Capybara final update

03/28/2017

Outline

- Goals
- Datasets
- HTM network
- Dimensionality reduction techniques
- Supervised classification
- Unsupervised classification
- Conclusion

Goals

Capybara goals

Main goal

- Create a canonical example for **online** and **semi-supervised classification** using a **HTM network**.
  - This example is intended to be in the same spirit as the Hot Gym examples (anomaly detection and prediction) but for classification.

Detailed goals

1. Show value added of HTMs for unsupervised sequence classification
2. Collect meaningful datasets
3. Create reusable frameworks for the NuPIC community
4. Create visualizations of classification results

Status

- Was not able to complete subgoal #1
- Completed subgoals 2, 3 and 4.

Goal of this update

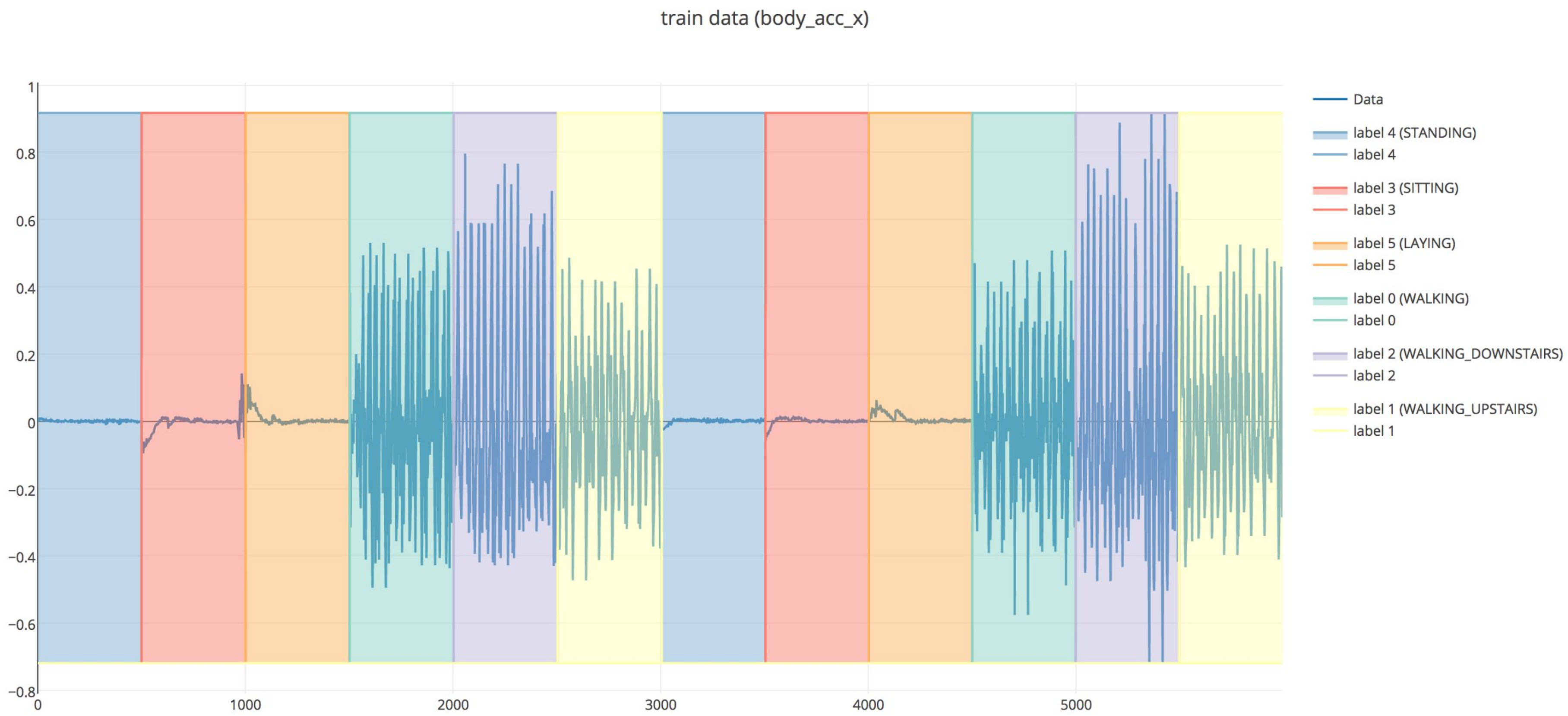
- Help anyone who would pick the project up again to:
  - Understand the project structure
  - Understand what has been tried
  - Understand what tooling is available

Datasets

Overview

- Code reference:
  - nupic.research/projects/capybara/datasets
- Dataset
  - UCI Human Activity Accelerometer data
  - UCR time series
  - Artificial data: motifs dataset and binary datasets
  - Sensortag data
  - Synapse.org data
- Tooling:
  - Scripts to download, format and visualize the input datasets.
  - Script to chunk datasets in smaller, homogeneous sequences

