Streaming Time Series Analysis for FPGAs

Eamonn Keogh

eamonn@cs.ucr.edu

Philip Brisk

philip@cs.ucr.edu

FCCM Tutorial

Welcome

The plan for today

- Dr. Keogh will speak *what* you can do with time series analytics (hint: almost everything) for about 70 minutes.
- Coffee Break
- Dr. Brisk will speak about *how* to do this time series analytics in an efficient and scalable way.

Format

- Dr. Keogh's part of this tutorial has almost no math or code.
- We want to give you the *intuition* behind the SOTA time series data mining.
- While these ideas are general, here we will make them concrete, by mostly considering a single running example (cows!).

- **Important note**: A few slides are a little too "dense", these are designed to be presented superficially today, and then read offline.

Our core claim

- What do you want to do with time series?
  - Classification, clustering, summarization, visualization, segmentation, anomaly discovery, repeated pattern discovery, rule discovery, emerging behavior discovery etc.
  - Amazingly, two simple related tools, the *Distance Profile*, and the *Matrix Profile*, is basically all you need to do all the above!

- To analyze time series, there are two different situations, which have almost no overlap:
  - Behaviors are conserved in *shape* (first ¾ of Keoghs talk)
  - Behaviors are conserved in *features* (last ¼ of Keoghs talk)

Bias Alert! We are the inventors of the Matrix Profile, so there is a danger this could be self indulgent… However, the Matrix Profile has exploded in popularity in the last 3 years

observations of the magnetosphere collected by the Cassini spacecraft in orbit around Saturn.. ..in this case, the best-performing method was the Matrix Profile.. Kiri L. Wagstaf et. al. NASA JPL.2020

(for an industrial IoT problem) *Matrix Profiles perform well with almost no parameterisation needed.* Anton et al ICDM 2018.

While there will never be a mathematical silver bullet, we have discovered that the Matrix Profile, a novel algorithm developed by the Keogh research group at UC-Riverside, is a powerful tool. Andrew Van Benschoten, lead engineer at Target.

If anybody has ever asked you to analyze time series data and to look for new insights then (the Matrix Profile) is definitely the open source tool that you'll want to add to your arsenal Sean Law, Ameritrade.

(for) intrusion detection in industrial network traffic, distances as calculated with Matrix Profiles rises significantly during the attacks. ..as a result, time series-based anomaly detection methods are capable of detecting deviations and anomalies. Schotten (2019).

The MatrixProfile technique is the state-of-the-art anomaly detection technique for continuous time series. Bart Goethals et. al. (ECML-PKDD 2019).

Based on the concept of Matrix Profile ..without relying on time series synchronization.. the Railway Technologies Laboratory of Virginia Tech has been developing an automated onboard data analysis for the maintenance track system Ahmadian et. al. JRC2019

Matrix Profile is the state-of-the-art similarity-based outlier detection method. Christian Jensen et. al. IJCAI-19

we use the exact method based on the Matrix Profile (to assess the effectiveness of therapy) Funkner et al Procedia 2019.

Recently, a research group from UCR have proposed a powerful tool - the Matrix Profile (MP) as a primitive...(we use it for) *fault detection* Jing Zhang et al. ICPHM 2019

Inspecting both graphs one can see that the matrix-profile algorithm was able to identify regions where there is a change on the power level over the observed band. F Lobao 2019.

RAMP builds upon an existing time series data analysis technique called Matrix Profile to detect anomalous distances...collected from scientific workflows in an online manner. Herath et. al. IEEE Big Data 2019

Based on obtained results for the considered data set, matrix profiles turned out to be most suitable for the task of anomaly detection Lohfink et al. VISSEC2019

The computation speed and exactness of the Matrix Profile make it a powerful tool and (our) results back this. Barry & Crane AICS 2019

(examining) *manufacturing batches considering raw amperage* (we found that the) *Matrix Profile highlights anomalies* Hillion & O'Connell of TIBCO Data Science. re:Invent 2019. [

we use the exact method based on the matrix profile to search for motifs can be used to monitor the patient's condition, to assess the effectiveness of therapy or to assess the physician's actions. Funknera et al.

(The Matrix Profile is a) *similarity join to measure the similarity between two given sequences. we opt for the median of the profile array as the representative distance* (3D Dancing Move Synthesis from Music)" Anh et al. IEEE Robotics and Automation Letters

We were amazed by the power of MP and seek to incorporate it into our framework Ye and Ageno.

..adopting the concept of (the) Matrix Profile, we conduct the first attempt to.. J. Zuo et. al. Big Data 20019

The accuracies obtained ...indicate that the Matrix Profile is useful for the task at hand instead of using the CNN features directly Dhruv Batheja

To speed up online bad PMU data detection a fast discovery strategy is introduced based on (the Matrix Profile) Zhu and Hill.

Specifically, ALDI uses the matrix profile method to quantify the similarities of daily subsequences in time series meter data, Zoltan Nagy, Energy & Buildings (2020)

Our two-fold approach first leverages the Matrix Profile technique for time series data mining.. Nichiforov 2020.

*the class of matrix profile algorithms.. ..is a promising approach, as it allows simplified post-processing and analysis steps by examining the resulting matrix profile structure*A. Raoofy et al.

We only require information about the time of several critical incidents to train our methods, as previously. To this end, we employ the Matrix Profle.. Bellas. et al.

a matrix-profile based algorithm applied across all trajectory data against a validation set revealed four significant motifs which we defined as motif A, B, C and D.. Fernandez Alvarez 2020.

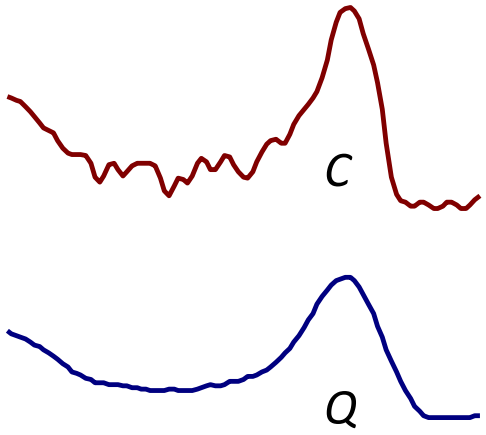
The main building block of this (game analytics) algorithm is the matrix profile, Saadat and Sukthankar AAAI2020

We leverage the Matrix Profile (MP),... to create a micro-service-based machinery monitoring solution Naskos et al 2021

SLMAD uses statistical-learning and employs a robust box-plot algorithm and Matrix Profile (MP) to detect anomalies Team from Huawei/UCD.

Comparing two time series

How similar are these two-time series?



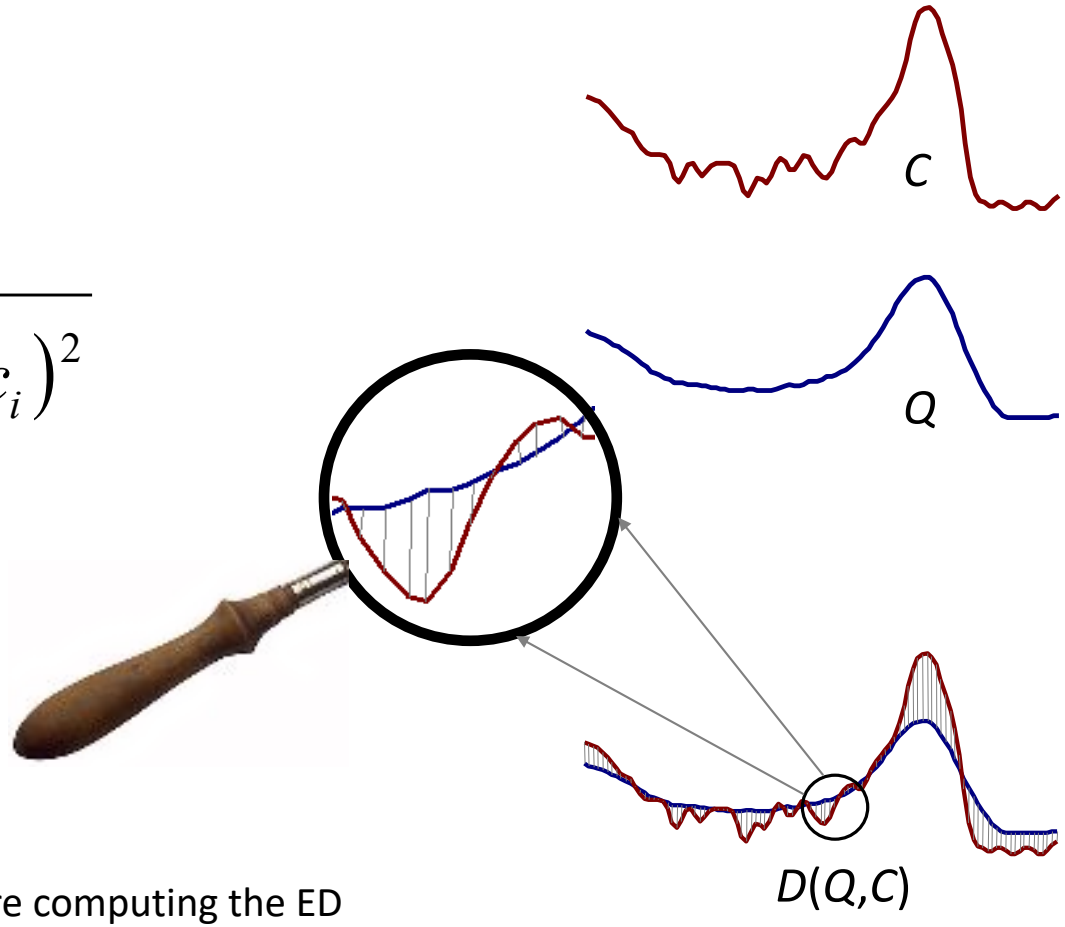
Euclidean Distance Metric (ED)

Given two time series:

$Q = q_1 \ldots q_n$

$C = c_1 \ldots c_n$

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



We always z-normalize the data before computing the ED

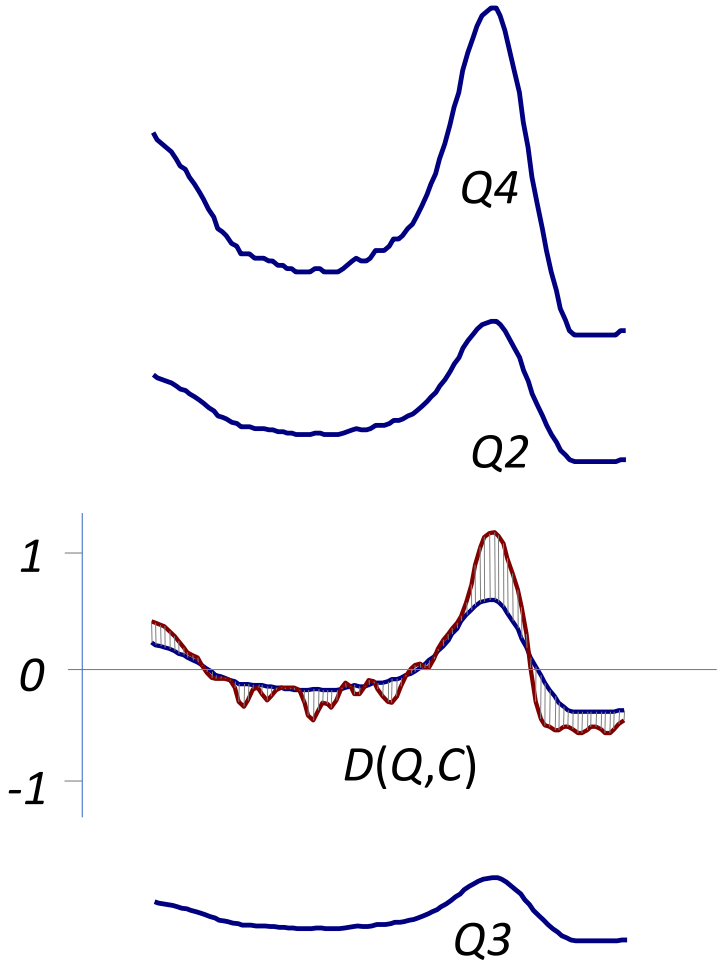
This is logically equivalent to Pearson's Correlation

Euclidean Distance Metric (ED)

Z-normalization means we are ignoring mean and standard deviation of the data.

At least 99.9% of the time, that is the right thing to do.

So here $D(Q,C) = D(Q2,C) = D(Q3,C) = D(Q4,C)$



Euclidean Distance Metric (ED)

Z-normalization means we are ignoring mean and standard deviation of the data.

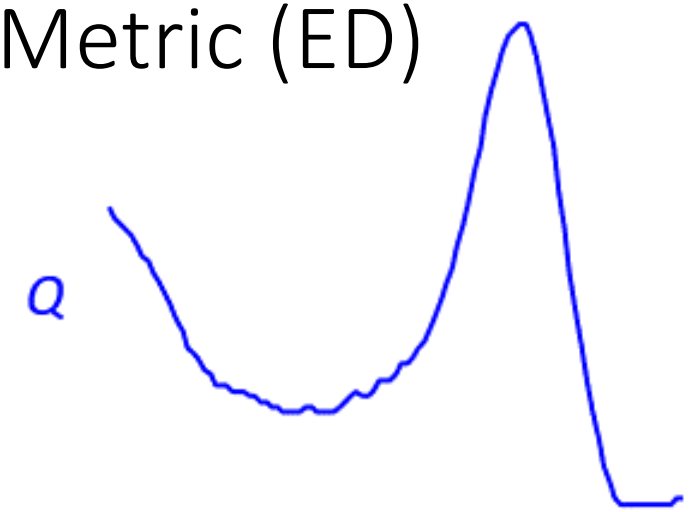
At least 99.9% of the time, that is the right thing to do.

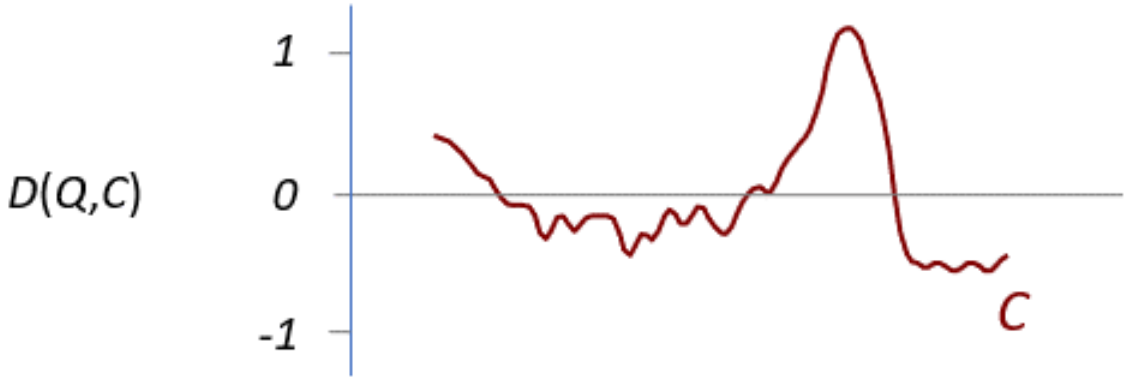
In Matlab

```
>> Q = zscore(Q);
```

Or

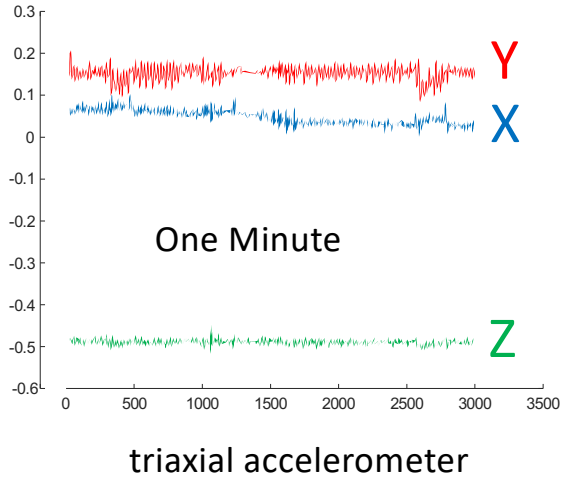
```
>> Q = (Q-mean(Q))/std(Q);
```






Let us use Apollo to make
a running example for
this tutorial

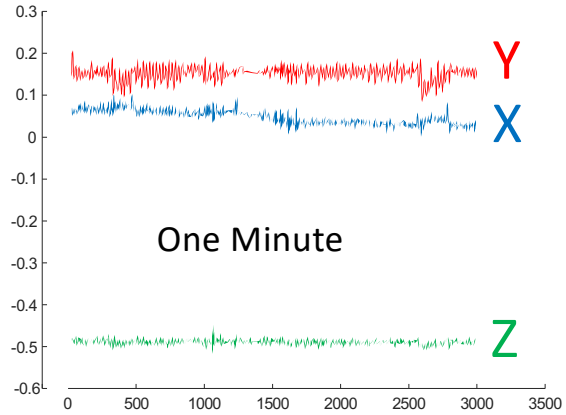
The IMU used in the collar tags is InvenSense MPU-9250[5] at 50 Hz



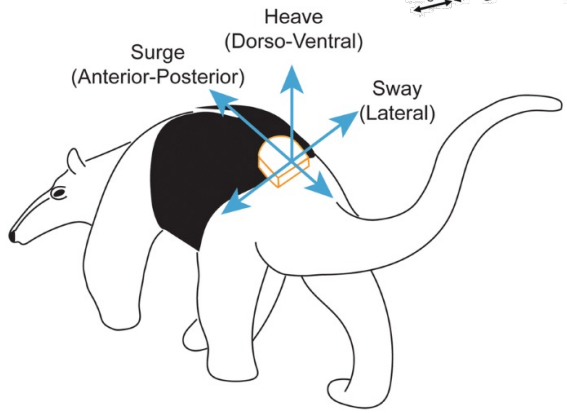
triaxial accelerometer

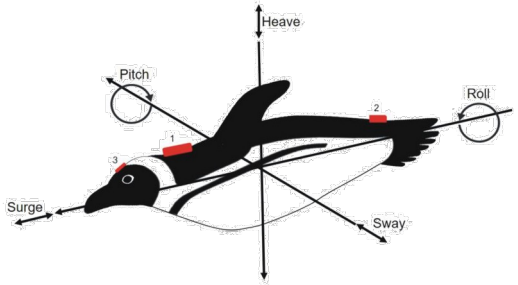
To be clear, X, Y and Z do not have any standard meaning for animal work. *Surge*, *Heave* and *Sway* are the biological terms for acceleration directions. (roll, pitch, yaw for rotation, are measures with a gyroscope).

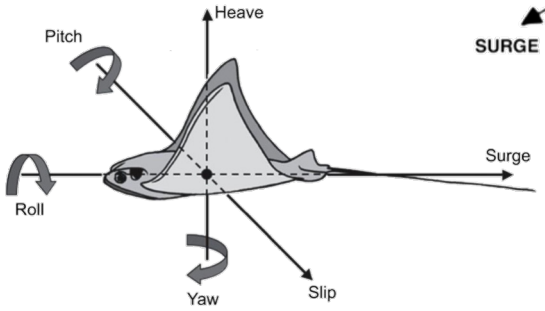
It is rare that X, Y and Z can be made to exactly map to Surge, Heave and Sway, because of the orientation of the sensors against the animal's body.

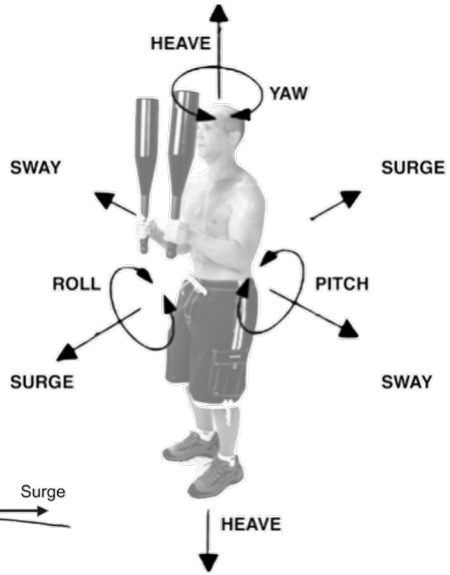


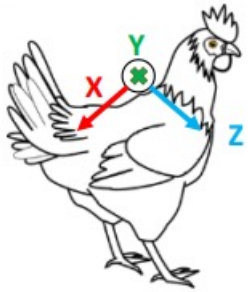
triaxial accelerometer

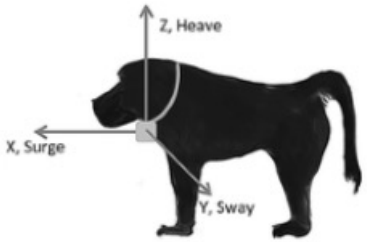












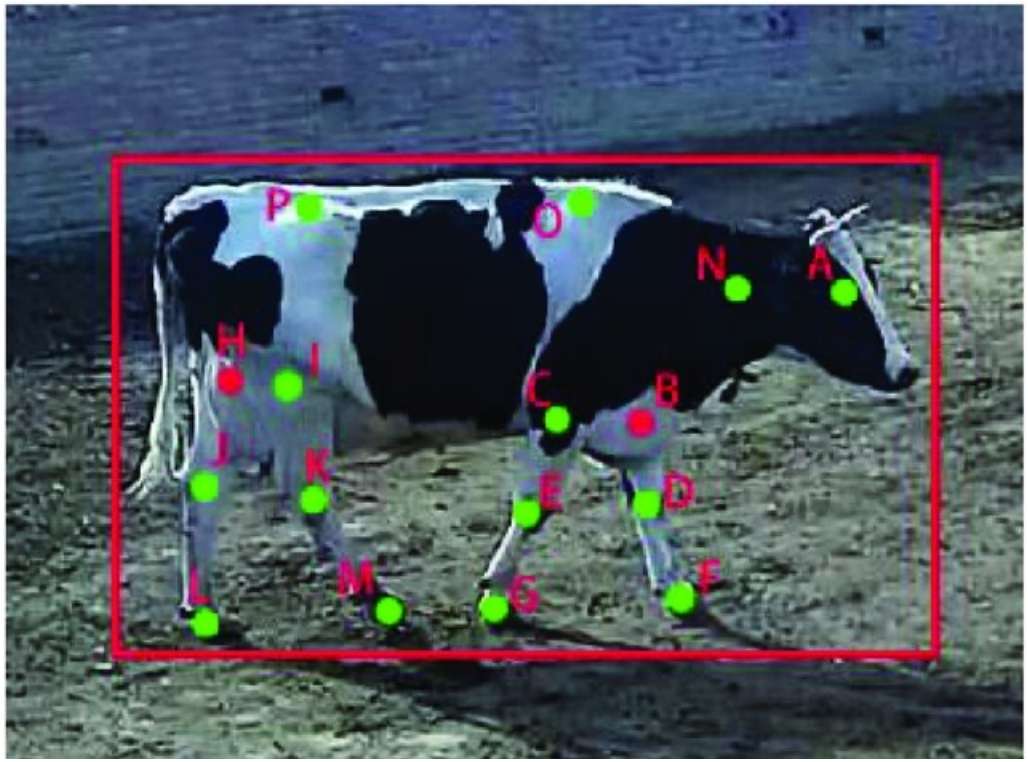
Alternatives to Accelerometers

Markerless tracking of animals has recently become *incredibly* robust and easy.

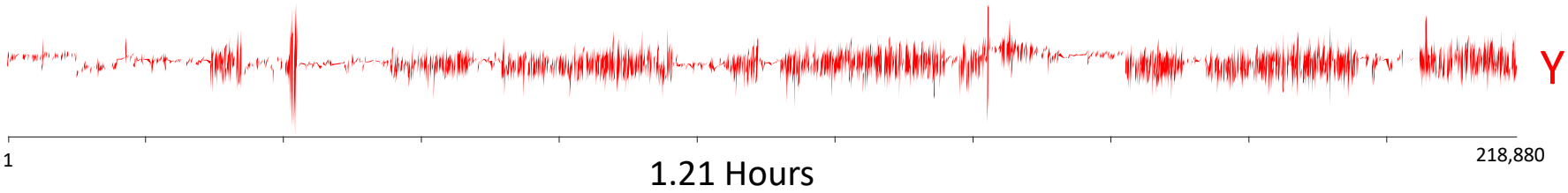
For this tutorial, I don't care how you got your time series.

http://www.mackenziemathislab.org/deeplabcut

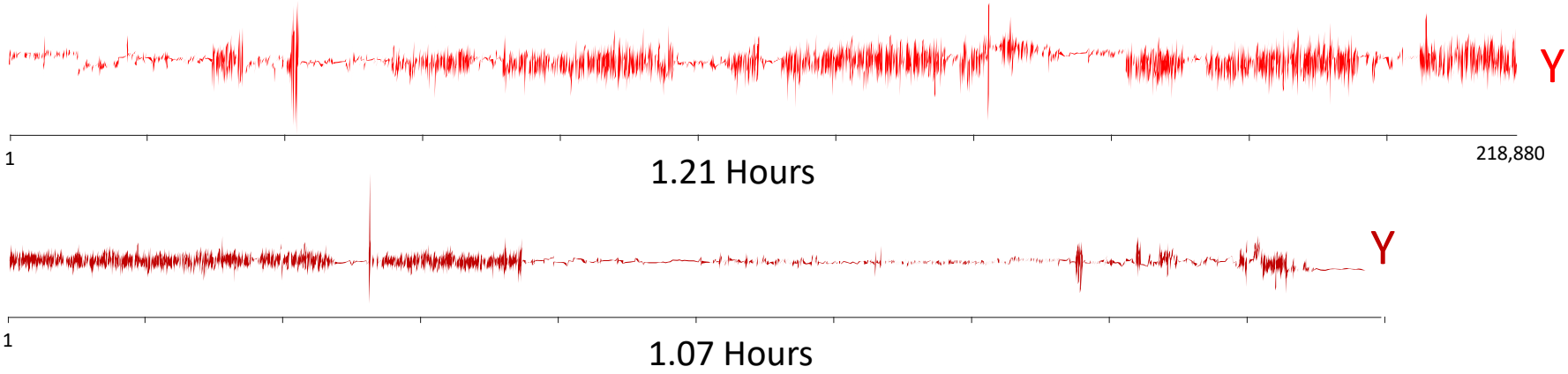
DeepLabCut™:
a software package for animal pose estimation



Apollo



Apollo



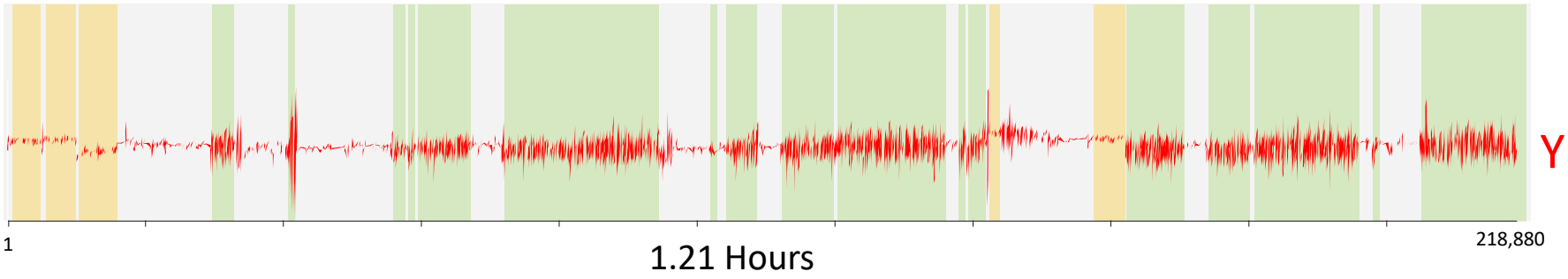
Dante

If possible, try to get *labeled* data.

- Could be
  - Apollo vs. Dante
  - Apollo before operation vs. Apollo after operation
  - Apollo on pasture vs. Apollo on grain
  - Apollo during winter vs. Apollo during summer
  - etc.

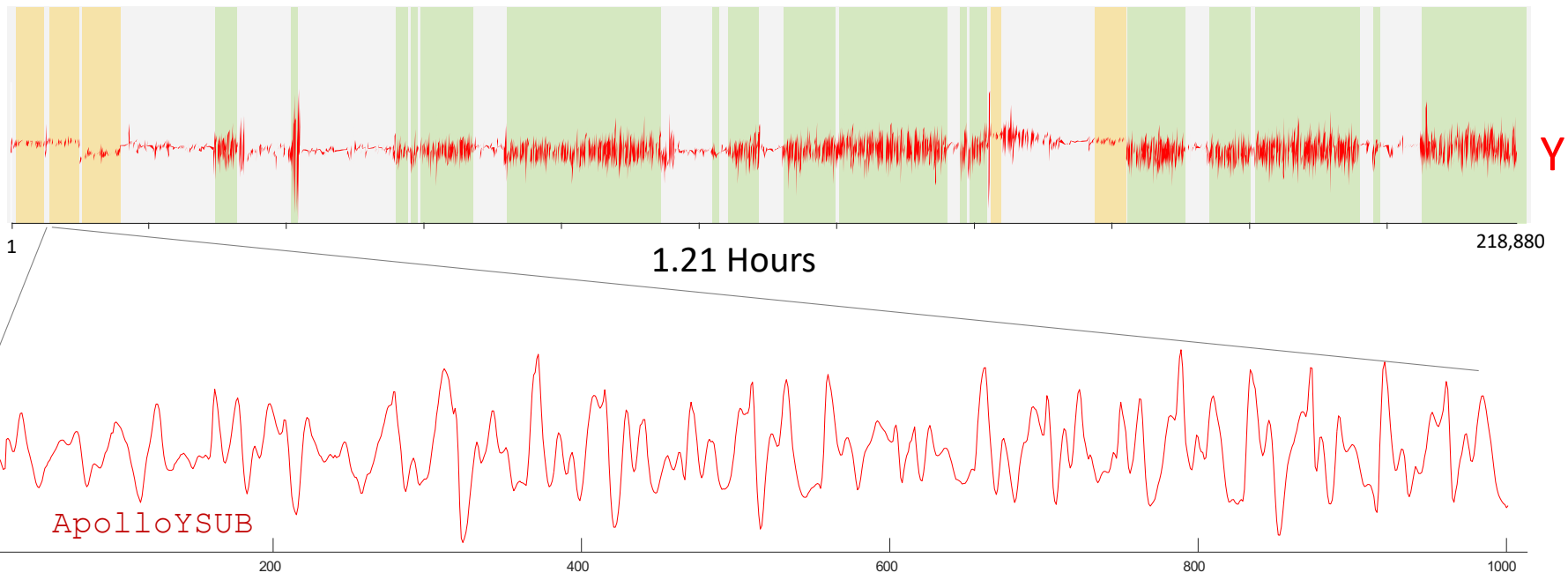
Try to get *all* labels, *all* metadata at data collection time.
It is very frustrating to find interesting structure in the data but have no way to "*go back in time*" to discover its cause.

1: grazing
2: ruminating
3: resting/other



As it happens, for this dataset, we have wonderfully detailed labels, a second-by-second annotation of the behavior.

1: grazing
2: ruminating
3: resting/other

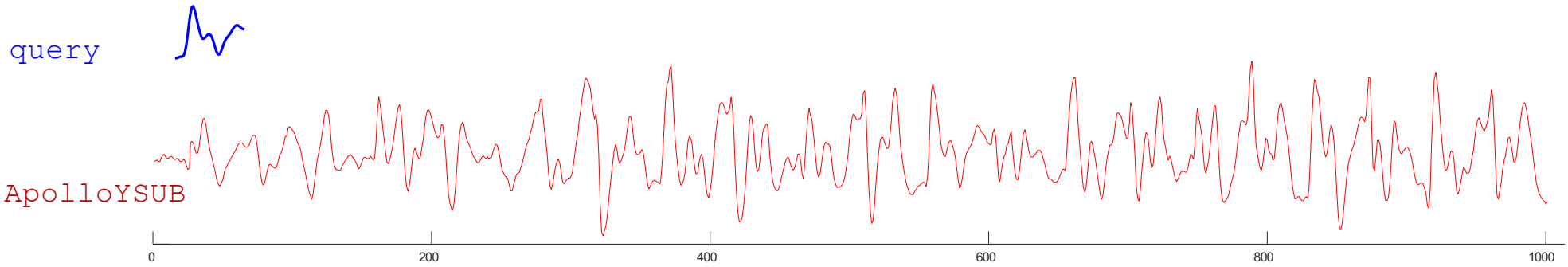


Lets zoom-in on 20 seconds of ruminating behavior
Call it `ApolloYSUB`

I have this one-second-long behavior  ,I am going to call is `query`.

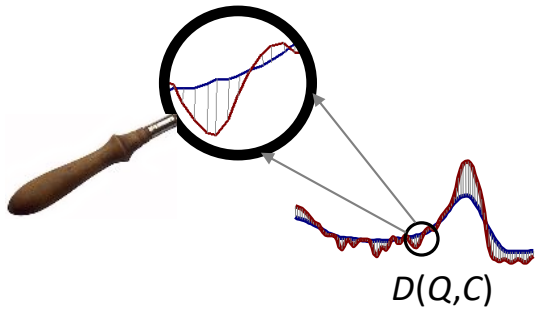
I have reason to think that it is indicative of Bovine spongiform encephalopathy (BSE)

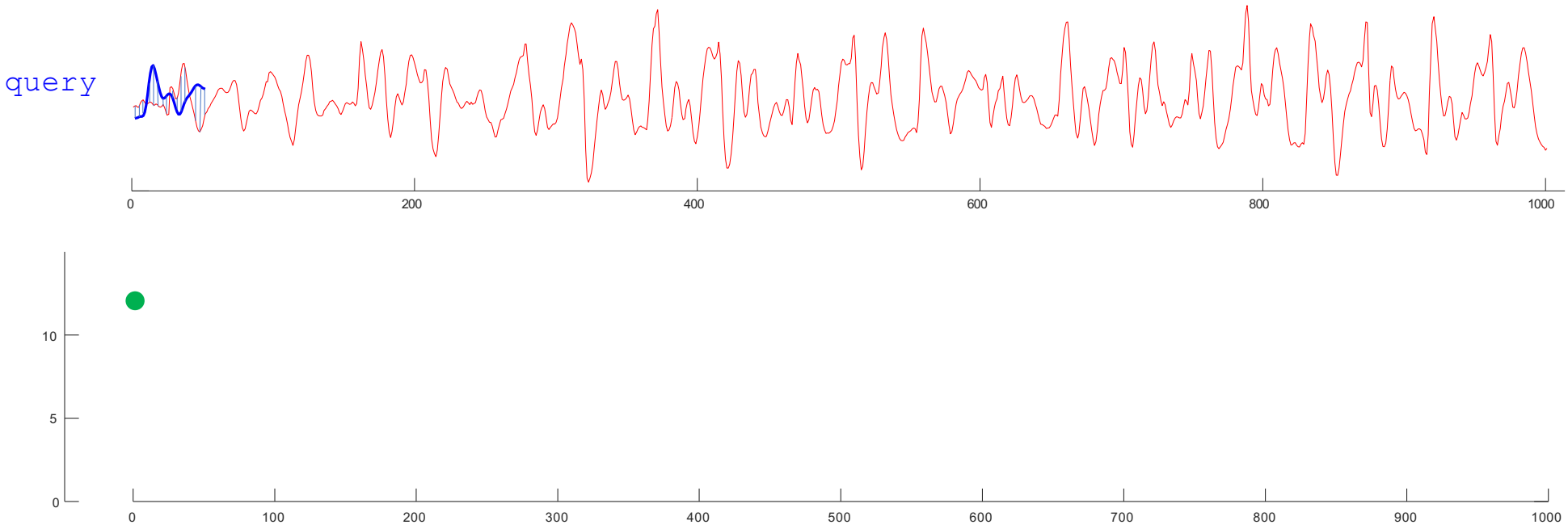
Does this query behavior exist in Apollo? To find out, we will build a *distance profile*.



To find out, we will build a *distance profile*.

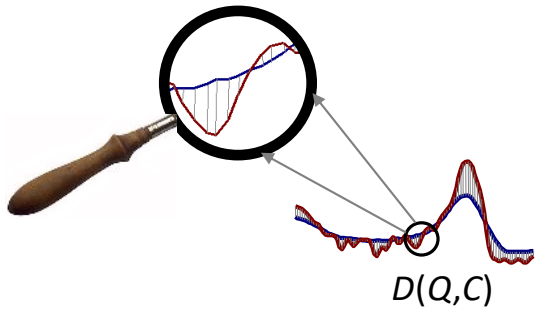
This is simply the z-normalized Euclidean distance between the query and every subsequence in the longer time series…

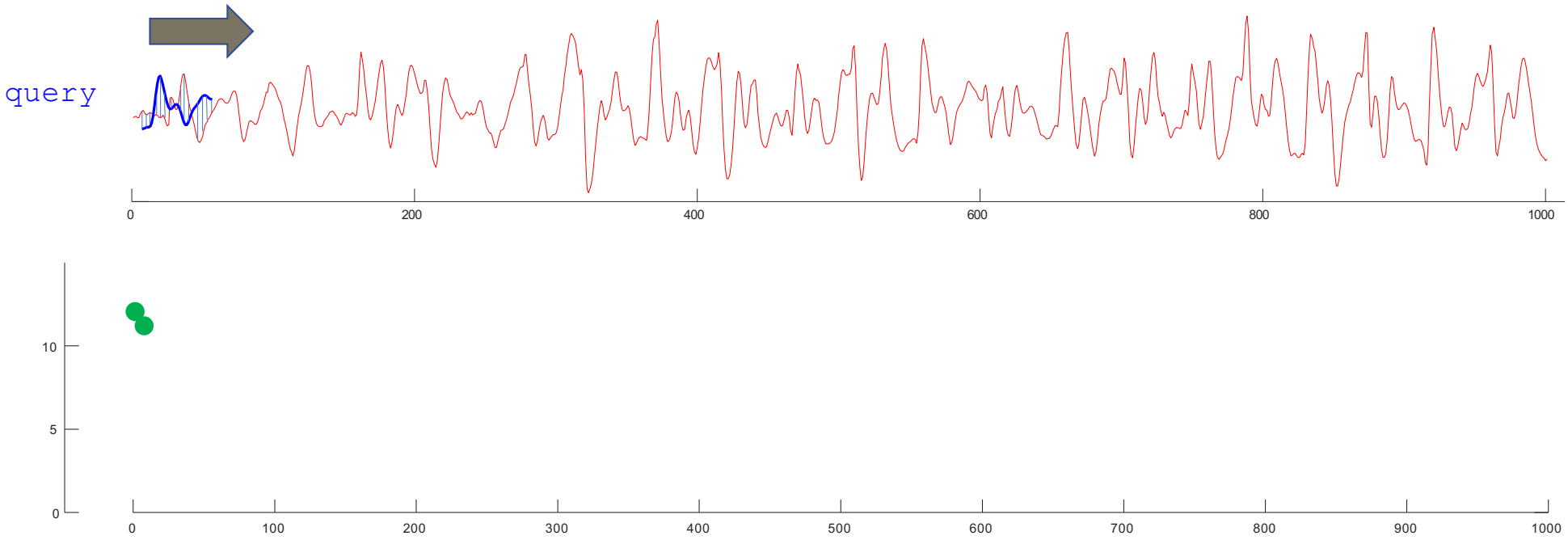




To find out, we will build a *distance profile*.

