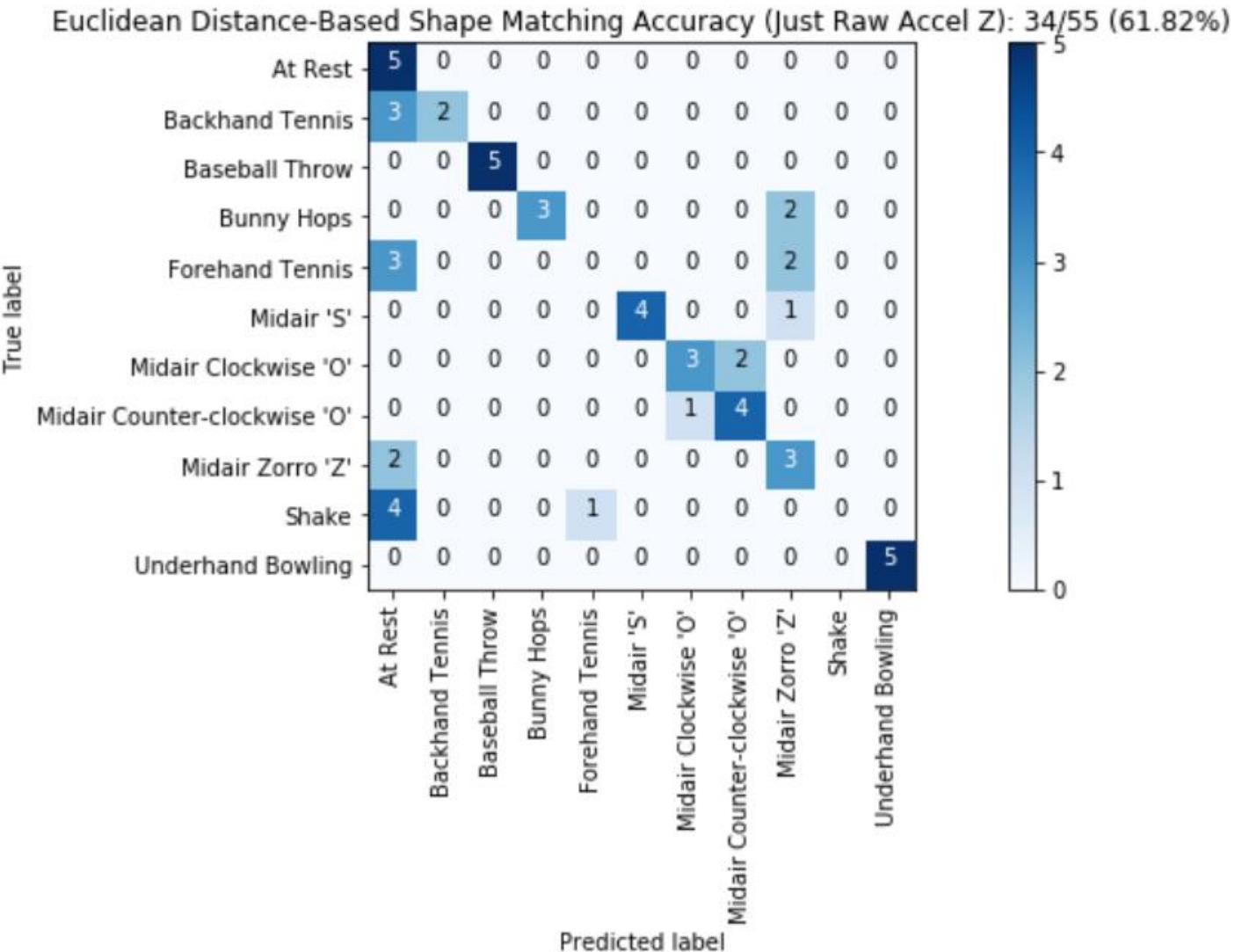
GENERAL APPROACH

1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms
 2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.
 3. **Play around with various transforms of the data.** Smoothed. Detrended. Scaled?
 4. **Determine strategy/algorithm for comparing two signals.** Euclidean distance? Dynamic time warping?
 5. **Setup experiment infrastructure to help you evaluate and compare performance.** In this case, we are using k-fold cross validation.
 6. **Create visualizations of your results to help inform your algorithm.** Iterate. Iterate.