Dataset 1: UCI data

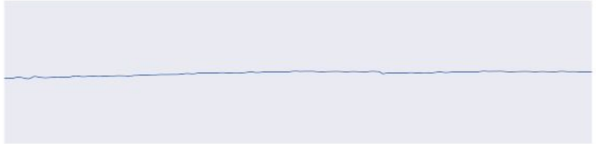


UCI data sequences

Label 1: Walking

Label 2: Walking upstairs

Label 3: Walking downstairs



Label 4: Sitting

Label 5: Standing

Label 6: Laying

Dataset 2: UCR data

Label 1: Random noise

Label 2: Cyclic

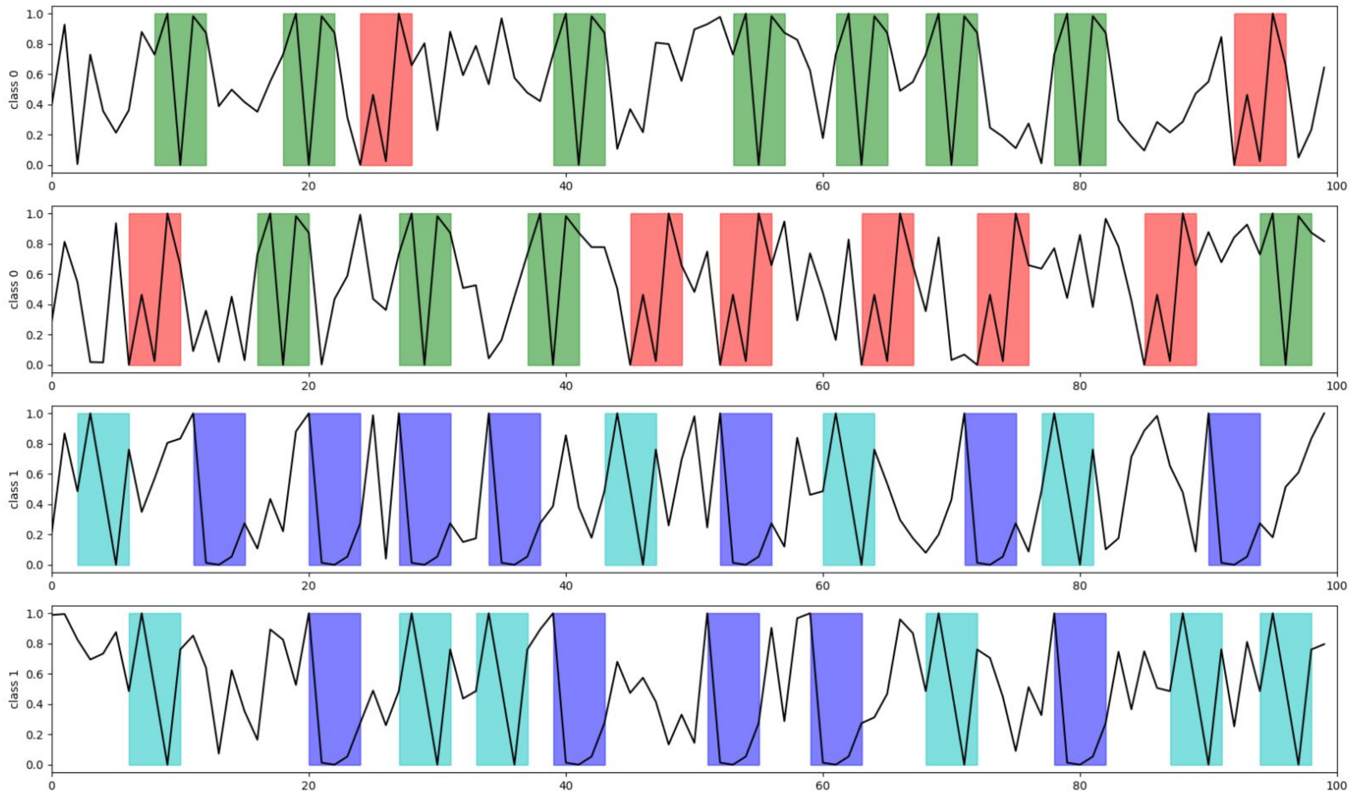
Label 3: Increasing trend

Label 4: Decreasing trend

Label 5: Upward shift

Label 6: Downward shift

Dataset 3: Motifs



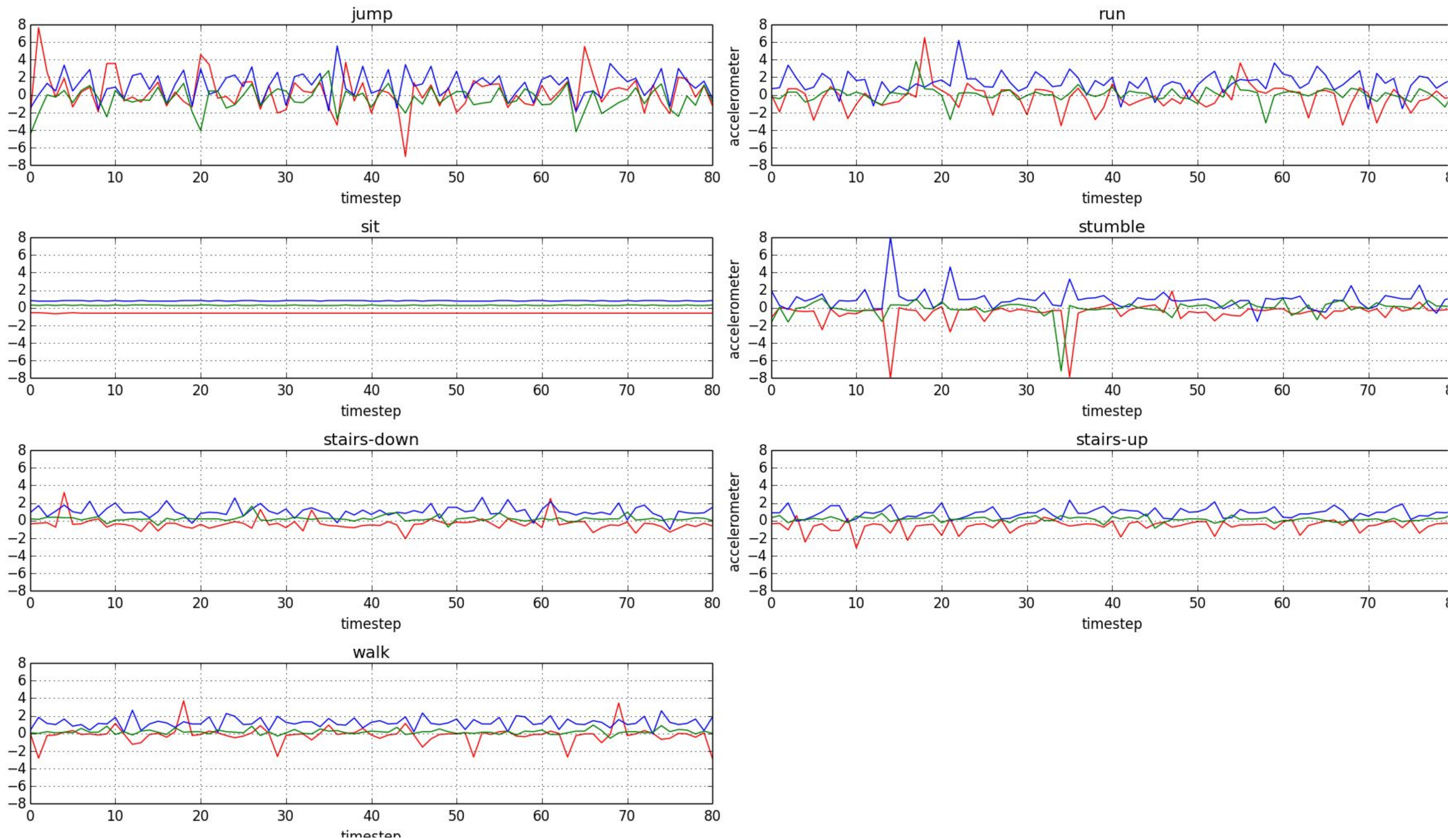
Label 1

Label 1

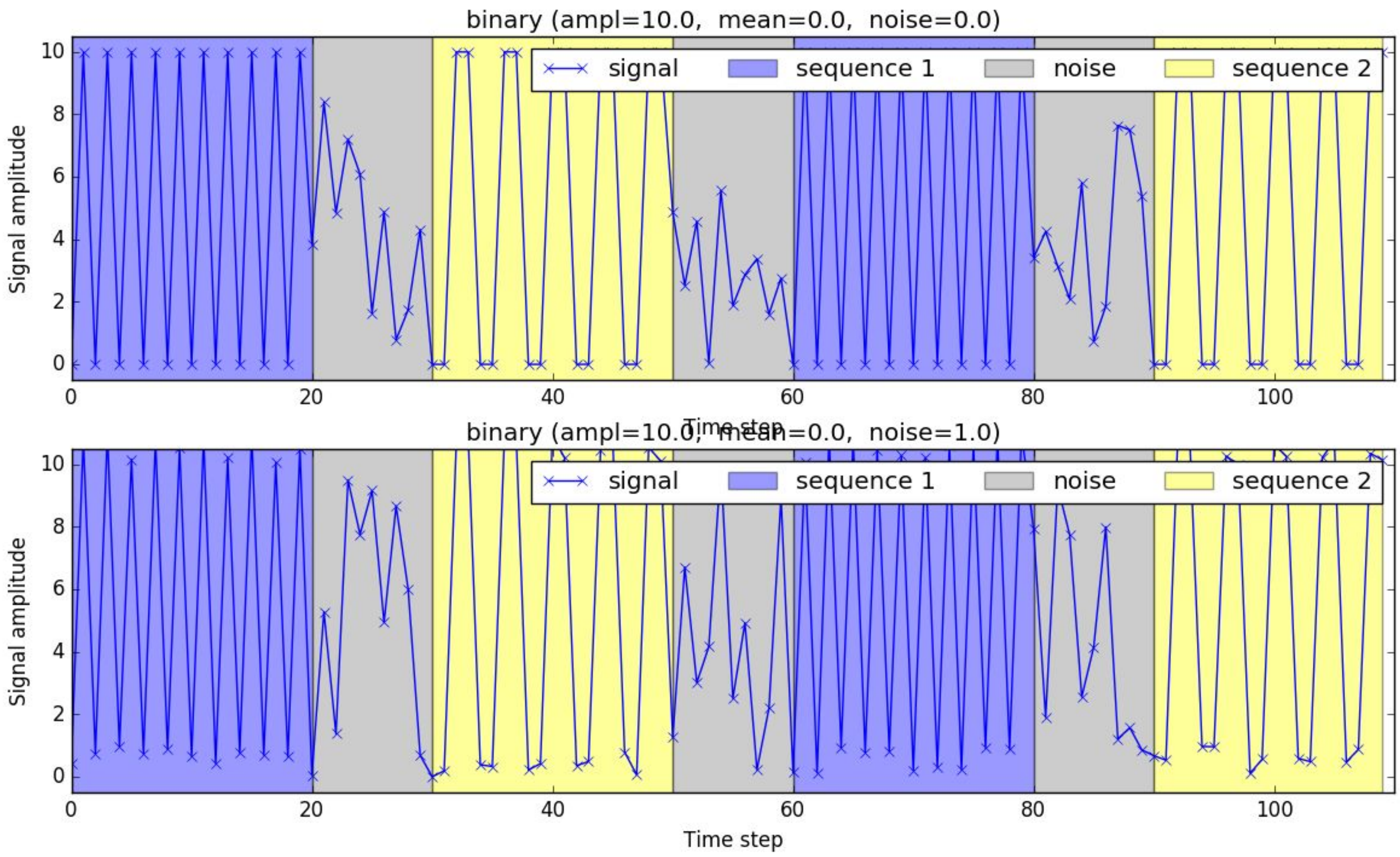
Label 2

Label 2

Dataset 4: Sensortag accelerometer data



Dataset 5: artificial binary data



HTM network

Network factory

- Code reference
  - htmresearch.frameworks.capybara.htm.network
- Goal
  - Run an HTM network on several datasets and generate HTM traces.
  - HTM traces: non-zero indices of SP and TM cells over time, saved as a CSV.
- Tooling
  - Network factory class that uses a YAML as input to create a simple network with the research API.
  - Can take 2 types of datasets as input:
    - Time-indexed
    - Sequence-indexed

Time-indexed VS sequence-indexed

- All datasets are labelled time series.