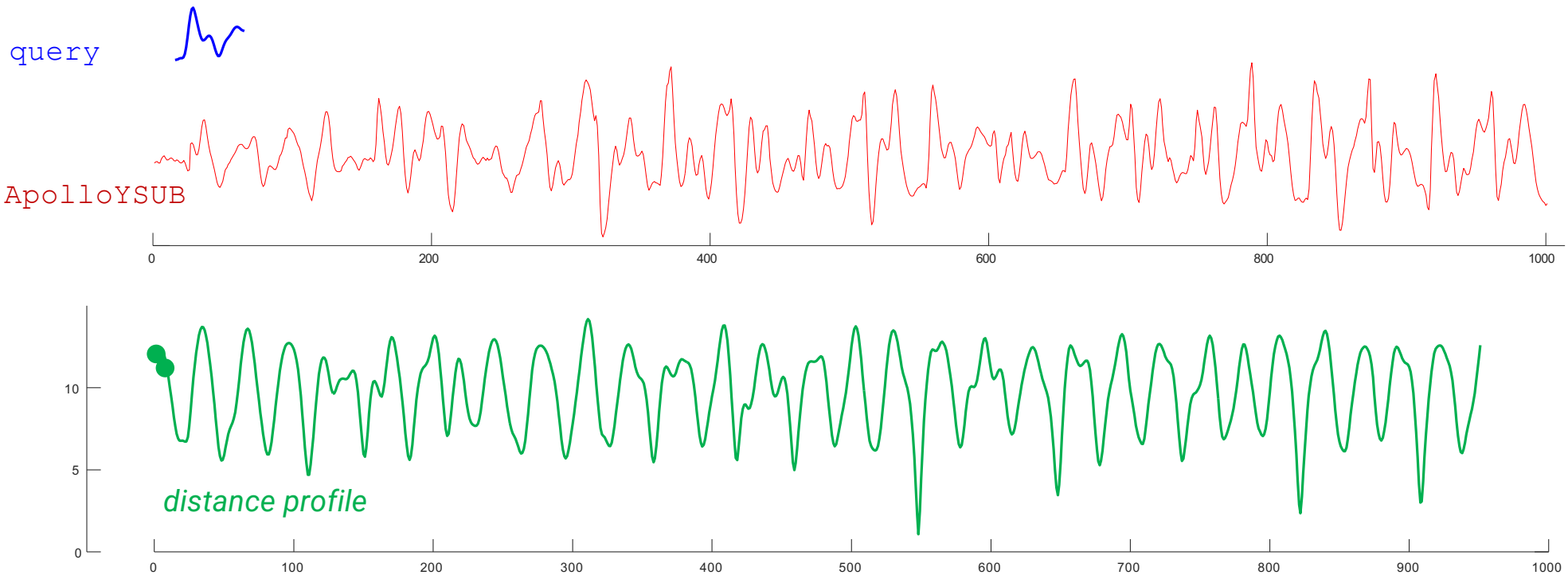
This is simply the z-normalized Euclidean distance between the query and every subsequence in the longer time series…





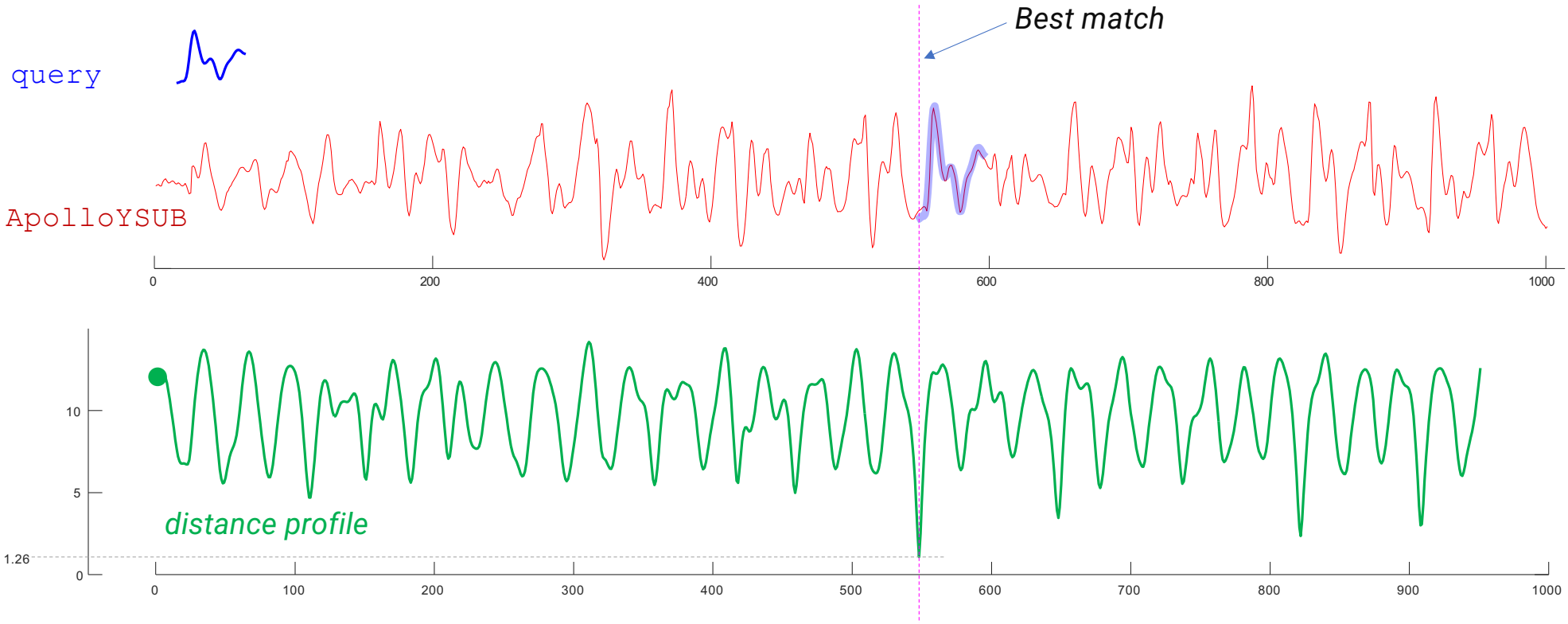
To find out, we will build a *distance profile*.

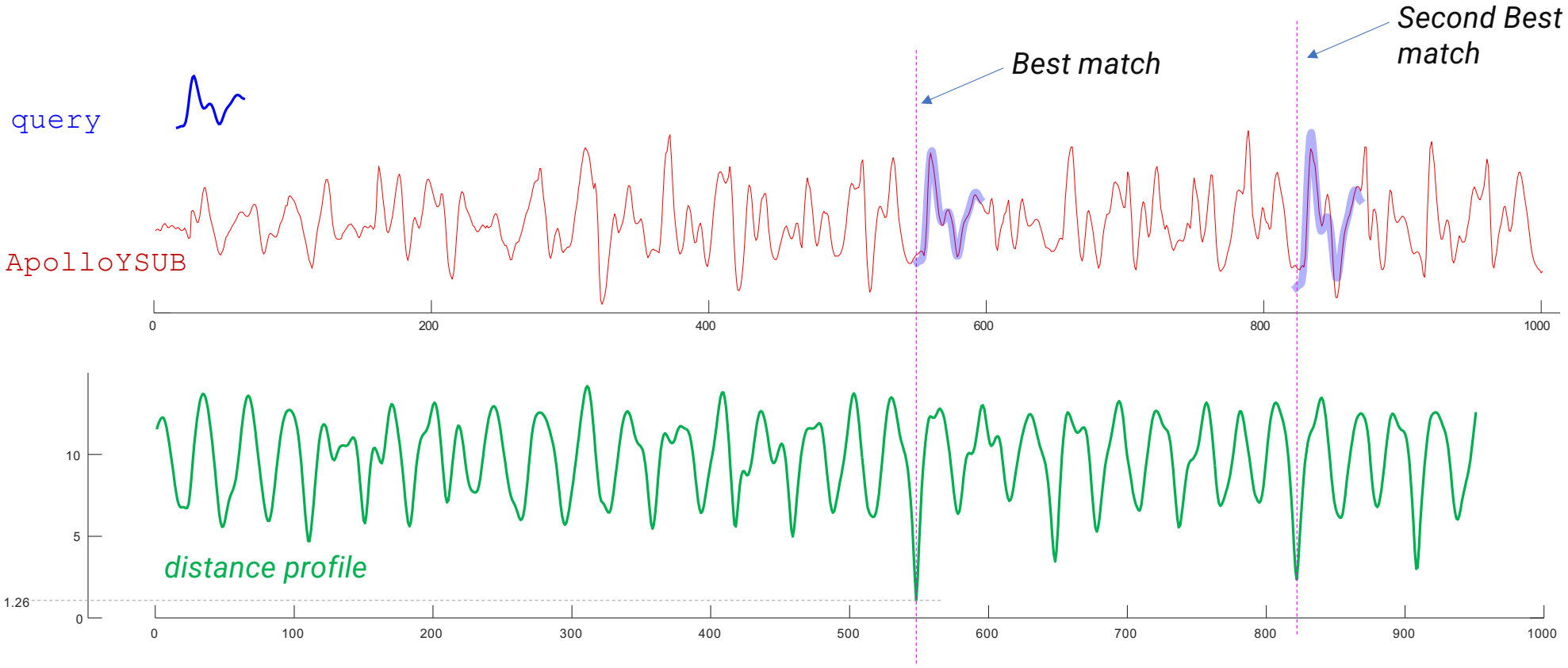
```
>> dist_profile = real(MASS_V2(ApolloYSUB,query));
>> plot(dist_profile)
```

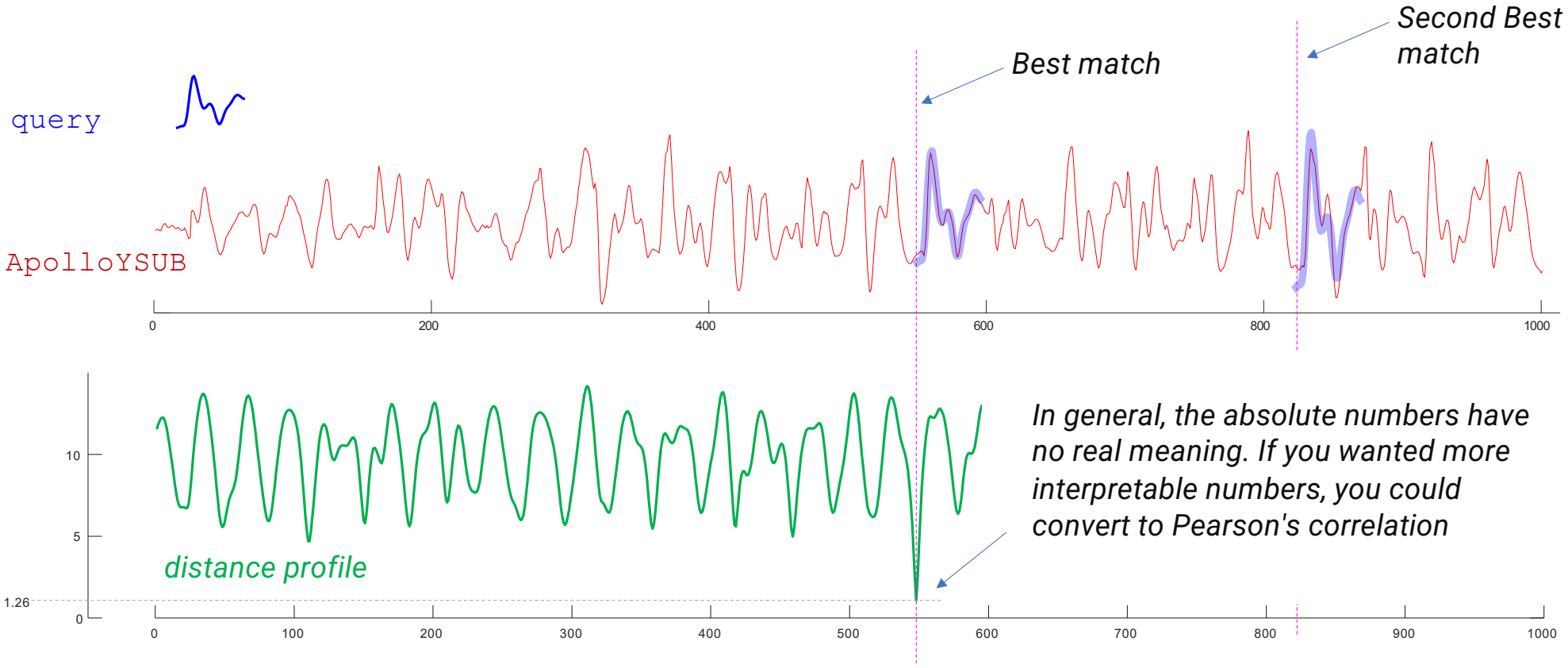


To find out, we will build a *distance profile*.

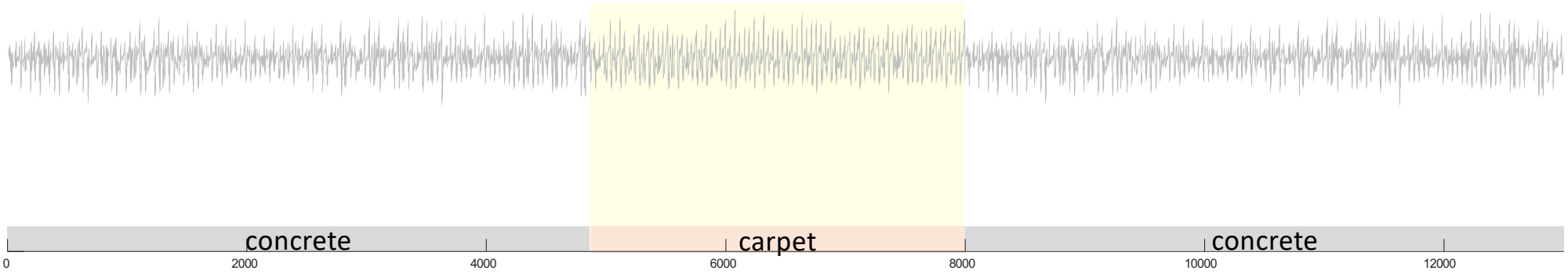
```
>> dist_profile = real(MASS_V2(ApolloYSUB,query));
>> plot(dist_profile)
```




query
ApolloYSUB
Best match
Second Best match
0
200
400
600
800
1000
10
5
1.26
0
distance profile
0
100
200
300
400
500
600
700
800
900
1000

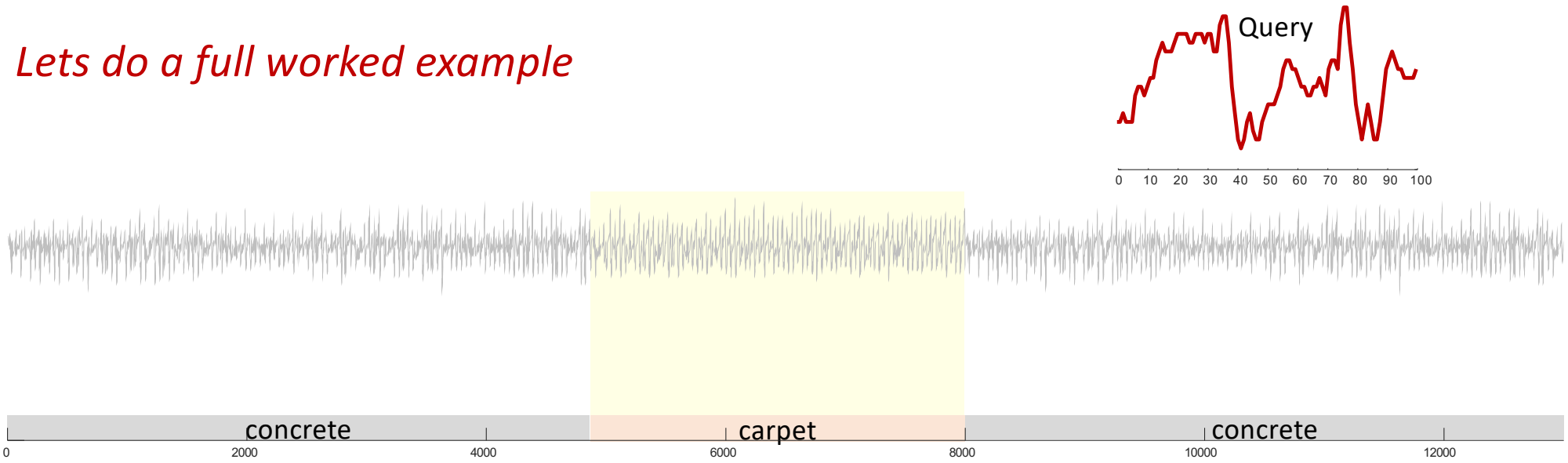

query
ApolloYSUB
Best match
Second Best match
0
200
400
600
800
1000
10
5
1.26
0
distance profile
In general, the absolute numbers have no real meaning. If you wanted more interpretable numbers, you could convert to Pearson's correlation
0
100
200
300
400
500
600
700
800
900
1000

Lets do a full worked example



I have this dataset, measuring the motion of a dog walking in my lab, first on concrete, then carpet, then concrete…

Lets do a full worked example

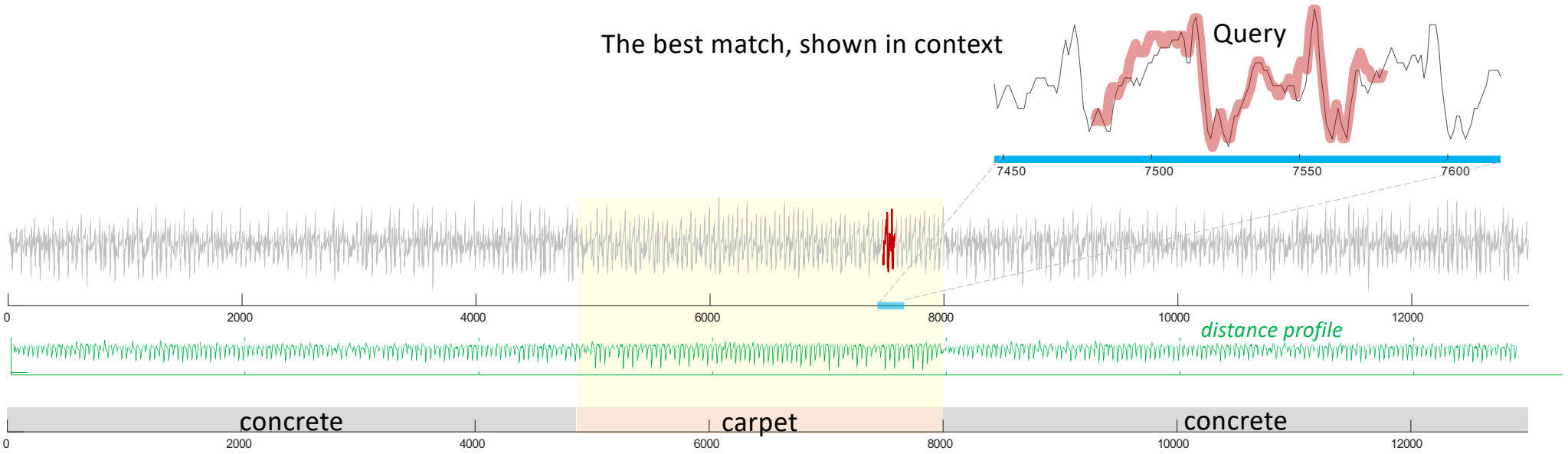


I have this dataset, measuring the motion of a dog walking in my lab, first on concrete, then carpet, then concrete…

I also have a query, for the same dog walking, on a different day.

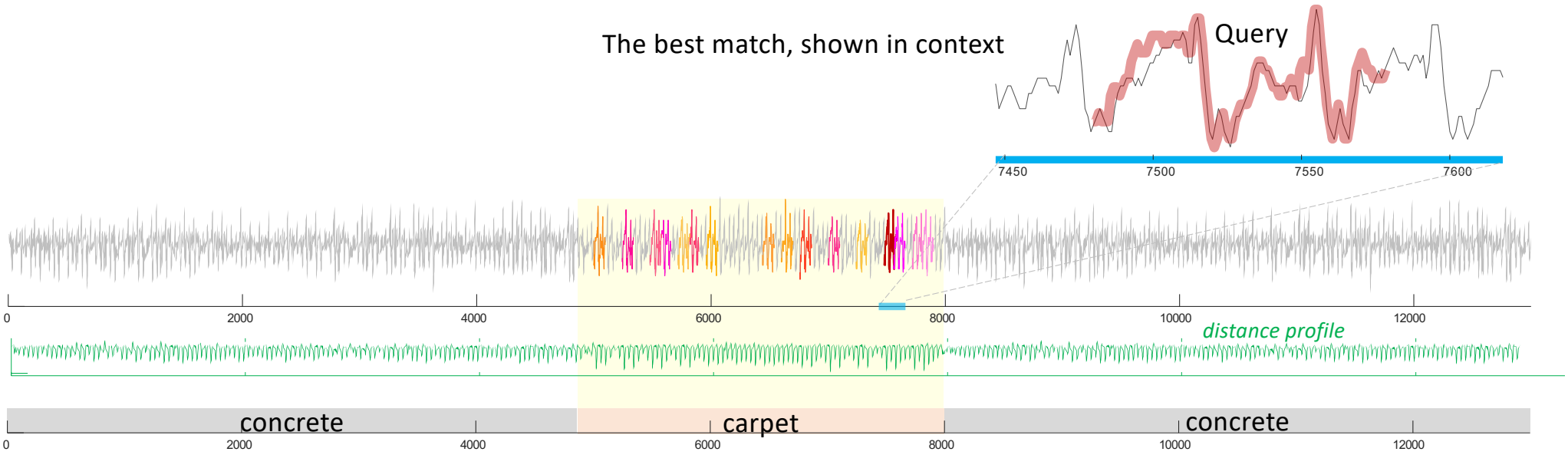
Do you think the query is from when the dog was walking on carpet or concrete or something else…

The best match, shown in context



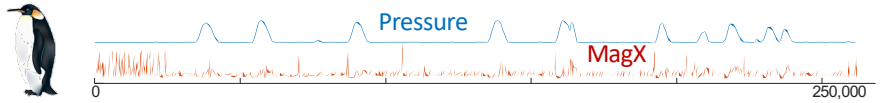
This task is trivial with MASS code…

```
>> dist = MASS(dog ,carpet_query );          % compute a distance profile
>> [val loc] = min(dist);                    % find location of match
>> disp(['The best matching subsequence starts at ',num2str(loc)])
The best matching subsequence starts at 7479
```



Below we plot the *16* best matches. Note that they all occur during the carpet walking period. This entire process takes about 1/10,000th of a second.

Note that this example is moving beyond nearest neighbor search, and is really performing *semantic segmentation* into regimes..

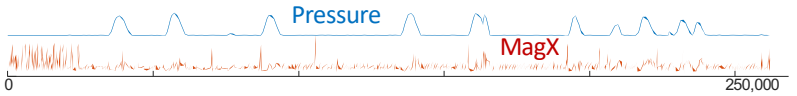


OK, we have seen a one-dimensional query is easy, but suppose I have multi dimensional time series?

Easy! The key insight is that the z-normalized time series is effectively unitless, so we can do each dimension individually, and just add them, to make a multi-dimensional distance profile.

Let's do a quick worked example…

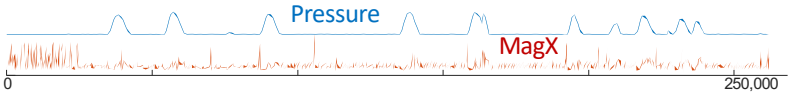
Multidimensional Nearest Neighbor (distance profile)



I have 262,144 data points that record a penguin's orientation (MagX) and the water/air pressure as he hunts for fish.

Question: Does he ever change his bearing leftwards as he reaches the apex of his dive?

Multidimensional Nearest Neighbor (distance profile)



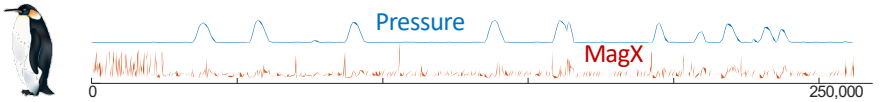
I have 262,144 data points that record a penguin's orientation (MagX) and the water/air pressure as he hunts for fish.

Question: Does he ever change his bearing leftwards as he reaches the apex of his dive.

This is easy to describe as a multidimensional search. The apex of a dive is just an approximately parabolic shape. I can create this with `query_pressure = zscore([[-500:500].^2]*-1)';` it looks like this 

I can create *bearing leftwards* with a straight rising line, like this `query_MagX = zscore([[-500:500]])';` It looks like this 

Multidimensional Nearest Neighbor (distance profile)



I have 262,144 data points that record a penguin's orientation (MagX) and the water/air pressure as he hunts for fish.

Question: Does he ever change his bearing leftwards as he reaches the apex of his dive.

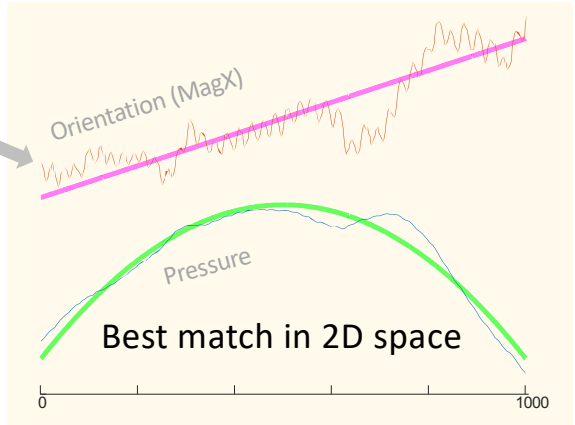
This is easy to describe as a multidimensional search. The apex of a dive is just an approximately parabolic shape. I can create this with `query_pressure = zscore([[-500:500].^2]*-1)';` it looks like this

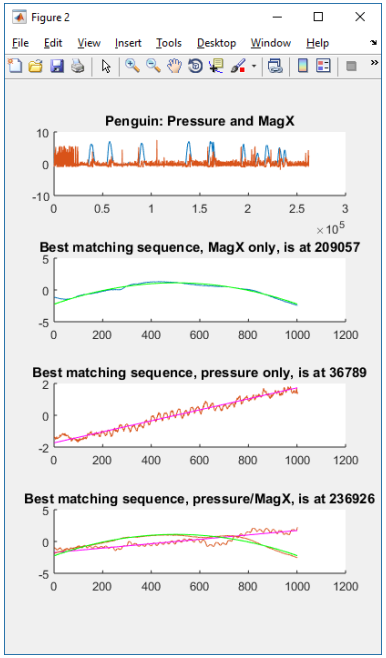
I can create *bearing leftwards* with a straight rising line, like this `query_MagX = zscore([[-500:500]])';` It looks like this

We have seen above how to search for a 1D pattern. For this 2D case, all we have to do is add the two distance profiles together, before we find the minimum value. Note that the best match location in 2D is different to either of the 1D queries.

```
load penguintest.mat
figure;, hold on;

  query_pressure = zscore([[-500:500].^2]*-1)';
  dist_p = MASS_V2(penguintest(:,1),query_pressure);
  query_MagX = zscore([[-500:500]])';
  dist_m = MASS_V2(penguintest(:,2),query_MagX);
  [val,loc] = min([dist_m + dist_p]);          % find best match location in 2D
  plot(zscore(penguintest(loc:loc+length(query_MagX),2)),'color',[0.85 0.32 0.09])
  plot(zscore(query_MagX),'m')
  plot(zscore(penguintest(loc:loc+length(query_pressure),1)),'b')
  plot(zscore(query_pressure),'g')
  title(['Best matching sequence, pressure/MagX, is at ', num2str(loc)])
```

What are the periodic bumps? They are wingstokes as the bird "flies" underwater





Mini-Review: The *distance profile*

- This is a simple idea called: query-by-content/similarity-search/nearest-neighbor search etc.
- You can use the distance profile to find the K-nearest neighbors to any query.
- This is an *incredibly* powerful and useful tool, limited only by your imagination.
- It is by far, the most important subroutine in all of time series data mining.

--

- Suppose you have multidimensional data? You can just compute the individual distance profiles, sum them, then find the lowest values as the multidimensional nearest neighbor! (penguin example)

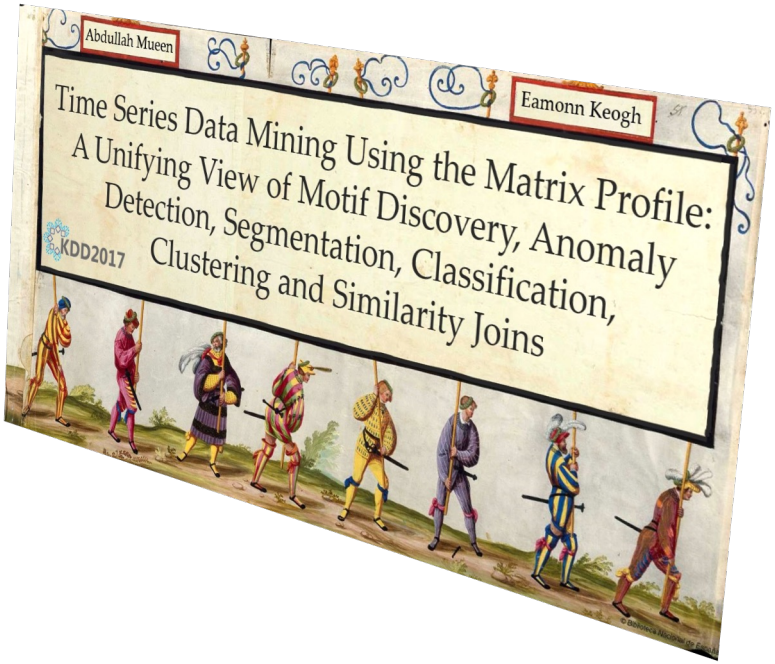
--

- The computation of the distance profile is *incredibly* fast.
- We can search the query in 24 hours of our bovine data (4,320,000 datapoints), in well under a second.
- This amazing speed is due to Mueen* (my former PhD student). His algorithm is called MASS*
- You can get MASS in most computer languages/platforms.

*https://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html

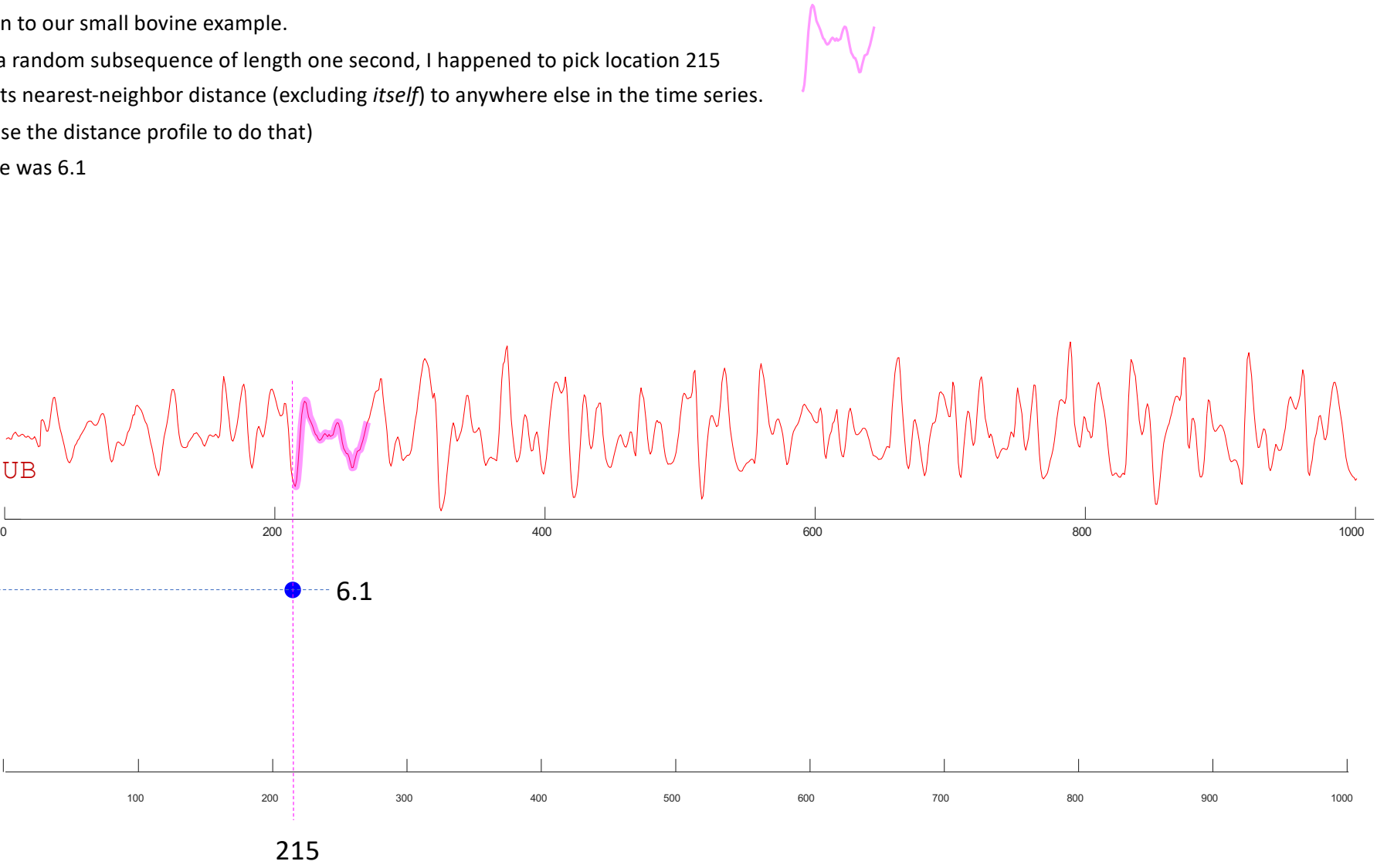
Preview: The *Matrix Profile*

- We are about to learn about the Matrix Profile*
- The Matrix Profile is the best idea in time series data mining in the last decade ;-)
- It is a stunningly simply idea.
- Our claim is that once you have the Matrix Profile, almost all time series problems are trivial.
- Longer tutorials are online

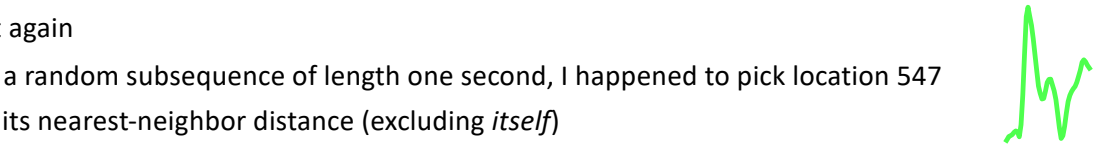


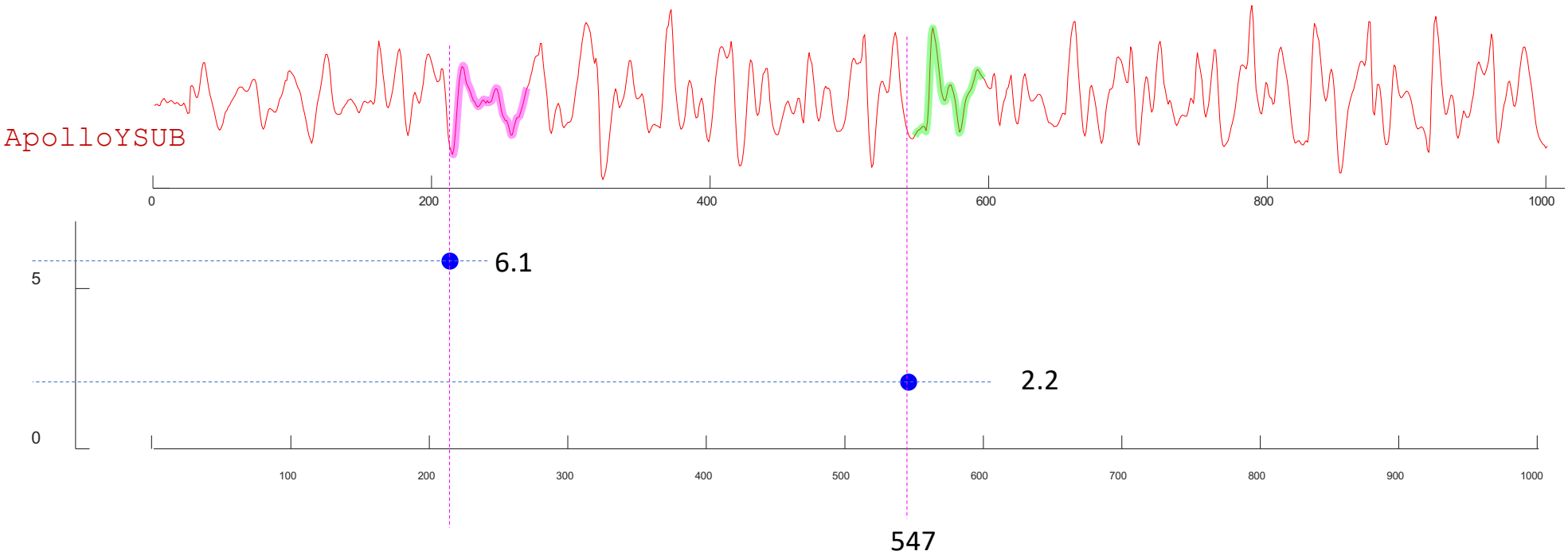
*https://www.cs.ucr.edu/~eamonn/MatrixProfile.html

- Let us return to our small bovine example.
- Let us pick a random subsequence of length one second, I happened to pick location 215
- Let us find its nearest-neighbor distance (excluding *itself*) to anywhere else in the time series.
- (we could use the distance profile to do that)
- The distance was 6.1

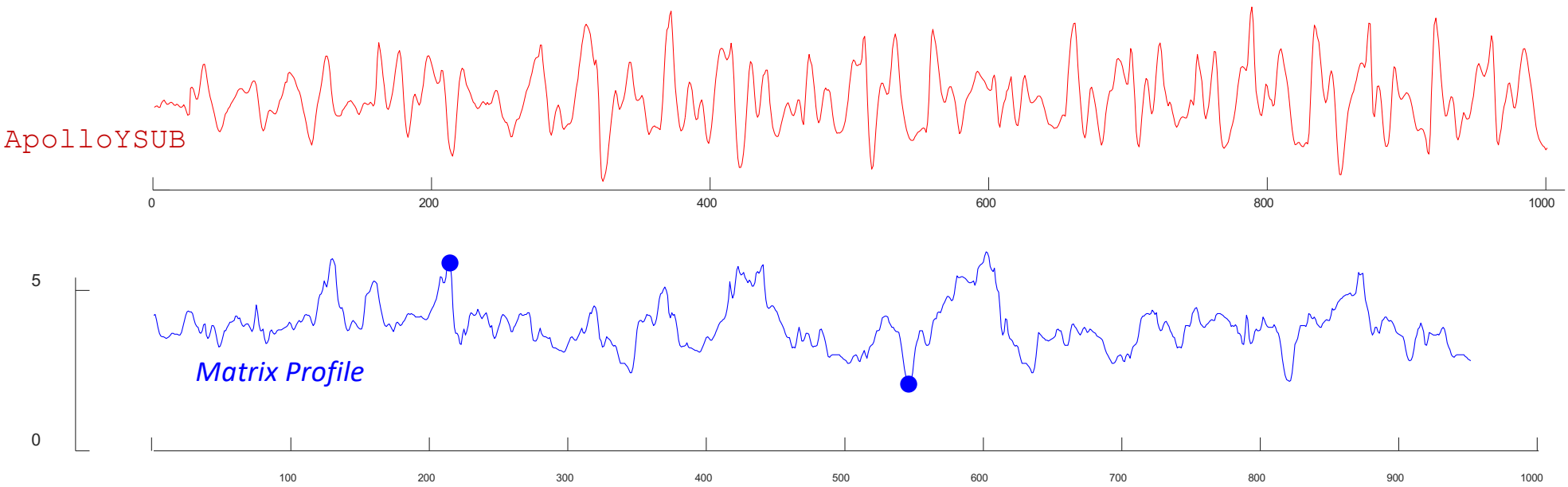


- Let do that again
- Let us pick a random subsequence of length one second, I happened to pick location 547
- Let us find its nearest-neighbor distance (excluding *itself*)
- (we could use the distance profile to do that)
- The distance was 2.2