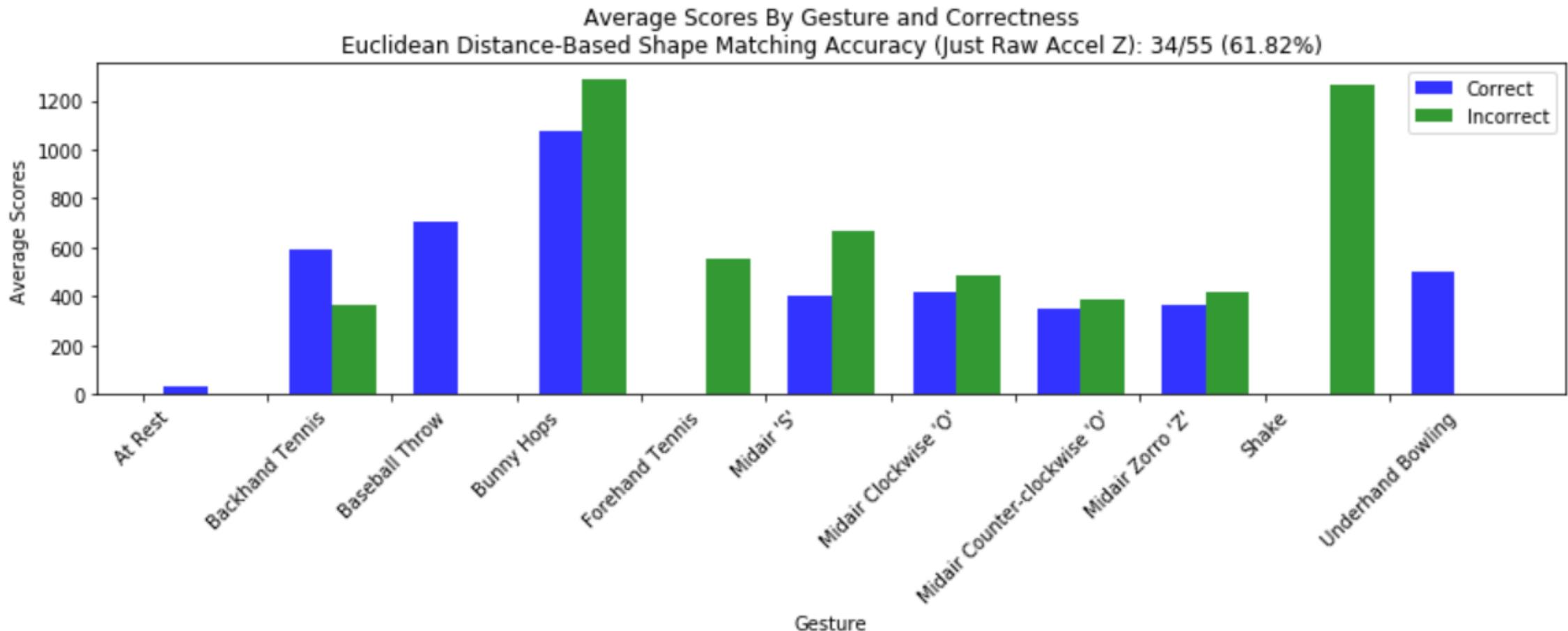
A3: SHAPE-BASED GESTURE RECOGNITION

VISUALIZATIONS: CONFUSION MATRIX