- For each time step, there is a value and a class (label) →

	body_acc_x	label
t0	0.000181	4
t1	0.010139	4
t2	0.009276	4
t3	0.005066	4
t4	0.010810	4

- Each dataset is split into smaller sequences (~100 points)
- nce have the same label)

	label	t0	t1	t2	t3	t4
sequence_0	4	0.000181	0.010139	0.009276	0.005066	0.010810
sequence_1	4	0.002162	-0.000946	-0.006476	-0.003423	-0.000610
sequence_2	4	-0.001637	-0.000097	0.001614	0.002619	0.004765
sequence_3	4	-0.001015	0.001832	0.001169	0.000362	-0.002587
sequence_4	4	-0.000353	0.000120	0.002108	0.002159	0.001069

Dimensionality reduction

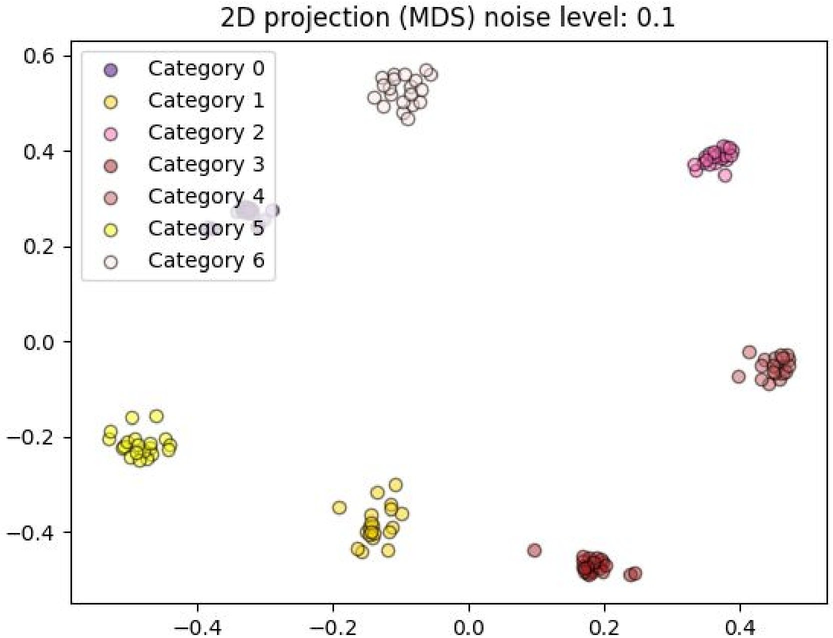
Overview

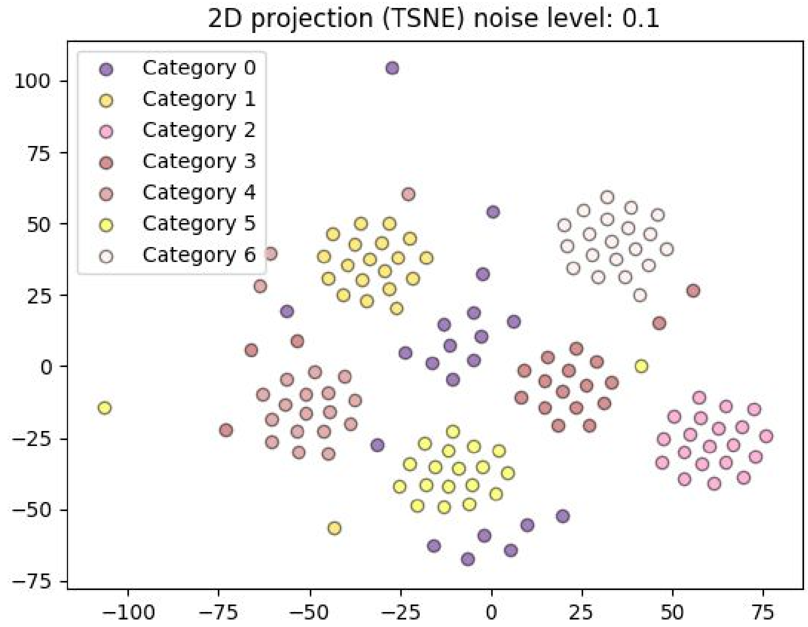
Code reference: nupic.research/projects/capybara/dim_reduction

Dimensionality reduction algorithms to visualize clusters of SDRs

- MDS
- tSNE

MDS & T-SNE on artificially generated SDRs





Supervised classification

Approach #1

Code reference

- nupic.research/projects/capybara/supervised_baseline/v1_no_sequences

Overview

- With time-indexed sequences
- Classify a rolling union of SDRs
- Voting mechanism to smooth out predictions.
- This approach did not work well