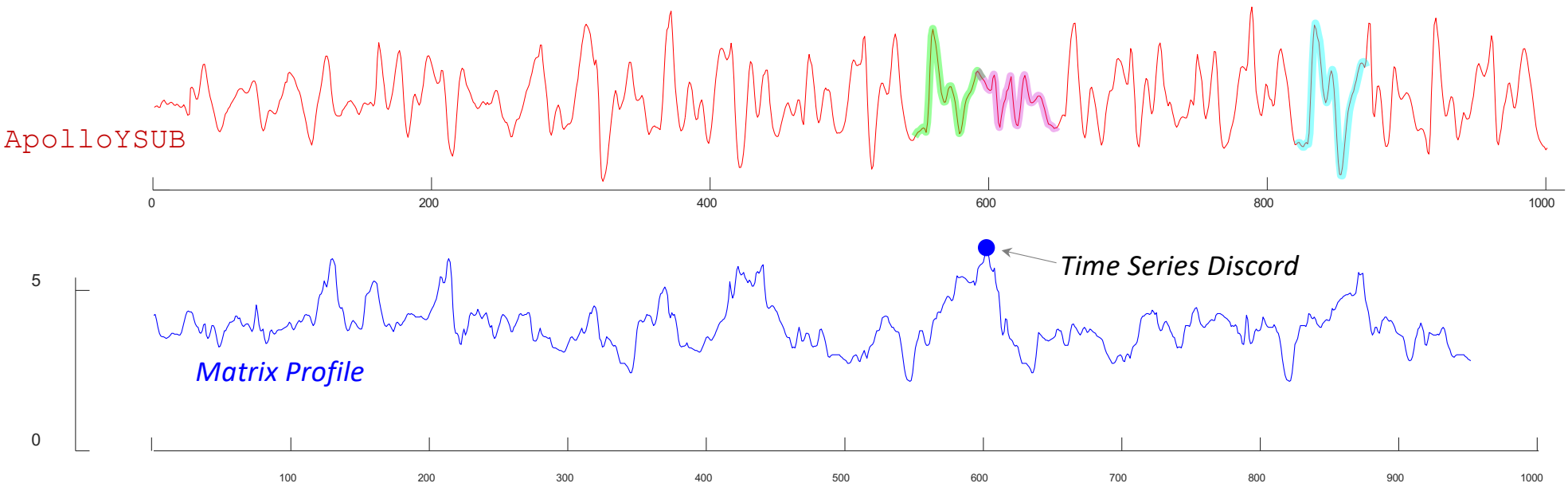




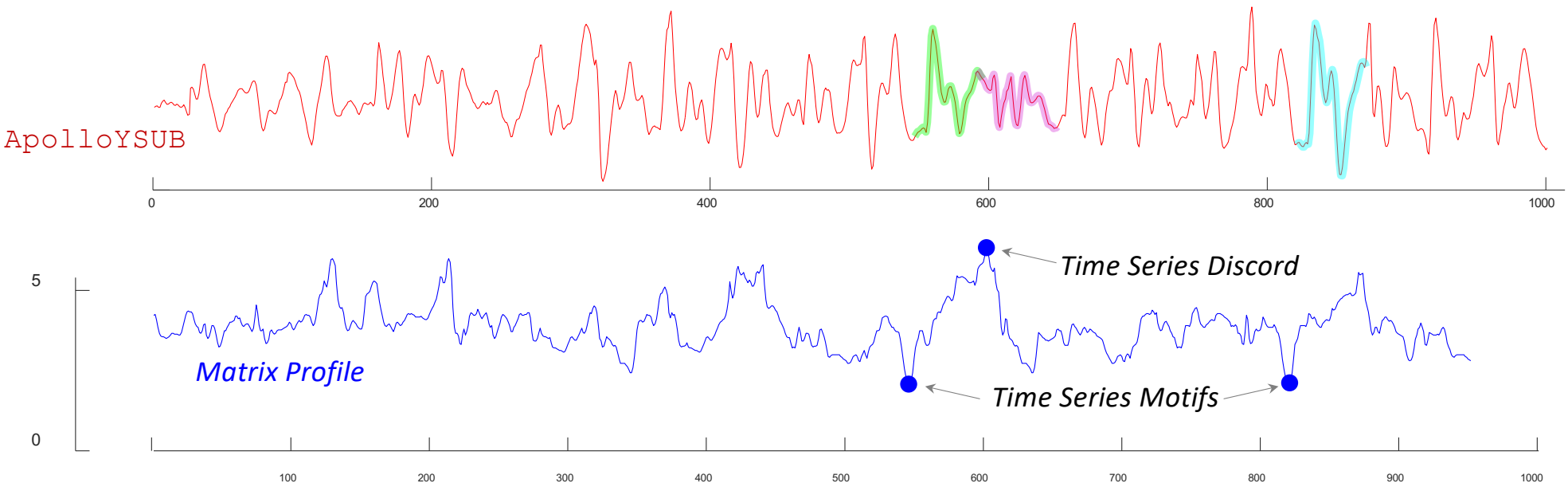
- Let us do this for every location!
- The resulting curve is called the *Matrix Profile*



- There are some parts of the Matrix Profile that have special names
- The highest location is called the *Time Series Discord*

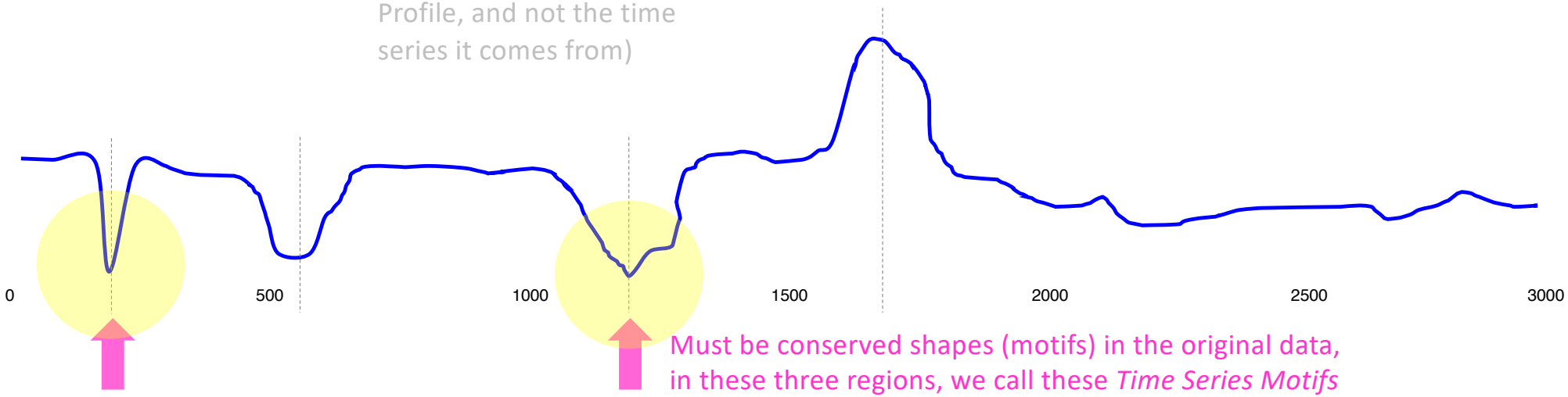


- There are some parts of the Matrix Profile that have special names
- The highest location is called the *Time Series Discord*
- The lowest locations (there will always be a tie) is called the *Time Series Motif* pair



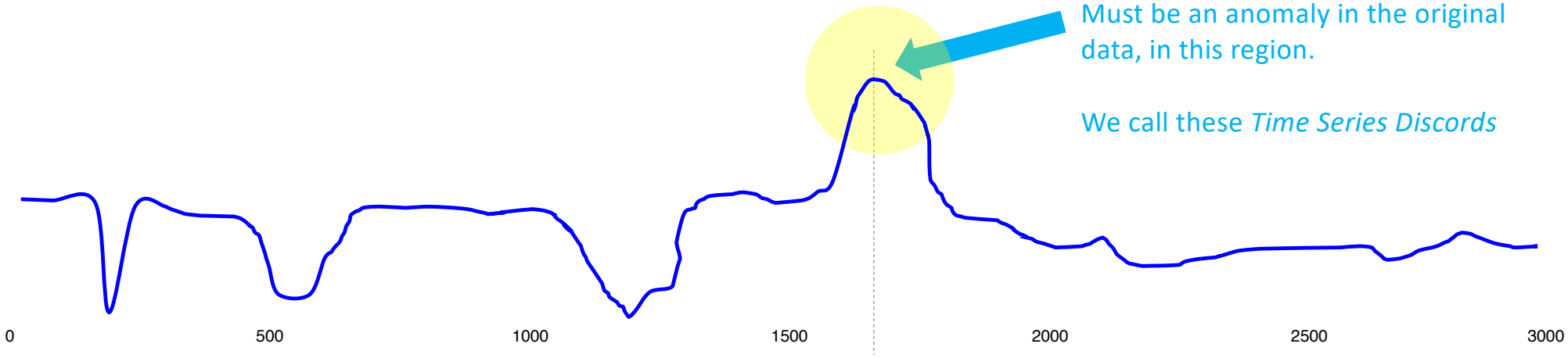
Reading a Matrix Profile

Where you see relatively low values, you know that the subsequence in the original time series must have (at least one) relatively similar subsequence elsewhere in the data (such regions are “motifs” or reoccurring patterns)



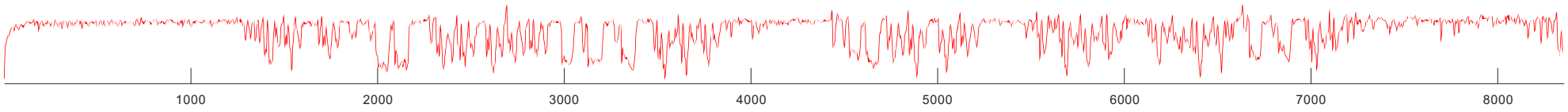
Reading the Matrix Profile

Where you see relatively high values, you know that the subsequence in the original time series must be unique in its shape (such areas are “discords” or anomalies).



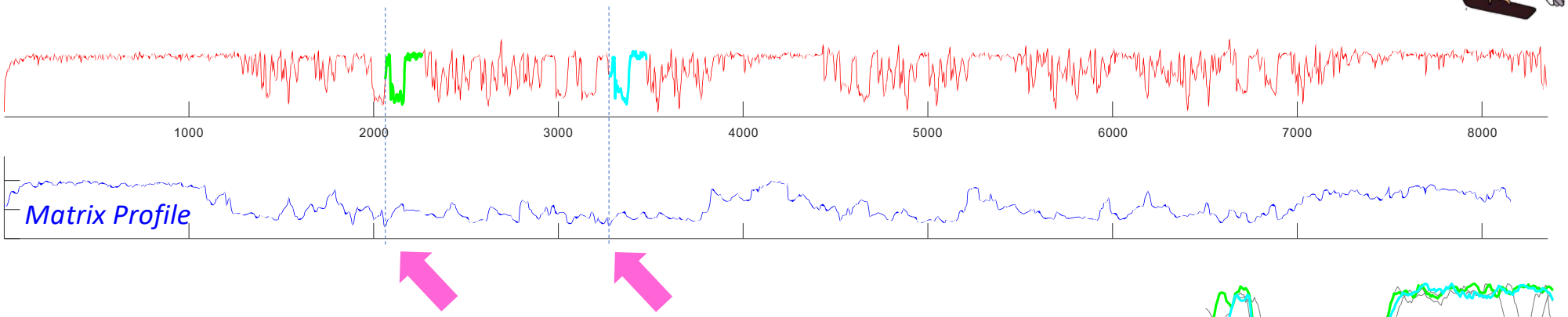
Zebra Finch

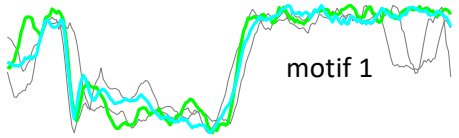
(Zebra Finch Vocalizations in MFCC, 100 day old male)

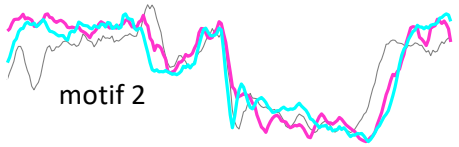


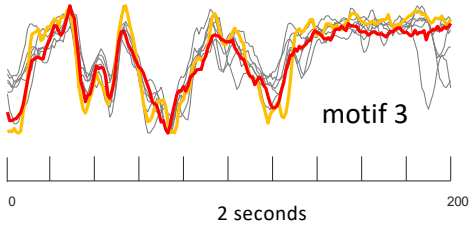
Can you see any conserved behavior here?

Zebra Finch









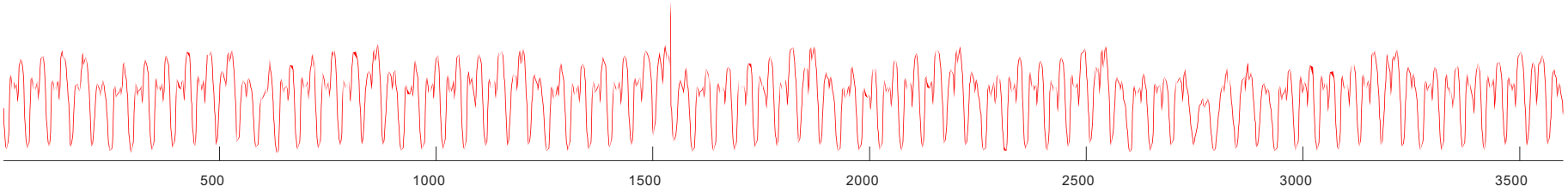
Motif discovery can often surprise you.

While it is clear that this time series is not random, we did not expect the motifs to be so well conserved or repeated so many times. There is evidence of a *vocabulary*, and maybe even a *grammar*…

Taxi Example: Part I

Below is the hourly average of the number of NYC taxi passengers over 75 days in Fall of 2014.

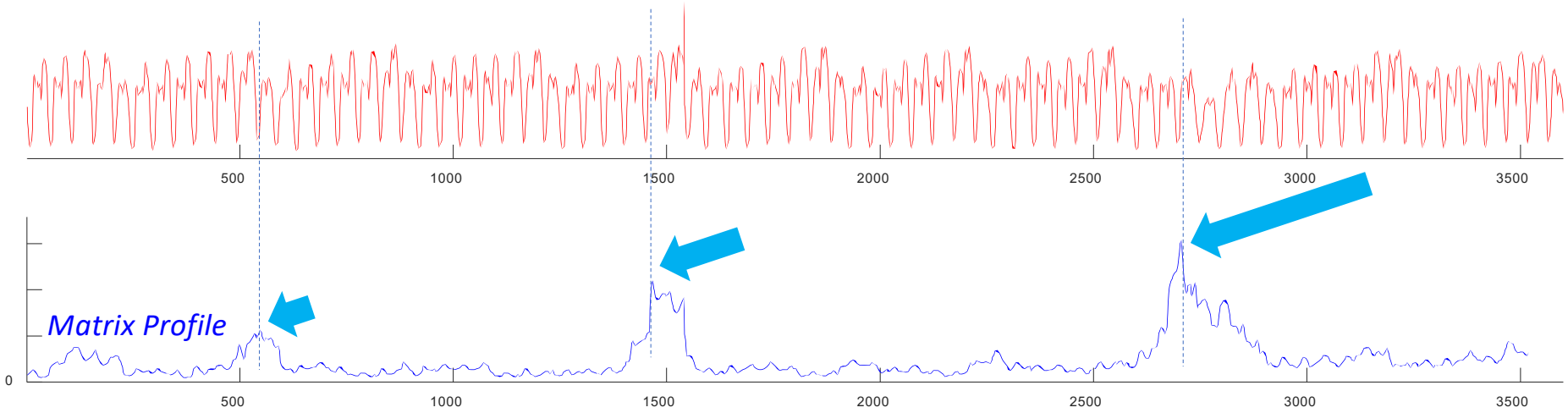
Lets compute the Matrix Profile for it, we choose a subsequence length corresponding to two days…. (next slide)



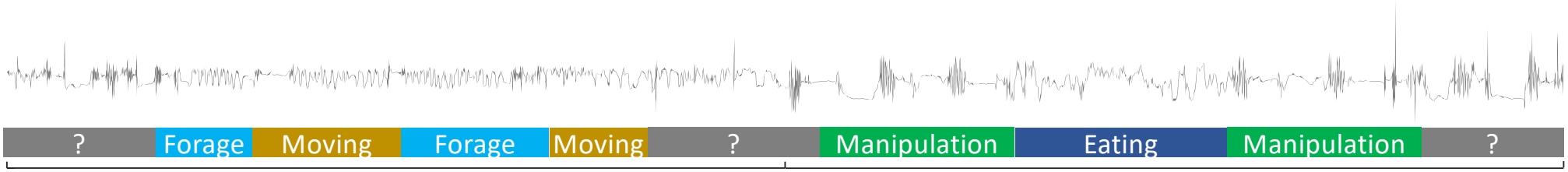
[a] http://futuredata.stanford.edu/ASAP/extended.pdf

Taxi Example: Part II

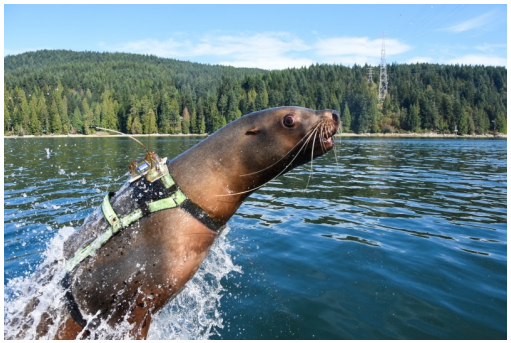
- The highest value corresponds to Thanksgiving
- We find a secondary peak around Nov 6th, what could it be? Daylight Saving Time! The clock going backwards one hour, gives an *apparent* doubling of taxi load.
- We find a tertiary peak around Oct 13th, what could it be? Columbus Day! Columbus Day is largely ignored in much of America, but still a big deal in NY, with its large Italian American community.

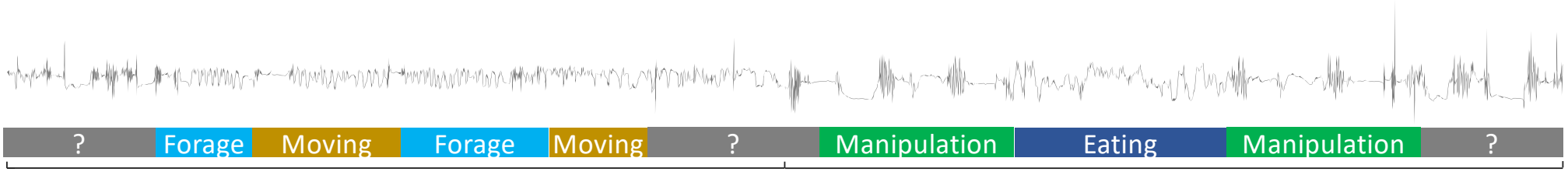


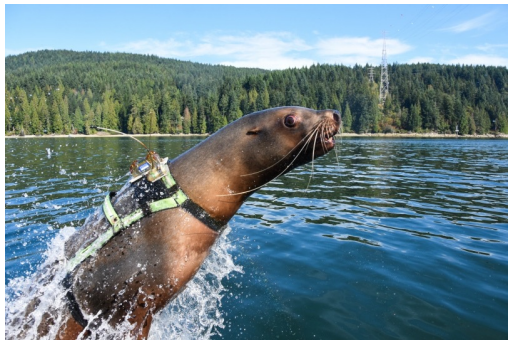

A quick example of the amazing utility of motif discovery (next 6 slides)
Australian Fur Seal (*Arctocephalus pusillus*)



This is an interesting dataset; can we find motifs in it?







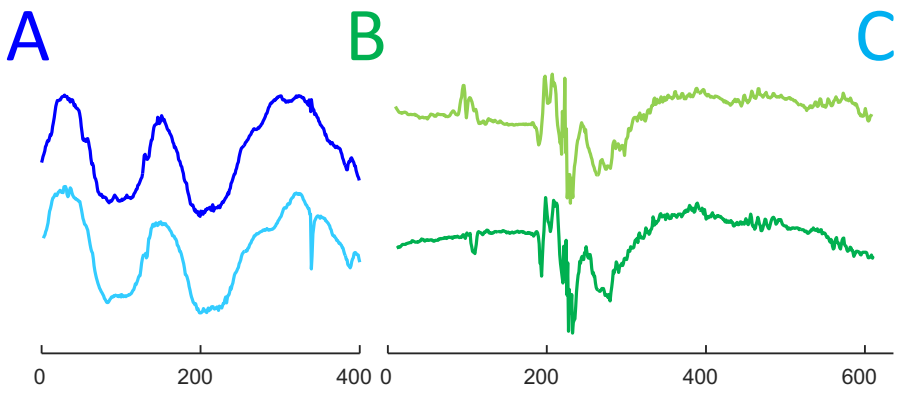
We can find *many* interesting motifs in this data.

Suppose we label them A, B, C , D etc.

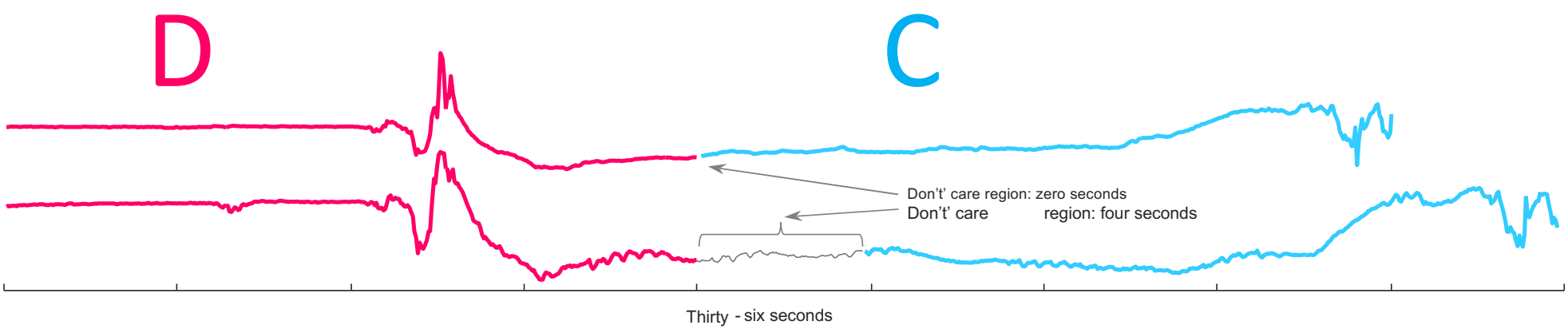
We can then ask the question, is there any patterns in the occurrence of these motifs?

In fact, there is, when we see D, we almost always see C within a few seconds…

If D Then C



What does this mean?



Foraging at uniform
speed, then a…

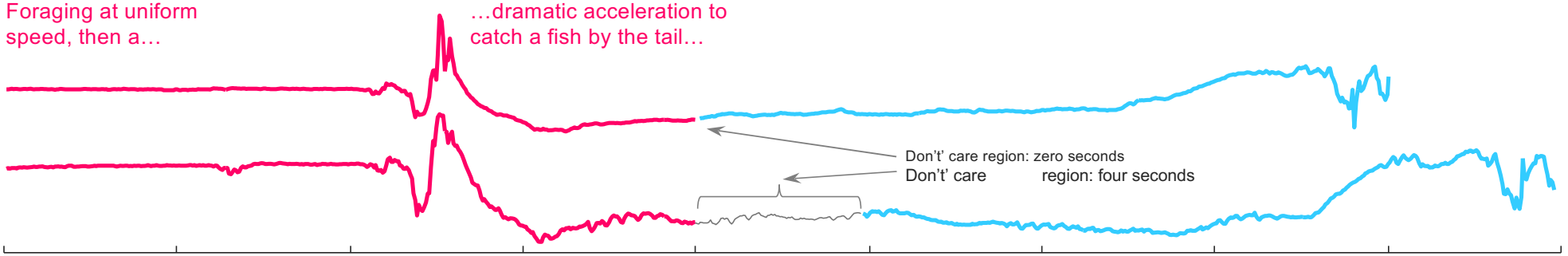
Don’t’ care region: zero seconds

Don’t’ care region: four seconds

Thirty - six seconds

Foraging at uniform speed, then a…

…dramatic acceleration to catch a fish by the tail…



Foraging at uniform speed, then a…

…dramatic acceleration to catch a fish by the tail…

…coast to surface…

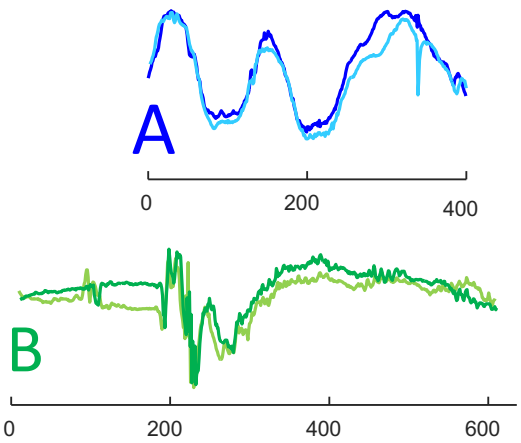
…to flip fish for head first swallowing

Don’t’ care region: zero seconds

Don’t’ care region: four seconds

Thirty - six seconds

This idea is very general

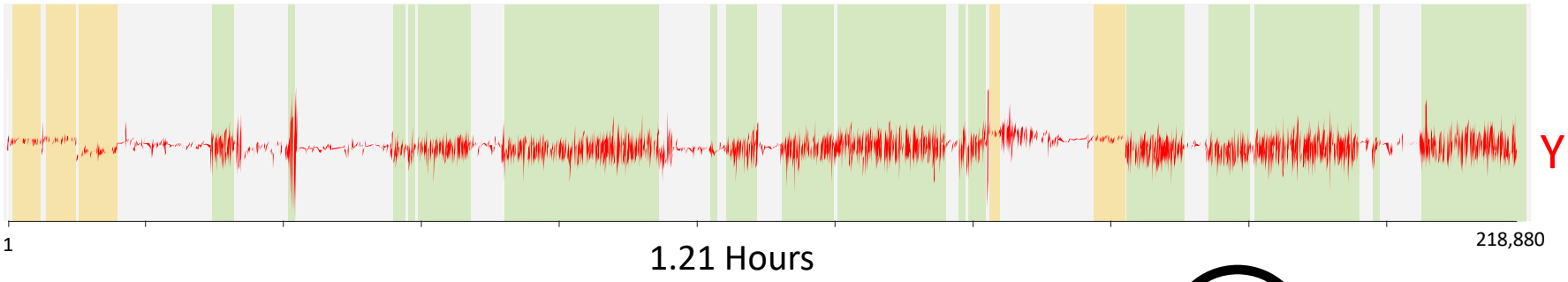


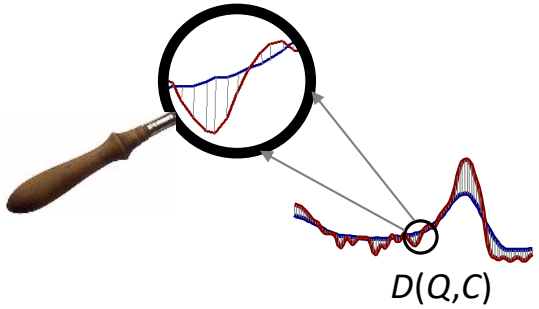
Find motifs, label them A, B, C , D , E etc.

Now you can ask lots of interesting questions…

- We see about 10 B's per hour in males, but only 1 or 2 B's per hour in females, why?
- If I give my cow flax instead of corn feed, does it change the frequencies of any of the motifs?
- Which motifs (if any) are associated with my more aggressive bulls?
- Are any of the motifs dependent on the outside temperature?
- It seems like motif D is much more common in good milkers. So, let me change these two things under my control, to see if it increases the frequency of the D motif.

1: grazing
2: ruminating
3: resting/other





Let us return to Apollo

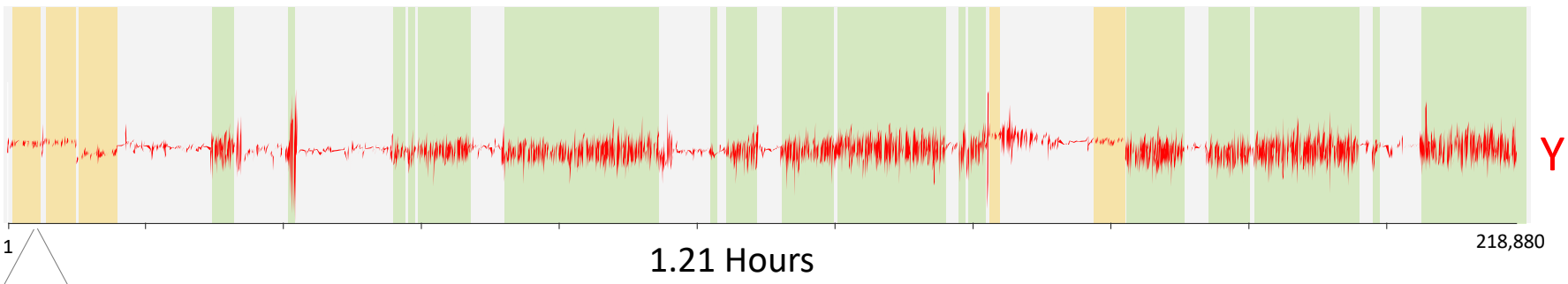
Let's compute the Matrix Profile for this dataset

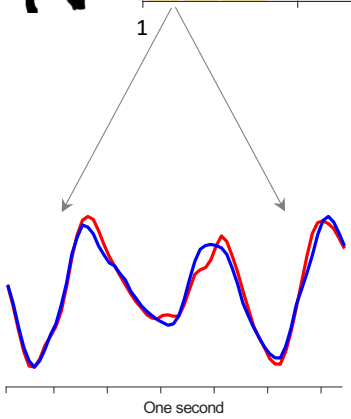
To do so, we have to compute the Euclidean Distance between the approximately 24 billion possible pairwise combinations of subsequences.

Because of an amazing algorithm called SCRIMP++, we can do this in seconds!

There are other algorithms: STAMP, STAMPI, STOMP, SWAMP, DAMP, SCAMP, GPU-STOMP, TranSCRIMP, TranSCAMP and TranSCAMPfpga…

1: grazing
2: ruminating
3: resting/other





Here is the best motif.

Note that both occurrences happen during **ruminating**

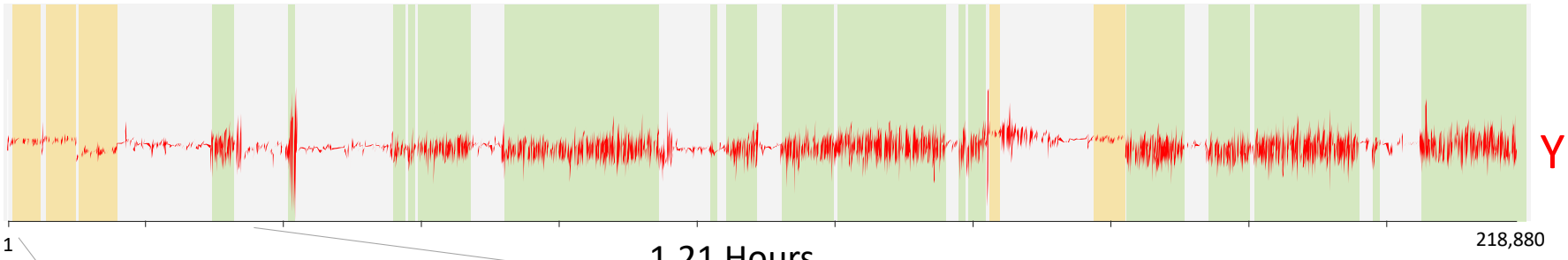
They are 1.29 units apart

This suggest that we may have found a *decision rule*

```
If you see a subsequence(X)such that
 D(X, ∿∿ ) < ( 3 * 1.29 )
Then Print('Bovine is ruminating!')
```

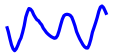
Here 3 is a magic constant that could be tuned, to change precision/recall

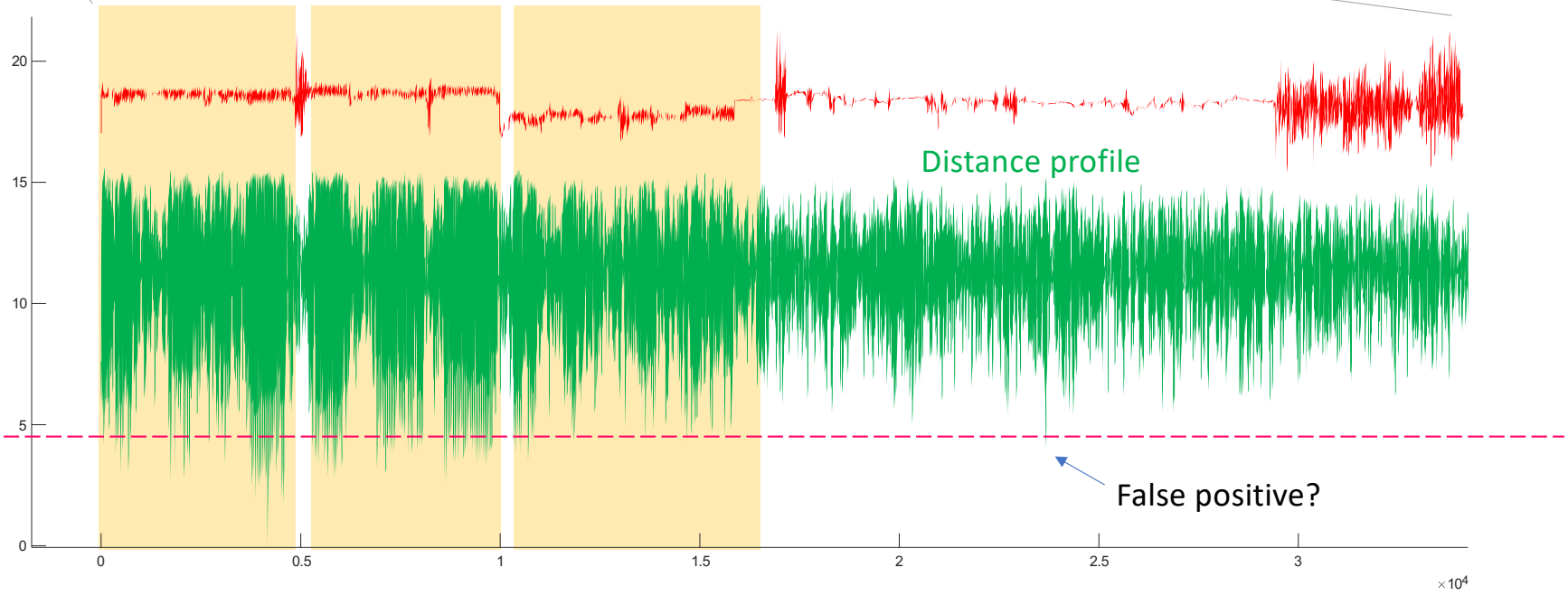
1: grazing
2: ruminating
3: resting/other



1.21 Hours

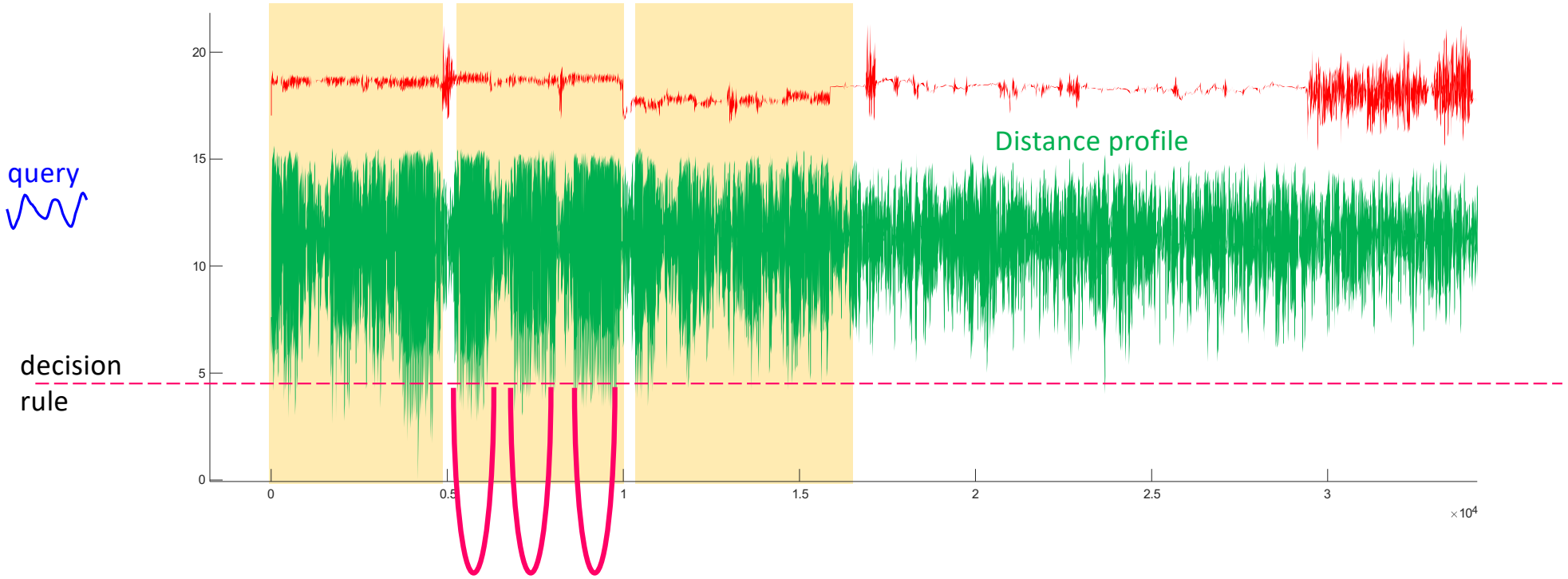
query





decision rule

False positive?



Quick aside

This might be a coincidence, but I see evidence of a *meta-pattern* here. There appears to be three “bursts” of fidelity to the template pattern.

