Version 1.12.0

- Includes a new tool recover_orderbooks. Refer to the document ./docs/co3-configuration.pdf for details.
- Adds the ability to record reward CSV files during runs of pytest.

Version 1.11.0

This release is largely a rationalization of YAML configuration for both playbacks and dataset generations.

CO3_CONFIG_PATH

This environment variable has been repurposed to now be a list of paths where configuration files are to be found. Each such path is separated with a ":" char; in this sense, it is similar PYTHONPATH. It is commonly referred to when loading the configuration for both playbacks and generates.

./co3-configurations/dev

This folder now contains a set of YAML files which, taken together, are meant to demonstrate typical configurations. These files should be referred to as further configurations are developed and existing ones are modified to accommodate this release.

- co3.yaml
  Noteworthy are the following:
  - console: can be set to logging or progress_bar; if not set, it defaults to logging.
  - defaults_ : used to identify a default configuration file (this file itself is discussed below). Note the underscore in the key name, which is required in order to distinguish from Hydra's use of 'defaults'.
  - playbacks: A collection of one or more playbacks.
    - <playback_name>:
      - agent: Parameters specific to the agent function. Many parameters are common to all agents and hence it is these that can be stored in the defaults_ file. The common set of parameters are discussed here. Agent-specific parameters are discussed in the section Agent-Specific Parameters.
        - action_noise

    - type
      To be selected from null | OrnsteinUhlenbeck | Gaussian
    - sigma
      Standard deviation of the noise source.
- batch_size
  Size of the batch to be sampled from experience replay buffer during training.
- buffer_size
  Size of the experiences replay buffer. batch_size <= buffer_size
- episodes
  The number of episodes that the process is to run.
- gamma
  Bellman discount factor
- gradient_clipping
  Further details available at:
  https://pytorch.org/docs/stable/generated/torch.nn.utils.clip_grad_norm_.html
    - max_norm
    - norm_type
- purge_network
  Boolean defining whether the network is to be purged before the run begins.
- network
  Defines a path name or a file name of a PyTorch network (extension: *.pt). If this parameter is not provided, then a new network file will be created in the location `CO3_PATH/networks/` (called the 'nominal location' below) with file name `<current datetime>.pt`. If this parameter is a path name and it already exists, then it must have content that is recognizable as a valid network file by PyTorch. If the path name does not currently exist, then a new network file will be created. On the other hand, if a file name is provided, then the software will check to see if the file already exists in the nominal location and, if so, it will expect that this file is one that is recognizable as a network file by PyTorch. If no such file is found, then a new file of that name will be created in the nominal location.
- training
  Boolean selecting whether the network is to be trained during the course of the run.

  - training_interval
    Number of steps between trainings of the network.
  - target_update_interval
    Number of trainings between updating the target

- misc: Parameters that are process-general and not associated with a specific function such as the agent or environment. These are discussed now:
  - csv_path
    Path to where the rewards file is to be recorded. If null, then the path name is automatically generated. If defined and a relative path, then the pathname must begin with ‘rewards’.
  - generate_csv
    A boolean variable defining whether a rewards csv is generated or not.
  - leave_progress_bar
    If set, then completed (test) progress bars are retained; otherwise complete test progress bars are discared.
  - log_interval
    The number of episodes between routine logging messages.
  - log_level
    The logging modules log level. To be selected from [DEBUG | INFO | WARNING | ERROR | CRITICAL]. See https://docs.python.org/3/library/logging.html for further details.
  - seed
    Seed to be used for all random number generators accessed by the software (Python, numpy, OpenAI gym, and PyTorch).
  - record_state
    Record (or not) state in the rewards CSV file.
- env_config
  - envId
  - exchange
  - reward_offset
  - randomize_dataset_reads
  - pdf

        - age_limit
          Expressed in days, can be decimal; e.g. 0.25 = 6 hrs
        - bins
          Further details can be found here:
          https://numpy.org/doc/stable/reference/generated/numpy.histogram.html
        - graph
          Boolean selecting whether to graph a newly generated PDF.