Approach #2

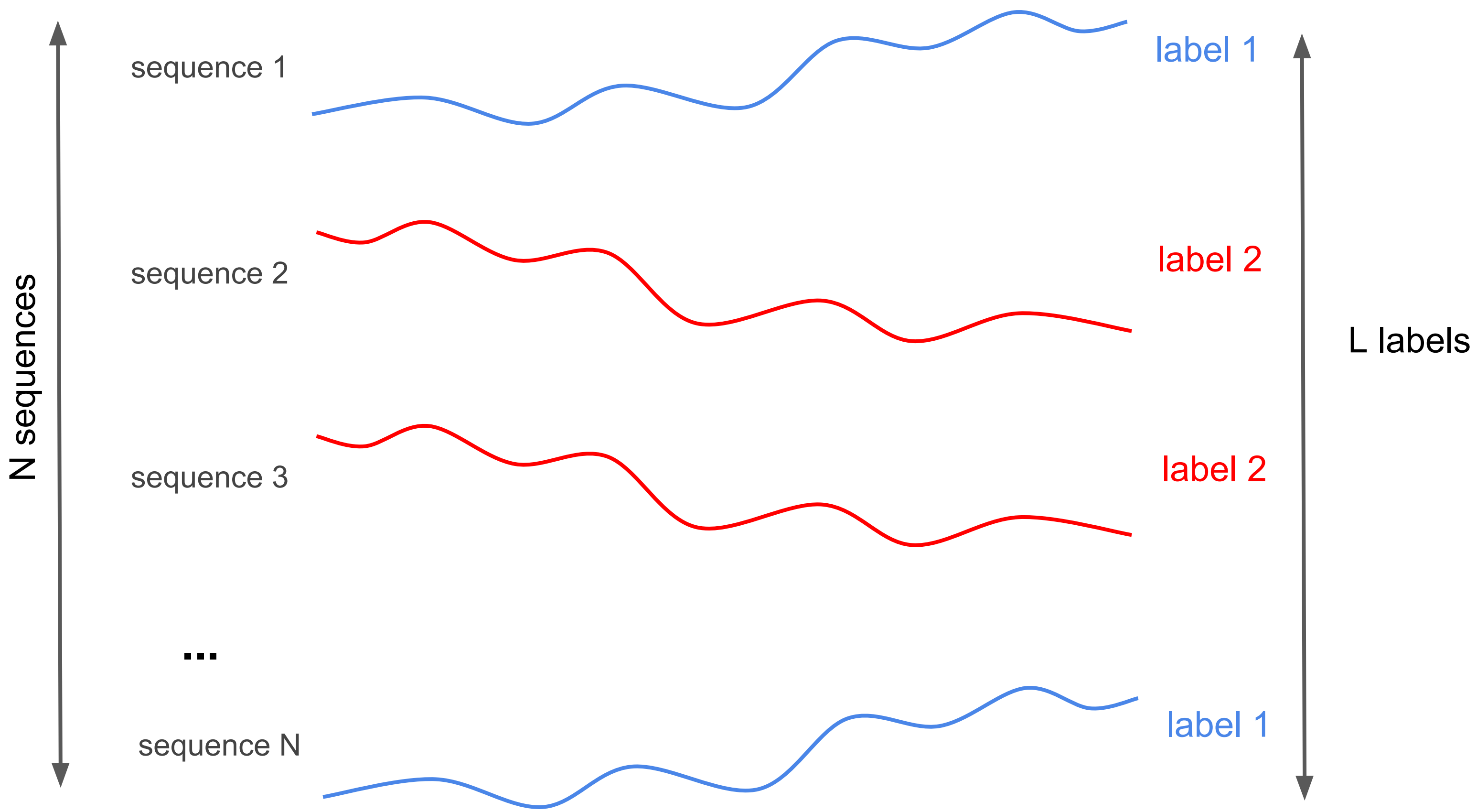
Code reference:

- nupic.research/projects/capybara/supervised_baseline/v2_with_sequences

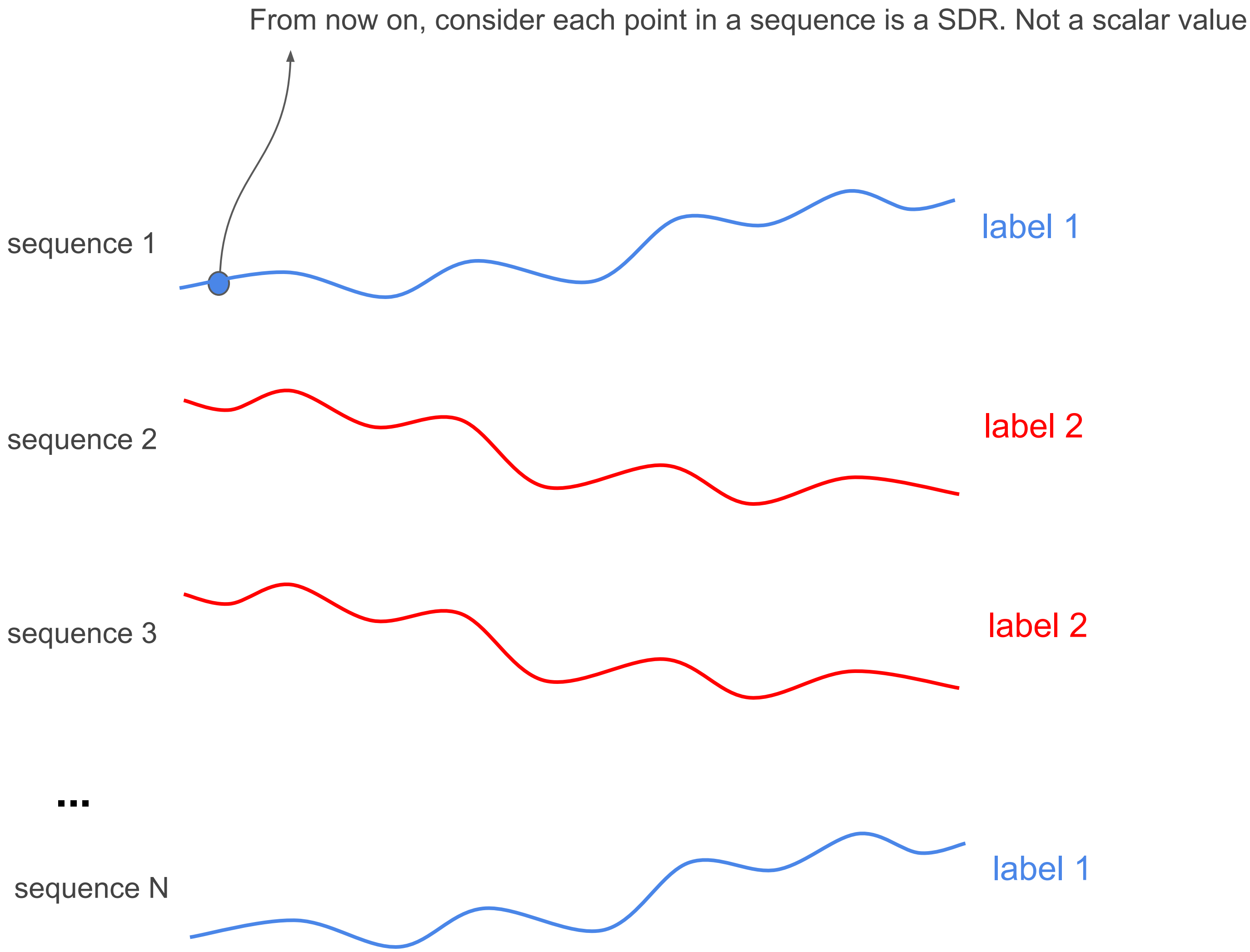
Instead of classifying a rolling union of SDRs:

- Chunk sequences
- Create an embedding of sequence chunks
  - Average
  - Logical and
- Use the euclidian distance to determine the distance between embeddings