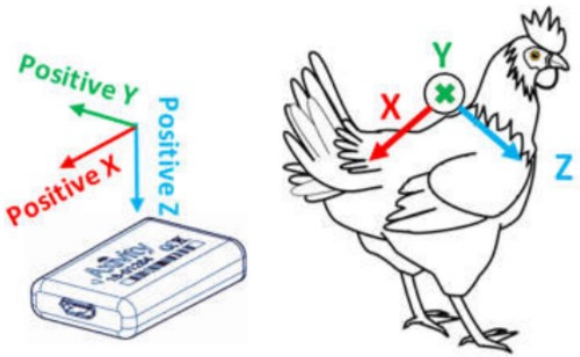
Mini Review I

- What I have shown you is *amazing* if you think about it.
  - Knowing *nothing* about bovine behavior…
  - Using less than eight lines of code in total.
  - Using one minute of brain power, and a few seconds of CPU power.
  - I built a tool to correctly annotate complex behavior in a bovine.

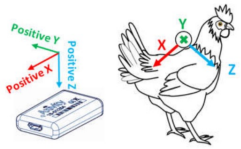
Mini Review I

- What I have shown you is *amazing* if you think about it.
  - Knowing *nothing* about bovine behavior…
  - Using less than eight lines of code in total.
  - Using one minute of brain power, and a few seconds of CPU power.
  - I built a tool to correctly annotate complex behavior in a bovine.

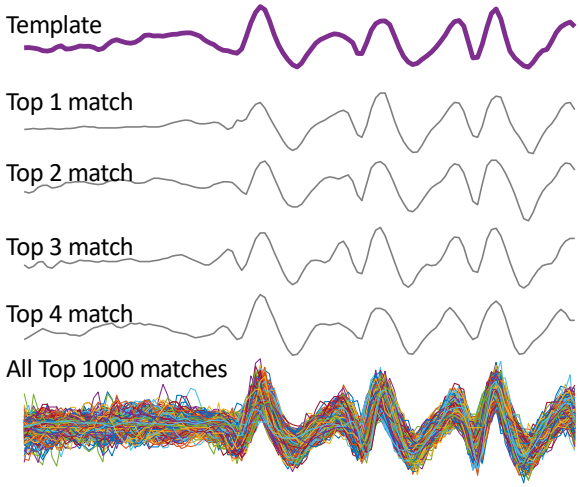
- Was this a fluke?
- Let quickly do it again!
- This time with chickens.
- This time I have *four years* of chicken data!
- (many different chickens in parallel, over months)



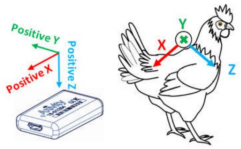
Mini Review II



- Here is the template that motif discovery found
- It was associated with *Dustbathing*
- I used it to search a 12,679,054,727 datapoint (four full years) archive of chicken behavior for the one thousand best matches. i.e. the 1,000 nearest neighbors

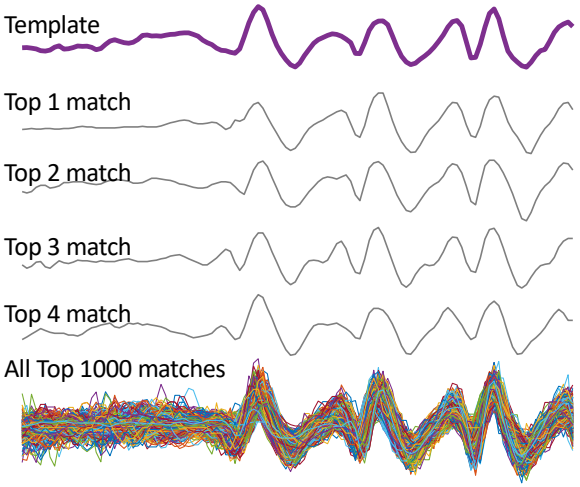


Mini Review II



- Here is the template that motif discovery found
- It was associated with *Dustbathing*
- I used it to search a 12,679,054,727 datapoint (four full years) archive of chicken behavior for the one thousand best matches. i.e. the 1,000 nearest neighbors

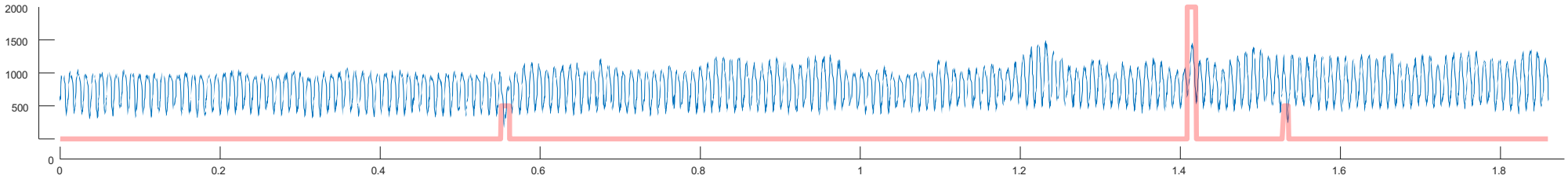
- It gets better.
- My data is annotated with two types of chicken `has-mites|no-mites`.
- I can see that Dustbathing is more common in the `has-mites` class, so have learned that chickens with mites will dustbath more.



A Strong Claim

- The Matrix Profile is the SOTA Time Series Anomaly Detector.
- This is a surprising claim, as there are about 100 papers a year published on this topic (including lots of deep learning methods)
- The Matrix Profile is simpler (one parameter) faster (by orders of magnitude) and on most bake-offs works much better.
- Lets see a quick example..

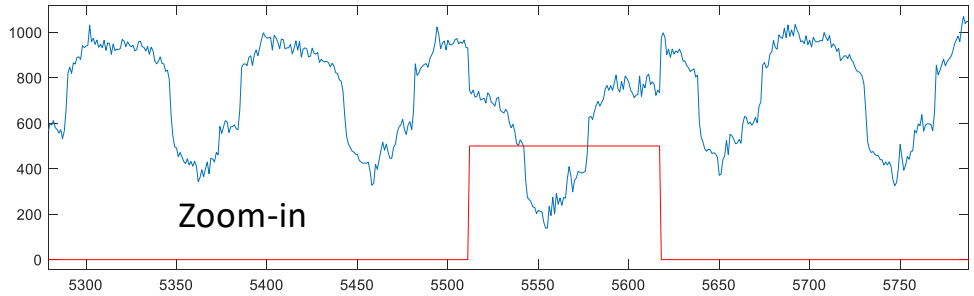
Dataset: Consumer electrical demand in Portugal

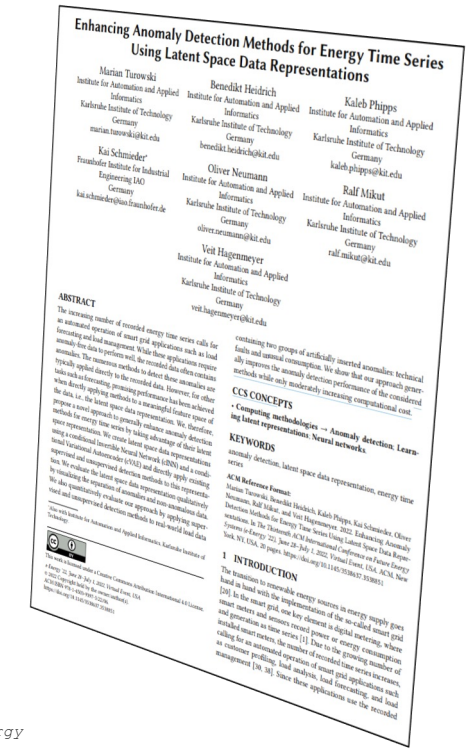


2/6/2014

8/19/2014

Anomaly Types: Technical faults in the metering infrastructure or unusual consumption.





Enhancing Anomaly Detection Methods for Energy Time Series Using Latent Space Data Representations

Marian Turowski, Benedikt Heidrich, Kaleb Phipps, Kai Schmieder, Oliver Neumann, Ralf Mikut, Veit Hagenmeyer

ABSTRACT

CCS CONCEPTS

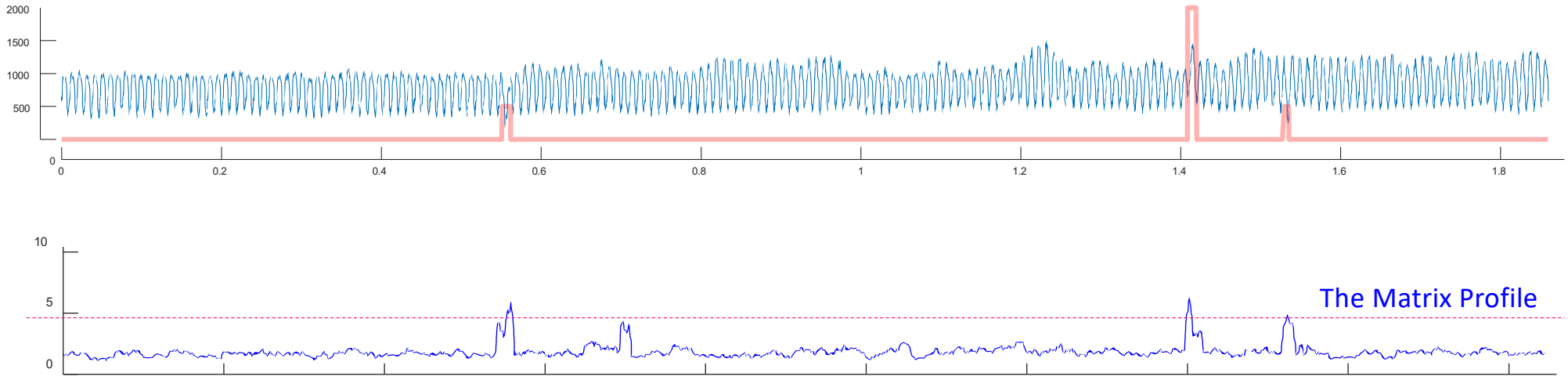
KEYWORDS

ACM Reference Format:

1 INTRODUCTION

Marian Turowski, Benedikt Heidrich, Kaleb Phipps, Kai Schmieder, Oliver Neumann, Ralf Mikut, and Veit Hagenmeyer. 2022. Enhancing anomaly detection methods for energy time series using latent space data representations. In Proceedings of the Thirteenth ACM International Conference on Future Energy Systems (e-Energy '22). Association for Computing Machinery, New York, NY, USA, 208–227. https://doi.org/10.1145/3538637.3538851

Dataset: Consumer electrical demand in Portugal



The Highly Desirable Properties of the Matrix Profile I

- It is **exact**: For motif discovery, discord discovery, time series joins etc., the Matrix Profile based methods provide no false positives or false dismissals.
- It is **simple** and **parameter-free**: In contrast, the more general algorithms in this space that typically require building and tuning spatial access methods and/or hash functions.
- It is **space efficient**: Matrix Profile construction algorithms requires an inconsequential space overhead, just linear in the time series length with a small constant factor, allowing massive datasets to be processed in main memory (for most data mining, *disk is death*).
- It allows **anytime algorithms**: While exact MP algorithms are extremely scalable, for extremely large datasets we can compute the Matrix Profile in an anytime fashion, allowing ultra-fast approximate solutions and real-time data interaction.
- It is **incrementally maintainable**: Having computed the Matrix Profile for a dataset, we can incrementally update it very efficiently. In many domains this means we can effectively maintain exact joins/motifs/discords on streaming data forever.

The Highly Desirable Properties of the Matrix Profile II

- It can **leverage hardware**: Matrix Profile construction is embarrassingly parallelizable, both on multicore processors, GPUs, distributed systems etc.
- It is **free of the curse of dimensionality**: That is to say, It has *time complexity that is constant in subsequence length*: This is a very unusual and desirable property; virtually all existing algorithms in the time series scale poorly as the subsequence length grows.
- It can be constructed in **deterministic time**: Almost all algorithms for time series data mining can take radically different times to finish on two (even slightly) different datasets. In contrast, given *only* the length of the time series, we can precisely predict in advance how long it will take to compute the Matrix Profile. (this allows resource planning)
- It can handle **missing data**: Even in the presence of missing data, we can provide answers which are guaranteed to have no false negatives.
- Finally, and subjectively: **Simplicity and Intuitiveness:** Seeing the world through the MP lens often invites/suggests simple and elegant solutions.

Mini Review III

- With just the *Distance Profile* and the *Matrix Profile*, you can solve many (most/all) problems in time series data analysis.
- Once installed on your machine, these are both one line of code!
- There is a large and growing community of Matrix Profile users, so these tools exist in most languages/platforms.

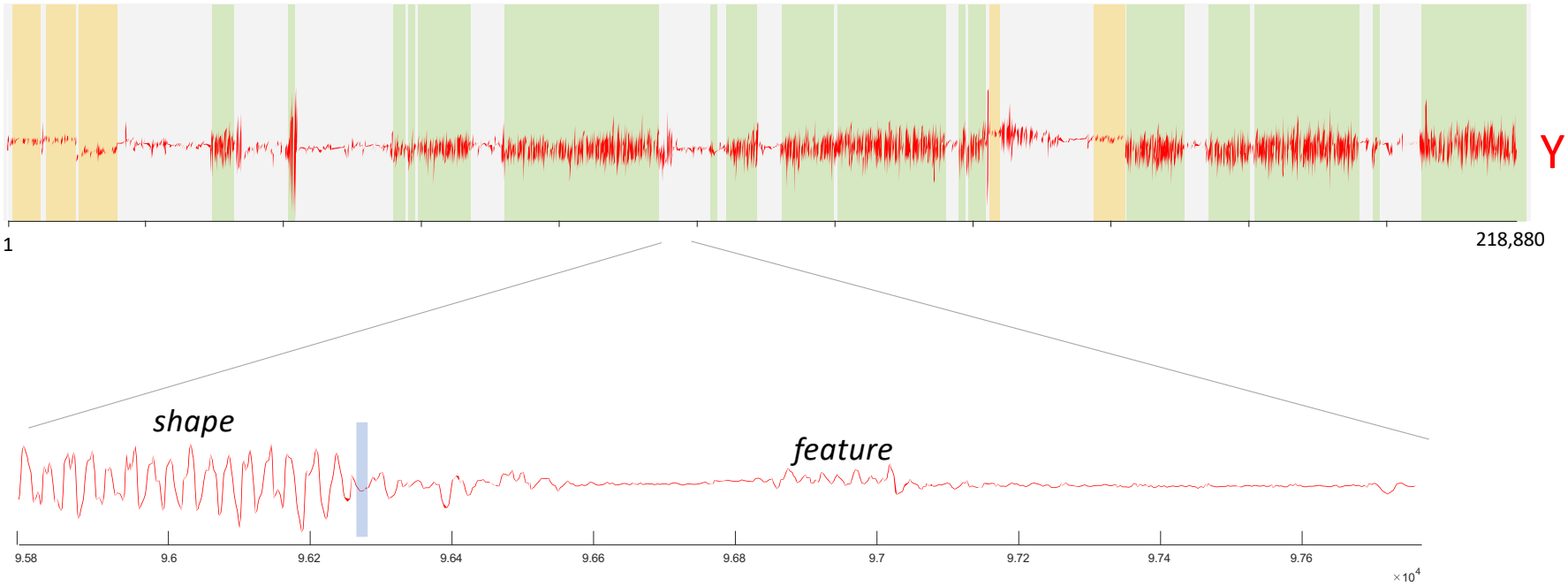
The *Matrix Profile* generalizes to multiple time series (i.e. joins) so you can ask questions that *compare* and contrast behaviors: (will not show this today)

- What patterns occur in *Males* **but not** *Females* (join *discord*)
- What patterns occur in *Hereford* **and** *Holstein* (join *motif*)
- What patterns occur before *Denuding* **but not** after *Denuding* (join *discord*)

The *Matrix Profile* generalizes to other useful primitives, *Chains*, *Novelets*, *Shapelets*, *Platos, Snippets, FLOSS*…. (will not show this today)

Switching Gears a Little

1: grazing
2: ruminating
3: resting/other



Bad news ;-(
Sometimes there are behaviors that are not captured well in *shape*.

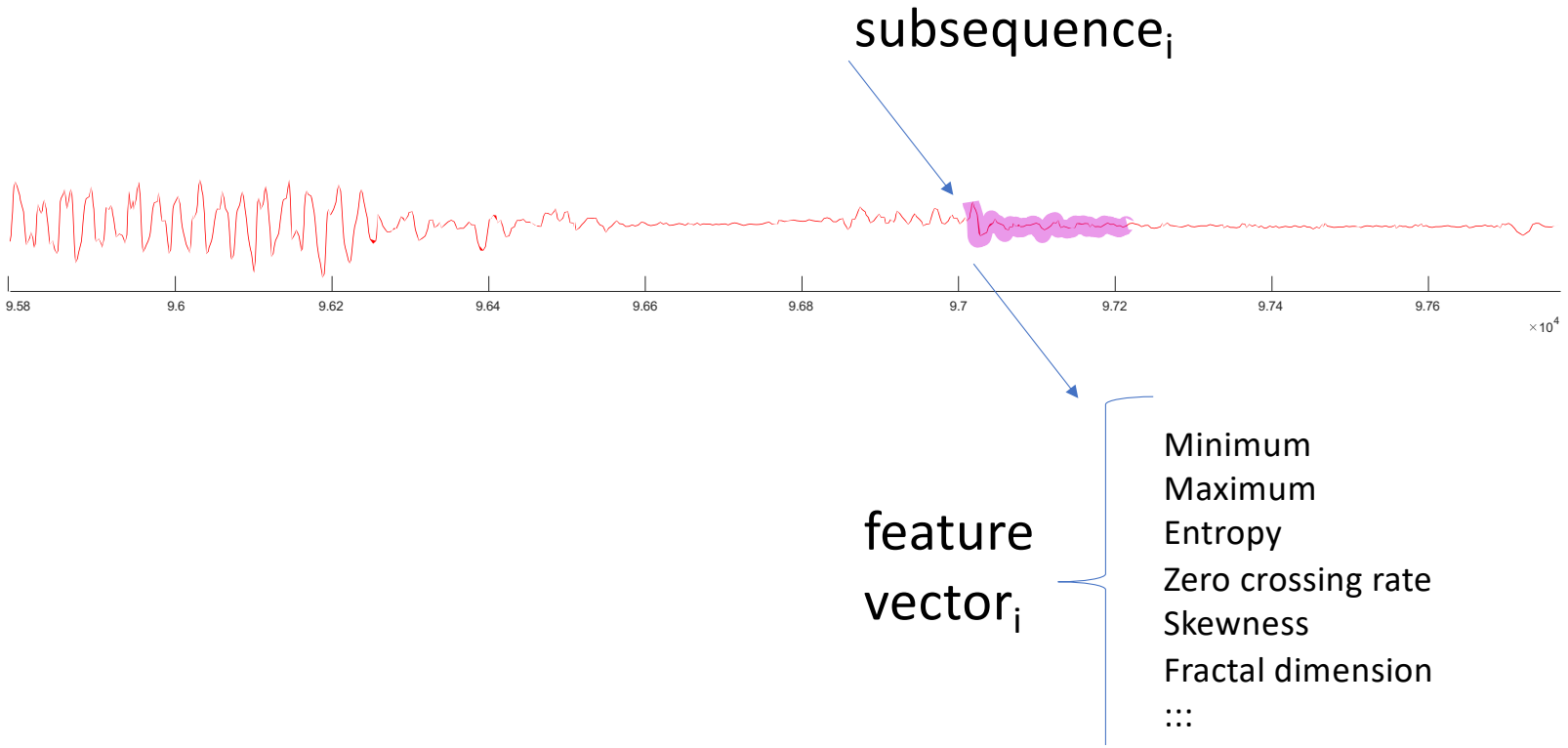
Think of human behaviors: walking/running/swimming/cycling all have characteristic *shapes*. *Dynamic Behaviors*

But there is no *shape* for reading/watchingTV/resting/sleeping etc.

Instead, we have to consider *features*.

Instead of pulling out subsequence *shapes*, lets pull out subsequences, and measure *features*…

We can then use the many algorithms that ingest *feature vectors*, nearest neighbor, decision trees, naïve Bayes,…



This begs the question, which features to use?

There are more than 9,000 suggested features for time series! [1]

Many of these features are redundant with each other, or just useless…

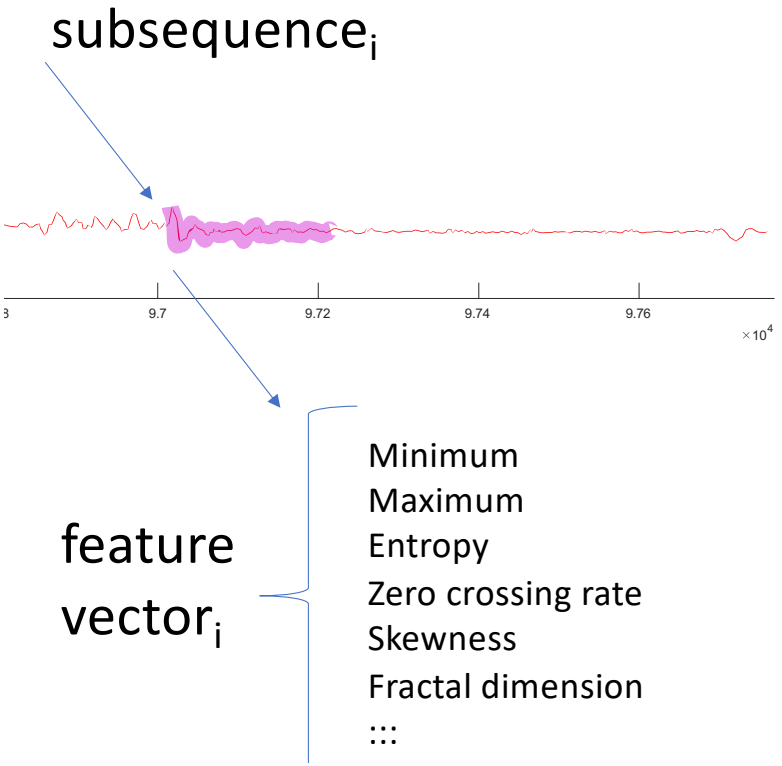
Jones and Fulcher searched through all these features to find a small subset that:

- Are mostly non-redundant
- Provide discrimination between semantically different classes in most real-world domains

The small set is called *Catch-22* [2]

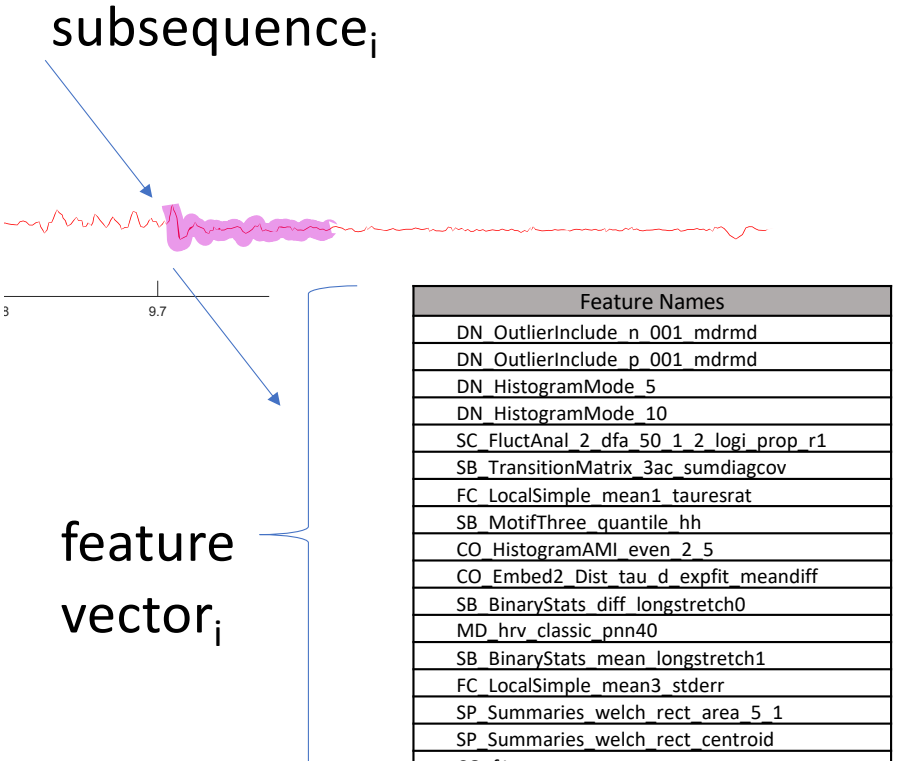
I think Catch-22 is one of the best ideas in time series data mining in the last decade..

1. Fulcher B.D., Little M.A., Jones N.S . Highly comparative time-series analysis: the empirical structure of time series and their methods. J. Roy. Soc. Interface 10, 20130048 (2013)
2. Lubba C, Sethi S, Knaute P, Schultz S, Fulcher B, Jones NS (2019) catch22: CAnonical Time-series CHaracteristics. Data Min Knowl Disc 33(6):1821–1852



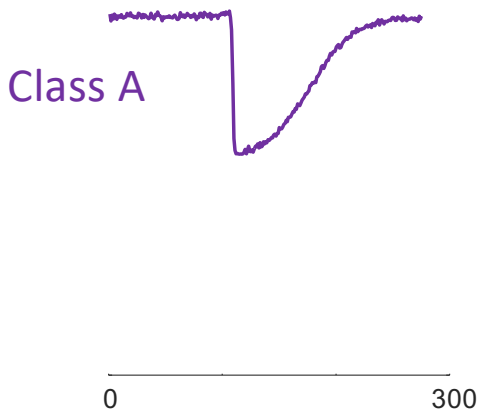
Unfortunately, the names of the Catch-22 features are poor, with no real mnemonic value.

1. Fulcher B.D., Little M.A., Jones N.S . Highly comparative time-series analysis: the empirical structure of time series and their methods. J. Roy. Soc. Interface 10, 20130048 (2013)
2. Lubba C, Sethi S, Knaute P, Schultz S, Fulcher B, Jones NS (2019) catch22: CAnonical Time-series CHaracteristics. Data Min Knowl Disc 33(6):1821–1852



Lets see how this works.

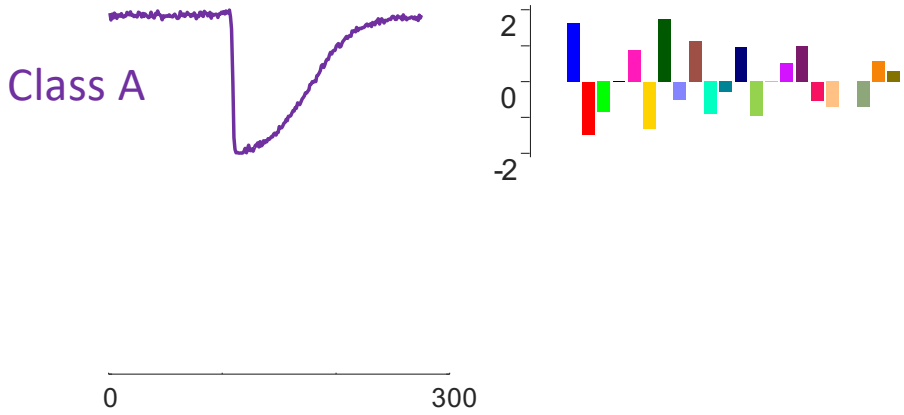
Lets take a time series subsequence from Class A

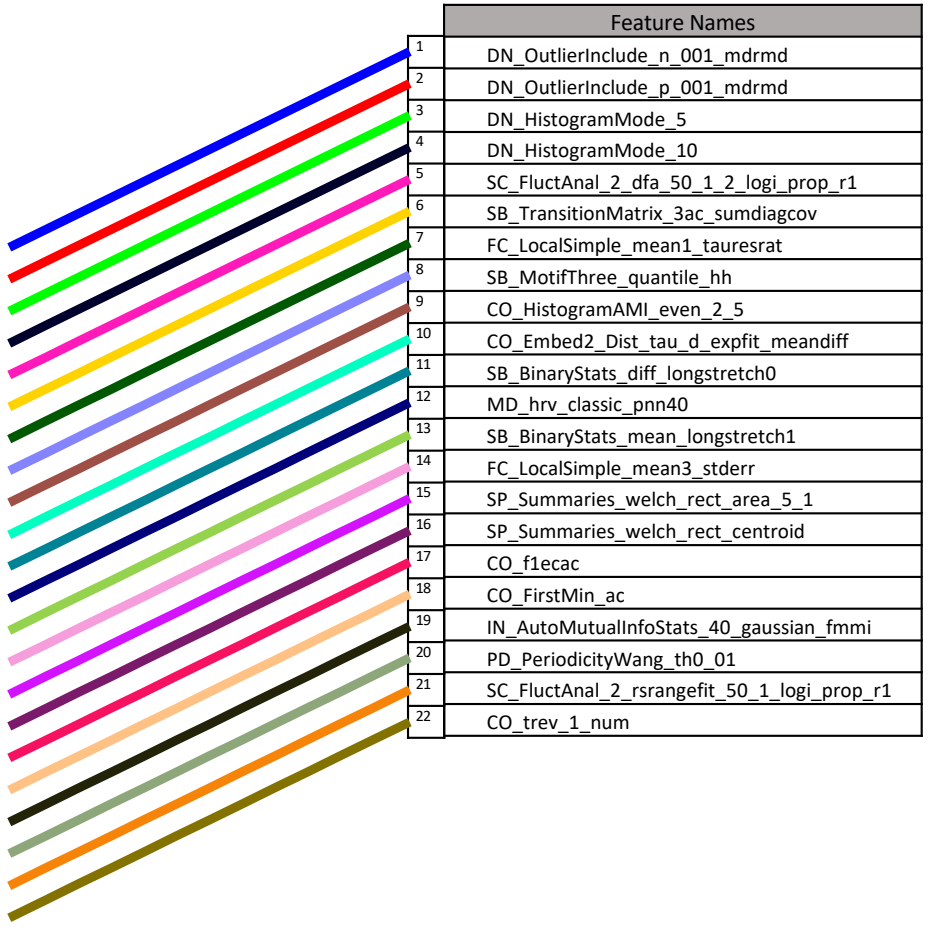


Lets see how this works.

Lets take a time series subsequence from Class A

Lets measure its 22 features, and summarize them with a color-coded bar chart…

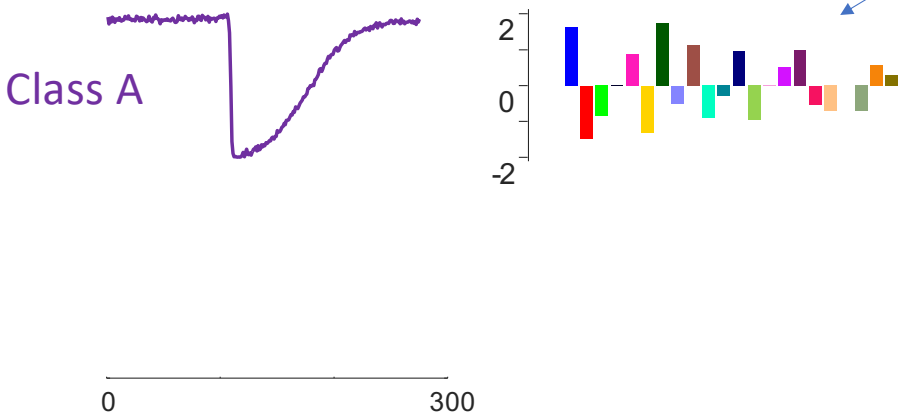




Lets see how this works.

Lets take a time series subsequence from Class A

Lets measure its 22 features, and summarize them with a color-coded bar chart…

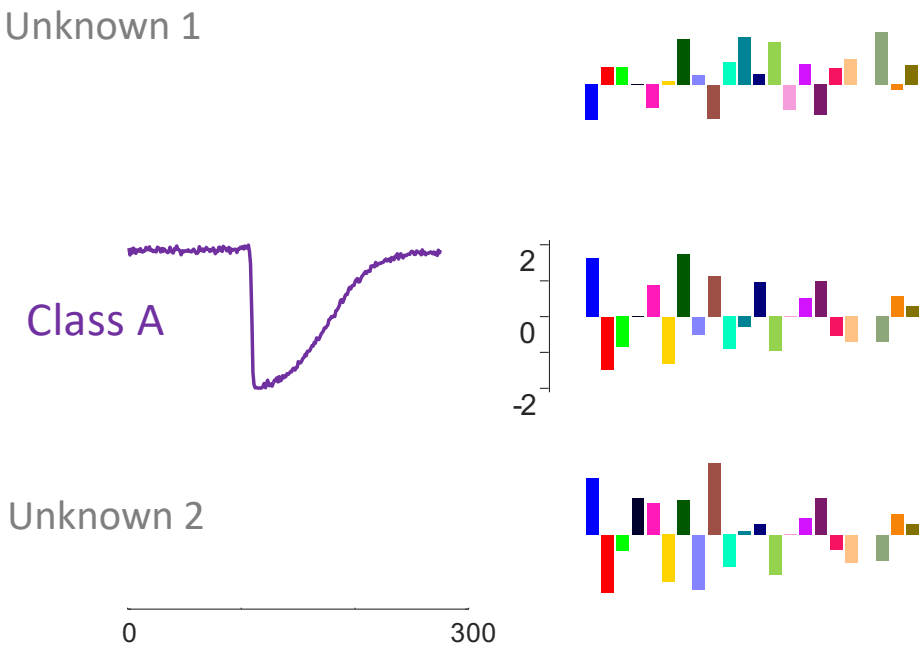


This is now a proxy for Class A

We will use this for classification, clustering, anomaly detection etc.

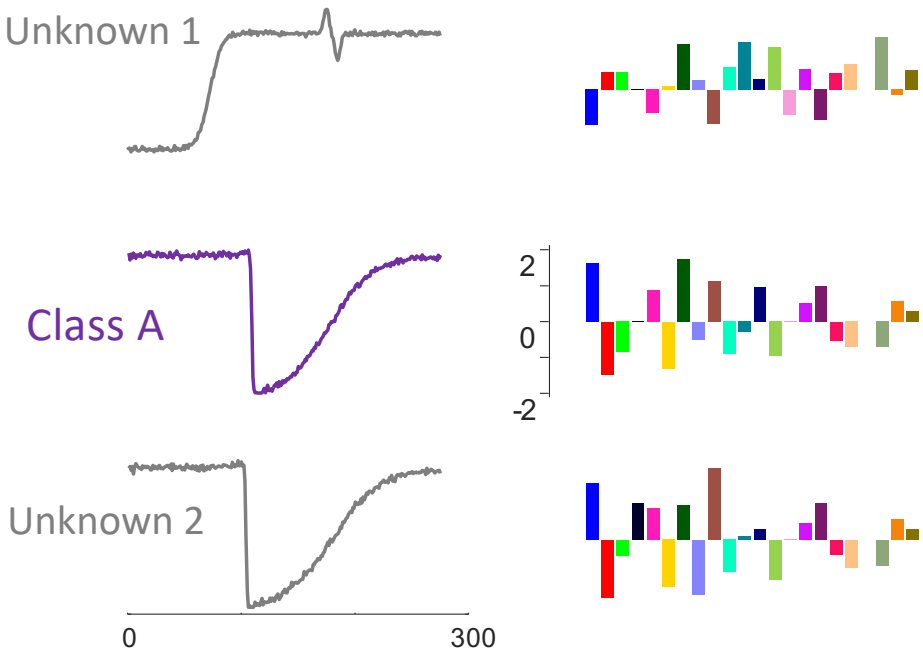
Now let us compare the feature vector to two unknown instances, ‘1’ and ‘2’

Which one is also in Class A?



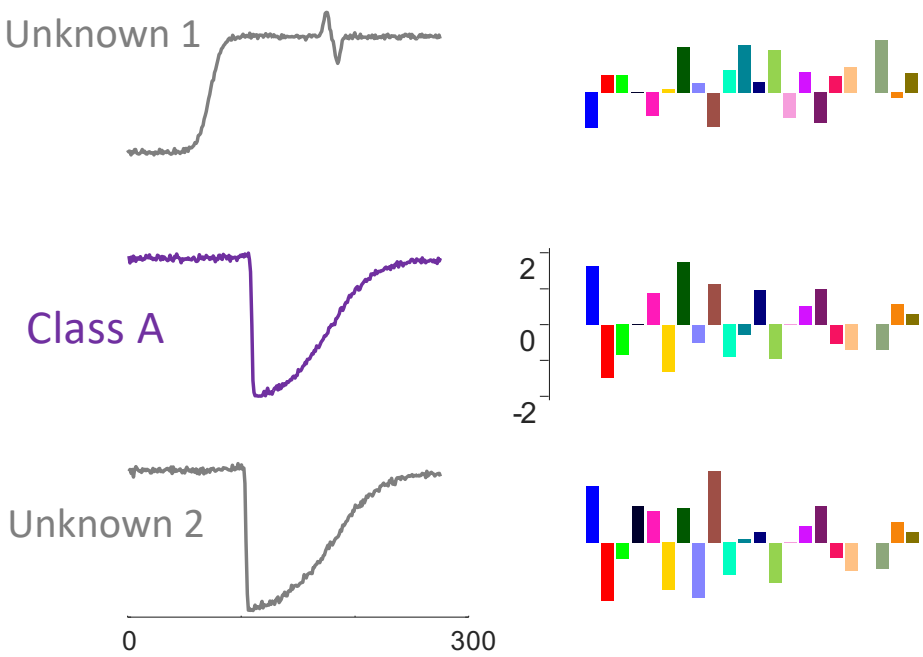
Which one is also in Class A?

It is of course, Unknown 2



Which one is also in Class A?

It is of course Unknown 2



An important note

In this example the class happens to be preserved in *both* shape and feature.

More generally, there are time series that are conserved *only* in features.