- child_process
  When a child process is created, its configuration parameters are a merge of the parents configuration parameters with the parameters belonging to the child_process key, although the merge has a number of limitations as follows:
    - A child process itself cannot have a child_process key.
    - agent.training is hardwired to be False (child process networks are never trained).
    - Only the following parameters can be set to different values from that of the parent (training) process:
      1. misc.csv_path
      2. misc.generate_csv
      3. misc.log_interval
      4. agent.episodes
      5. agent.nn_trace
      6. env_config.datasets
      7. env_config.randomize_dataset_reads

Version 1.2.0

Nominal Configuration

Although the general approach to managing configuration has been retained, there are a number of specific changes that could cause confusion. A good reference is to review the example configurations found in the file:

$CO3_PATH/co3-configuration/dev/nominal-config/co3.yaml

Environments

The concept of an environment has always been central to the co3 software, although largely a background concept with little visibility to the user. Version 1.2 gives more presence to the concept in that each algorithm of interest must be configured to use one of the available environments.

A list of the available environments are:

- SenitientTrading:EvaluateTradeHistoryEnv-v0
- SenitientTrading:EvaluateDateFrameEnv-v0
- SenitientTrading:BalanceTradeHistoryEnv-v0 [1]
- SenitientTrading:BalanceDataFrameEnv-v0 [2]
- SentientTrading:BuyOrderbookDataFrameEnv-v0
- SentientTrading:SellOrderbookDataFrameEnv-v0

The configuration parameter, env_name, is no longer available at file scope. But it lives on in the configuration of each algorithm. Hence, all of the algorithms now have an algorithm-specific env_name parameter. Failure to assign env_name to a given algorithm will trigger a Python exception. As an example:

```
GAC:
  env_name: SentientTrading:SellOrderbookDataFrameEnv-v0
```

1 *Not available with the first release of 1.2*

2 *Not available with the first release of 1.2*

Environment Configuration

The longer term plan is to capture all environment configuration under a new environments parameter. Currently only two of the environments are configured in this way. Configuration for the other (and older) environments is unchanged.

environments:

BuyOrderbookDataFrameEnv-v0:

pdf: pdf1

Name of the pdf found in the Mongo database "configurations".

Ql: 0.2

Quantity Limit

precision: 8

Rate precision. Internally tick = $10^{-precision.}$

SellOrderbookDataFrameEnv-v0:

pdf: pdf1

ql: 0.2

precision: 8

Dataset Generation

Configuration for dataset generation has been changed. All such configuration is defined within a new hierarchical configuration parameter generate. Dataset configuration is discussed in this section. See section Launch Commands for details on how to launch dataset generations.

generate:

log_interval: 200

specific to dataset generation

ob_vectors:

3 types of ob vector generations, as defined by side.

n: 4

*After compression, this is the number of compressed orderbook entries to be recorded to the dataset. Hence, in the selected 'side' is buy or sell, each vector will be of length 2 * n. If the selected side is "ob" or if side is not selected, then each vector will be of length 2 * 2 * n = 4n.*

k: 1.5

k is a number expressed in base currency that indicates that if an order quantity is smaller that k*, that quantity is to be rationalized in the entry ahead of it.*

side: sell

[buy | sell | ob]

folder: datasets/ob_vectors/test

The folder to record the dataset into. Note that if, for example, 'side' = 'sell', then the dataset will be recorded to the file defined by <folder>/sell.json.

th_vectors:

evaluate: True

Either True or False. If True, then the dataset will be populated with vectors derived from Trade History records reaching back in time according to the configuration parameter time_breaks, where a merge is done of buy and sell trade history. If False, then dataset is populated with records that separately distinguish the buy side and sell side trade history.

destination: test-th-vector-dataset

The file name to store the output dataset. Note that it will be stored in the location $CO3_PATH/datasets/evaluate, if evaluate = True or to $CO3_PATH/datasets/balance if evaluate = False.

Playbacks

The typical workflow is to generate one or more datasets and then play them back. This section discusses playback configuration. See section Launch Commands for details on how to launch a playback.

A new configuration parameter, playbacks, is now available and it has the following structure:

```
playbacks:
    ob_vectors:
        buy_ob: [datasets/ob_vectors/test/buy.json]
            Currently only applicable to the SentientTrading:BuyOrderbookDataFrame-v0 environment.
        sell_ob: [datasets/ob_vectors/test/sell.json]
            Currently only applicable to the SentientTrading:SellOrderbookDataFrame-v0 environment.
        ob: [datasets/ob_vectors/test/ob.json]
            Not yet used by any environment, but planned to be used by the
            SentientTrading:OrderbookDataFrame-v0 environment.
    th_vectors:
        evaluate: [datasets/evaluate/training_set_snt.json]
            Currently only applicable to the SentientTrading:EvaluateDataFrame-v0 environment.
        balance: null
```

In each case, the listed JSON file(s) define which files are to be used during playback for each environment. Such configuration is only relevant to the data frame environments (i.e. those that are driven from a dataset).