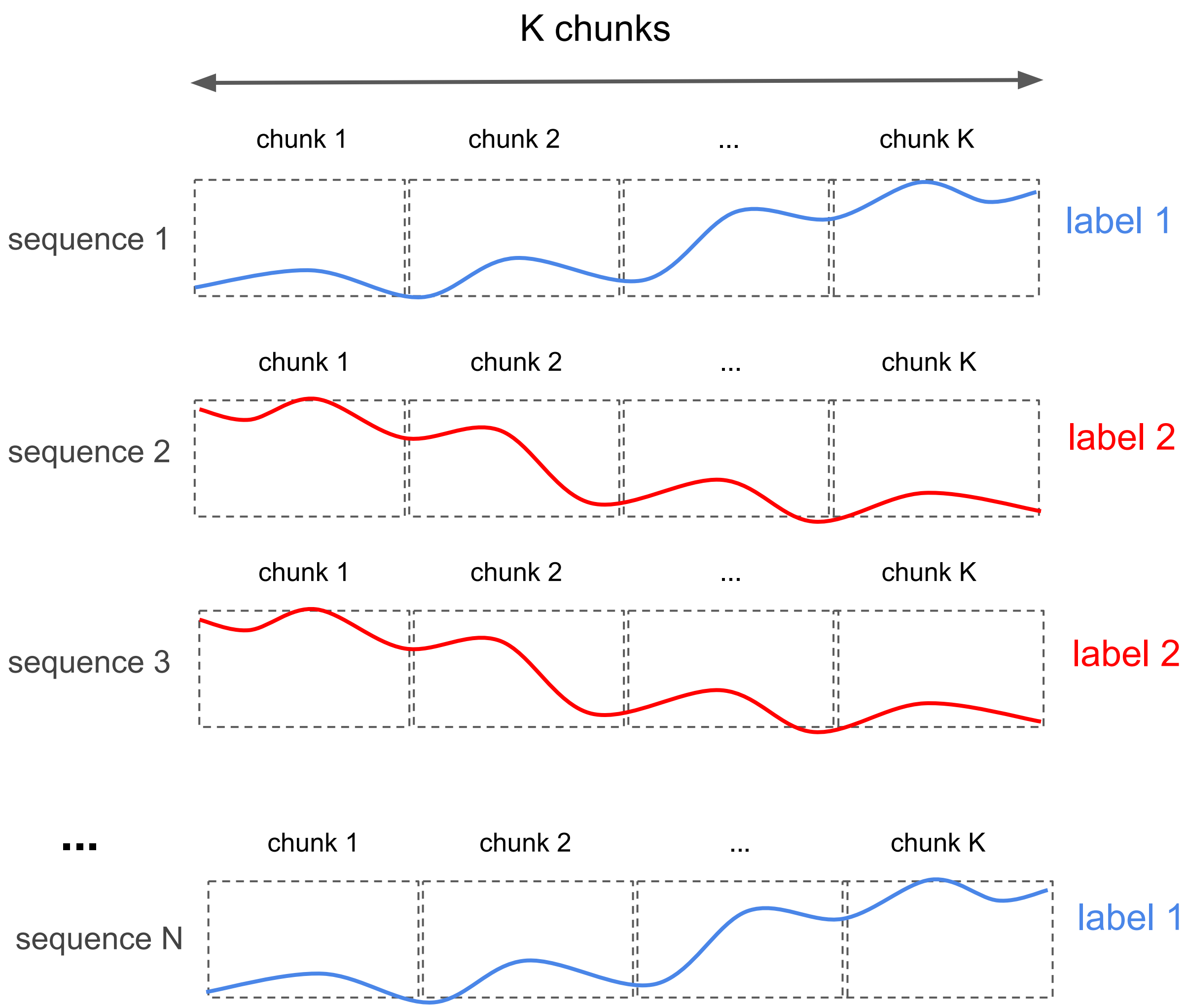
Example


N sequences
sequence 1
sequence 2
sequence 3
...
sequence N
label 1
label 2
label 2
label 1
L labels

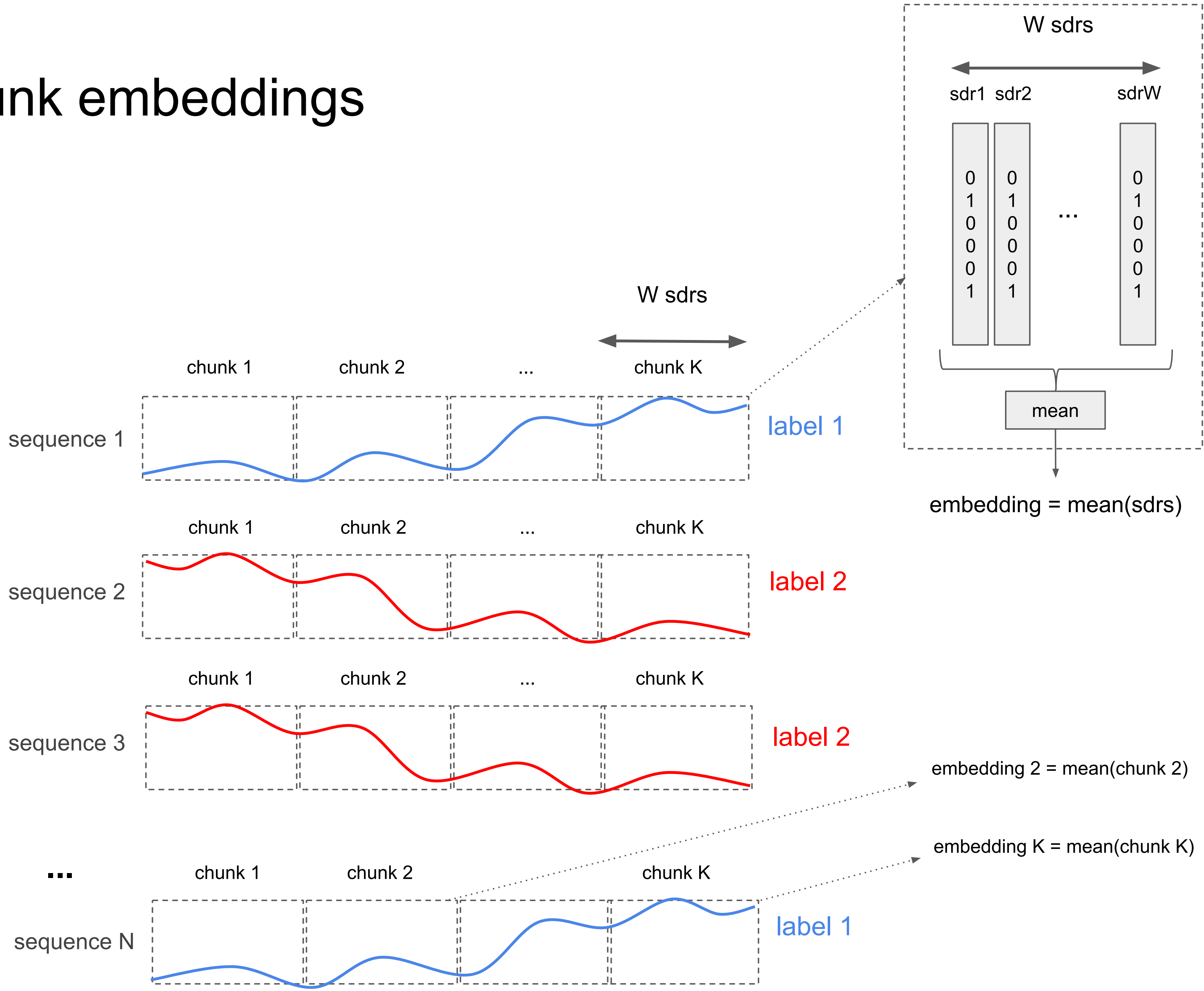
Run HTM on each sequence



Sequence chunking



Chunk embeddings



Classification

Training data:

- Sequence 1: **label**, **[embedding 1, …, embedding K]**
- Sequence 2: **label**, **[embedding 1, …, embedding K]**
- Etc .

Classifier:

- 1-NN
- Distances
  - Euclidian
  - Logical and

Results

- Increasing the number of chunks increases the accuracy
- In some cases (like the UCR synthetic data), adding the TM does not make a difference

Test accuracy (SP)

- UCI accelerometer data
  - # chunks = 1: 51.85%
  - # chunks = 2: 77.78%
  - # chunks = 5: 85.19%
- UCR synthetic control
  - # chunks = 1: 27.33%
  - # chunks = 2 75.67%
  - # chunks = 5: 80.67%
- Motifs
  - # chunks = 1: 97.00%
  - # chunks = 2 96.00%
  - # chunks = 5: 98.00%

Test accuracy (TM)

- UCI accelerometer data
  - # chunks = 1: 44.44%
  - # chunks = 2: 44.44%
  - # chunks = 5: 40.74%
- UCR synthetic control
  - # chunks = 1: 18.67%
  - # chunks = 2 47.33%
  - # chunks = 5: 73.33%
- Motifs
  - # chunks = 1: 100.00%
  - # chunks = 2 100.00%
  - # chunks = 5: 93.00%

Plot associated with these results live here:
https://github.com/numenta/nupic.research/tree/master/projects/capybara/supervised_baseline/v2_with_sequences/plots_assume_aligned

Problem?