Think of human behaviors:

Conserved in **shape**:
walking/running/swimming/cycling

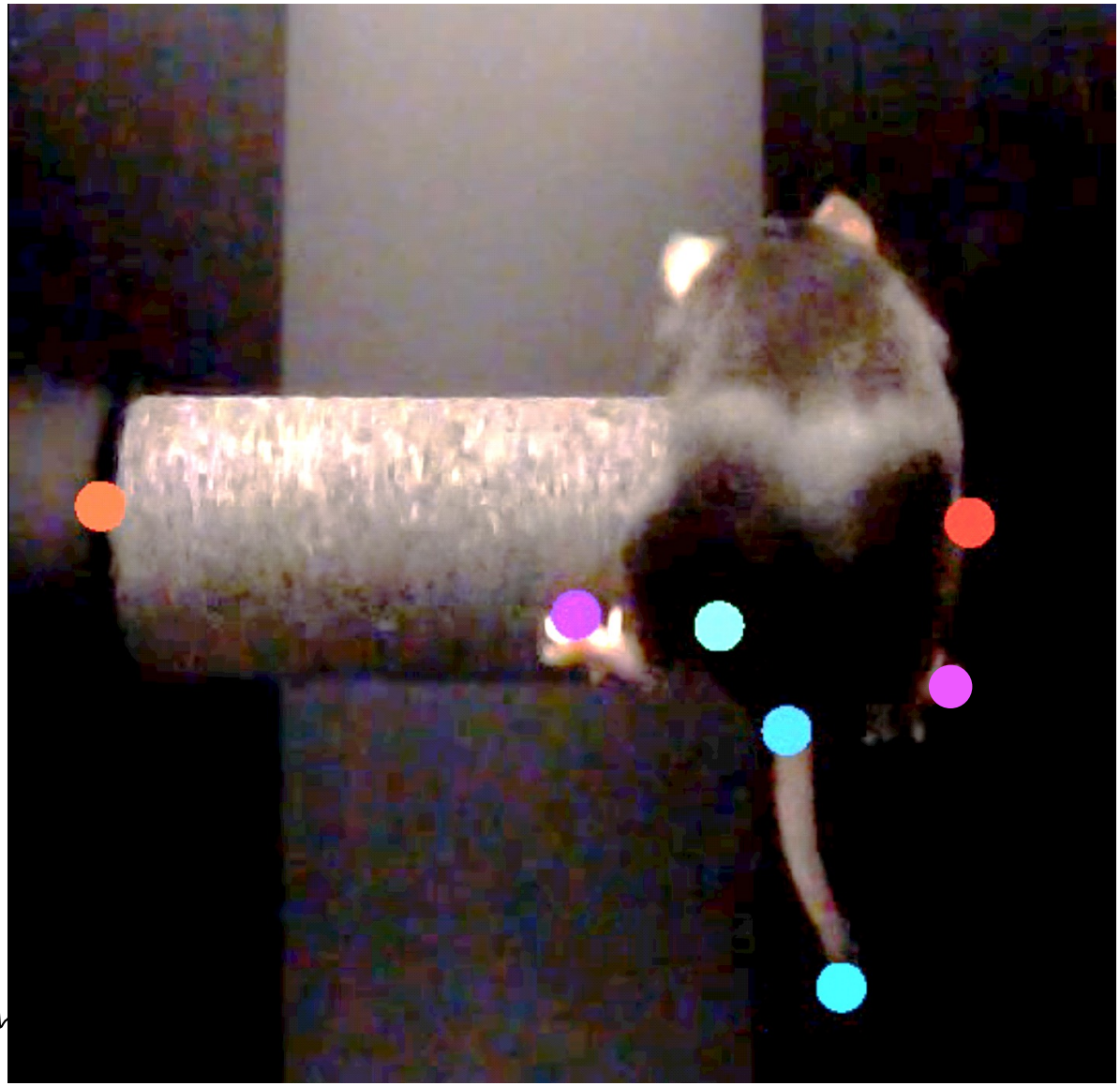
Conserved in **feature**:
reading/watchingTV/resting/sleeping

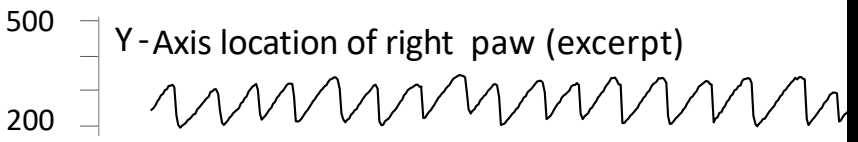
Once you have your feature vectors, you can (among *many* other things) do the analogue of Distance Profile and Matrix Profile.

Lets see a quick example:

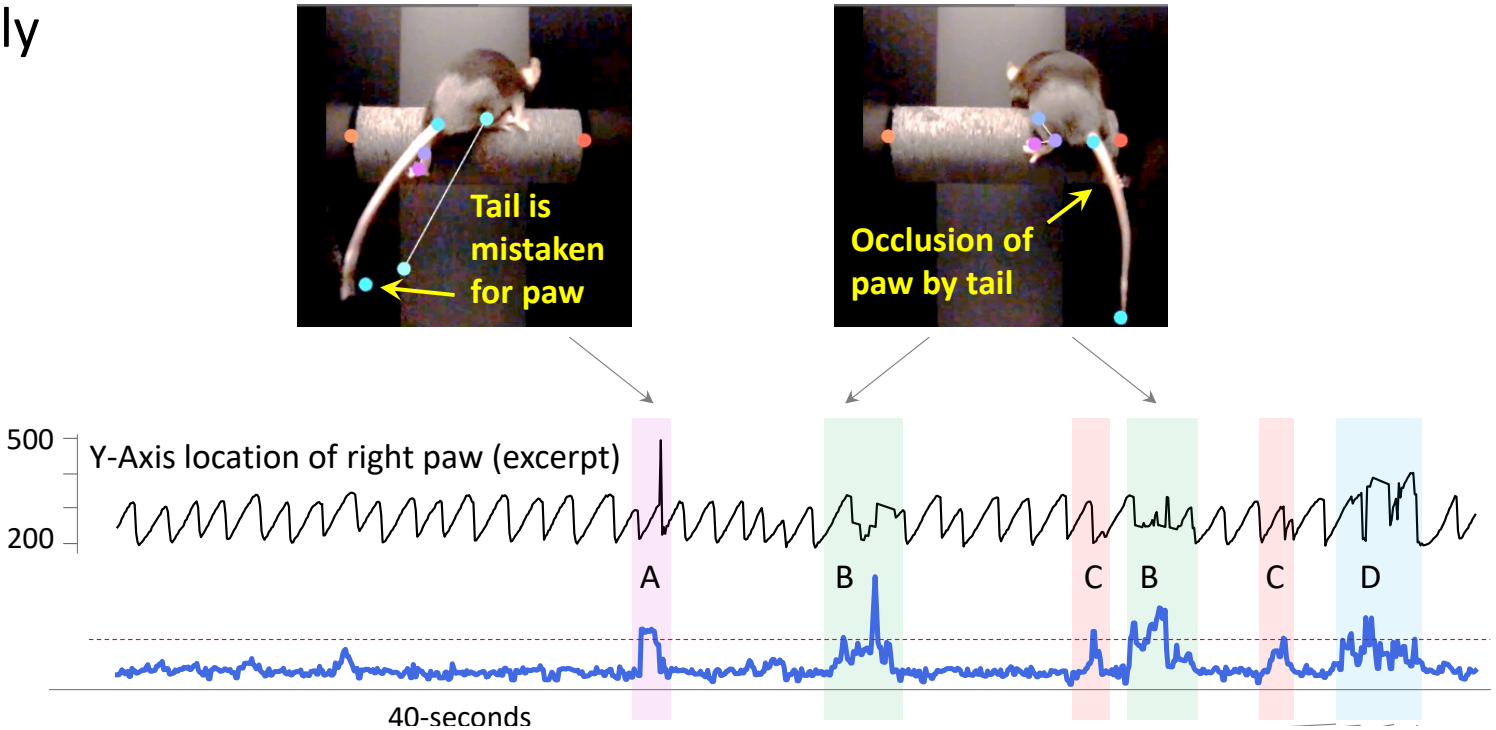
We have a mouse walking on a circular treadmill..

Let's build a catch-22 Matrix Profile. The high values should be when there are anomalies in the data…

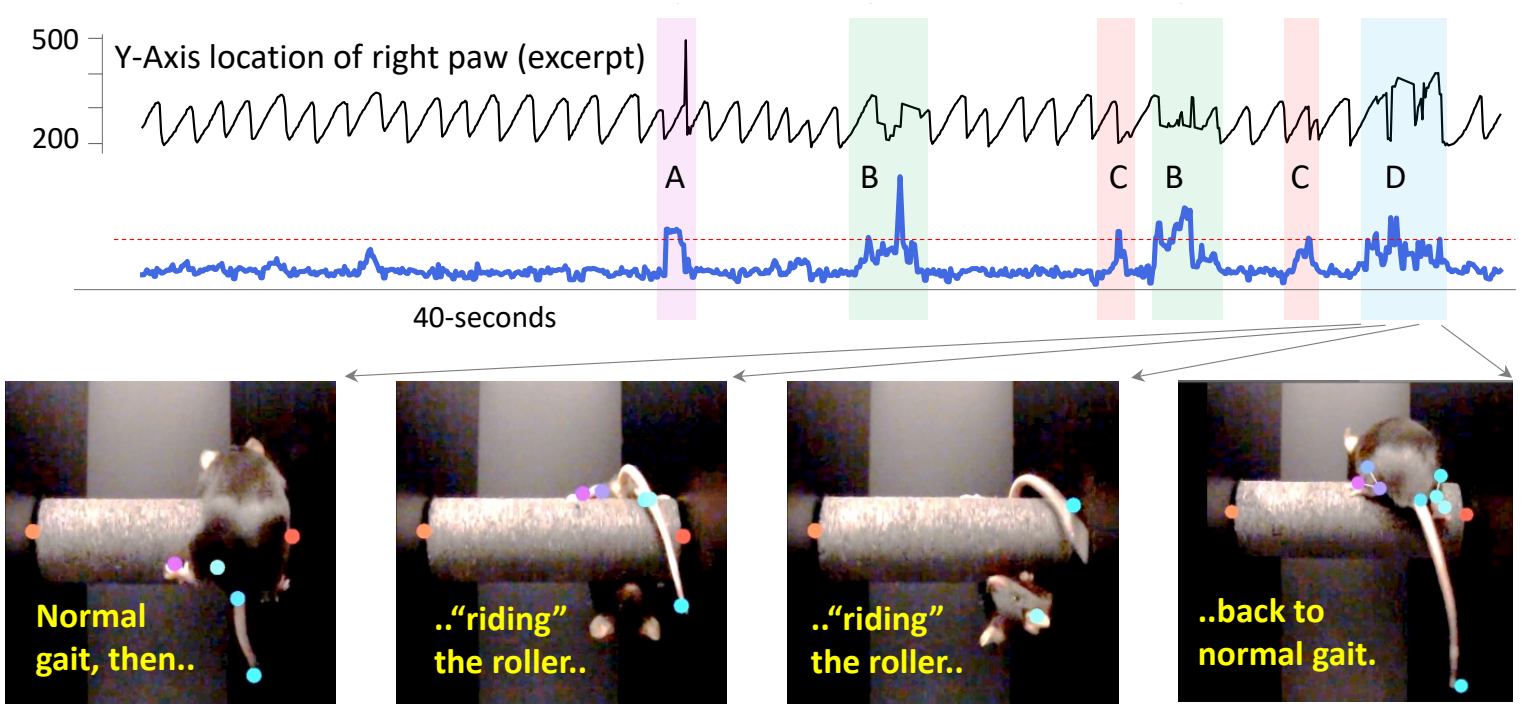




This works, beautifully



This works, beautifully



Summary

- For data that has conserved *shape*, the Distance Profile and the Matrix Profile will solve 90 to 100% of your task at hand.
- For data that has conserved *features*, using catch-22 with an appropriate algorithm will get you a long way. You can do a little better sometimes:
  - Reducing the 22 features to an even smaller subset
  - Augmenting the features with one or two custom features.

- We only looked at univariate data, but you normally have many dimensions, say X, Y, Z ,roll, pitch, yaw, temperature, pressure..
  - The Distance Profile and the Matrix Profile trivially generalize to multidimensions.
  - However, in most cases, you can solve your problem using only one dimension.
  - Using catch22 with multidimensional data is possible, but less well understood.

FCCM Tutorial

Streaming Time Series Analysis for FPGAs

Eamonn Keogh

eamonn@cs.ucr.edu

Phillip Brisk

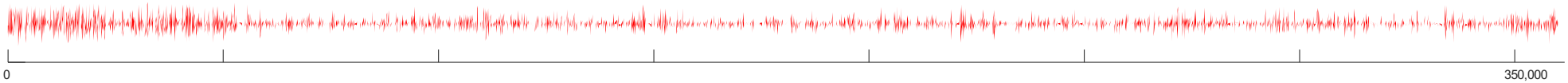
philip@cs.ucr.edu

Questions?

Backup Slides Below

Are there any repeated patterns in my data?

The dataset is an hour of EOG (eye movement) data of a sleeping patient, sampled at 100 Hz. It looks very noisy, it is not obvious that there is any repeated structure…



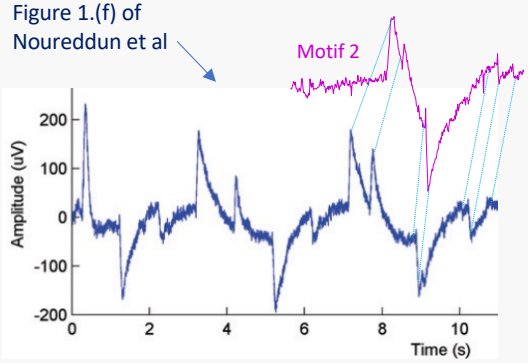
Let us run the Matrix Profile, looking for four-second long motifs…

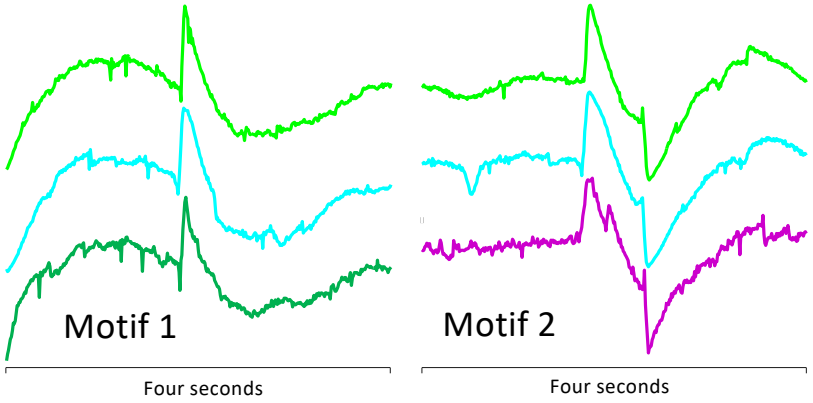
```
>> load eog_sample.mat
>> [matrixProfile profileIndex, motifIndex, discordIndex] = interactiveMatrixProfileVer3_website(eog_sample, 400);
```

The code takes a while to fully converge, but in just a few seconds, we see some stunningly well conserved motifs…

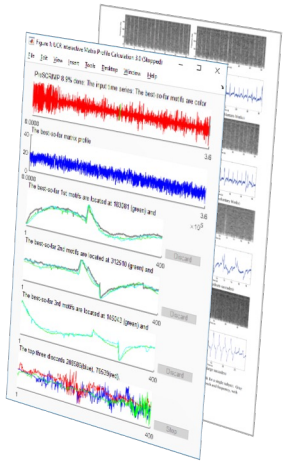
Having found the motifs, we can ask, what are they? A quick glance at a paper by Noureddin et. al. locates a very similar pattern (with some time warping) called `eye-blink-artifact`.

Figure 1.(f) of Noureddun et al

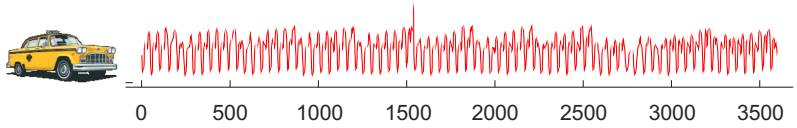




Note that there may be more examples of each motif. We should take one of the above, and use MASS to find the top 100 neighbors… See *Have we ever seen a pattern that looks just like this?*. We can also adjust the range parameter *r* inside the motif extraction code.



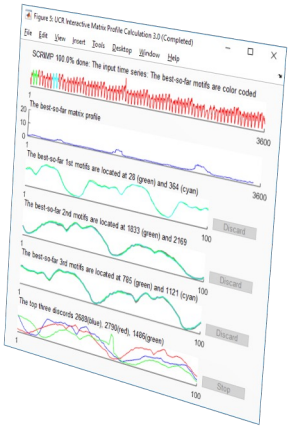
What are the three most unusual days in this three-month long dataset?



The datasets is Taxi demand, in New York City, in the last three months of the year.

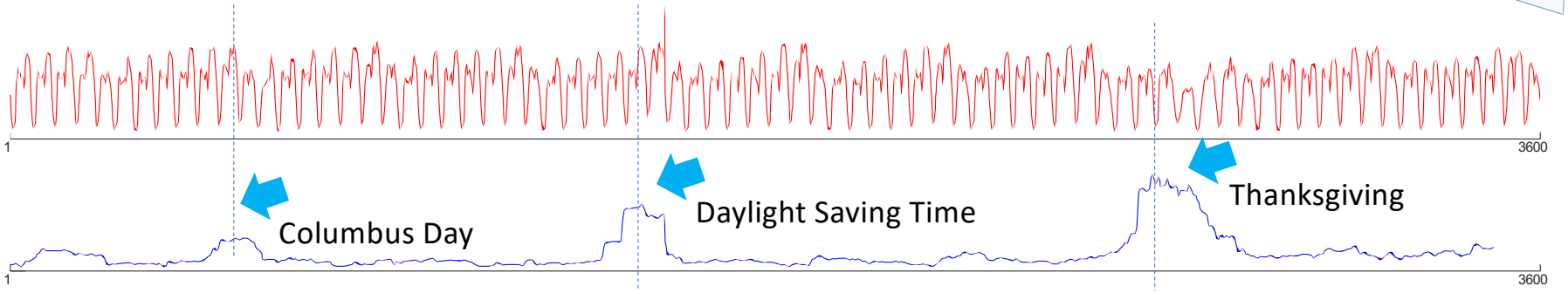
We choose 100 datapoints, which is about *two* days long (the exact values do not matter much here).

```
>> load taxi_3_months.txt
>> [matrixProfile, profileIndex, motifIndex, discordIndex] = interactiveMatrixProfileVer3_website(taxi_3_months ,100);
```

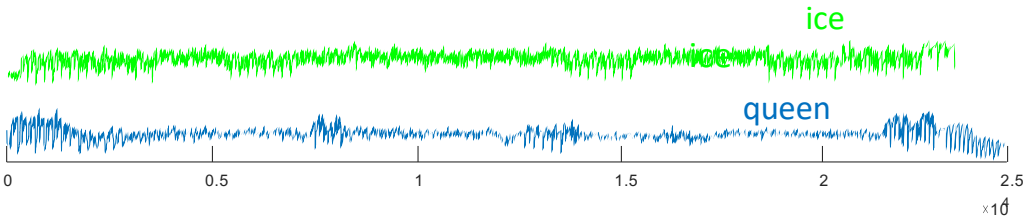


The code pops up the matrix profile tool, and one second later, we are done! The three most unusual days correspond to the three highest values of the matrix profile (i.e. the *discords*), but what are they?

- The highest value corresponds to Thanksgiving
- We find a secondary peak around Nov 6th, what could it be? Daylight Saving Time! The clock going backwards one hour, gives an *apparent* doubling of taxi load.
- We find a tertiary peak around Oct 13th, what could it be? Columbus Day! Columbus Day is largely ignored in much of America, but still a big deal in NY, with its large Italian American community.



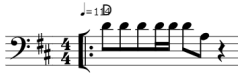
Is there any pattern that is common to these two time series?

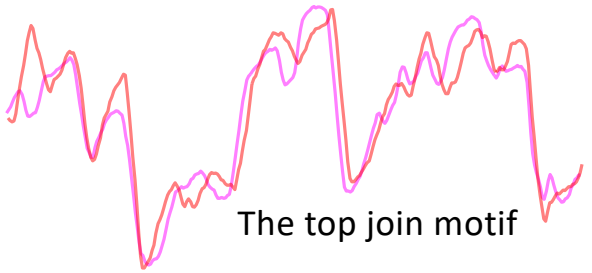


Lets assume that the common pattern is 3 seconds, or 300 datapoints long.
Let us concatenate the two time series, and smooth them (just for visualization purposes, we don't really need to)
Now let us find the top motif, but insist that one motif comes before 24289, and one after…

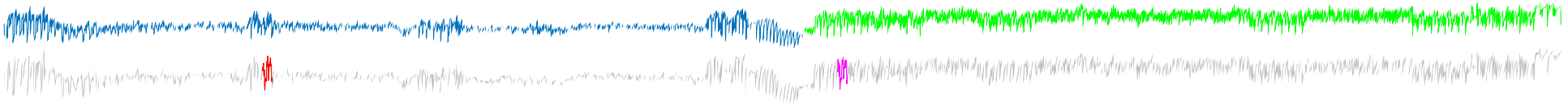
```
>> load('Queen_vs_Ice.mat')
>> whos
  Name                      Size                Bytes  Class     Attributes
  mfcc_queen                1x24289            194312  double
  mfcc_vanilla_ice          1x23095            184760  double
>> interactiveMatrixProfileAB(smooth([mfcc_queen , mfcc_vanilla_ice]), 300, 24289); % This will spawn this plot ->
```

The top join motif shows a highly conserved pattern.

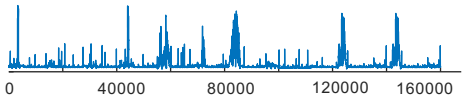
It is the famous bass line  from *Under Pressure* by Queen which was plagiarized by Vanilla Ice.



The concept for this example comes from Dr. Diego Furtado Silva.



Find the most conserved pattern that happens at least once every two days in this dataset



The question is a little underspecified, as the length for the conserved patterns was not given. Let us try two hours, which is about 800 data points.
The full 20,000 datapoints represents about 14 days of electrical demand data for a house in the U.K. Thus we first need to divide it into approximate 2 day chunks.

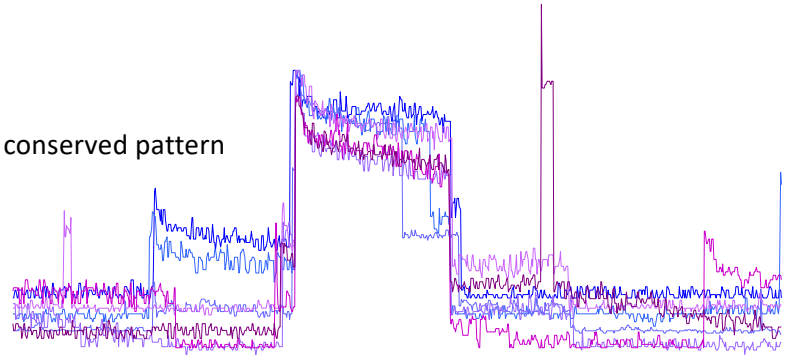
```
>> load TwoWeekElectrical
>> seven_two_day_chunks = divide_data(T);
```

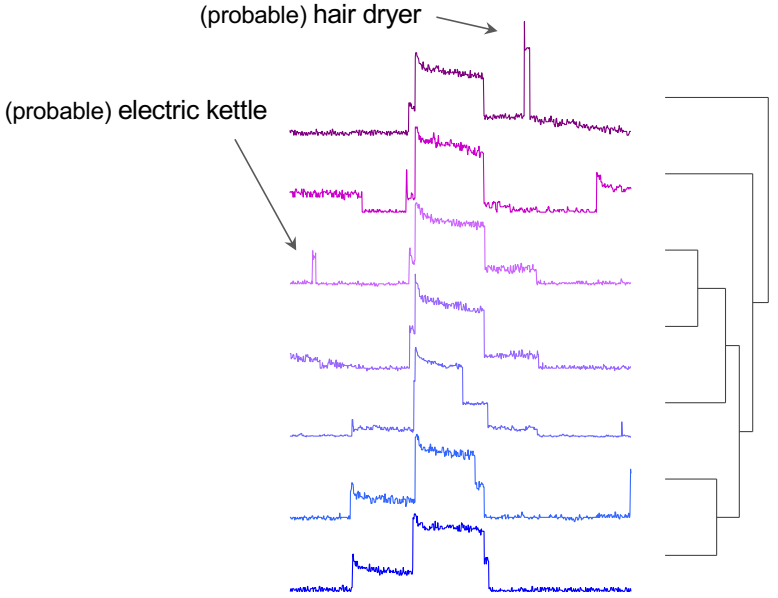
Now we just need to call the consensus motif code.

```
>> consensus_motifs = consensusMotifs(seven_two_day_chunks,800); % 800 is the length of subsequence
```

The code returns the seven time series below. Note that the basic pattern is highly conserved, given how noisy the data is.
The similarity between the items can be better seen if we cluster the time series with a single linkage dendrogram.

The most conserved pattern





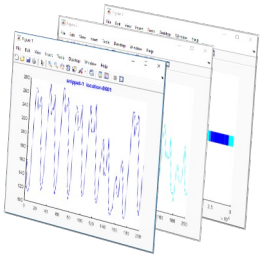
If you had to summarize this long time series with just two shorter examples, what would they be?

The dataset is 3 years of Italian power demand data which represents the hourly electrical power demand of a small Italian city for 3 years beginning on Jan 1st 1995.

Jan/1/1995 May/31/1998

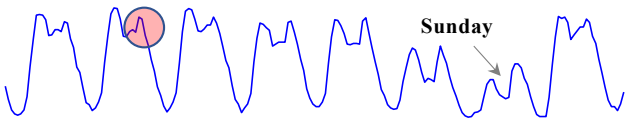
We just need to call Time Series Snippets algorithm…

```
>> load('ItalianPowerDemand.mat')
>> [fraction,snippet,snippetidx]= snippetfinder(data(:,4),2,200,30);
```

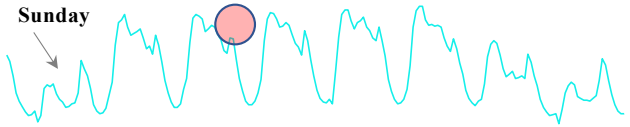


It will pop open three windows, which are snippet 1, snippet 2 and the regime bar.

We searched for the top-2 snippets of length 200. This was our quick "eyeballing" guess as to the length of a week, but it is actually about 8.3 days. Note that the snippets are not align to start at the same day of the week (this is a trivial constraint to add if desired).

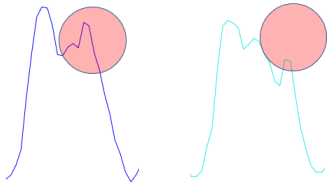


Snippet 1 8 days

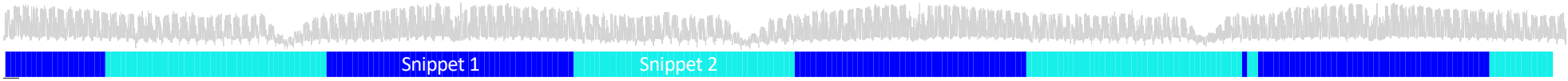


Snippet 2 8 days

What makes the snippets different? (tentative answer) In the winter, people go home after work (and turn on heaters/appliances). In the summer, people do more leisure activities after work and don't return home until it is cooler.

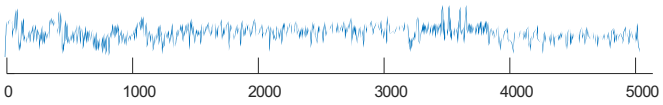


We obtain the "regime bar," which tells us which snippet "explains" which region of data. As it happens, Snippets seem to represent *summer* and *winter* regimes respectively.



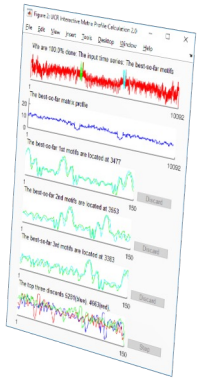
Jan/1/1995 May/31/1998

Are there any patterns that appear as time reversed versions of themselves in my data?



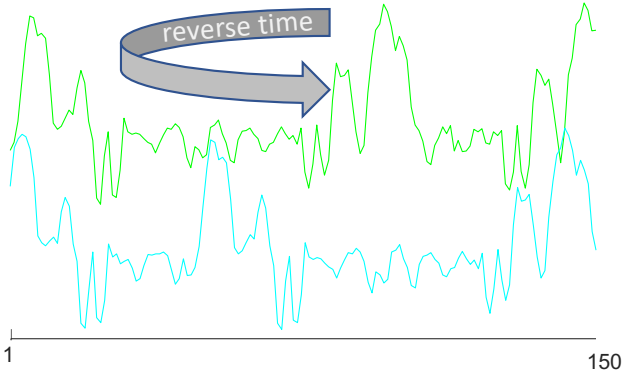
Lets us load the data, and concatenate it to itself, after flipping left to right.
We can then search for a join motif, that spans 5046, the length of the original time series.
If we find a good join motif, it means that the conserved pattern is time reversed!

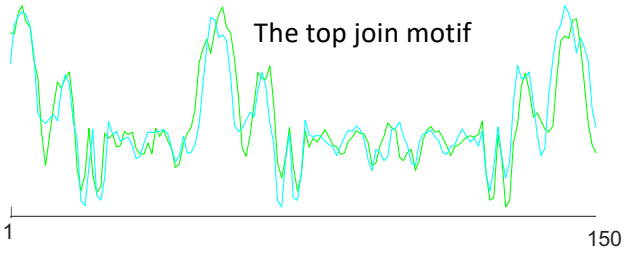
```
>> load('mfcc.mat')
>> length(mfcc1(1,:))
ans = 5046
>> interactiveMatrixProfileAB(([mfcc1(1,:)'; flipud(mfcc1(1,:)')]), 150, 5046); % This will spawn this plot ->
```



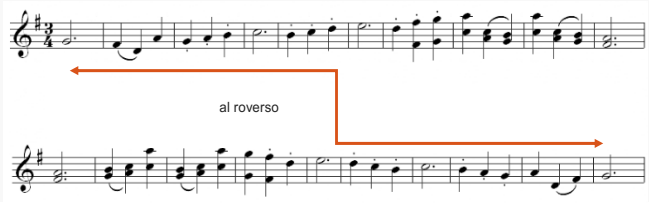
The top join motif shows a highly conserved pattern.

Why would a pattern occur time reversed?

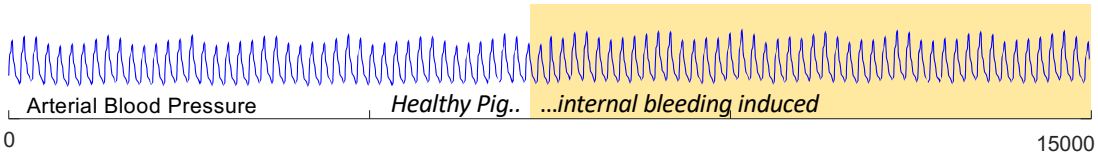




"The most extraordinary of all canonic movements from this time is of course from Symphony No. 47. Here Haydn writes out only one reprise of a two-reprise form, and the performer must play the music 'backward' the second time around".
The data is the 1st MFCC of this piece of music.



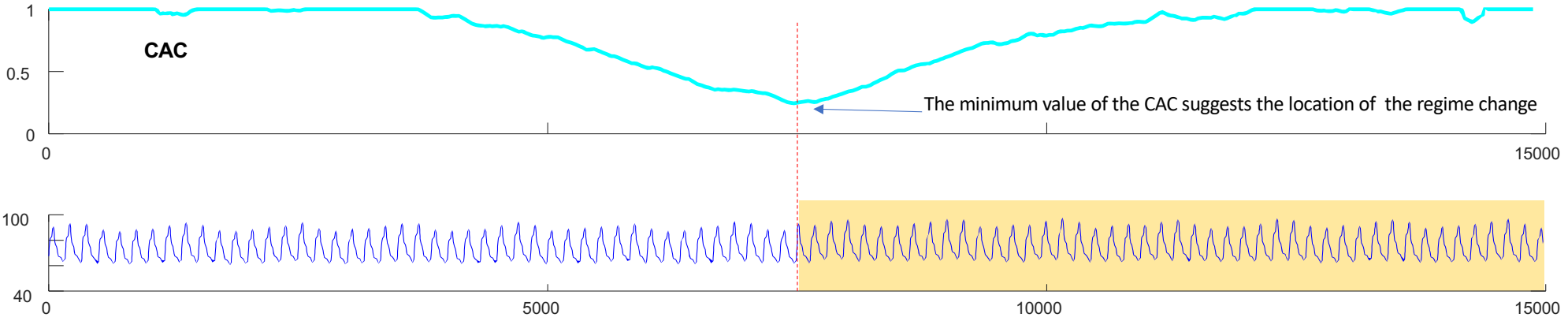
When does the regime change in this time series?



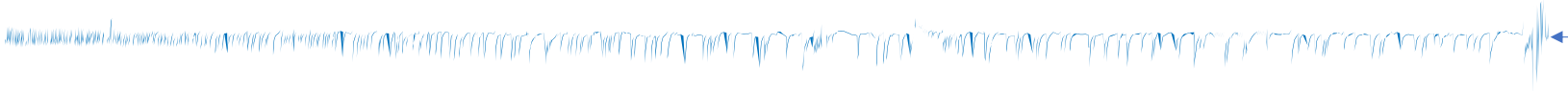
In this dataset, at time stamp 7,500, bleeding was induced in an otherwise healthy pig. This changes the pig's APB measurement, but only *very* slightly. Could we find the location of the change, if we were not told it? Moreover, can we do this with no domain knowledge? In other words, can we detect regime changes in time series?

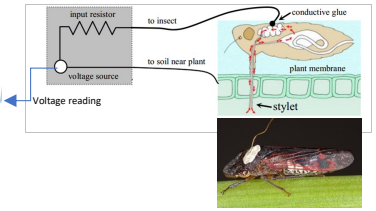
```
>> TS = load('PigInternalBleedingDatasetArtPressureFluidFilled_100_7501.txt');
>> CAC = RunSegmentation(TS, SL);  %SL is the length of subsequence
>> plot(CAC,'c')
>> [~, loc] = min(CAC) %value of loc is 7460 which is the approximation of exact value 7500
```

Here, we choose SL to be 100, approximately the length of one period of arterial pressure (or the period of whatever repeated patterns you have in your data), however, up to half or twice that value would work just as well. The output curve, the CAC, minimizes at just the right place.
How does it do it? In brief, if we examine the pointers in the Matrix Profile Index, we will find that very few will cross *over* the location of a regime change (most healthy beats have a nearest neighbor that is another healthy beat, most "bleeding" beats have a nearest neighbor that is another "bleeding" beat), it is this lack of pointers that cross over the regime change that is what the CAC is measuring.



Are there any patterns that repeat in my data, but at two distinct lengths?





See also "*Is there any pattern that is common to these two time series?*"

We can solve this with a quick and dirty trick. The code `interactiveMatrixProfileAB(T,m,crossover)` searches time series `T` for a motif of length `m`, such that one of the motif pair occurs before `crossover` and one occurs after `crossover`. We can take a time series and append it to a *rescaled* copy itself, setting the to the length of the original time series. Now when we find motifs, we are finding one at the original scale, *and* one at the rescaled size.

In this case, I want to know if any of my insect behaviors happens at length 5,000 and at 10,000, so I type…

```
>> load insectvolts.mat   % load some insect epg data
>> interactiveMatrixProfileAB([insectvolts ; insectvolts(1:2:end)], 5000, length(insectvolts));  % search the appended data
```

No need to let it converge, after a few seconds we have our answer…

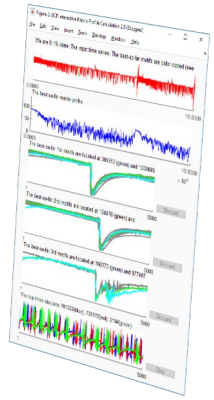
Note you *can* do this for non integer rescaling. Matlab will warn you, but it is defined and allowed.

Note that the bottom motif is discovered in spite of having a lot of noise in one of the occurrences.

Note that the dimensionality of the motifs is 5,000! This would have been unthinkable before the Matrix Profile.

Two motifs in the rescaled space

Two motifs in the original, true space



How can I optimize similarity search in a long time series?

see also "*How do I quickly search this long dataset for this pattern, if an approximate search is acceptable?*"

Suppose you want to find a query inside a long time series, say of length 67,000,000.

First trick: MASS (and several other FFT and DWT ideas) have their best case when the data length is a power of two, so pad the data to make it a power of two (padding with zeros works fine).

Second trick: MASS V3 is a piecewise version of MASS that performs better when the size of the pieces are well aligned with the hardware. You need to tune a single parameter, but the parameter can only be a power of two, so you can search over say 2^{10} to 2^{20}. Once you find a good value, you can hardcode it for your machine.

```
rng('default')  % Set seed for reproducibility
data= cumsum(randn(1,67000000));  % make data
query= cumsum(randn(1,2^13));     % make a long query
tic
  dist = MASS_V2(data ,query );
  [val,loc] = min(dist);          % find best match location
  hold on
  plot(zscore(data(loc:loc+length(query))))
  plot(zscore(query),'r')
  disp(['Best matching sequence is at ', num2str(loc)])
toc

figure
data = [data zeros(1,2^nextpow2(67000000) -67000000)]; % pad data to
tic                                                    % next pow of 2
  dist = MASS_V2(data ,query );
  [val,loc] = min(dist);           % find best match location
  hold on
  plot(zscore(data(loc:loc+length(query))))
  plot(zscore(query),'r')
  disp(['After padding: Best matching sequence is at ', num2str(loc)])
toc

figure
tic
  dist = MASS_V3(data ,query, 2^16 );
  [val,loc] = min(dist);      % find best match location
  hold on
  plot(zscore(data(loc:loc+length(query))))
  plot(zscore(query),'r')
  disp(['MASS V3 & padding: Best matching sequence is at ', num2str(loc)])
toc
```

Naive

Trick 1

Trick 1 & 2

If you run this code, it will output…

Best matching sequence is at 32463217

Elapsed time is 14.30 seconds.

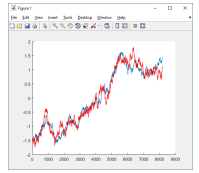
After padding: Best matching sequence is at 32463217

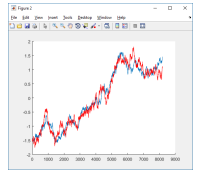
Elapsed time is 12.31 seconds.

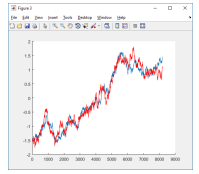
MASS V3 & padding: Best matching sequence is at 32463217

Elapsed time is 5.82 seconds.

Note that it outputs the exact same answer, regardless of the optimizations, but it is fast, then faster, then super fast.







What is the right length for motifs in this dataset?

See also Are there any repeated patterns in my data?

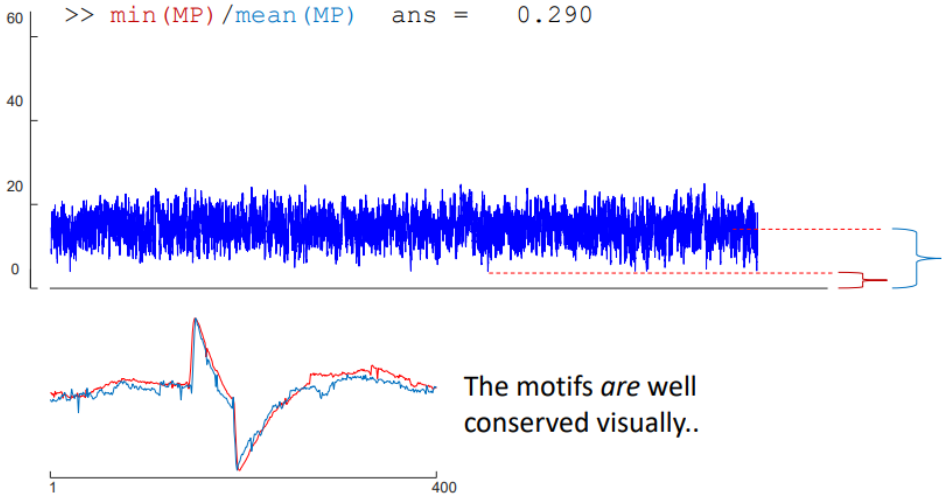
This is a very interesting question, which more than most, deserves a long explanation. However, to be brief and pragmatic. Let us revisit the EOG dataset. Recall that we choose 4 seconds as the motif length, which I happen to know (from reading papers on the topic) is a good choice.

```
>> load eog_sample.mat
>> [MP profileIndex, motifIndex, discordIndex] = interactiveMatrixProfileVer3_website(eog_sample, 400);
```

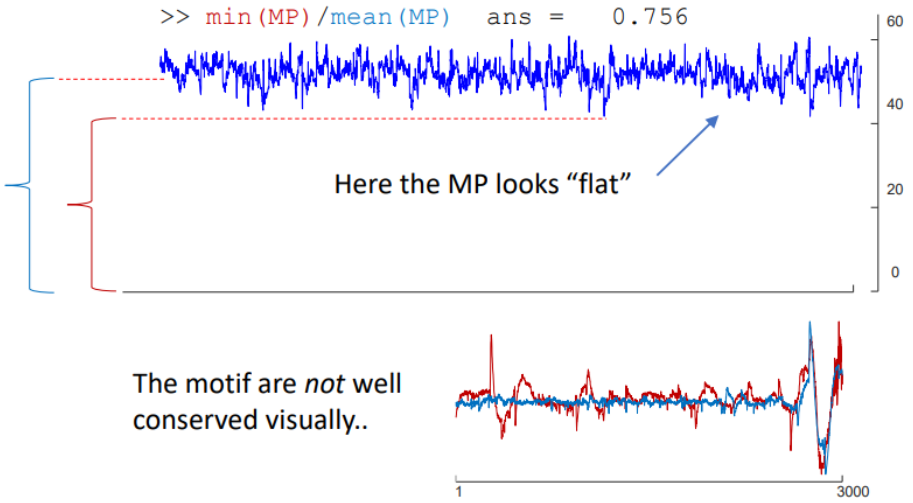
Let us look at the Matrix Profile, and the top motif we find (bottom left). The results seem to make sense.