Launch Commands

The basic co3 launch commands are listed here. Configuration for each of these commands continues to reside in the YAML file identified by the environment variable $CO3_CONFIG_PATH and, as before, all parameters can be overridden at the command line with the syntax param=XYZ.

- $ generate-th-vectors
  Generates a dataset containing th-vector/IT pairs. Such datasets are suitable for the discrete action space environments.
- $ generate-ob-vectors
  Generates a dataset containing ob-vector/market-price pairs. Such datasets are suitable for the continuous action space environments.

- $ playback
  Replaces $ play03

Deprecated:

- $ co3
  This command is no longer supported, although it is still available and may even be operable for some algorithm / environment pairs. This command has the effect of both generating th-vector/IT pairs on the fly and then playing them back all in one go. The recommended approach now is to first generate datasets and then play them back.

New Algorithms

A general comment about all new and future algorithms: all new algorithms provide their own configuration area. For example, in the case of the new GAC algorithm, there exists a hierarchical GAC configuration parameter. Some of the GAC's configuration parameter names are the same as those that have been used before and sometimes some of these parameters are replicated at file scope. For example, buffer_size, exists now at both file scope and within GAC. In such a case, the one at file scope is not used; only the one defined within GAC is relevant.

The plan is to remove all algorithm configuration parameters out of file scope down into algorithm-specific configuration parameters. Hence, this general principle will eventually apply to all supported algorithms. The one such parameter that exists now only at the algorithm-specific level is env_name, which received special attention in this regard.

Generative Actor Critic (GAC)

The Generative Actor Critic (GAC) uses generative actor training. Contrary to parametric distribution functions, a generative neural network acts as a universal function approximator, enabling the possibility to represent arbitrarily complex distributions. The GAC supports continuous action space environments.

Soft Actor Critic (SAC)

This algorithm's status is experimental and is not recommended for production use.

Soft Actor Critic (SAC) is an algorithm that optimizes a stochastic policy in an off-policy way, forming a bridge between stochastic policy optimization and DDPG-style approaches. The SAC supports continuous action space environments.

Deep Deterministic Policy Gradient (DDPG)

This algorithm's status is experimental and is not recommended for production use.

DDPG is a reinforcement learning technique that combines both Q-learning and Policy gradients. The DDPG supports continuous action space environments.

Version 1.0.0

- Start using version numbers for the co3 project.
- To every recurring log message add the time in seconds between it and the last recurring log message; e.g. “+x.xxx”.
- Move all docs into a new folder ./docs
- In order to execute all pytests, enter `$ pytests`
- The co3 environment now supports three (3) commands:
  - co3: Direct execution. Whether or not CSV generation is provided is dependent on the value of the boolean config.generate-csv.
  - gen03: Generates datasets to be used subsequently by play03. gen03 can run generate the following dataset types:

    ```
    (th_vector/IT)                  # evaluate=True
    (ob_vector/IT)                  # evaluate=True
    (th_vector/(buy_IT, sell_IT))   # evaluate=False
    (ob_vector/(buy_IT, sell_IT))   # evaluate=False
    ```

    The following config entries indicate new config elements that are exclusively applicable to gen03. Note that in a single run, gen03 can do generate both an ob-vector type generation and a th-vector type generation:

    ```
    generate:
        ob-vectors:
            active: False
            n: 4
            k: 1.5
            destination: test-ob-vector-dataset

        th-vectors:
            active: False
            destination: test-th-vector-dataset
    ```

  - play03: Exploits the datasets that have been generated by gen03. Whether CSV generation is provided is dependent on the value of the boolean config.generate-csv.

- “state” is no longer being used as dataframe column. It is replaced by “th_vector”.
- config.output-dataset is deprecated. Ignored if set.
- config.db-action is deprecated. Ignored if set.
- Child processes are always run using play03.
- A memory leak problem has been resolved.