HTM.core Streamer

Python Module

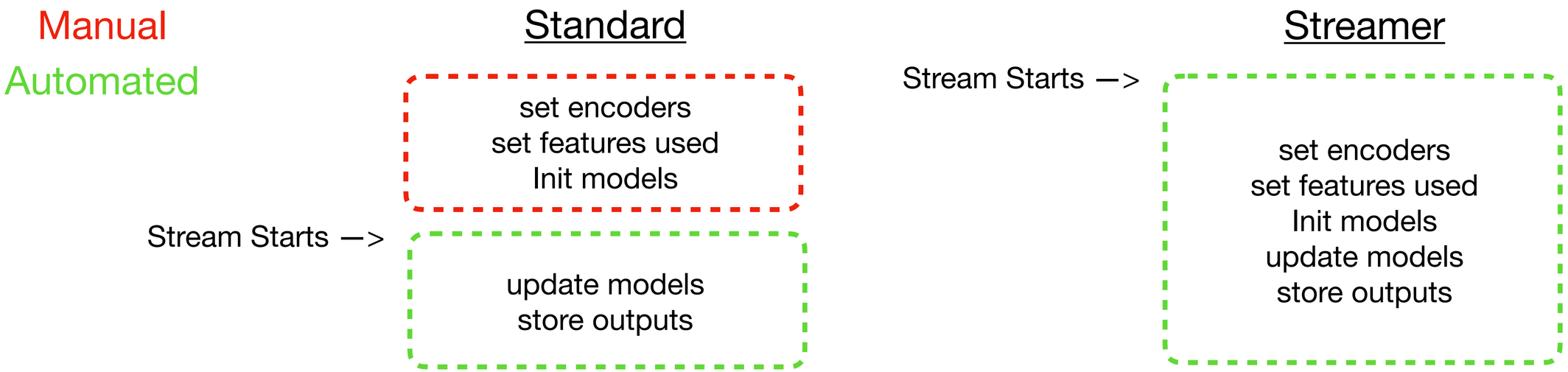
Sam Heiserman — Jan, 2022

Table of Contents

- Motivation
- Module Overview & Pseudocode
- Config
- Limitations & Next Steps
- Demo
- Implementation (**htm.core**)
- Discussion!