- This assumes sequence alignment.
- New distance: does not assume sequence alignment
- The same distance is used in the clustering experiments.
- Results are worse:

Test accuracy (SP)

- UCI accelerometer data
  - # chunks = 1: 51.85%
  - # chunks = 2: 77.78%
  - # chunks = 5: 55.56%
- UCR synthetic control
  - # chunks = 1: 27.33%
  - # chunks = 2 49.33%
  - # chunks = 5: 48.67%
- Motifs
  - # chunks = 1: 97.00%
  - # chunks = 2 91.00%
  - # chunks = 5: 90.00%

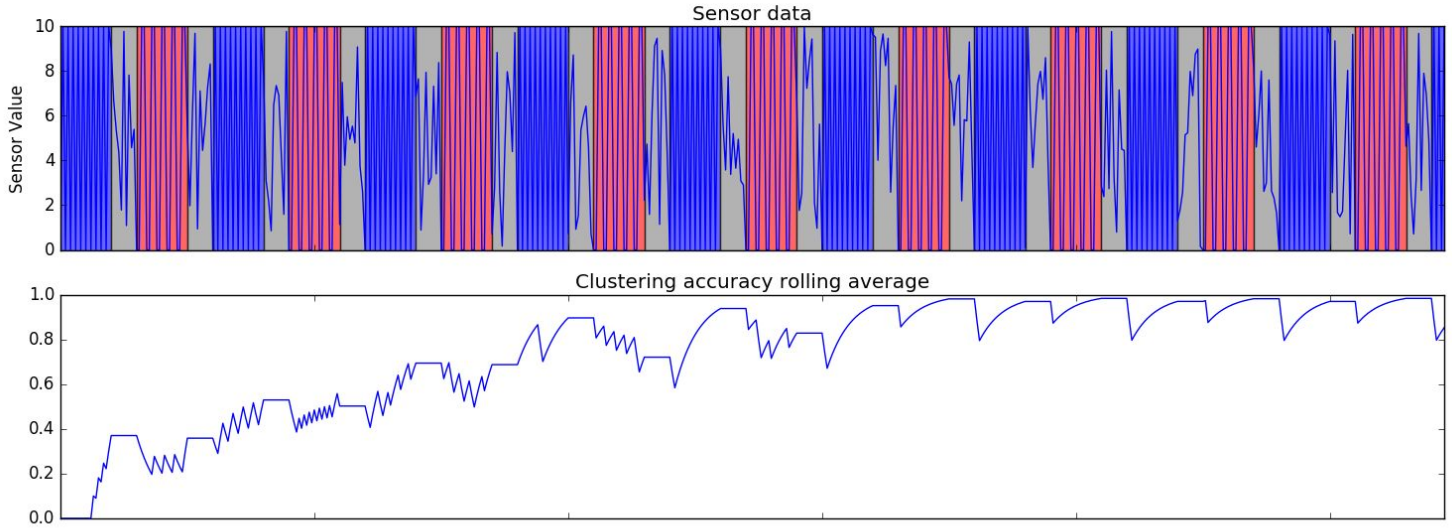
Test accuracy (TM)

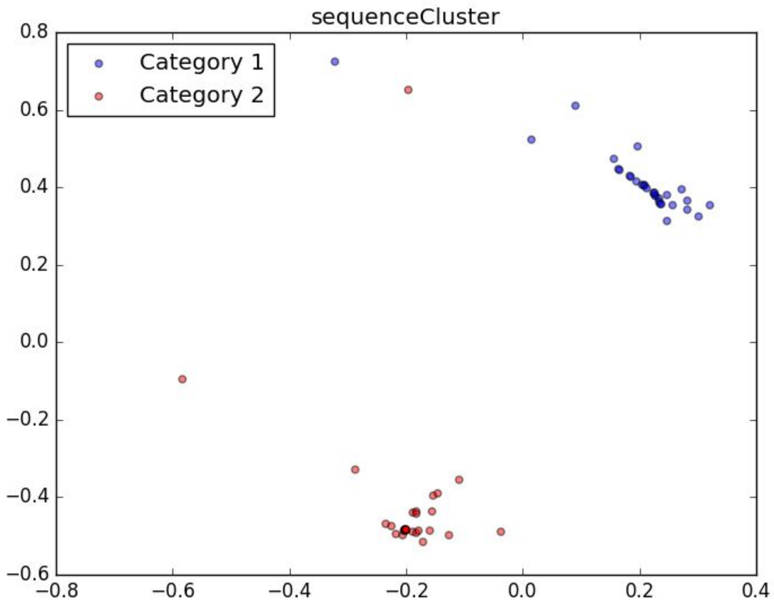
- UCI accelerometer data
  - # chunks = 1: 44.44%
  - # chunks = 2: 44.44%
  - # chunks = 5: 44.44%
- UCR synthetic control
  - # chunks = 1: 18.67%
  - # chunks = 2 23.00%
  - # chunks = 5: 29.00%
- Motifs
  - # chunks = 1: 100.00%
  - # chunks = 2 100.00%
  - # chunks = 5: 100.00%

Plot associated with these results live here:
https://github.com/numenta/nupic.research/tree/master/projects/capybara/supervised_baseline/v2_with_sequences/plots_assume_not_aligned/