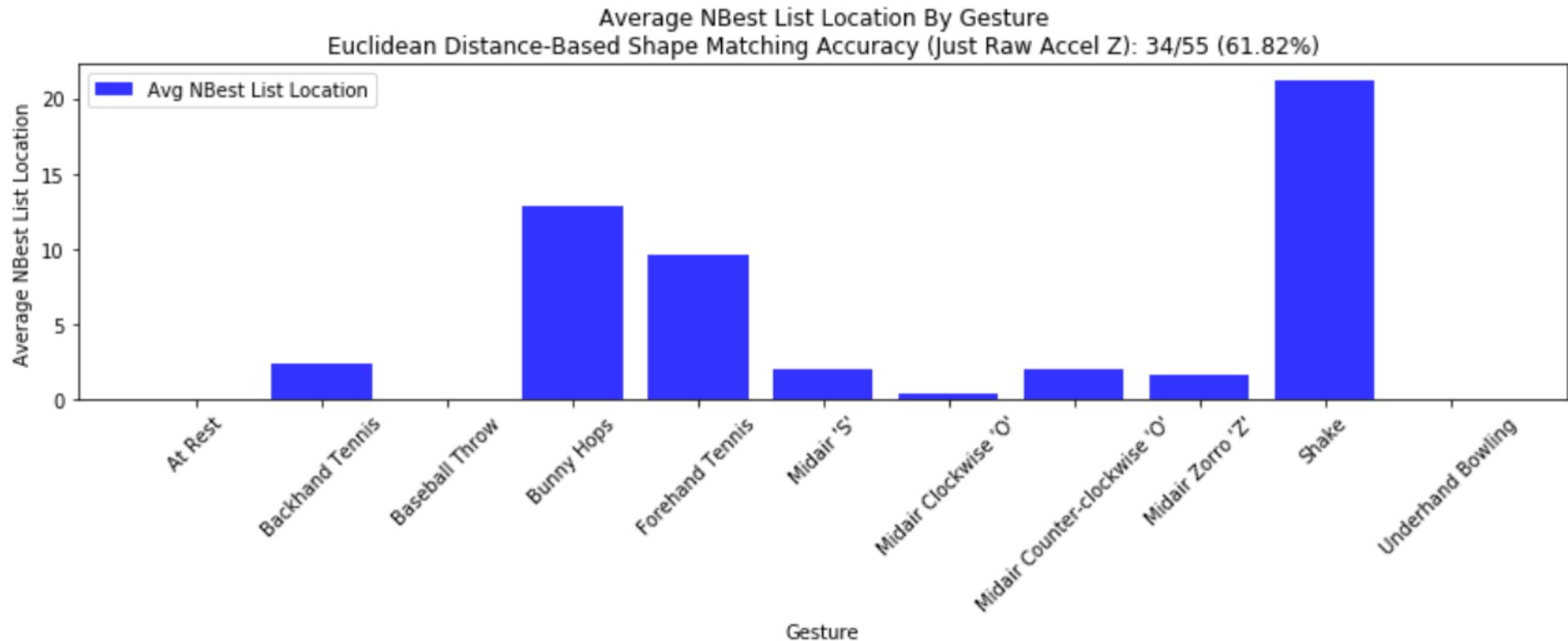
A3: SHAPE-BASED GESTURE RECOGNITION

VISUALIZATIONS: AVERAGE SCORES

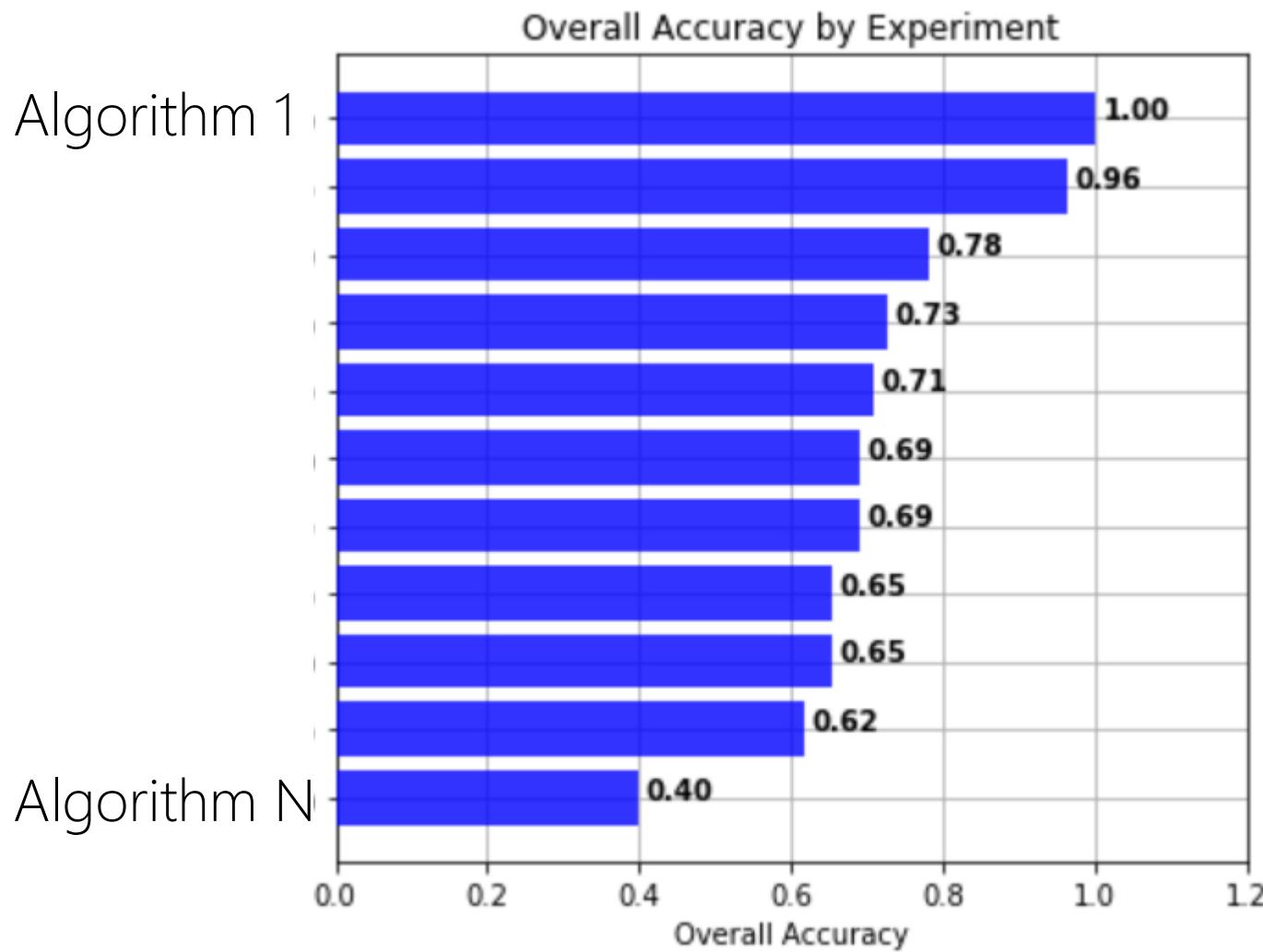