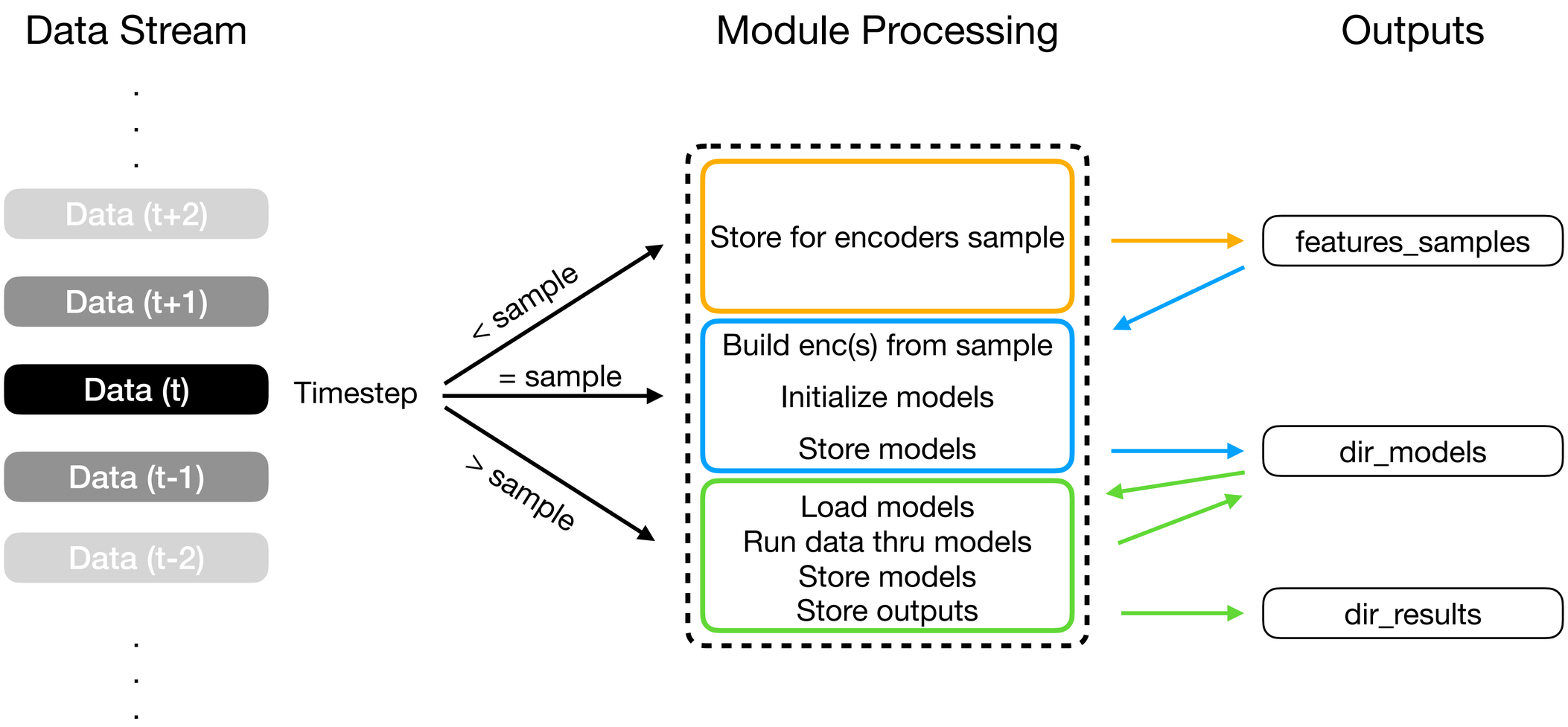
Motivation

- Achieve max scalability for **htm.core** thru —> **100% streaming** functionality
  - Eliminate need for EDA by —> generating encoders from stream sample
- Easily control modeling meta-parameters:
  - Whether to —> model each feature separately or all concatenated into 1 model
  - Whether to —> include timestamp feature
  - Whether to —> use predictor
  - Whether to —> disable learning at any point



Module Overview

1) Sample 2) Initialize 3) Run



Pseudocode

Main Functions

htm_stream_runner

1. Load —> Config
2. Load —> Batch Data
3. For Row in Batch Data:
    1. Generate —> Stream Data
    2. Store —> Stream Data
    3. Run —> **stream_to_htm**()

stream_to_htm

1. Load —> Config
2. Load —> Data
3. Validate —> Config

if mode == Sample:

4. Store —> Data

elif mode == Initialize:

4. Store —> Data
5. Build —> Encoder Params
6. Build —> HTM model(s)
7. Store —> HTM model(s)

else: (mode=Run)

4. Load —> HTM model(s)
5. Run —> Data thru HTM model(s)
6. Store —> HTM outputs
7. Store —> HTM model(s)

8. Store —> Config

Config Structure

Set by user

```yaml
features:
  3.3_Bus_Current:
    max: null
    min: null
    type: float
    weight: 1.0
  category1:
    type: category
    weight: 1.4
  satellite_time:
    custom: 0
    dayOfWeek: 0
    format: '%m/%d/%y %H:%M'
    holiday: 0
    season: 0
    timeOfDay:
    - 21
    - 9.49
    type: timestamp
    weekend: 0
    weight: 1.5
```

For each modeled feature:

- type
- weight
- encoding settings

```yaml
timesteps_stop:
  sampling: 50
  learning: 100
  running: 110
```

Time steps module will stop:

- sampling
- learning
- running

```yaml
models_state:
  model_for_each_feature: true
  return_pred_count: true
  use_sp: false
```

learn a separate model for each feature or combine into 1
measure number of predictions made by TM
use spatial pooler