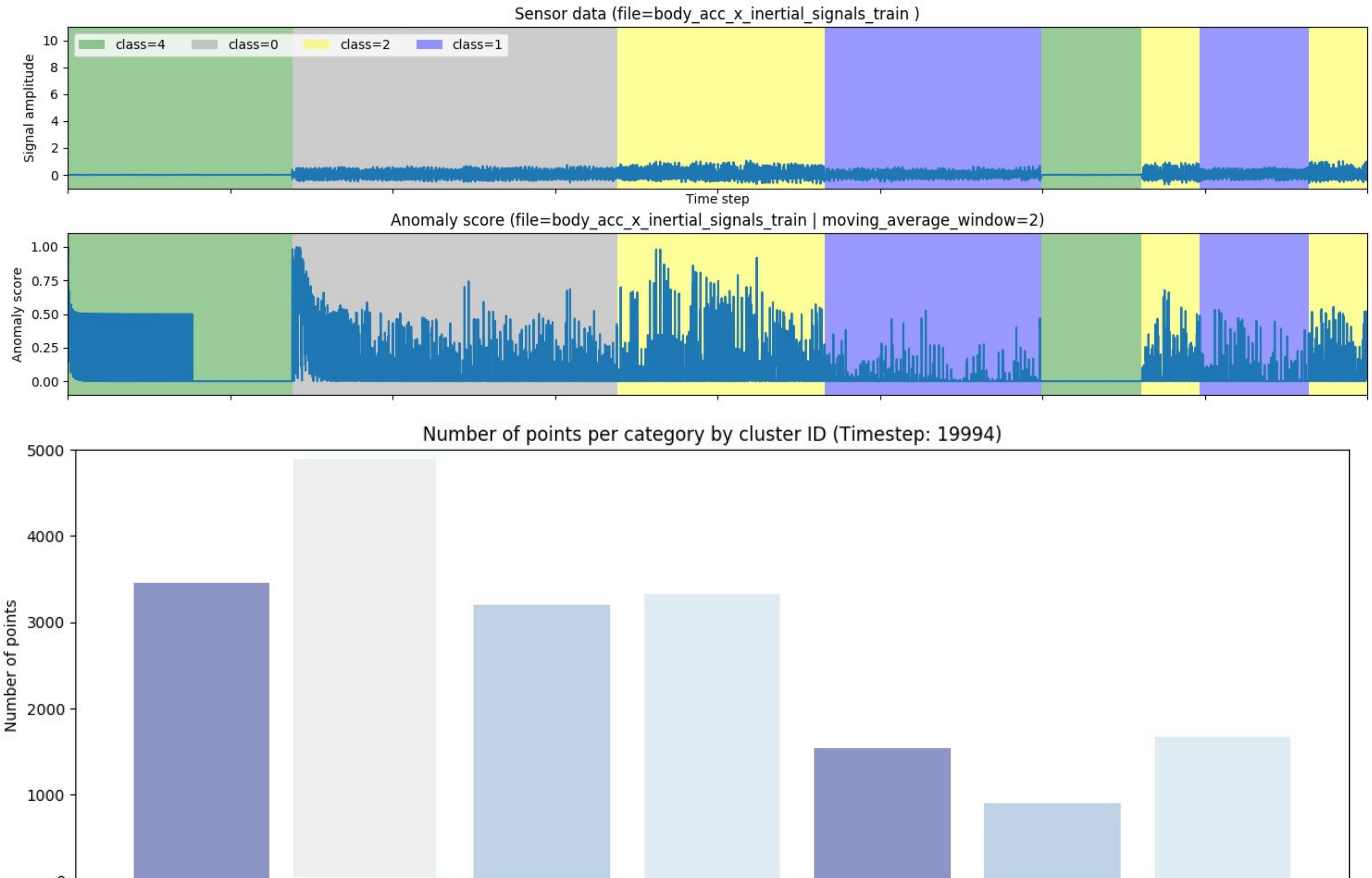
Unsupervised classification

Binary artificial data



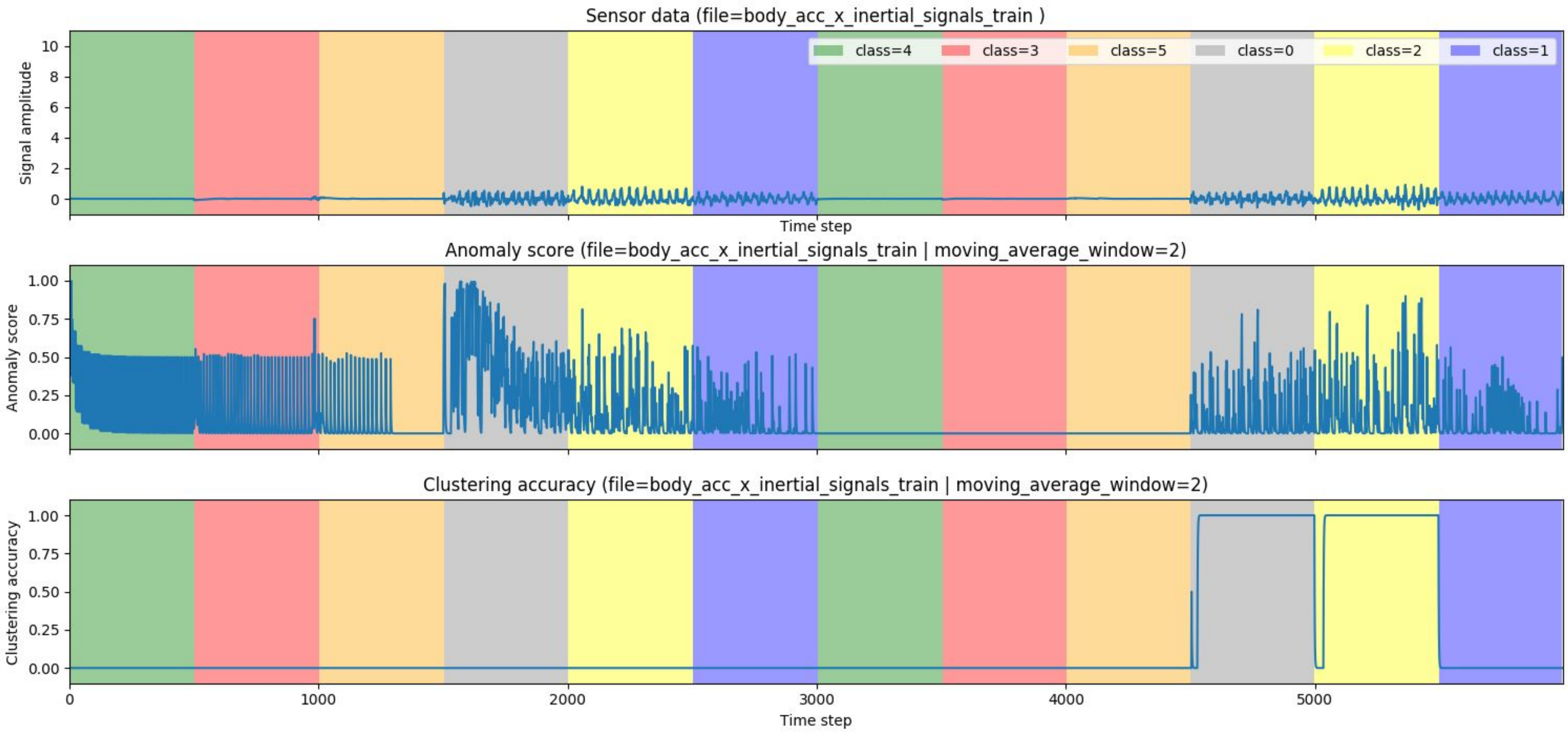


UCI accelerometer data with 4 classes

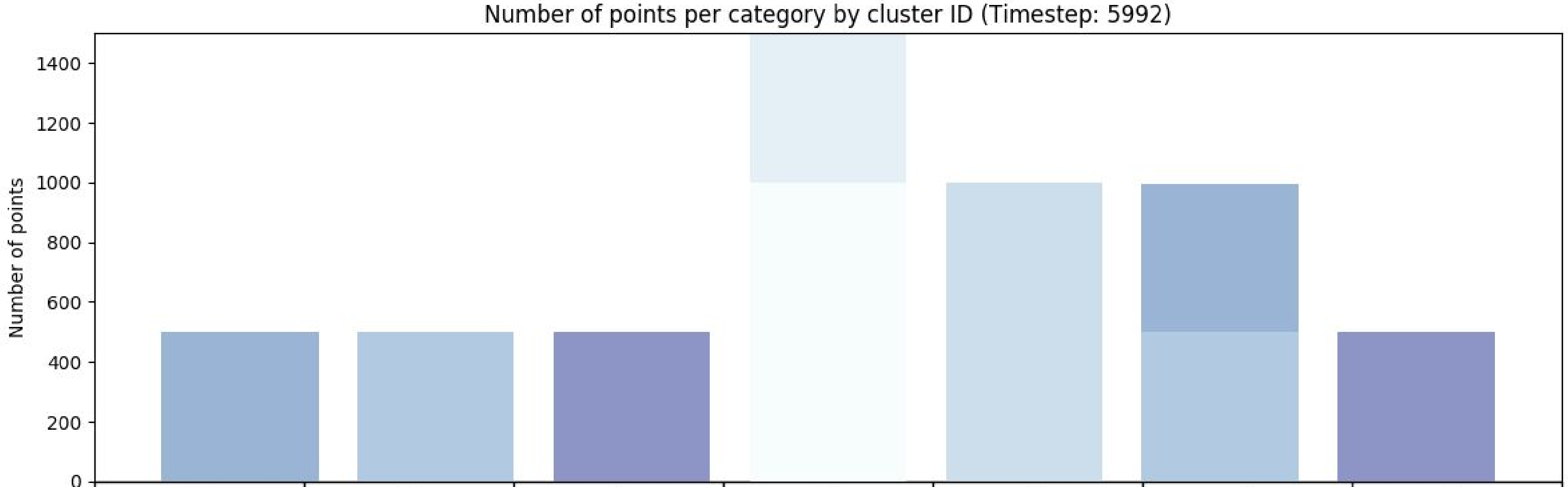


	c1	c2	c3	c4	c5	c6	c7
category 0	0	4999	0	0	0	0	0
category 1	0	0	0	3328	0	0	1671
category 2	0	0	3200	0	0	896	0
category 4	3456	0	0	0	1543	0	0