A3: SHAPE-BASED GESTURE RECOGNITION

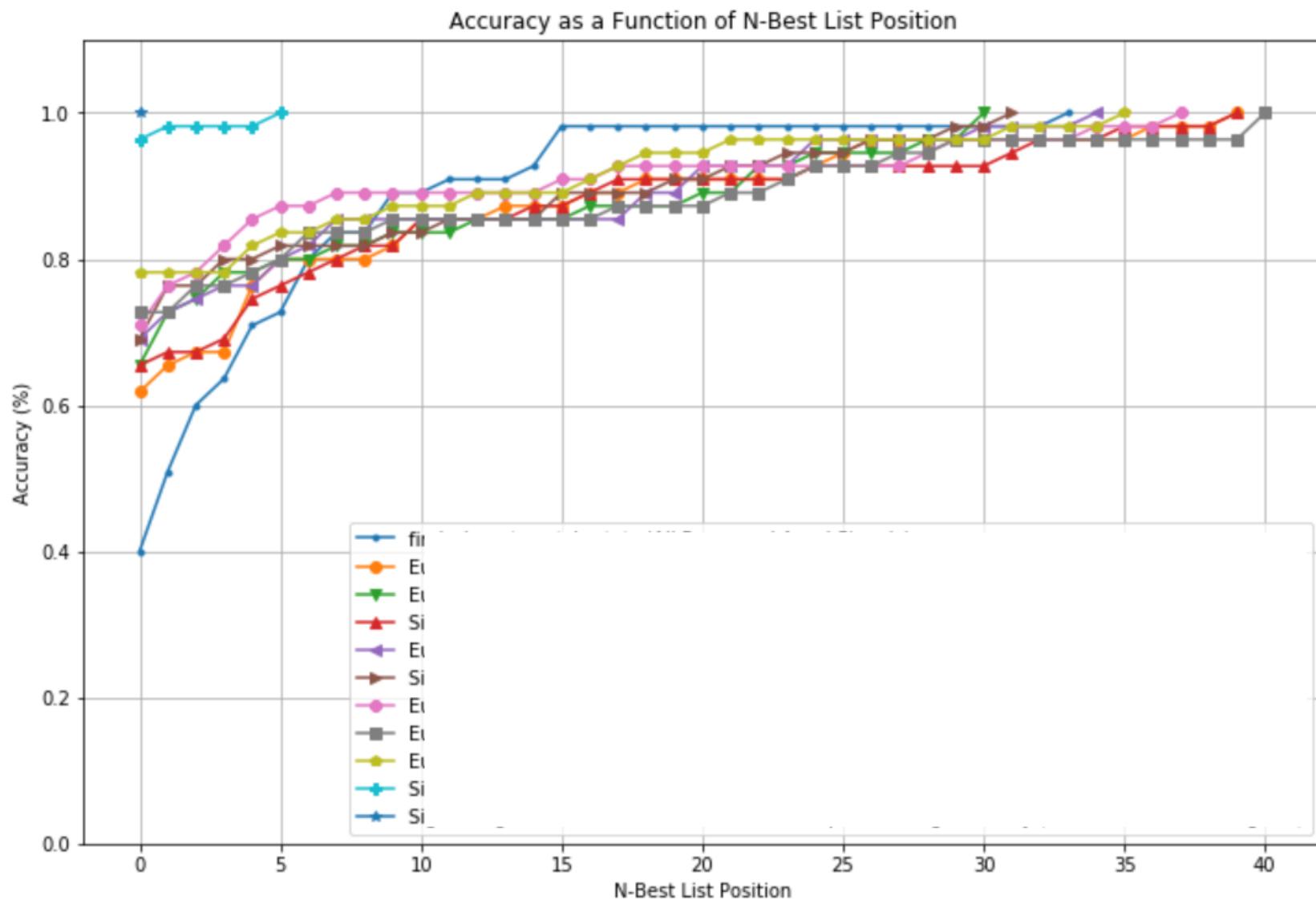
VISUALIZATIONS: AVERAGE NBEST LIST LOCATION