However, suppose in contrast that we knew nothing about the domain, and had chosen a motif length that was much too long, say length 3,000 (bottom right). How could we know that we had picked a length that was too long? There are two clues:

- Obviously, the motifs themselves will be less well conserved visually.
- The Matrix Profile itself offers useful clues. It tells us how "specially well conserved" the motif is (`min(MP)`) relative to the *average* subsequence (`mean(MP)`). As the ratio of these two numbers is approaches zero, it suggests a stunningly well conserved motif in the midst of others unconserved data. However, as the ratio of these two numbers is approaches one, it suggest that the "motif" is no better conserved than we would expect by random chance. In practice, we rarely compute these ratios, as is visually obvious that the MP looks "flat".



The motifs *are* well conserved visually..



The motif are *not* well conserved visually..

I need to find motifs faster! Part I

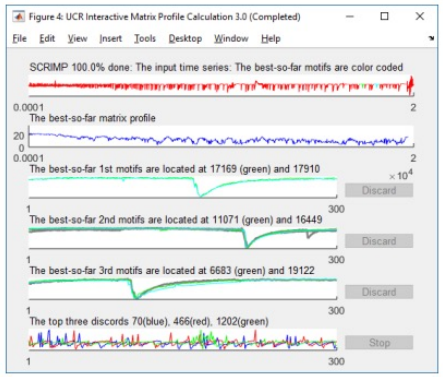
Part of the solution might be to use GPUs, see [a][b].

Moreover, it is important to understand, we almost *never* need to compute the Matrix Profile to completion, the anytime SCRIMP++ converges so fast, that in general we just run it 1% (or less) of convergence.

Nevertheless, sometimes you might want to compute the converged Matrix Profile. There is a faster algorithm for this. It exploits some of the ideas in [b], and it exploits the fact that it does not need to waste the overhead needed to make anytime updates, to achieve about an order of magnitude speedup.

For consistency with our other tools, when the fast code finish, it pops open the same plot.

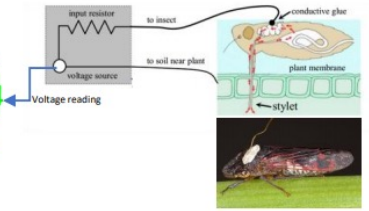
```
>> tic,  [MP MPIndex, mIndex, dIndex] = interactiveMatrixProfileVer3_website(insectvolts(1:35:700000), 300);, toc
Elapsed time is 61.44 seconds.
```




```
>> tic, [MP MPIndex, mIndex, dIndex] = mpx(insectvolts(1:35:700000), 200, 300); , toc
Elapsed time is 6.22 seconds.
```

To make this compatible with 2016 MATLAB, we replace the built-in "maxk" with a third party version called "maxk1" (by Salam Ismaeel). If you have later versions of MATLAB, you may wish to undo this change.

Have we ever seen a pattern that looks just like this, but possibly at a different length?



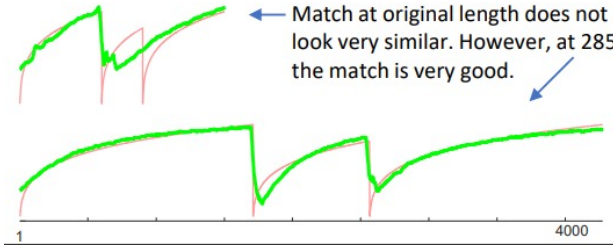
In our insect data, a basic feeding primitive looks like this:  , we can model it with something like: [1:600].^0.2

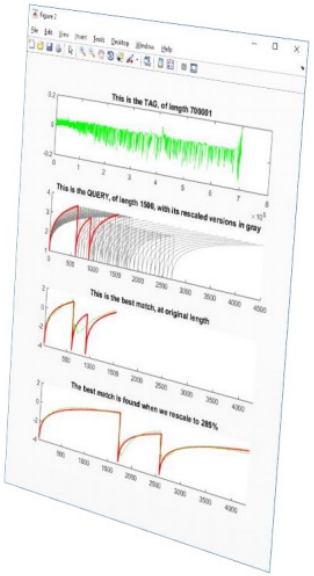
We have a theory that a certain higher level behavior will result in "*A long primitive, followed by a shorter and smaller primitive, followed by another long primitive*", like this..  , we can model it with: [[[1:600].^0.2] [[1:300].^0.2] [[1:600].^0.2]]

However, we don't know how long the whole pattern could be…

The function to the right can solve this problem. It simply brute forces a MASS test for all lengths within a range (here 100 to 300%) at a given step size (here 5%).
There may be faster techniques, but MASS is so fast, they may not be worth bothering with.
A critical trick is to *normalize* the comparisons at different lengths (see Appendix).

```
>> load insectvolts.mat % load some insect epg data
>> query = ([[[1:600].^0.2] [[1:300].^0.2] [[1:600].^0.2]]);
>> uniform_scaling_search(smooth(insectvolts,10), query);
```



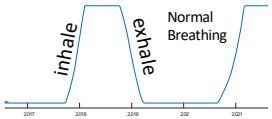


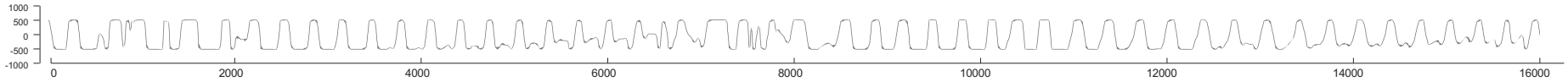
Even with this unoptimized approach. We can search two hours of data (at 100hz), with a long query (upto to 4,500 datapoints), in well under 10 seconds

This looks like a lot of code, but most of it is for plotting

```
function [] = uniform_scaling_search(TAG, QUERY)
figure;                                          % Spawn a blank figure
subplot(4,1,1);                                  % Plot panel 1
plot(TAG,'g')                                    % Plot the TAG/time series in green
title(['This is the time series, of length ',num2str(length(TAG))]);

subplot(4,1,2);                                  % Plot panel 2
hold on;
title(['This is the QUERY, of length ',num2str(length(QUERY)), ', rescaled versions in gray']);
for i = 110:10:300                               % Plot the rescaled queries
    plot(QUERY(1:100/i:end),'color',[0.5 0.5 0.5])
end
plot(QUERY,'LineWidth',2,'color','r');           % Plot the query

subplot(4,1,4);                                  % Plot panel 4
hold on;
best_match_val = inf;
for i = 100:5:300                                % Loop over all scalings
    NewQUERY = (QUERY(1:100/i:end));
    distprofile = MASS_V3(TAG, NewQUERY, 1024);
    [val,loc] = min(distprofile);
    val = val * 1/sqrt(i);                       % This normalizaiton step is critical see Appendix
    if val < best_match_val                      % Record the best scaling
        best_match_val = val;
        best_match_loc  = loc;
        best_match_scale = i;
    end
end

plot(zscore(QUERY(1:100/best_match_scale:end)),'LineWidth',2,'color','r');   % Plot bestscaled match
plot(zscore(TAG(best_match_loc:best_match_loc+length(QUERY(1:100/best_match_scale:end))-1)),'g');
set(gca,'Xlim',[1 length(QUERY(1:100/best_match_scale:end))]);
title(['The best match is found when we rescale to ',num2str(best_match_scale),'%']);

subplot(4,1,3);                                  % Plot panel 3
hold on;
distprofile = MASS_V3(TAG, QUERY, 1024);         % Compute the distance profile
[val,loc] = min(distprofile);                    % Find were the best match was
plot(zscore(QUERY),'LineWidth',2,'color','r');   % Plot the query
plot(zscore(TAG(loc:loc+length(QUERY)-1)),'g');
set(gca,'Xlim',[1 length(QUERY(1:100/best_match_scale:end))]);
title(['This is the best match, at original length'])

end
```

Are there any evolving patterns in this dataset (time series chains)

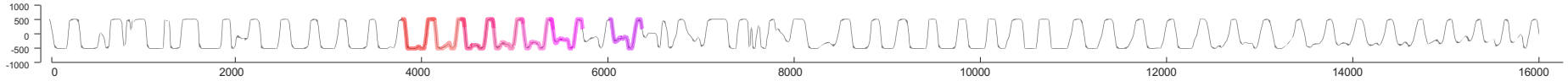




This is a dataset of respiration from a sleep study. Each breath appears to be about 360 data points long. So lets search *for time series chains* of length 360…

```
>> load respiration.mat
>> TSC1_demo(respiration , 360);
```

The algorithm finds the highlighted chains below.



Let us zoom in on the chains, to better see what is going on…

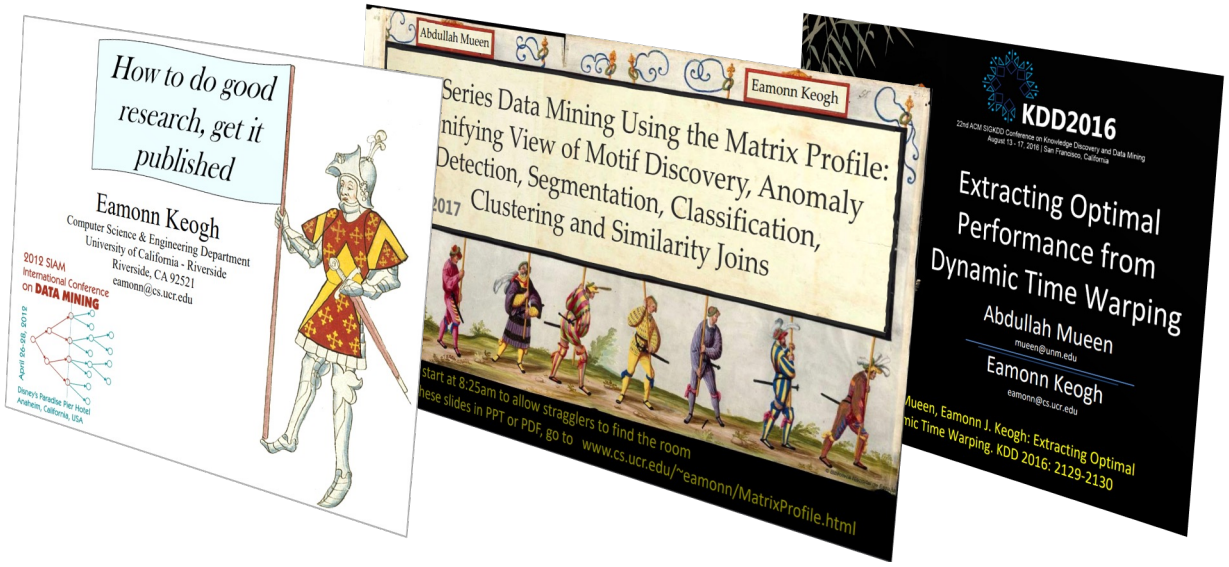
Note the increasing “gulp” artifact that happens between cycles. Also note that it begins to happen earlier and earlier in the cycle. What does this mean?

Here is the (lightly edited) annotation of Dr. Gregory Mason (LA BioMed/UCLA) an expert on cardiopulmonary interactions.

“The gulps are attempts to inspire against an obstruction coming the back of the tongue. The large signals are from the machine which do not necessarily reach the patient, the small gulps are pathologic attempts to breathe. Why does it increase? With each successive breath the patient tries harder to inspire. It finally is 'synchronized' and you don't see the small patient signal, and this event cycles over and over. The cycling is best seen without treatment if one looks up "crescendo snoring," a hallmark of obstructive sleep apnea.”

Welcome

- Congrats! About 90% of the best minds in data science are implicitly or explicitly working on the problem of getting people to click on ads! You are doing something more interesting and noble.
- If you like this tutorial, you may enjoy my others ;-)



Streaming Time Series Analysis for FPGAs

Eamonn Keogh

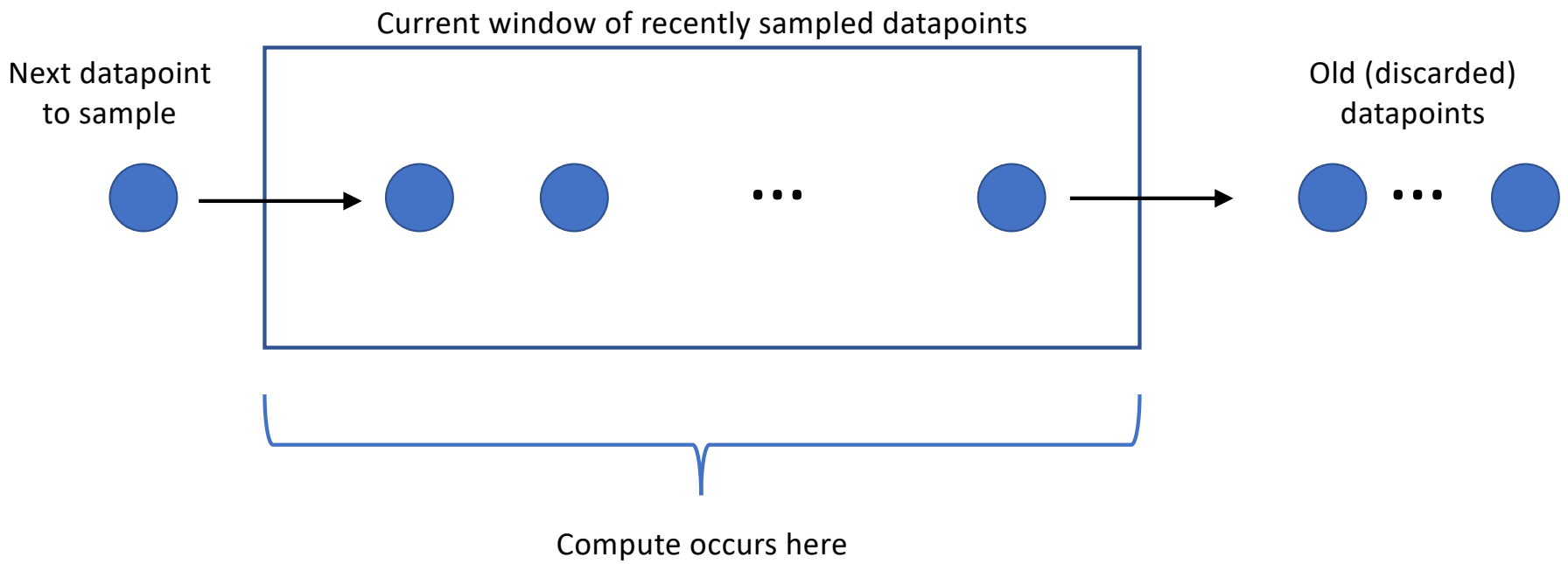
eamonn@cs.ucr.edu

Philip Brisk

philip@cs.ucr.edu

FCCM Tutorial

Streaming Time Series

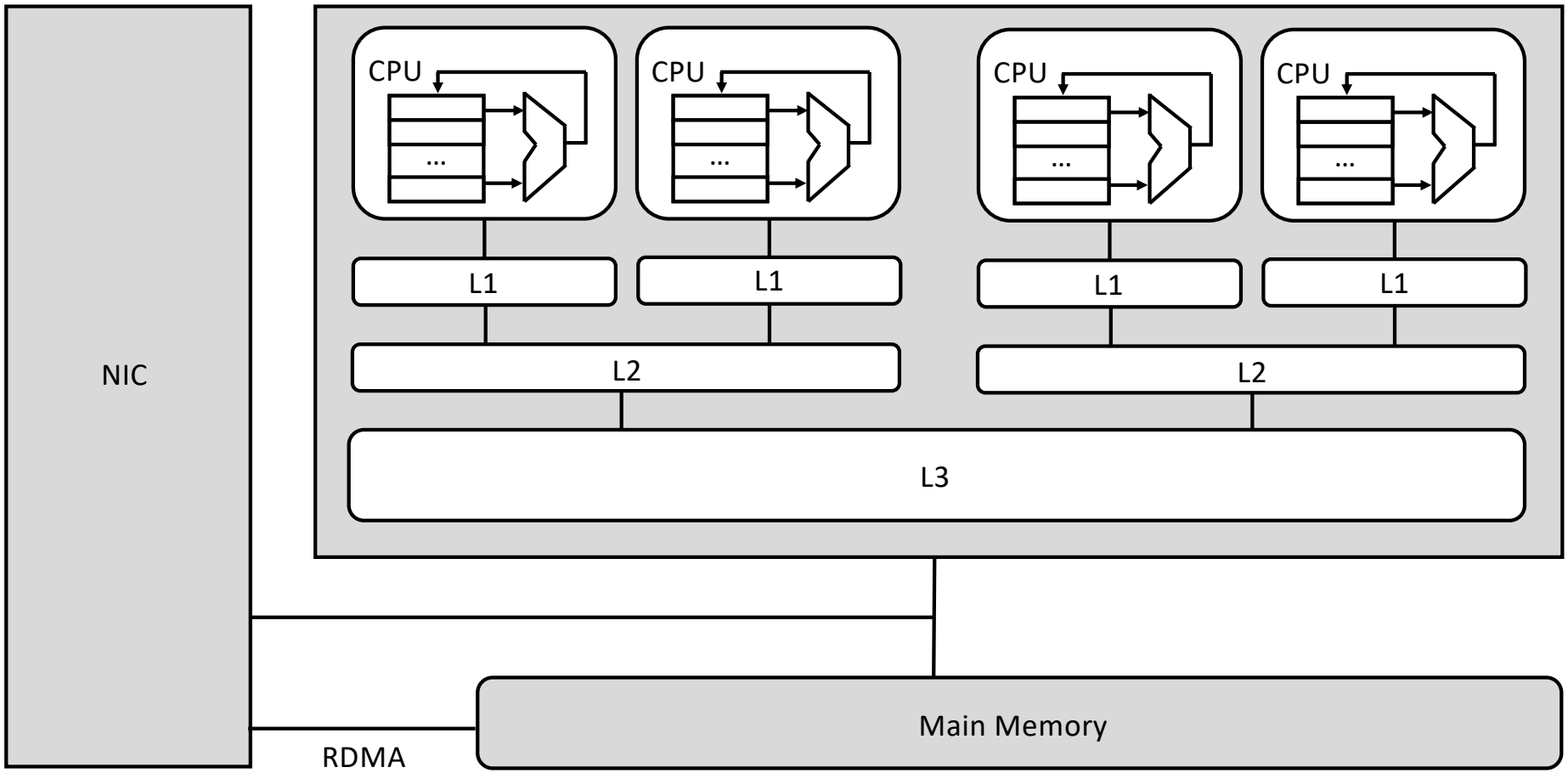


Offline vs. Streaming

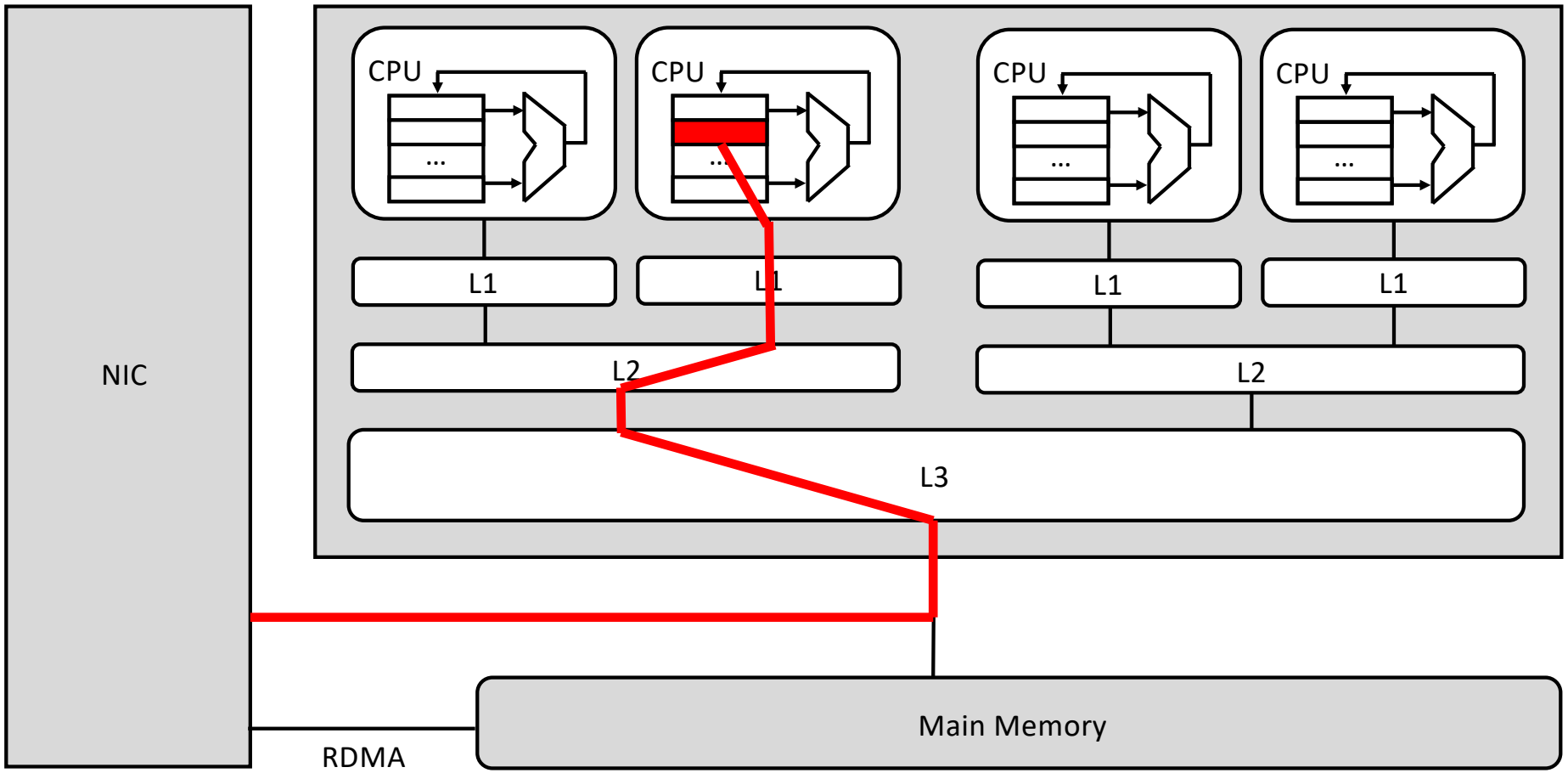
- Offline
  - Entire time series is available in memory or on disk
  - Example: Read time series from disk
    Compute Matrix Profile or catch-22 features
    Write Matrix Profile or catch-22 features to disk

- Streaming
  - Time series length is unknown (treat as infinite)
  - Datapoints stream from a sensor or across a network
  - Computed result for the current window may be consumed immediately

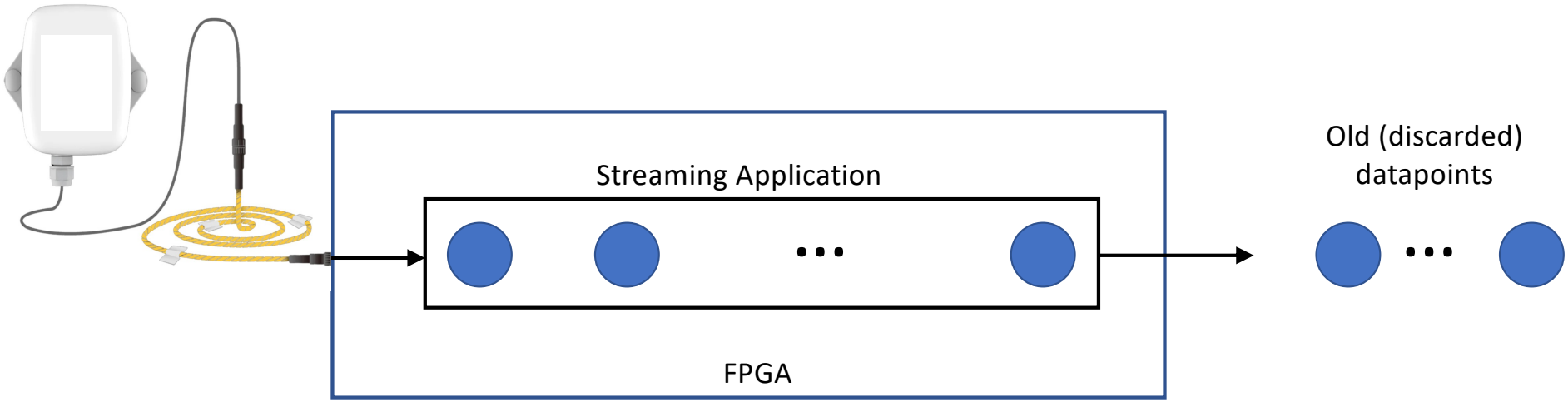
Streaming with your CPU (or GPU)



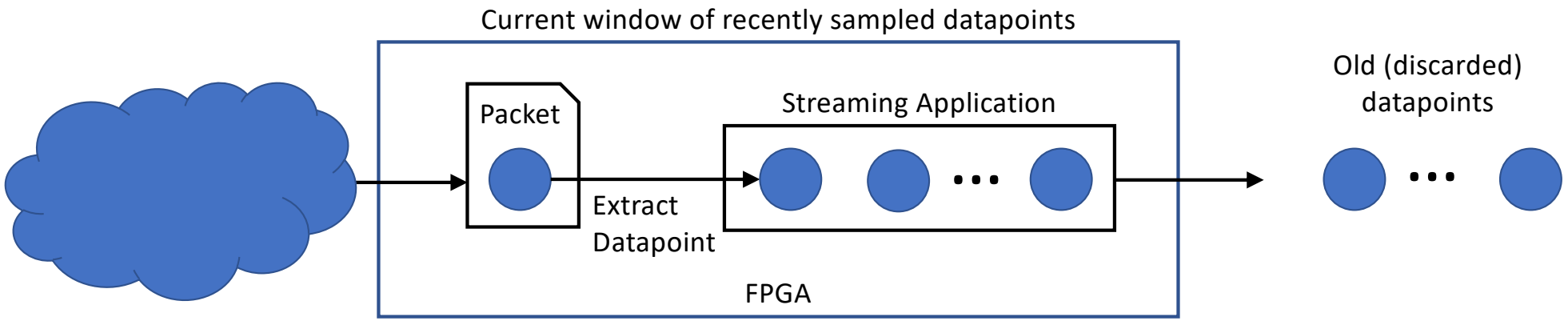
Streaming with your CPU (or GPU)



Streaming Time Series: Sensor



Streaming Time Series: Network



Offline → Streaming

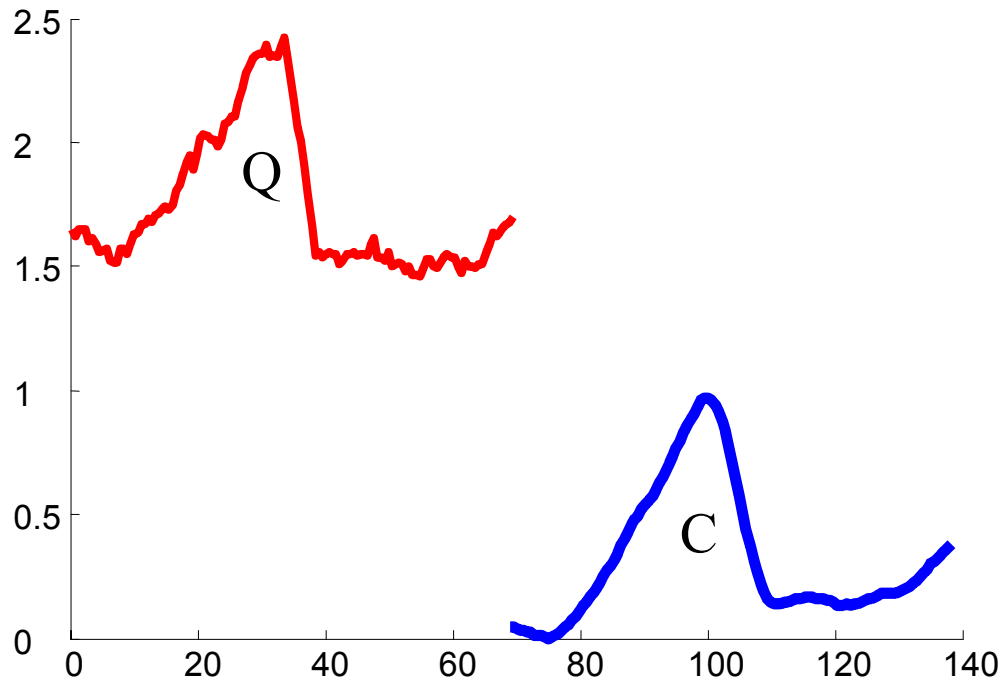
- Many time series algorithms are originally developed and evaluated in an offline context and development goes no further than a paper published at KDD, ICDM, etc.

- Very little work on how to convert an algorithm from offline to streaming and how to quantify or evaluate algorithmic differences.

Simple Example: Similarity

Objective Measure:

- How "similar" are the shapes of Q and C?



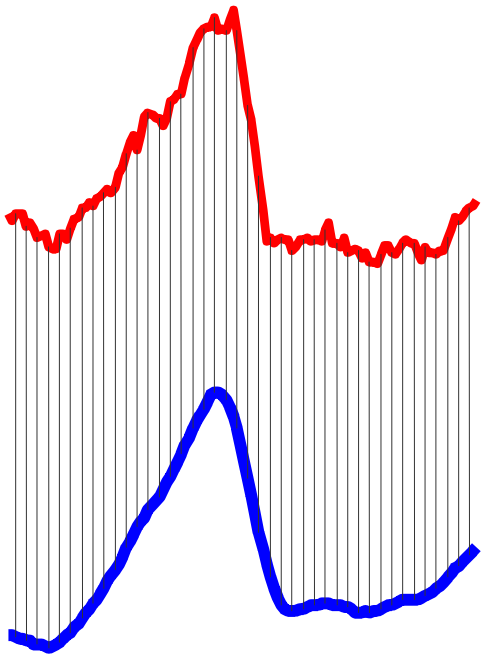
ole Example: Similarity

Objective Measure:

- How "similar" are the shapes of Q and C?

Euclidean Distance:

- $ED = \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$



Simple Example: Similarity

Objective Measure:

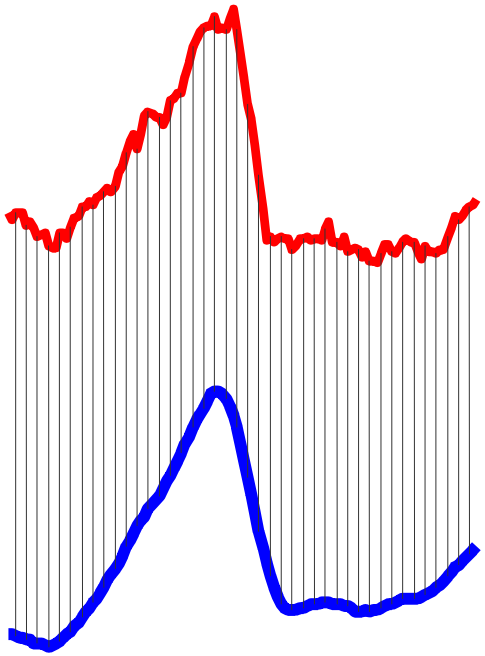
- How "similar" are the shapes of Q and C?

Euclidean Distance:

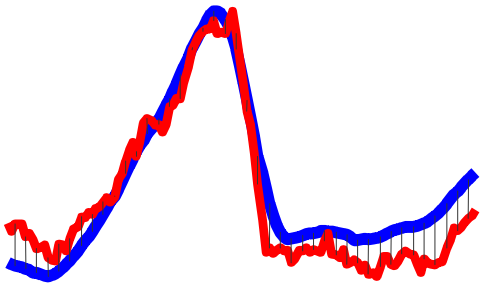
- $ED = \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$

Normalization needed!

Unnormalized



Normalized



Z-Normalization

- Many offline algorithms start by Z-normalizing a time series

$$T = \langle t_1, t_2, \ldots, t_n \rangle \qquad T_{Norm} = \left\langle \frac{t_1 - \mu}{\sigma}, \frac{t_2 - \mu}{\sigma}, \ldots, \frac{t_n - \mu}{\sigma} \right\rangle$$

$$\mu = \frac{1}{n} \sum_{i=1}^{n} t_i \qquad \sigma = \sqrt{\frac{\sum_{i=1}^{n} (t_i - \mu)^2}{n-1}}$$

Z-Normalization

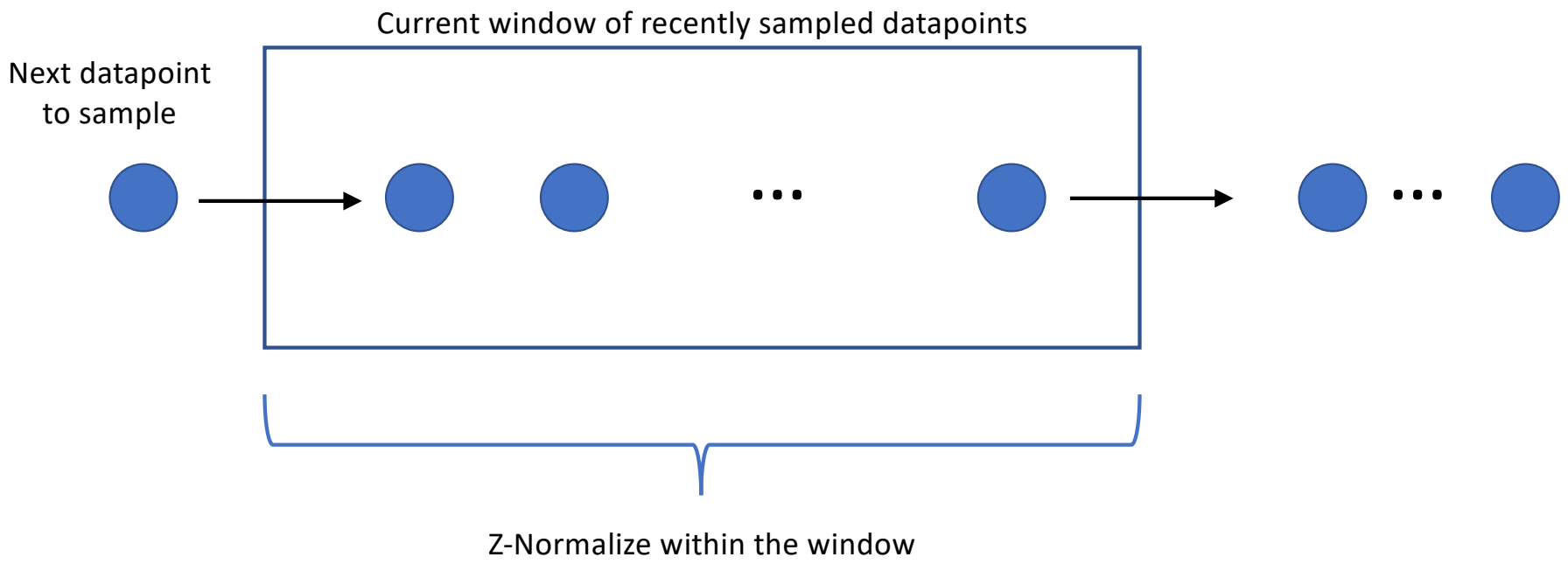
- Many offline algorithms start by Z-normalizing a time series

$$T = \langle t_1, t_2, \ldots, t_n \rangle \qquad T_{Norm} = \left\langle \frac{t_1 - \mu}{\sigma}, \frac{t_2 - \mu}{\sigma}, \ldots, \frac{t_n - \mu}{\sigma} \right\rangle$$

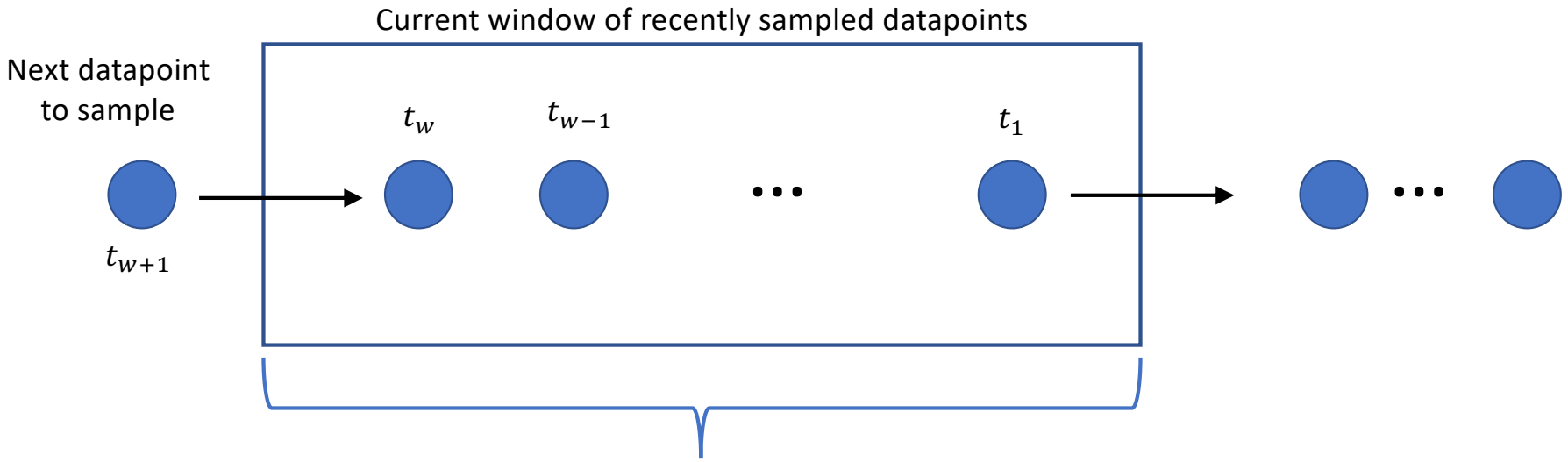
$$\mu = \frac{1}{n} \sum_{i=1}^{n} t_i \qquad \sigma = \sqrt{\frac{\sum_{i=1}^{n} (t_i - \mu)^2}{n-1}}$$