UCI dataset with 6 classes (1/2)



UCI dataset with 6 classes (2/2)



	c1	c2	c3	c4	c5	c6	c7
category 0	0	0	0	999	0	0	0
category 1	0	0	0	501	0	0	0
category 2	0	0	0	0	999	0	0
category 3	0	501	0	0	0	498	0
category 4	500	0	0	0	0	499	0
category 5	0	0	501	0	0	0	498

Summary

Binary data

✓ Detection of new clusters, online and in an unsupervised manner

✓ Good classification accuracy

✓ Distance metric is pretty good at distinguishing between different **predictable sequences**

Accelerometer data

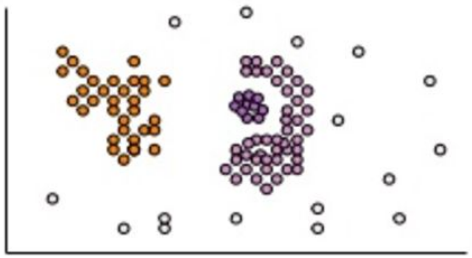
X Detects too many clusters

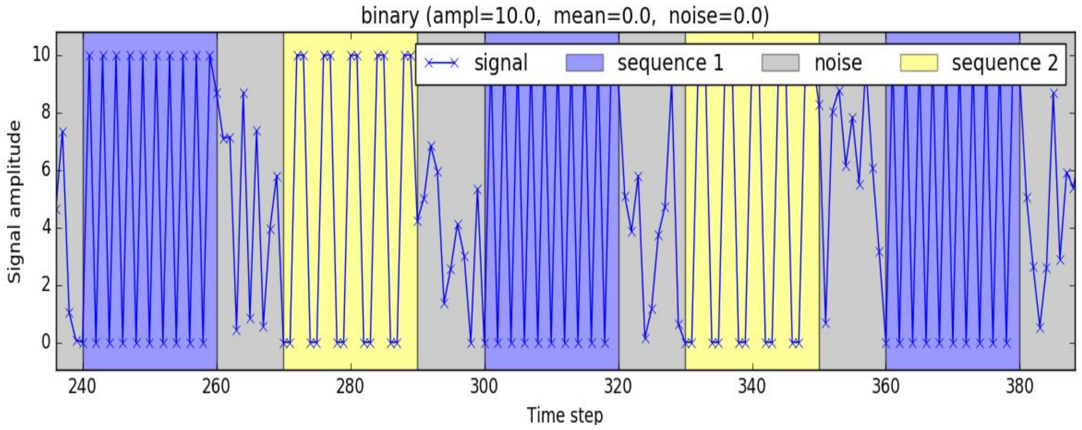
X Bad classification accuracy

X Distance metric struggles to distinguish between sequences with different levels of predictability

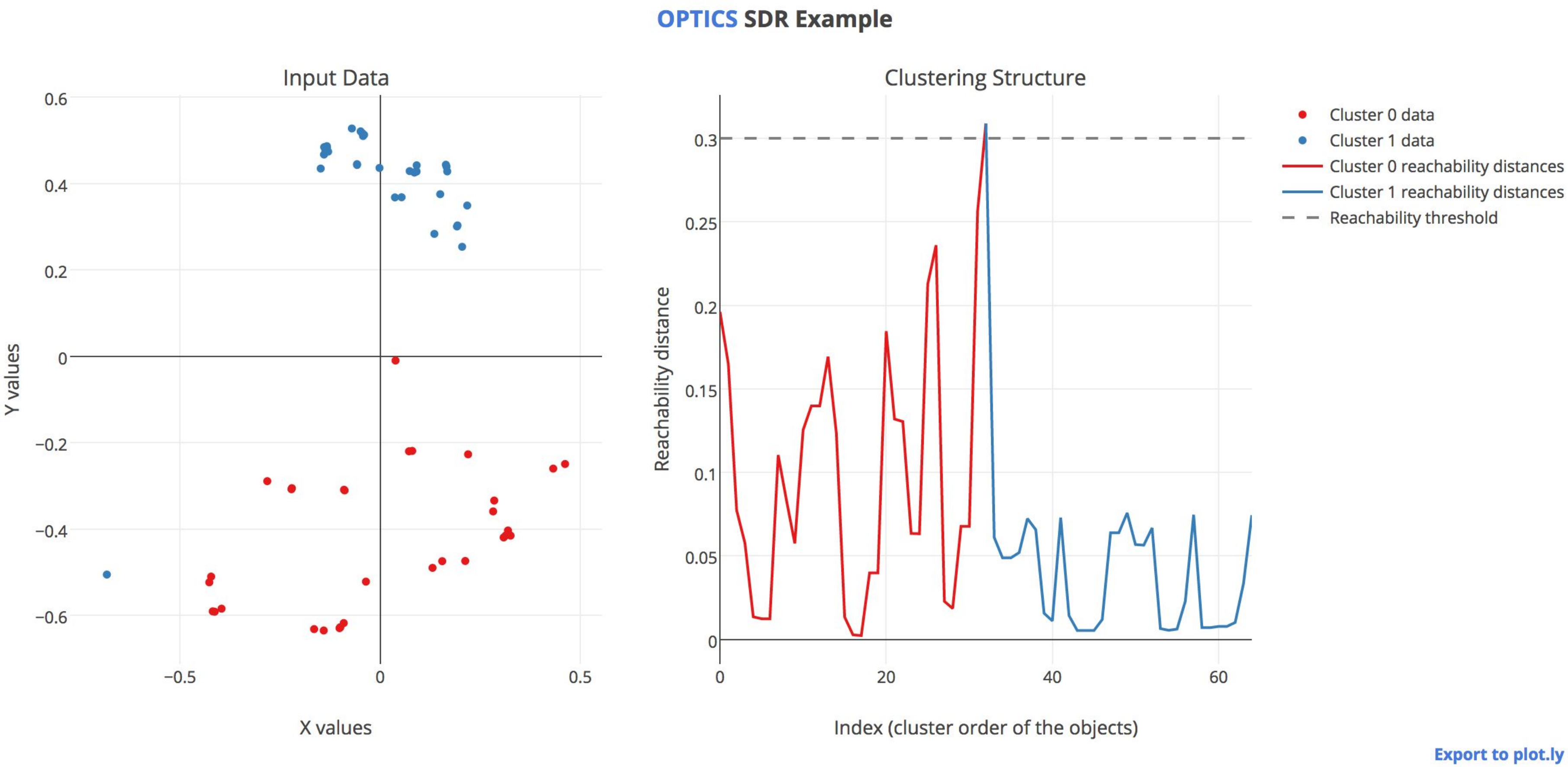
SDR clustering with OPTICS

- OPTICS algorithm
  - Group objects into meaningful subclasses
    - Density-based Clustering locates regions of high density that are separated from one another by regions of low density.
    - Density = number of points within a specified radius
- Demo 2D
- Demo with SDR
  - Average SDRs by sequence
  - Run OPTICS to find valleys
  - tSNE / MDS to project in 2D





OPTICS on binary data SDRs



Takeaways

Conclusion

- Re-usable frameworks:
  - Data cleaning and visualizations
  - HTM network factory to generate traces
  - Cluster visualizations: OPTICS, tSNE, MDS
  - Supervised classifier for HTM traces
  - Unsupervised clustering for HTM traces
- List of things that have been tried