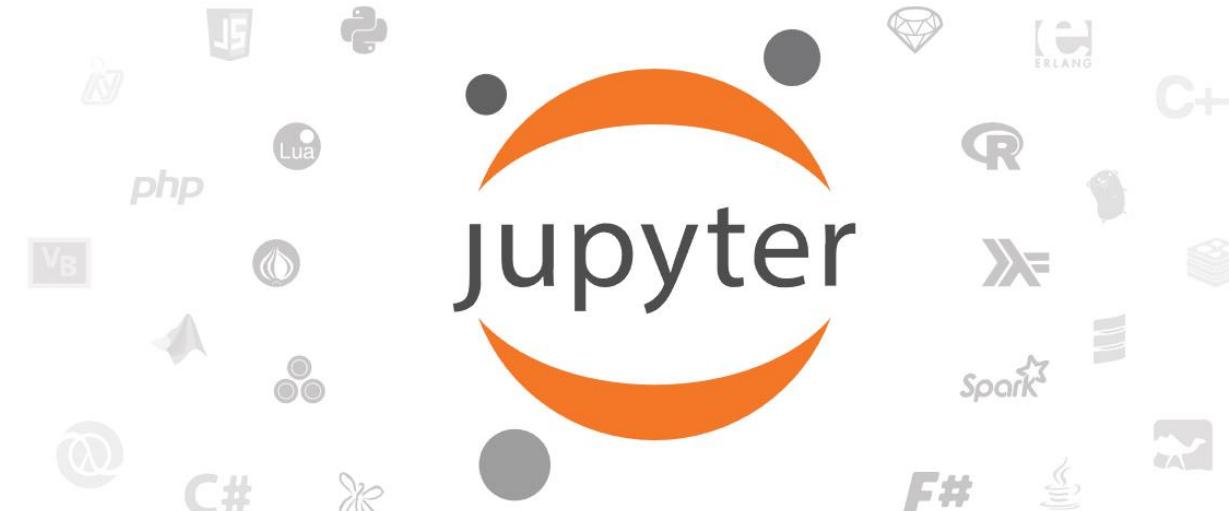


VISUALIZATIONS: COMPARING ALGORITHMS/APPROACHES



VISUALIZATIONS: COMPARING ALGORITHMS/APPROACHES



[Install](#) [About Us](#) [Community](#) [Documentation](#) [NBViewer](#) [Widgets](#) [Blog](#)

Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

JupyterCon 2018

August 21 - 25

JUPYTER NOTEBOOK

TUTORIAL NOTEBOOKS

 [jonfroehlich / CSE599Sp2019](#)

 Unwatch ▾

7

 Star

4

 Fork

0

 Code

 Issues 1

 Pull requests 0

 Projects 0

 Wiki

 Insights

 Settings

Branch: master ▾

[CSE599Sp2019](#) / Notebooks /

Create new file

Upload files

Find file

History



jonfroehlich updated comment

Latest commit 6347c62 2 hours ago

..