Config Structure

Set by default

```yaml
models_params:
  spatial_anomaly:
    enable: true
    tolerance: 0.05
    perc_min: 0
    perc_max: 100
    anom_prop: 0.3
    window: 100000
  anomaly_likelihood:
    probationaryPeriod: 500
    reestimationPeriod: 100
  sp:
    potentialPct: 0.8
    columnCount: 2048
    globalInhibition: true
    boostStrength: 0.0
    localAreaDensity: 0.0  #0.1
    stimulusThreshold: 0.0
    numActiveColumnsPerInhArea: 40  #0
    synPermActiveInc: 0.003
    synPermConnected: 0.2
    synPermInactiveDec: 0.0005
    wrapAround: true
    minPctOverlapDutyCycle: 0.001
    dutyCyclePeriod: 1000
    seed: 0  #1956
  tm:
    activationThreshold: 20
    cellsPerColumn: 32
    columnDimensions: 2048
    initialPerm: 0.24
    maxSegmentsPerCell: 128
    maxSynapsesPerSegment: 128
    minThreshold: 13
    newSynapseCount: 31
    permanenceDec: 0.008
    permanenceInc: 0.04
    permanenceConnected: 0.5
    predictedSegmentDecrement: 0.001
    seed: 0  #1960
  predictor:
    sdrc_alpha: 0.1
```

hyperparams for simple spatial threshold

hyperparams for nupic.AnomalyLikelihood

hyperparams for htm.core.SpatialPooler

hyperparams for htm.core.TemporalMemory

```yaml
models_predictor:
  enable: false
  resolution: 1
  steps_ahead:
  - 1
  - 2
```

hyperparams for htm.core.Predictor

```yaml
models_encoders:
  n: 400
  w: 21
  n_buckets: 130
  p_padding: 20
  seed: 0
```

hyperparams for htm.core encoders

Limitations

- Encoders
  - Rely on feature distribution stationarity
- Runtime
  - Grows linearly with feature count (assuming 1 model per feature)
  - Slows down a lot when **predictor** active
- Memory
  - Grows a lot when **predictor** active

Next Steps

- Model Monitoring
  - TM connectivity
    - Density of permanent synapses
    - Rate of growth over time
    - Distribution of permanent synapses over columns
  - Feature distributions
    - Drift from original samples — could invalidate encoders
  - Anomaly scores
    - Long periods of 0 or 1.0
  - Prediction counts
    - Long periods of 0 or too high (> 10 ?)

- Quantify performance variation
  - When predictor active
  - When feature counts get big

Function Call Tree

source.pipeline.htm_stream**.stream_to_htm()**

load_config
load_json
validate_config

if mode == sample:
    extend_features_samples

elif mode == initialize:
    extend_features_samples
    build_enc_params
        get_rdse_resolution
    init_models
        HTMModel.init_model()
            HTMModel.init_encs()
                htm.core.RDSE_Parameters()
                htm.core.RDSE()
                htm.core.DateEncoder()
            HTMModel.init_sp()
                htm.core.SpatialPooler()
            HTMModel.init_tm()
                htm.core.TemporalMemory()
            HTMModel.init_anomalyhistory()
                htm.core.AnomalyLikelihood()
            HTMModel.init_predictor()
                htm.core.Predictor()
    save_models

elif mode == run:
    load_models
        load_pickle_object_as_data
    run_models
        HTMModel.run()
            HTMMode.get_encoding()
                htm.core.encoder.encode()
                htm.core.SDR.concatenate()
            htm.core.sp.compute**()**
            HTMModel.get_predcount()
                htm.core.tm.activateDendrites()
                htm.core.tm.getPredictiveCells()
            htm.core.tm.compute()
            HTMModel.get_preds()
                htm.core.predictor.infer()
                htm.core.predictor.learn()
    save_outputs
    save_models