This sum assumes that the whole time series is available

Online Z-Normalization



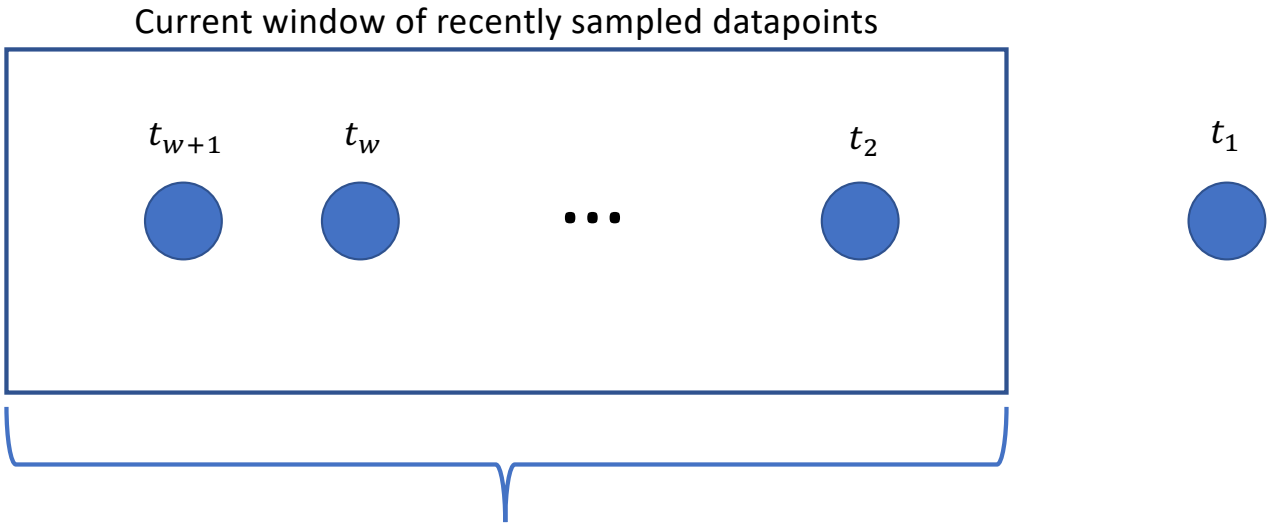
Streaming Sum and Streaming Mean



$$\Sigma_{sum} = \Sigma_{sum} + t_{w+1} - t_1$$

$$\mu = \mu + \frac{t_{w+1}}{w} - \frac{t_1}{w}$$

Streaming Z-Normalizer

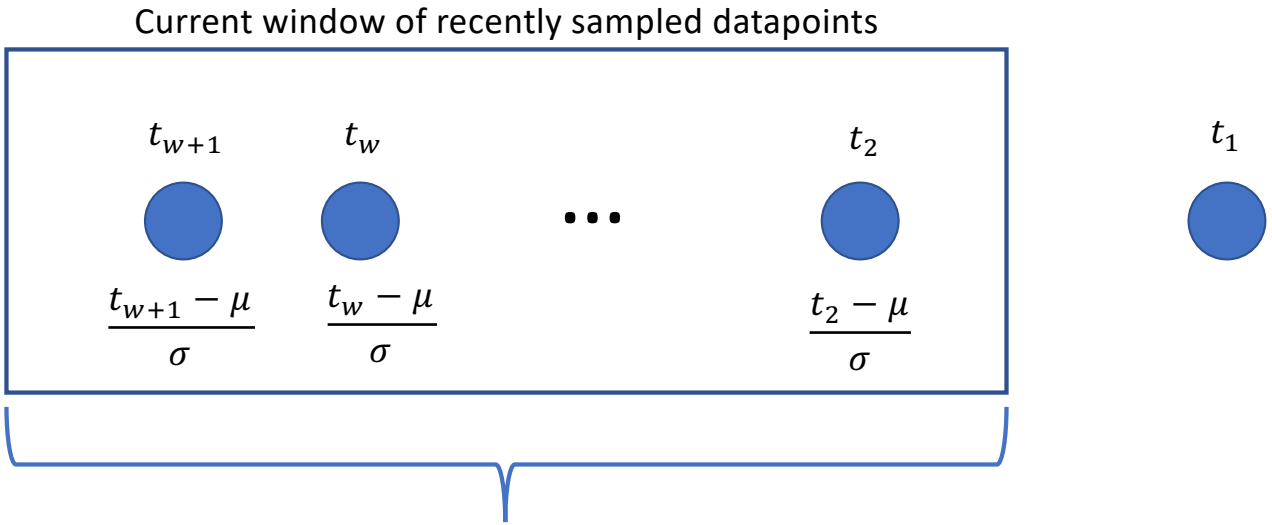


$$\Sigma_{sum} = \Sigma_{sum} + t_{w+1} - t_1$$

$$\mu = \mu + \frac{t_{w+1}}{w} - \frac{t_1}{w}$$

$$\sigma = \sqrt{\frac{\sum_{i=2}^{w+1}(t_i - \mu)^2}{w-1}}$$

Streaming Z-Normalizer



$$\Sigma_{sum} = \Sigma_{sum} + t_{w+1} - t_1$$

$$\mu = \mu + \frac{t_{w+1}}{w} - \frac{t_1}{w}$$

$$\sigma = \sqrt{\frac{\sum_{i=2}^{w+1}(t_i - \mu)^2}{w - 1}}$$

Z-Normalization: Offline vs. Streaming

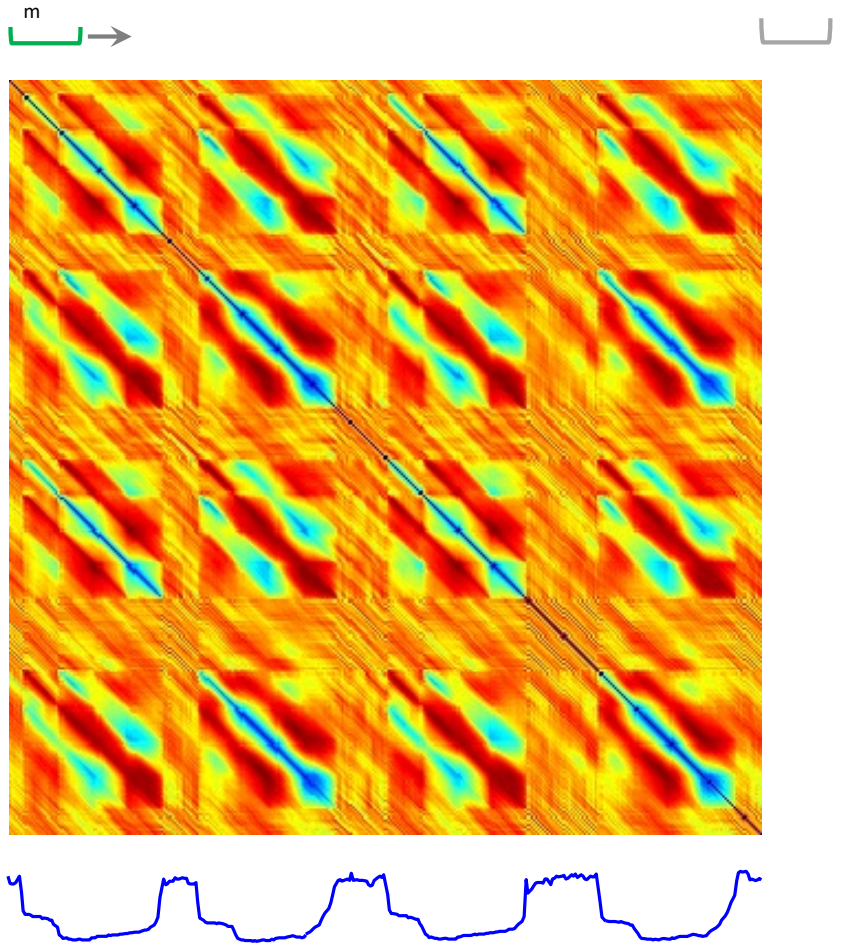
- Many offline time series algorithms begin by Z-normalizing the entire time series
  - This cannot work in a streaming context!

- If applied in an offline context, there may be numerical differences between results obtained from offline vs. streaming Z-normalization
  - Understanding and bounding these differences is key to effectively converting an offline algorithm to streaming, prior to FPGA implementation

- This phenomena is not exclusively limited to Z-Normalization

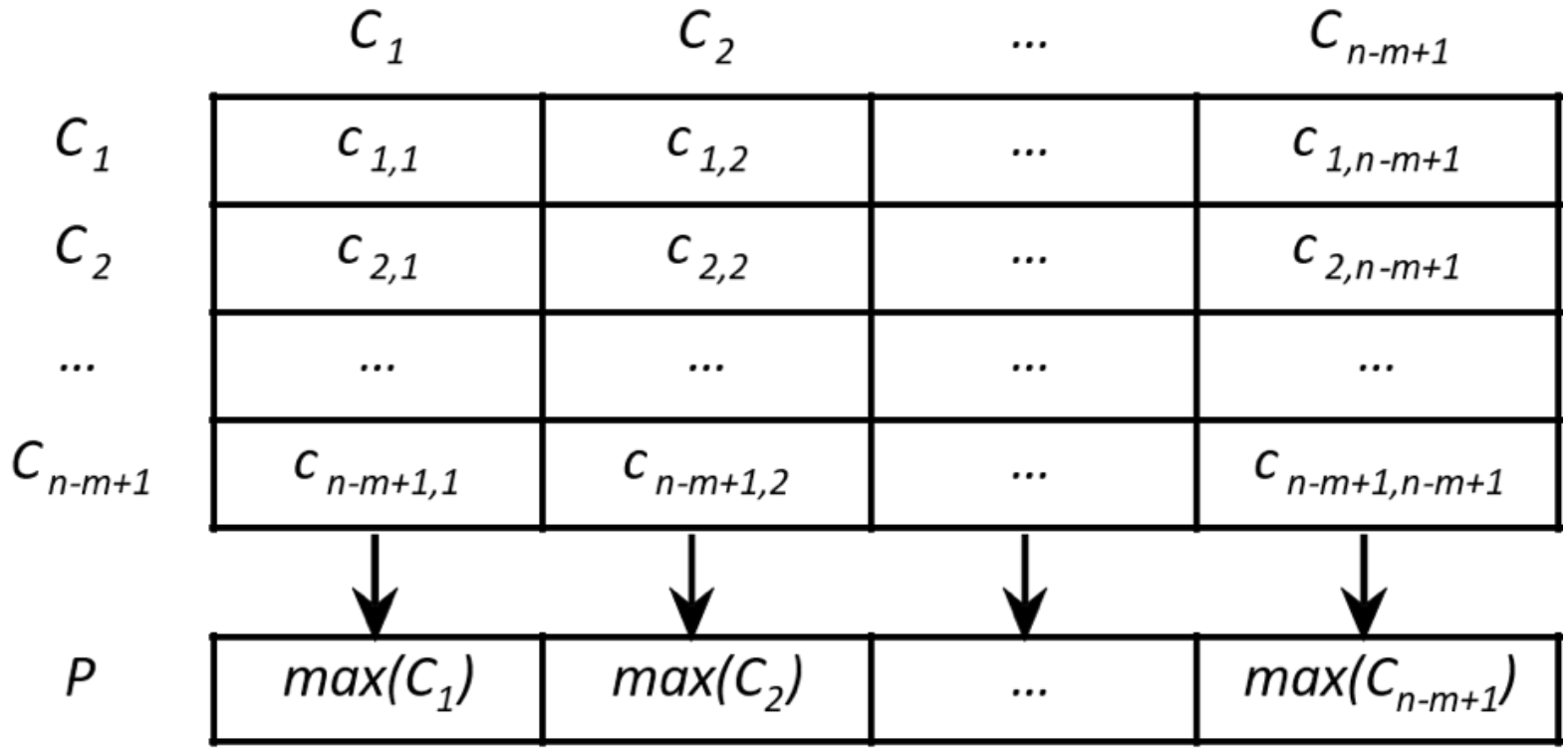
The Matrix Profile

- Need to compare every length-m subsequence in the time series
- This means we need the whole time series before we can compute the Matrix Profile
- Is there a streaming solution?

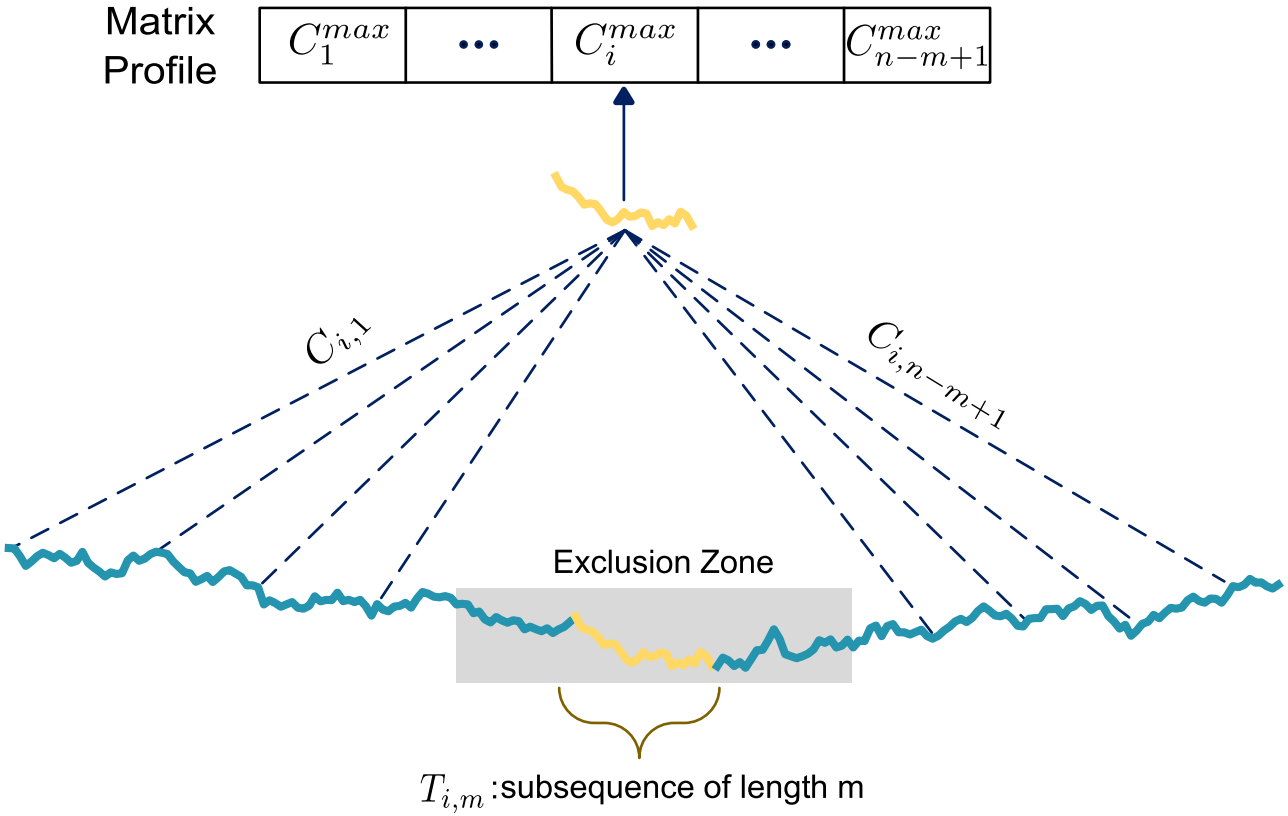


Key:
Small distances are blue
Large distances are red
Dark stripe is excluded

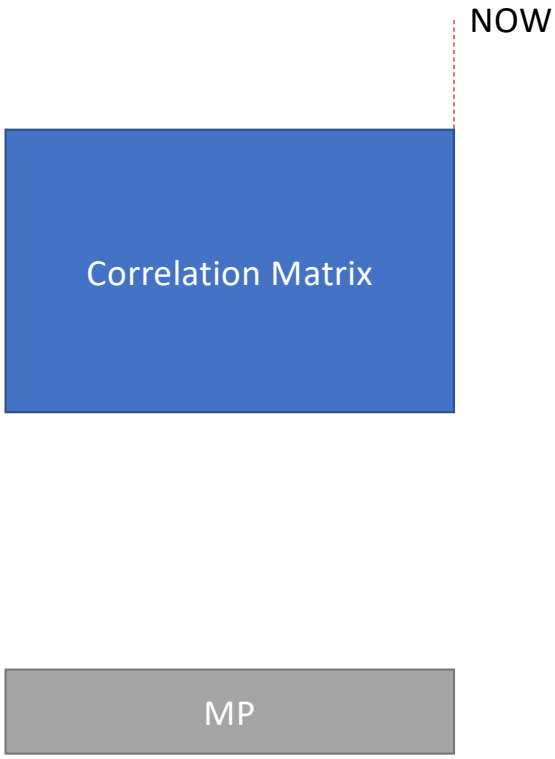
The Matrix Profile: Correlation



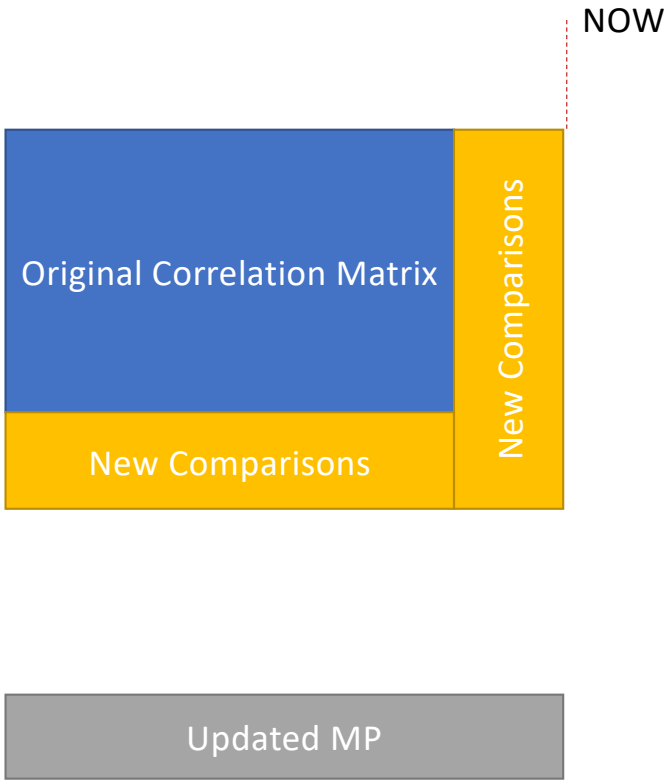
The Matrix Profile: Correlation



The Matrix Profile: Streaming

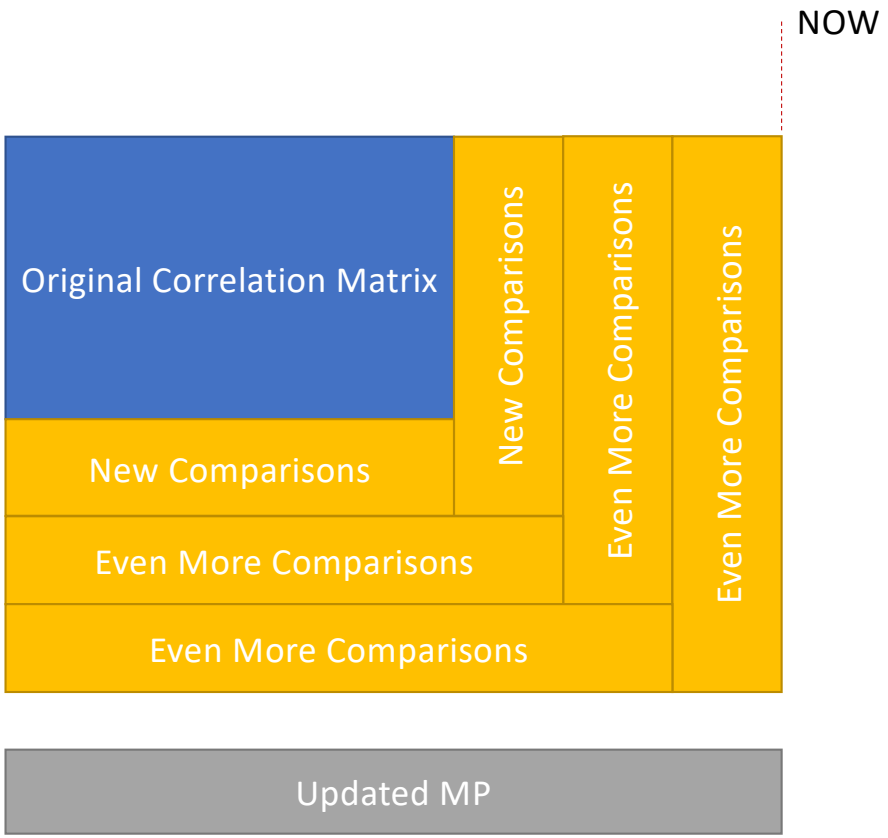


The Matrix Profile: Streaming

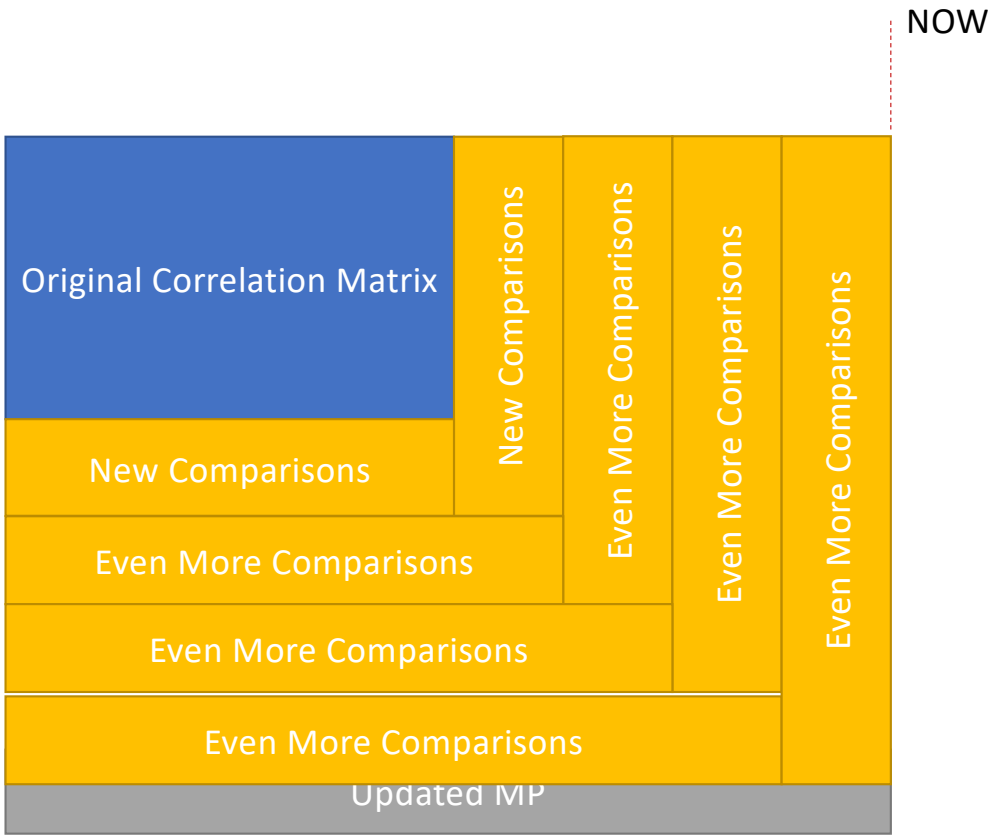


The Matrix Profile: Streaming

The Matrix Profile: Streaming



The Matrix Profile: Streaming



How can we fix this?

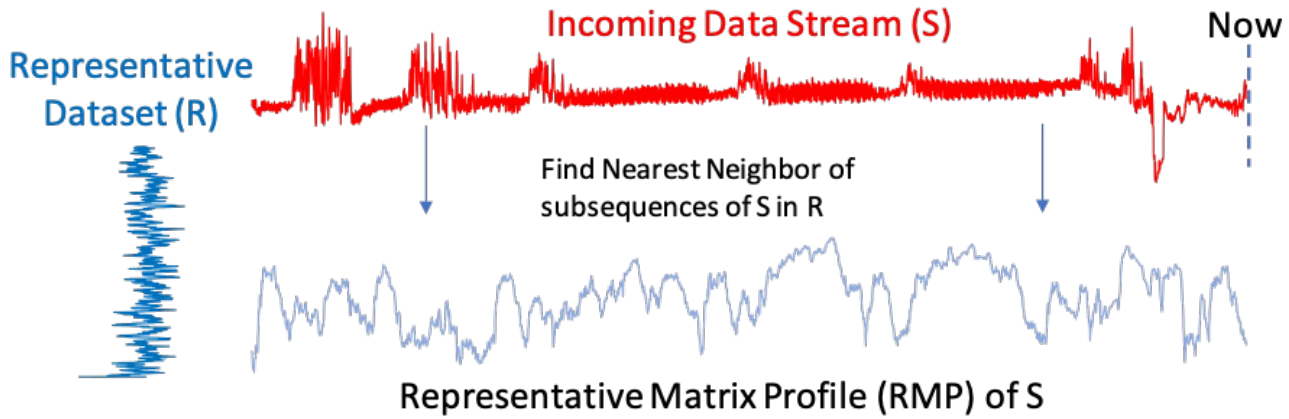
The Matrix Profile: Problems with the Existing Method

- This method computes the EXACT Matrix Profile values; however…

- There are three significant downsides:
  - It is too slow for any meaningful analysis on long or fast streams
  - It depends on the availability of the previous data
  - The runtime grows with the number of observed sequences.

Representative Matrix Profile

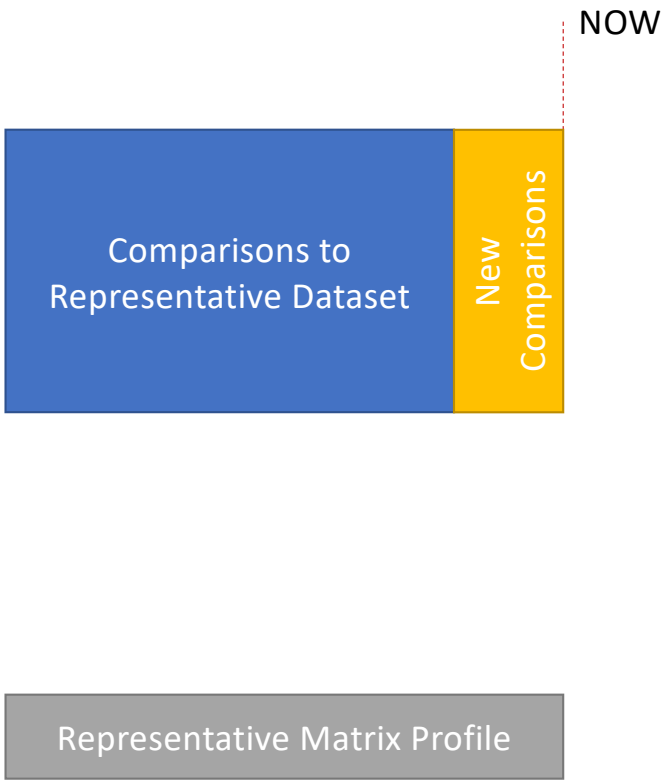
- **Key Assumption:** From a stream **S** we have prior-observed data **R** that is representative of future observations.



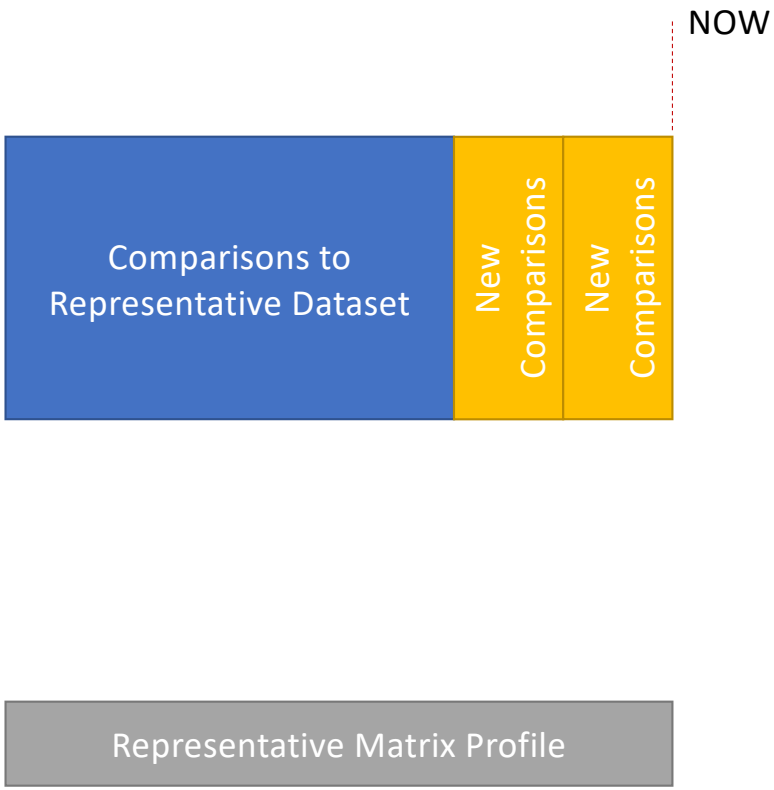
- This results in a good approximation of the Exact (Oracle) MP.

Representative MP Example

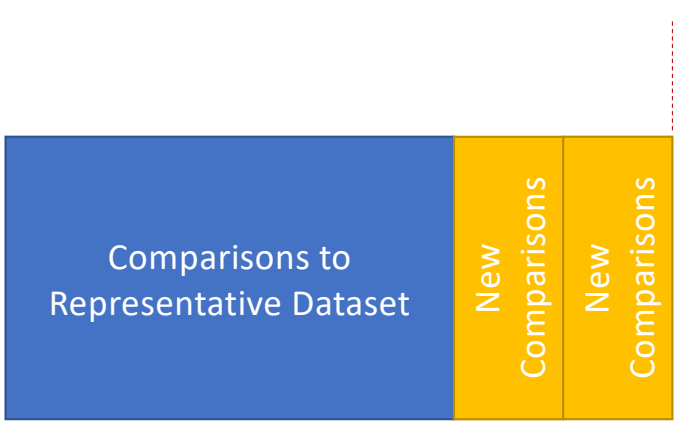
Representative MP Example



Representative MP Example



Representative MP Example

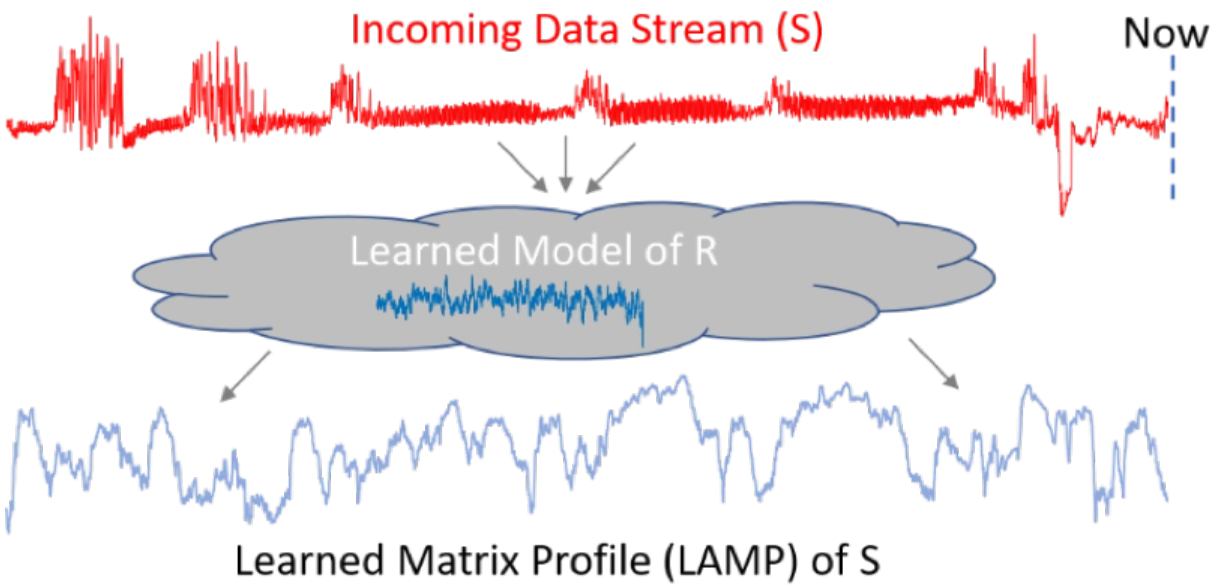


STILL O(|R|) per new subsequence!

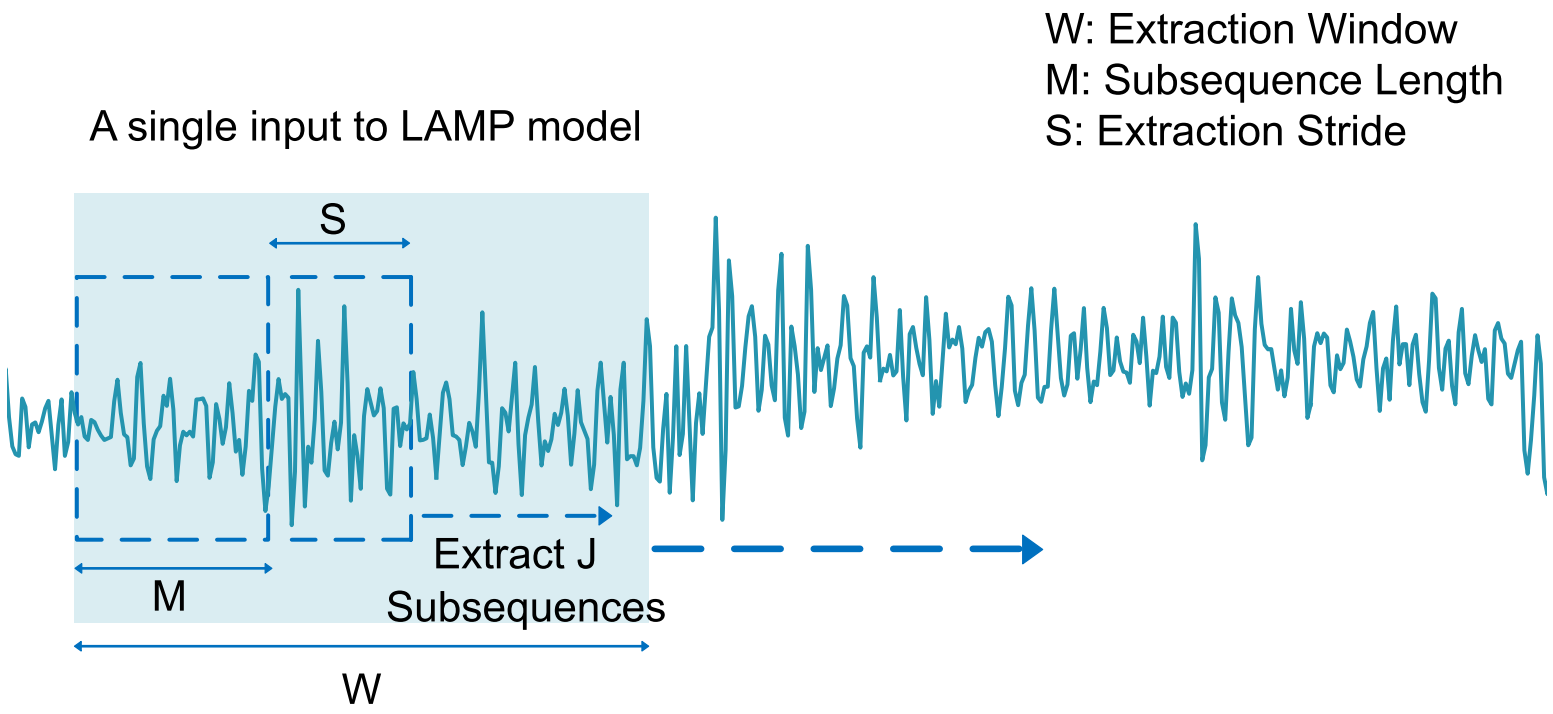
Learned Approximate Matrix Profile (LAMP)

<u>Objectives</u>

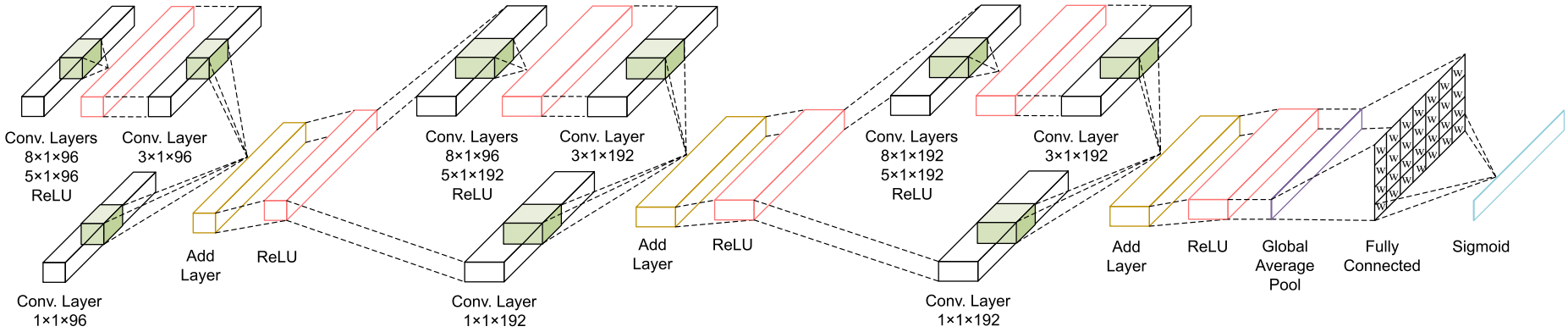
- Be fast enough to operate in real-time
- Be able to operate with power/memory/disk constraints.
- Be dataset agnostic