 [MatplotlibPyplotTutorial.ipynb](#)

added figure to matplotlib tutorial

16 hours ago

 [NumpyTutorial.ipynb](#)

Added in more dynamic invocation examples

2 hours ago

 [PythonTutorial.ipynb](#)

updated comment

2 hours ago

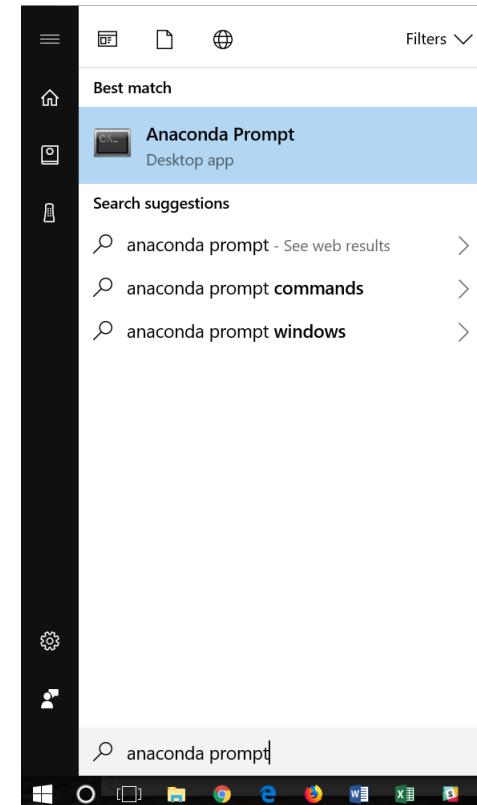
<https://github.com/jonfroehlich/CSE599Sp2019/tree/master/Notebooks>

STARTING A FRESH JUPYTER NOTEBOOK

On a Mac

```
> mkdir funtimes  
> cd funtimes  
> jupyter notebook
```

On Windows





Logout

Files Running Clusters

Select items to perform actions on them.

Upload New ▾

0 /

Name Last Modified

The notebook list is empty.



Logout

Files Running Clusters

Select items to perform actions on them.

0 /

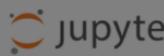
The notebook list is empty.

Upload New ▾

Notebook:
Python 3

Other: Create a new notebook with Python 3

Text File
Folder
Terminal



File Edit View Insert



In []:

Rename Notebook

X

Enter a new notebook name:

This is my first Jupyter Notebook

Cancel

Rename

localhost:8888/notebooks/This%20is%20my%20first%20Jupyter%20Notebook.ipynb

jupyter This is my first Jupyter Notebook Last Checkpoint: 2 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run

```
In [1]: "Hello World"
Out[1]: 'Hello World'
```

In []:

localhost:8888/notebooks/This%20is%20my%20first%20Jupyter%20Notebook.ipynb

jupyter This is my first Jupyter Notebook Last Checkpoint: 3 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

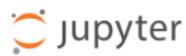
In [1]: "Hello World"

Out[1]: 'Hello World'

In [3]: myList = [1, 2, 3, 4, 5]
print(myList)

[1, 2, 3, 4, 5]

In []:



File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3



In [1]: "Hello World"

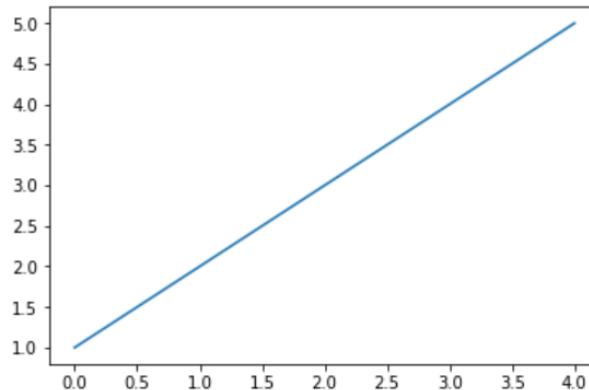
Out[1]: 'Hello World'

In [3]: myList = [1, 2, 3, 4, 5]
print(myList)

[1, 2, 3, 4, 5]

In [6]: import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(myList)

Out[6]: [<matplotlib.lines.Line2D at 0x1e822d0e8d0>]



In []:



File Edit View Insert Cell Kernel Widgets Help

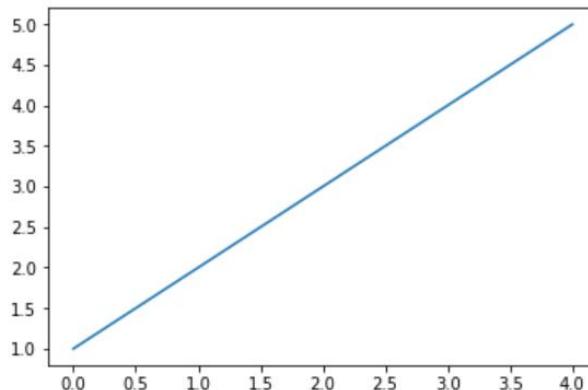
Trusted

Python 3



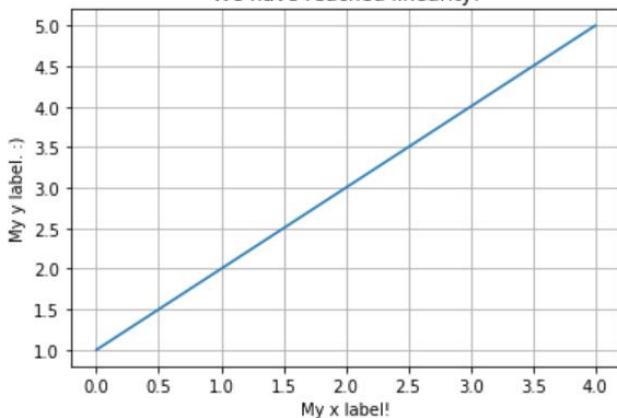
```
fig, ax = plt.subplots()
ax.plot(myList)
```

Out[6]: [`<matplotlib.lines.Line2D at 0x1e822d0e8d0>`]



```
In [8]: fig, ax = plt.subplots()
ax.plot(myList)
ax.set(xlabel='My x label!', ylabel='My y label. :)',
       title='We have reached linearity!')
ax.grid()
```

We have reached linearity!



File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3



Code

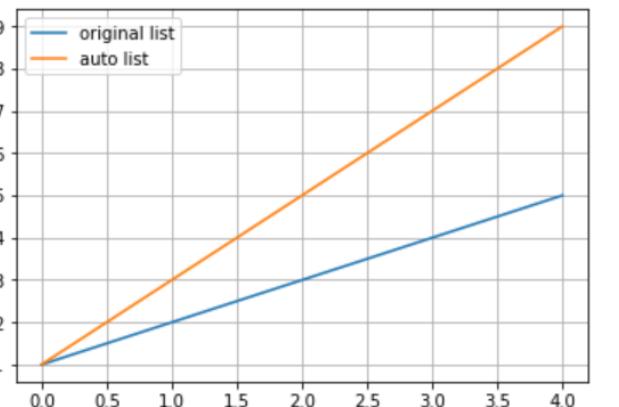


```
# range takes a start val, end val, and step
myList2 = list(range(1, 10, 2))
print(myList2)
```

```
[1, 3, 5, 7, 9]
```

In [16]: # Let's plot both of these lists

```
fig, ax = plt.subplots()
ax.plot(myList, label="original list")
ax.plot(myList2, label="auto list")
ax.legend()
ax.grid()
```



In [19]: # We can also have HashMaps or dictionaries--Python calls them dicts

```
mapNamesToAges = {"Camden": 1.5, "Gavin": 3.3, "Susan": 4.5, "Billy": 4.4, "Frankie": 6}
print(mapNamesToAges)
```



```
{'Camden': 1.5, 'Gavin': 3.3, 'Susan': 4.5, 'Billy': 4.4, 'Frankie': 6}
```