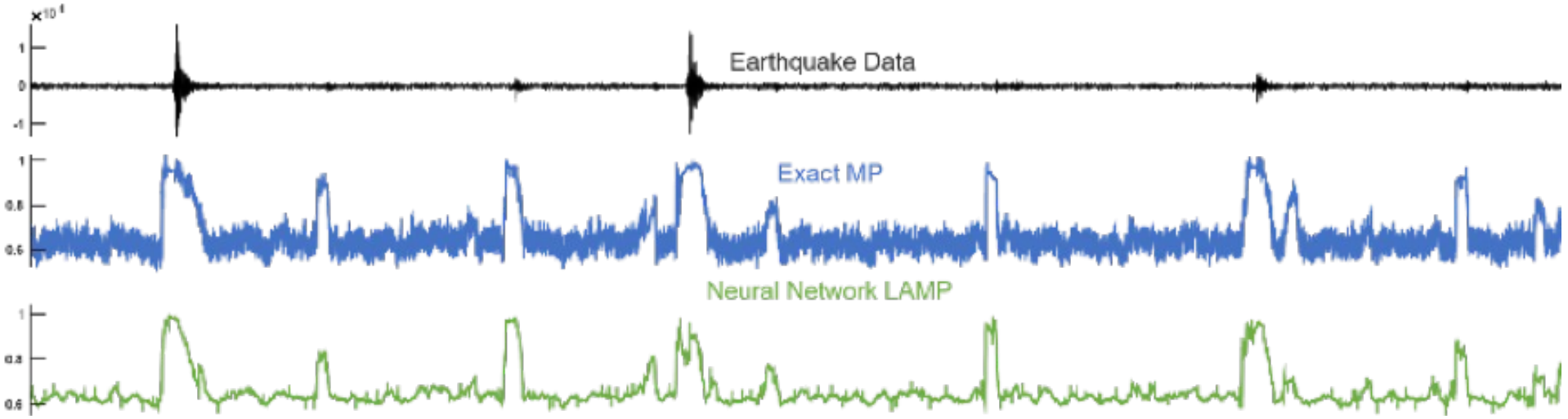
Approximate the Matrix Profile…



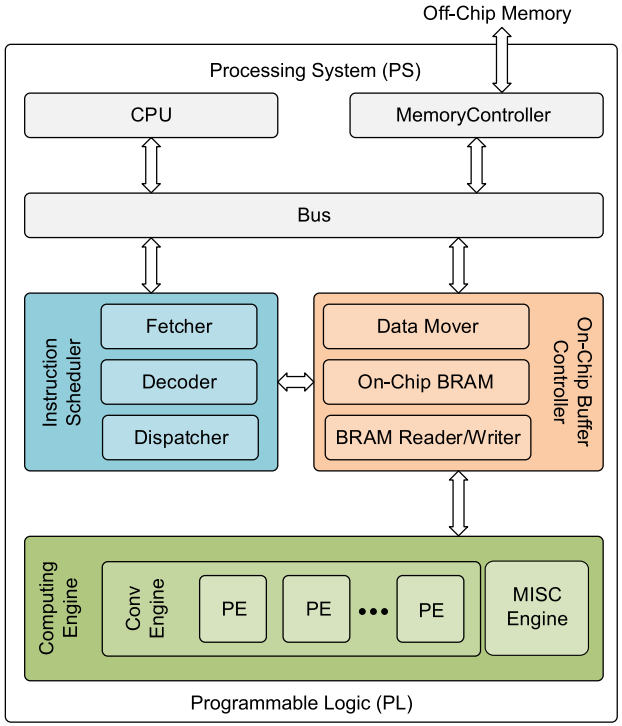
… Using a Convolutional Neural Network



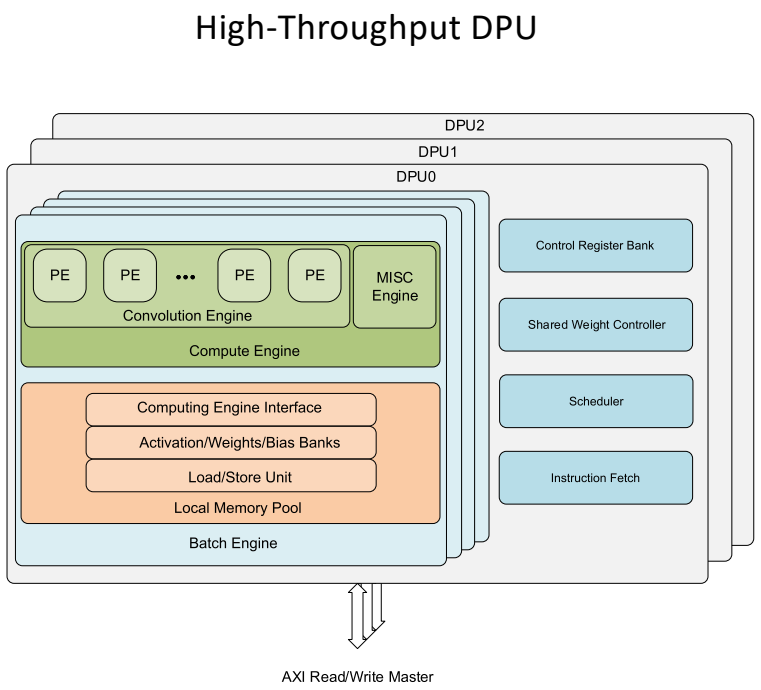
Assessment: Seismic Dataset



Edge-Scale Deployment: AMD/Xilinx Zynq

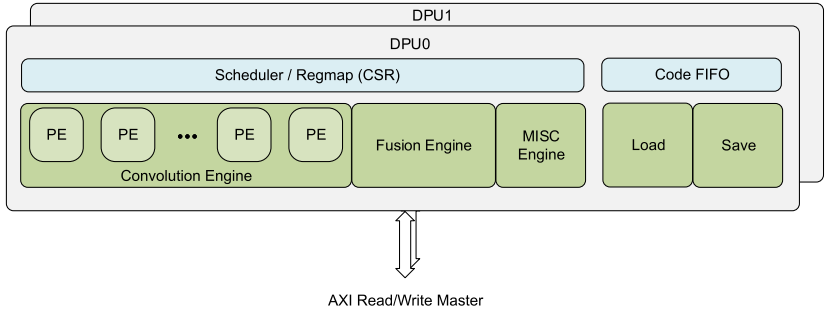


Cloud Deployment: AMD/Xilinx Alveo U280

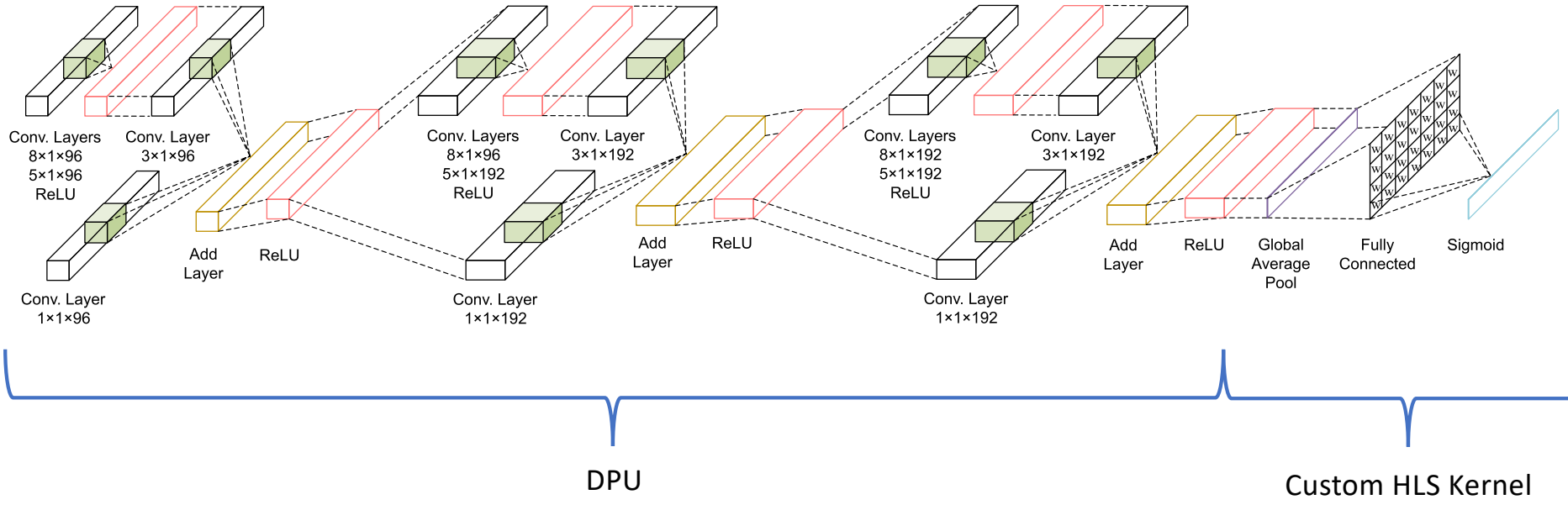


Cloud Deployment: AMD/Xilinx Alveo U280

Low-Latency DPU

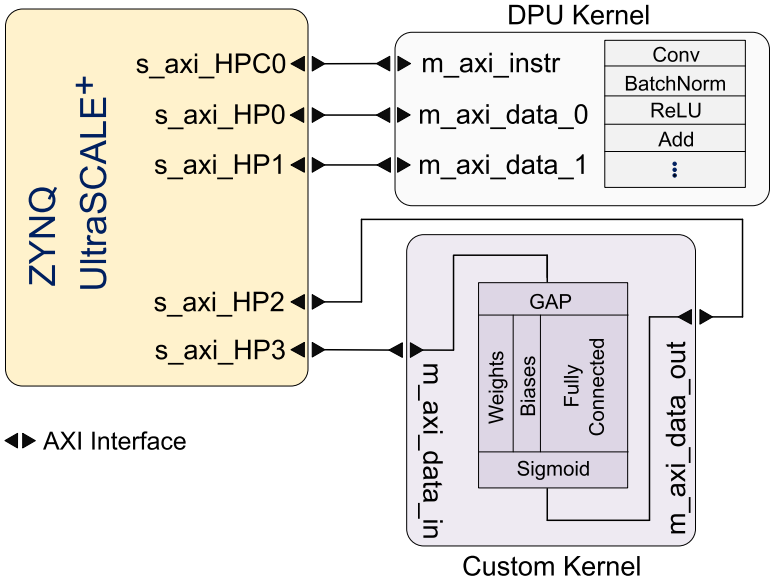


CNN Partitioning

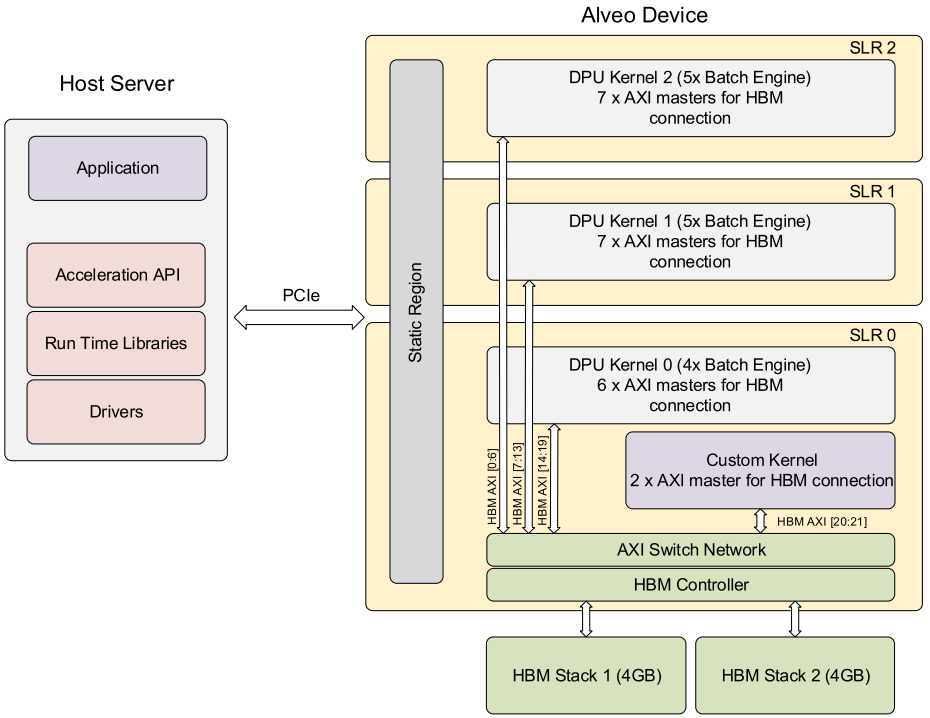


- Note: When the work was done, the DPU did not support Global Average Pool or Sigmoid Layers. Newer DPUs support Global Average Pool.

Zynq Implementation



Alveo U280 Implementation



Global Average Pool (GAP) Kernel

Input: Array of feature maps $D \in \mathbb{R}^{M \times N}$ (N: # of channels)

Output: N-dimensional vector $q \in \mathbb{R}^{N}$ consisting of the average value of each feature map.

$$q_j \leftarrow \frac{1}{M} \sum_{i=1}^{M} D_{i,j}, \quad 1 \leq j \leq N$$

Fully-Connected Layer

Input: Feature vector $q \in \mathbb{R}^N$

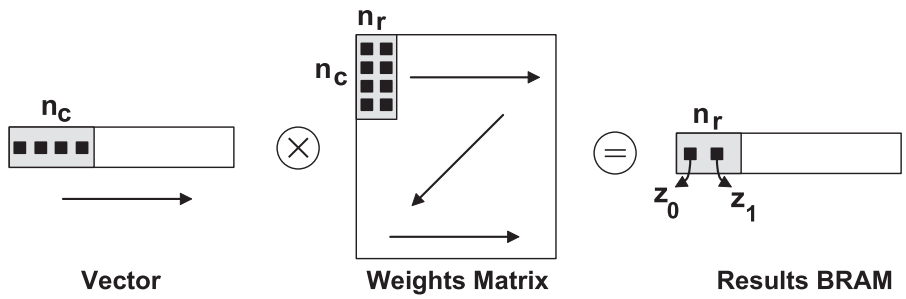
Output: Feature vector $z \in \mathbb{R}^M$

$$z \leftarrow qW + b$$

Weight Matrix: $W \in \mathbb{R}^{N \times M}$

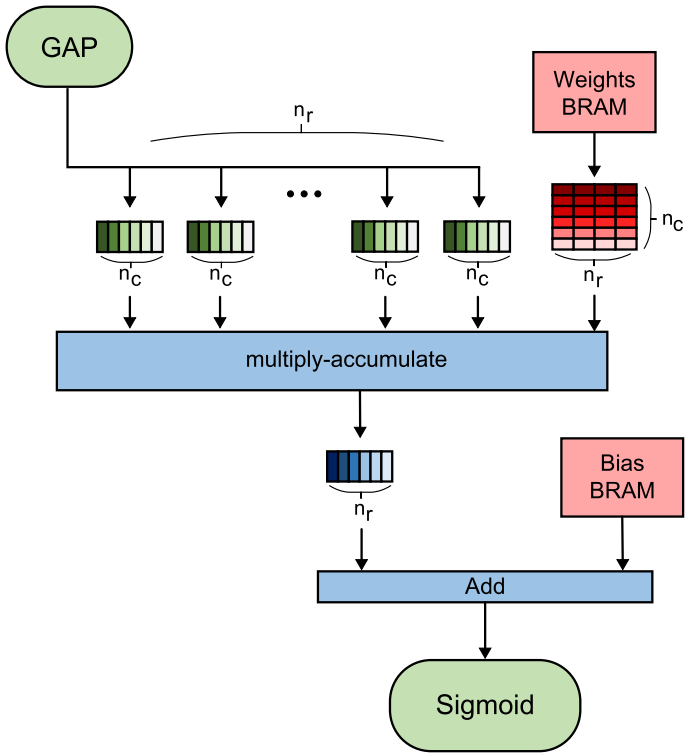
Bias Vector: $b \in \mathbb{R}^M$

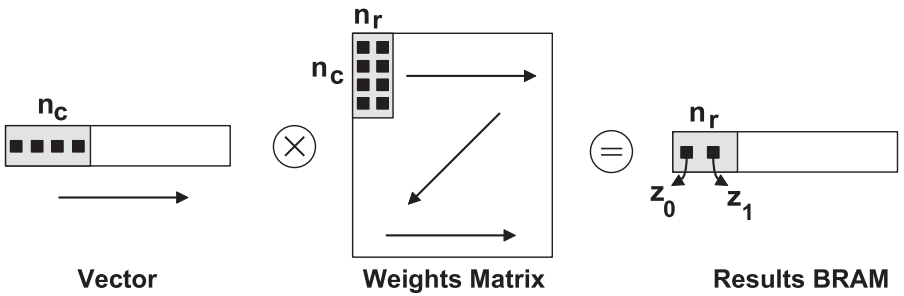
Tiling Scheme



Note: Bias vector initialized in BRAM

Fully Connected Layer: Architecture





Sigmoid Layer

- Logistic Sigmoid Function: $f(x) = \frac{1}{1+e^{-x}}$
- Challenges: Exponentiation and Reciprocal (Division) are expensive
- Solution: HW-friendly approximations

Sigmoid Approximations

ultra_fast_sigmoid (Theano library)

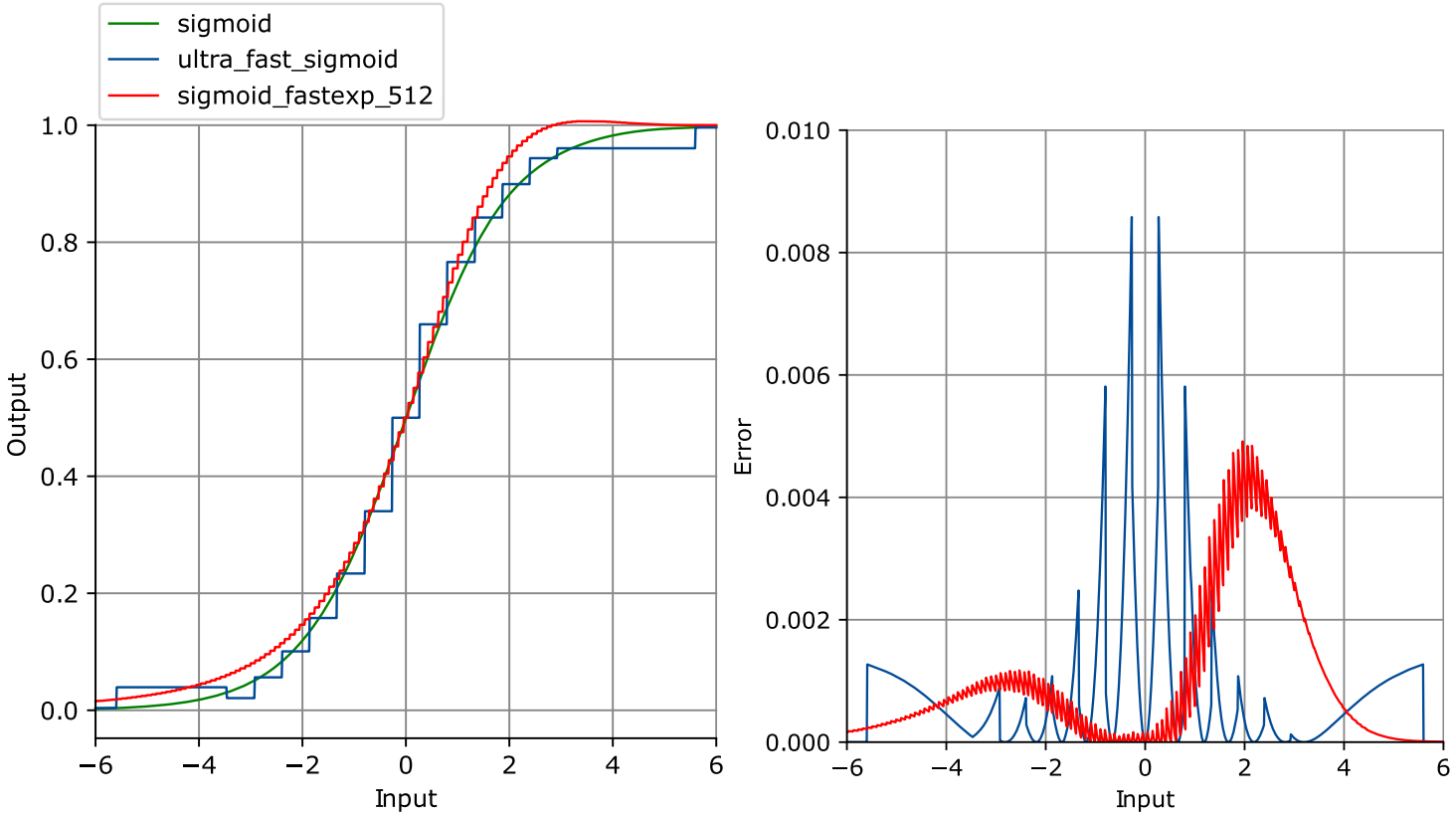
$$f(x) = \begin{cases} 0.5\left(\frac{\frac{1.5x}{2}}{1+\frac{x}{2}} + 1\right) & 0 \le \frac{x}{2} < 1.7 \\ 0.5(1 + 0.935 + 0.045\left(\frac{x}{2} - 1.7\right)) & 1.7 \le \frac{x}{2} < 3 \\ 0.5(1 + 0.995) & \frac{x}{2} \ge 3 \\ 0.5\left(-\frac{\frac{-1.5x}{2}}{1-\frac{x}{2}} + 1\right) & -1.7 \le \frac{x}{2} \le 0 \\ 0.5\left(1 - \left(0.935 + 0.045\left(-\frac{x}{2} - 1.7\right)\right)\right) & -3 < \frac{x}{2} \le -1.7 \\ 0.5(1 - 0.995) & \frac{x}{2} \le -3 \end{cases}$$

fast_exp_512

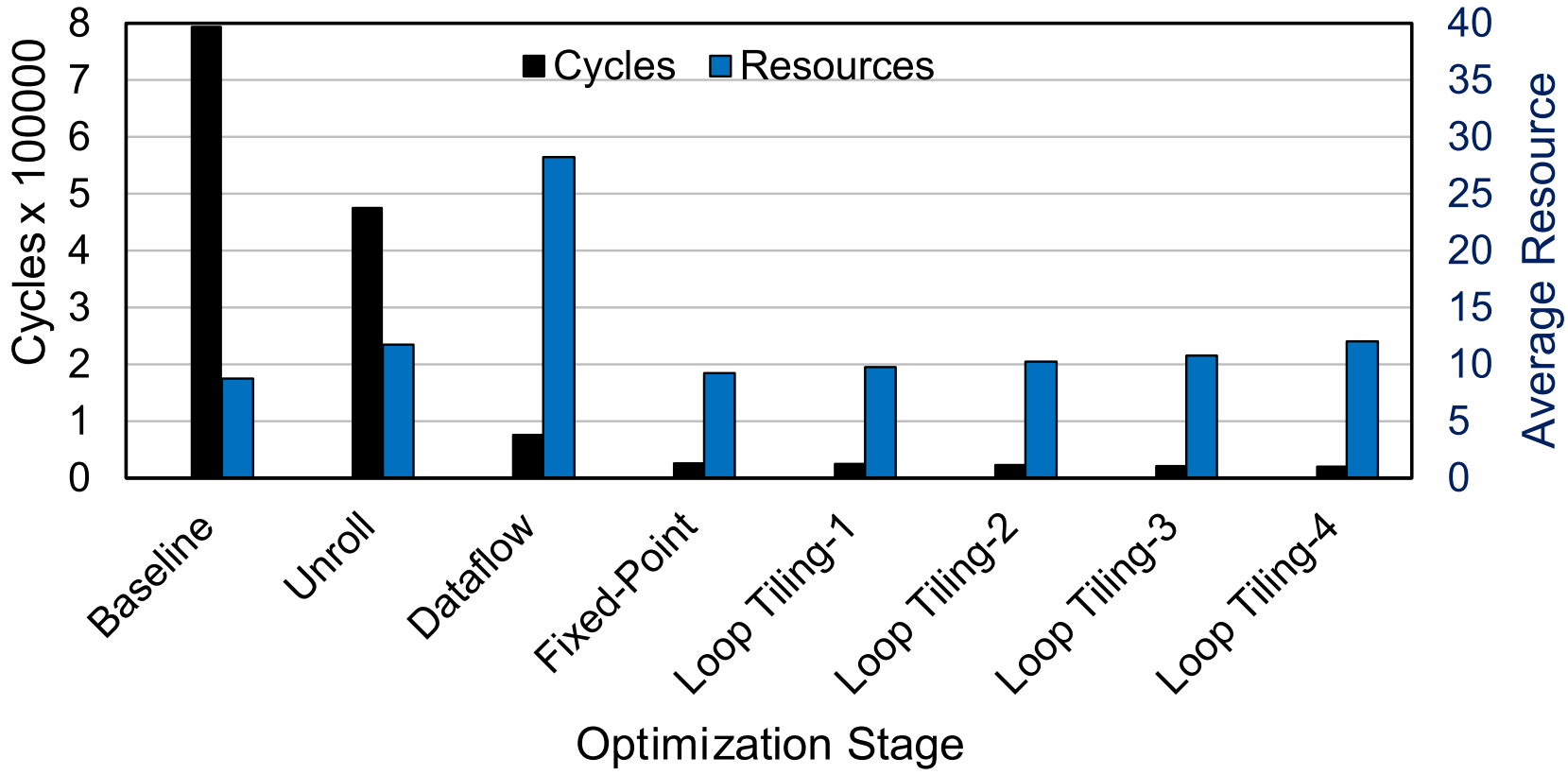
$$\exp(x) = \prod_{k=1}^{lg(n)} \left(1 + \frac{x}{k}\right)^k, \quad n = 512,$$

$$\mathrm{sigm}(x) = \frac{1}{1 + \exp(-x)}$$

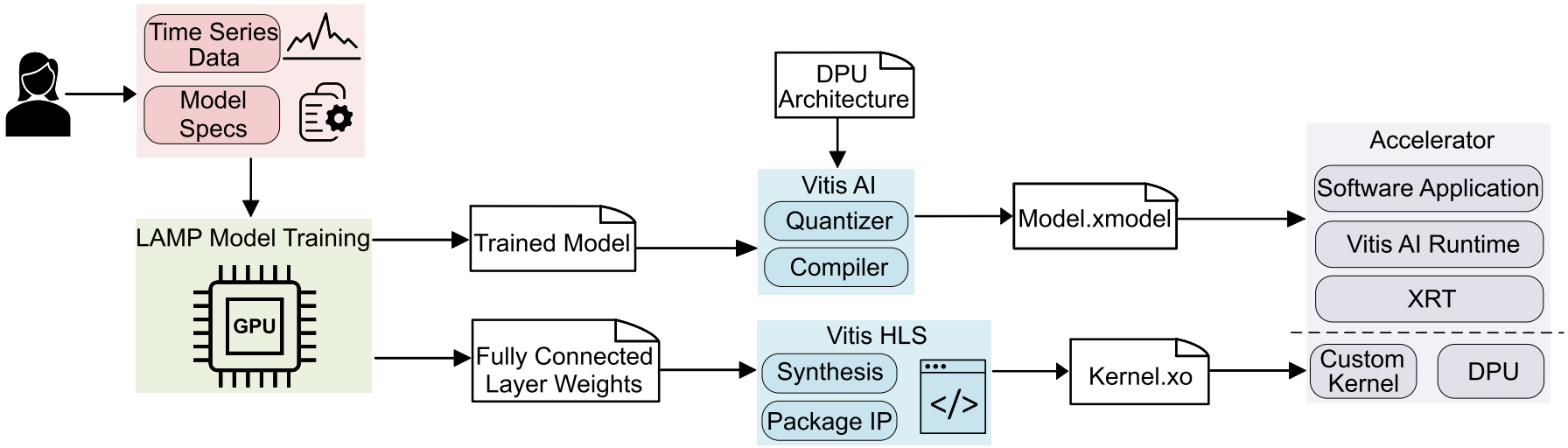
Sigmoid Approximations and their Error (8-bit Fixed-point)



HLS Optimizations

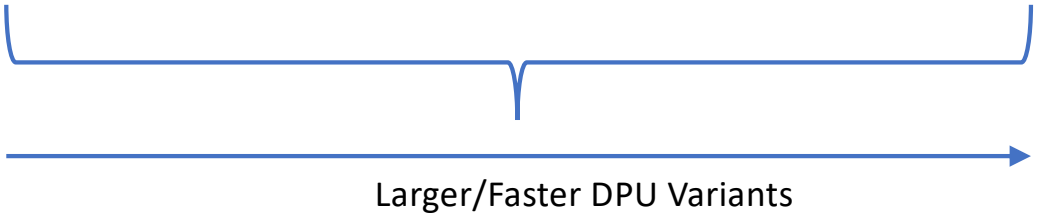


Model Training Flow



Results: Edge Prototype

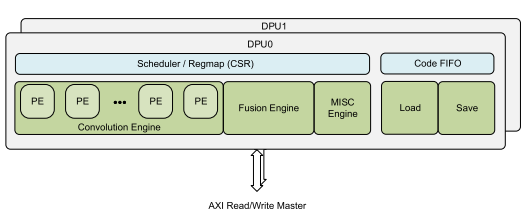
	DPU + ARM						DPU + HLS Kernel	
	DPU + Arm (B512)	DPU + Arm (B800)	DPU + Arm (B1024)	DPU + Arm (B1152)	DPU + Arm (B1600)	DPU + Arm (B2304)	DPU + IP (ultra_fast)	DPU + IP (fastexp_512)
Logic Usage	39K (56%)	42K (59%)	46K (65%)	44K (62%)	49K (70%)	52K (75%)	57K (82%)	60K (86%)
Register Usage	50K (36%)	57K (40%)	65K (46%)	64K (45%)	77K (54%)	87K (62%)	95K (67%)	100K (71%)
DSP Usage	78 (21%)	117 (32%)	154 (42%)	164 (45%)	232 (64%)	289 (79%)	290 (80%)	326 (90%)
On-chip RAM Usage	77 (35%)	95 (44%)	109 (50%)	127 (58%)	131 (60%)	171 (79%)	174 (81%)	174 (81%)
Throughput (GOPS)	70.4	107.0	154.2	167.6	220.2	367.1	**453.5**	**428.3**
Peak Throughput (GOPS)	153	240	307	345	480	691	691	691

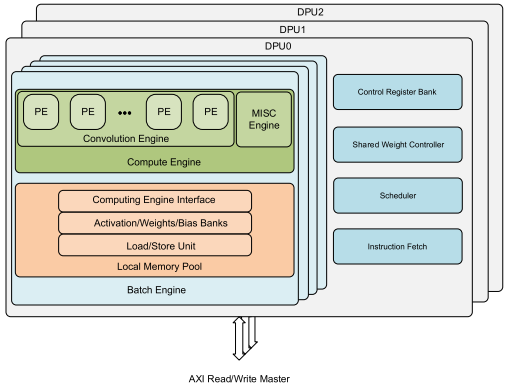


Results: Edge vs. TPU / Raspberry Pi

	Edge TPU	Raspberry Pi 3	DPU + Arm	DPU + IP ultra_fast	DPU + IP fastexp_512
Inf. Rate (Hz)	824	2.6K	12.1K	15.0K	14.2K
Energy (J)	161.4	58.8	7.2	6.7	9.1
GOPs/Watt	5.8	10.4	107.9	146.1	135.8

Results: Cloud Prototype





	GPU	Server CPU	Desktop CPU	LL + CPU	LL + IP (ultra_fast)	LL + IP (fastexp_512)	HT + CPU	HT + IP (ultra_fast)	HT + IP (fastexp_512)
Throughput (TOPS)	92.52	69.57	4.91	2.26	3.04	2.53	4.27	5.53	4.83
Latency (ms)	356	227	296	5.68	3.25	3.49	6.29	3.85	4.09
Inference Rate (KHz)	3,079	2,298	163	75	101	84	142	184	160
Energy (J)	118.32	72.20	38.43	9.15	6.81	8.20	4.86	3.75	4.29
GOPs/Watt	1,210	740	68	44	57	46	77	98	89

LL = Alveo Low-latency, HT = Alveo High-throughput.

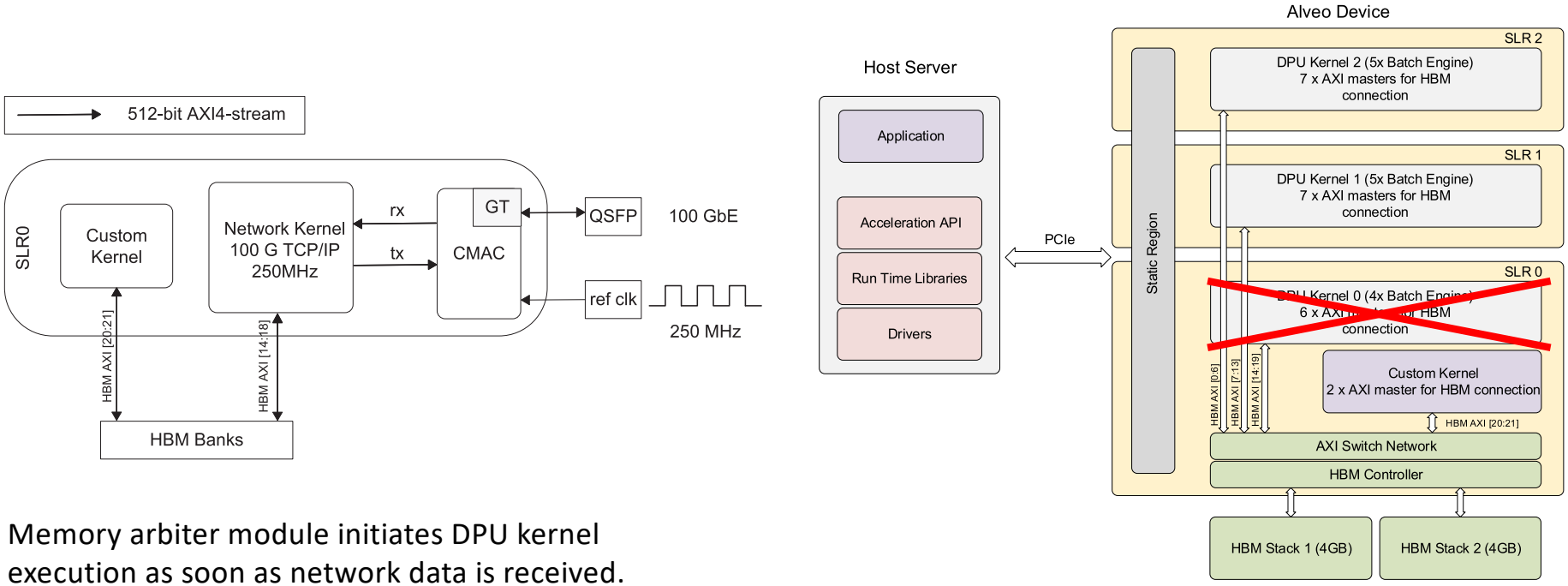
Results: Inference Accuracy

Time Series Dataset		FA-LAMP Inference Accuracy						
Name	Train/Test Split	32-bit float	edge: ultra_fast	edge: fastexp	qa_edge: ultra_fast	qa_edge: fastexp	qa_cloud: ultra_fast	qa_cloud: fastexp
Earthquake	120M/30M	97.4%	91.4%	92.5%	93.8%	94.7%	94.3%	95.1%
Insect EPG	2.5M/5M	97.2%	90.8%	93.2%	91.9%	94.4%	92.5%	94.8%
Chicken Accel.	6M/2M	95.8%	86.9%	91.1%	89.5%	93.1%	90.2%	93.7%

qa = quantization-aware Training, edge = Ultra96, Cloud = Alveo.

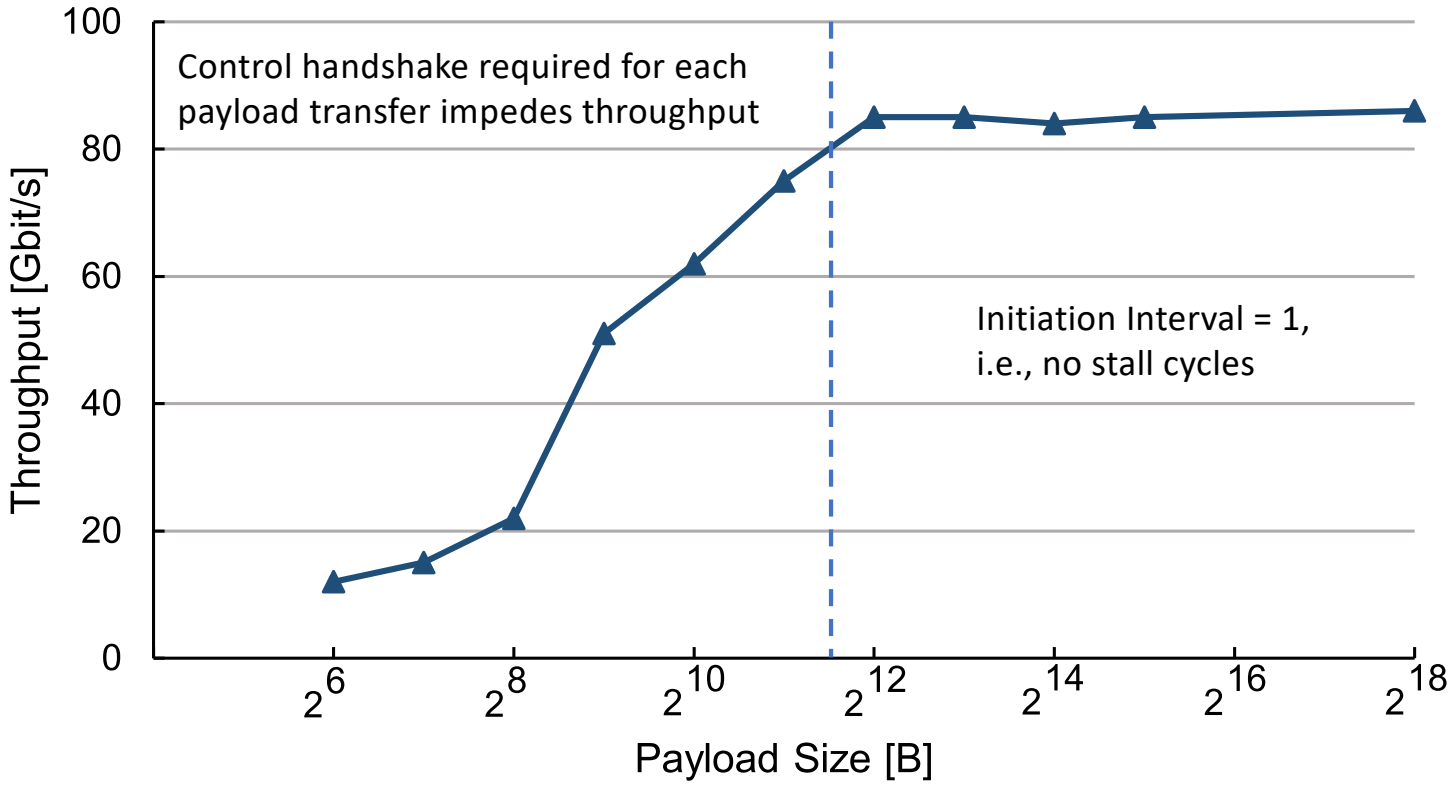
Future Work:
Can we train sigmoid approximations, rather than inserting them after training with the exact sigmoid?

Ethernet Integration (Cloud/Alveo U280)



Memory arbiter module initiates DPU kernel execution as soon as network data is received.

Throughput as a Function of Payload Size



Recap

- FPGAs are highly amenable to streaming applications
- Offline time series algorithms often need significant modifications to work in a streaming context
- Using a neural network to predict the output of an offline algorithm is one of many possible approaches to convert offline to streaming
  - Simpler approaches may make smaller-scale modifications, such as replacing offline normalization with streaming normalization
- Think about context: is a given streaming application better deployed in the cloud or in the edge?

Future Directions

- Can the catch-22 features be computed in a streaming fashion?
- What can be done with streaming catch-22 features once they are computed?
  - This tutorial: combine with the Matrix Profile
  - Immediate Thought: does it make sense to predict the catch-22 features in a manner similar to LAMP?
  - Future Work: Real-time machine learning using catch-22?
    - What type of machine learning models? (Neural nets? Something else?)
    - Can we detect concept drift in streaming data from catch-22 observations?
    - When concept drift occurs can we incrementally retrain our model(s) to compensate?

Numerical Precision Challenges

- Most papers on time series (offline or online) assume 32-bit floats or 64-bit doubles
- If you use a neural network you can train with other formats (e.g., bfloat) and quantize down to fixed-point
- Many numerical stability problems exist for very large time series (e.g., billions of datapoints) for both online and offline algorithms
  - e.g., see Rakthanmanon et al., KDD 2012 for online normalization
  - Many downstream implications for online algorithms that process an online-normalized time series

Approximate Arithmetic Opportunities

- Benefits: smaller, faster, more energy-efficient than exact arithmetic operators
- Challenges: arithmetic operators will be incorrect for some input bit combinations
- Key questions:
  - Does approximate arithmetic impact Matrix Profile computation?
  - Does approximate arithmetic impact catch-22 feature computation?
  - Does approximate arithmetic impact the conclusions that can be drawn?