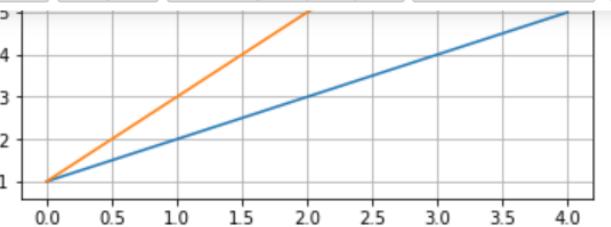
In [20]: # Let's plot them all!

```
fig, ax = plt.subplots()
ax.plot(myList, label="original list")
ax.plot(myList2, label="auto list")
ax.plot(mapNamesToAges.values(), label="ages")
ax.legend()
ax.grid()
```

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

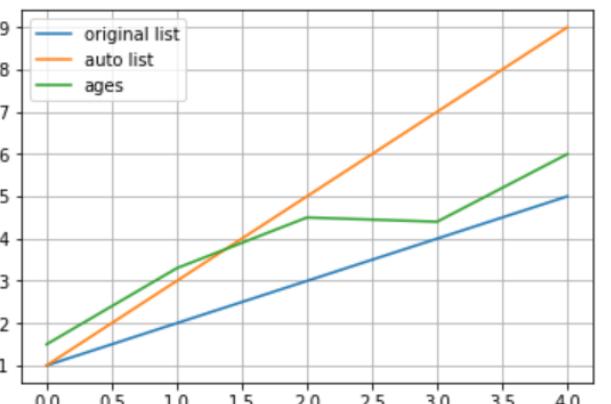
Python 3



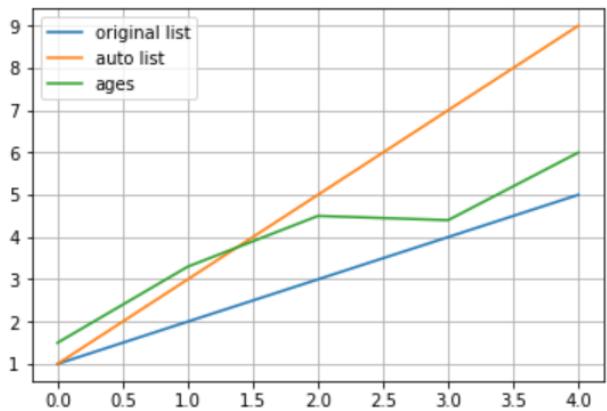
```
In [19]: # We can also have HashMaps or dictionaries--Python calls them dicts
mapNamesToAges = {"Camden" : 1.5, "Gavin" : 3.3, "Susan": 4.5, "Billy": 4.4, "Frankie" : 6}
print(mapNamesToAges)

{'Camden': 1.5, 'Gavin': 3.3, 'Susan': 4.5, 'Billy': 4.4, 'Frankie': 6}
```

```
In [20]: # Let's plot them all!
fig, ax = plt.subplots()
ax.plot(myList, label="original list")
ax.plot(myList2, label="auto list")
ax.plot(mapNamesToAges.values(), label="ages")
ax.legend()
ax.grid()
```



Markdown Support!



Markdown Support!

Jupyter Notebook also supports markdown! From the dropdown menu, select Cell->Cell Type->Markdown

Markdown Support!

Jupyter Notebook also supports markdown! From the dropdown menu, select Cell->Cell Type->Markdown

NOTEBOOK SUPPORTS MARKDOWN



1

I am a new to Jupyter Notebook too, must say that their documentation is awful as they say what we **can do** instead of **HOW** to do it. I have opened already made '.ipynb' notebook and figured it out, I will list a couple of them from **unexplained documentation**:

0

1



1. Make text **ITALIC**: *Italic*
2. Make text **BOLD**: **Bold**
3. List item as a bullet: dash and space -
4. List item as a number: Simple as number and dot 1.
5. Indenting text: Greater than and space >
6. Inline code span: Back quotation mark `` ``
7. Block of code: Triple back quotation marks " `` `` "
8. Link a section: [Title of Section](#title-of-section)
9. Hyperlink: [Text](URL)

0

Hot I

 Wc
for

Ho

Co

 Ca
"m

An

 Ho
fro

Ho

 Wt
inp

I hope everyone has found their solution.

[share](#) [improve this answer](#)

answered Apr 26 at 22:48



Uki.

11 • 1

[add a comment](#)