CryptoEnv

Release 0.1.1

Crinstaniev

May 27, 2022

CONTENTS:

CryptoEnv is a Python library for cryptocurrency researches by agent-based methods, including non-reinforcement learning and reinforcement learning algorithms.

Note: This project is under development.

CHAPTER

ONE

INSTALLATION

1.1 Download From Source

You can directly download the project from our GitHub repository:

```
git clone https://github.com/crinstaniev/cryptoenv.git
```

1.2 Install Using *pip*

Or you can use *pip* to install

```
pip install crenv
```

CHAPTER

TWO

API

This section we provide the API definition and usages for our *CryptoEnv* package.

2.1 CryptoEnv

class crypto_env.core.**CryptoEnv**(*max_sell*, *max_buy*, *min_sell*, *min_buy*, *dataloader:* crypto_env.dataloader.dataloader.DataLoader, *recorder:* crypto_env.recorder.Recorder)

Bases: gym.core.Env, abc.ABC

This is the core module of *CrytoEnv*. It provide environment for agents to perform buy and sell actions and provide market states.

__init__(*max_sell*, *max_buy*, *min_sell*, *min_buy*, *dataloader:* crypto_env.dataloader.dataloader.DataLoader, *recorder:* crypto_env.recorder.Recorder)

Parameters

- **max_sell** (*float*) – maximum crypto to sell
- **max_buy** (*float*) – maximum crypto to buy
- **min_sell** (*float*) – minimum crypto to sell
- **min_buy** (*float*) – minimum crypto to buy
- **dataloader** (DataLoader) – the *crypto_env.dataloader.dataloader.DataLoader* instance
- **recorder** (Recorder) – the Recorder instance

buy(*value*, *verbose=0*)

The agent buy some amount of crypto.

Parameters

- **value** (*float*) – number of crypto to buy
- **verbose** (*int, optional*) – whether to print out debug info. Defaults to 0.

Returns same return as *step()*

first_observation()

Return the first observation

Returns return a dictionary structured dict(features, index)

Return type dict

abstract classmethod get_reward()

Returns the reward for agent after taking an action

Return type float

hold(*verbose=0*)
The agent does not want to do anything in this step

Parameters **verbose** (*int, optional*) – whether to print out debug info. Defaults to 0.

Returns same return as *step()*

meta()
Return the meta information of the environment

Returns the meta of the env

Return type dict

render(*mode='human'*)
Placeholder. Not implemented yet.

Parameters **mode** (*str, optional*) – Defaults to "human".

reset()
Reset the environment to prepare for a new episode

Returns

Return type *CryptoEnv*

sell(*value*, *verbose=0*)
The agent sell some amount of crypto.

Parameters

- **value** (*float*) – number of crypto to sell
- **verbose** (*int, optional*) – whether to print out debug info. Defaults to 0.

Returns same return as *step()*

step(*action=None*)

Parameters **action** (*dict, optional*) – action to take. Defaults to None.

Returns agent's observation after taking the action (numpy array), reward of the action (float), whether the episode is to the end (bool), and diagnostic information for debugging (any).

2.2 DataLoader

class crypto_env.dataloader.**DataLoader**(*start_idx*, *end_idx*)
Bases: abc.ABC

The DataLoader module is for user to map arbitrary data source to form that the environment can recognize.

abstract __init__(*start_idx*, *end_idx*)

Parameters

- **start_idx** (*int*) – Start index

- **end_idx** (*int*) – End index

__iter__()
This object is iterable. See https://www.w3schools.com/python/python_iterators.asp for more details.

abstract __len__()
Return the length of the iterable

Raises **NotImplementedError** –

abstract __next__()
See https://www.w3schools.com/python/python_iterators.asp for more details

abstract get_duration()
Get length of the data source.

Raises **NotImplementedError** –

abstract get_feature(*feature_name*)
Get input variables (features)

Parameters **feature_name** (*str*) – name of the feature

Raises **NotImplementedError** –

get_idx()
Get current index

Raises **NotImplementedError** –

get_transaction_fee(*idx=None*)
Return the transaction fee list

Parameters **idx** (*int, optional*) – Number of transaction fee to return. Defaults to None.

Returns list

get_transaction_fee_type()
Return the name of transaction fee type

Returns str

property idx

load_transaction_fee(*values*, *fee_type='percentage'*)
Load the transaction fee list

Parameters

- **values** (*list*) – Transaction fee list
- **fee_type** (*str, optional*) – 'percentage' or 'fix'. Defaults to 'percentage'.

abstract reset()
Reset the dataloader

Raises **NotImplementedError** –

class crypto_env.dataloader.**ETHLoader**(*base_dir*, *start_idx*, *end_idx*, *features: list*, *dropna=False*, *download=True*, *url='https://raw.githubusercontent.com/coinmetrics/data/master/csv/eth.csv'*)
Bases: *crypto_env.dataloader.dataloader.DataLoader*

Our example implementation of *DataLoader* class. We use the Ethereum history data from the coinmetrics repo. See https://raw.githubusercontent.com/coinmetrics/data for more details.

__init__(*base_dir*, *start_idx*, *end_idx*, *features: list*, *dropna=False*, *download=True*, *url='https://raw.githubusercontent.com/coinmetrics/data/master/csv/eth.csv'*)

Parameters

- **base_dir** (*str*) – Directory to save the download data
- **start_idx** (*int*) – Where to start in the data source
- **end_idx** (*int*) – Where to end in the data source
- **features** (*list*) – Input variables for the environment
- **dropna** (*bool, optional*) – Whether to drop lines including empty values. Defaults to False.
- **download** (*bool, optional*) – Whether to re-download the data. Defaults to True.
- **url** (*str, optional*) – Link to the data source. Defaults to "https://raw.githubusercontent.com/coinmetrics/data/master/csv/eth.csv".

__len__()
Number of items

Returns int

get_duration()
Get length of the data source.

Raises **NotImplementedError** –

get_feature(*feature_name*)
Get input variables (features)

Parameters **feature_name** (*str*) – name of the feature

Raises **NotImplementedError** –

get_idx()
Get current index

Raises **NotImplementedError** –

reset()
Reset the dataloader

Raises **NotImplementedError** –

2.3 Recorder

class crypto_env.recorder.**Recorder**(*price_list*, *crypto_cap=0*, *fiat_cap=1000*)
Bases: object

This is the recorder class. A recorder object is able to record agent's action in both training and production mod. The data collected can later be used for plotting or analyzing.

__init__(*price_list*, *crypto_cap=0*, *fiat_cap=1000*) → None

Parameters

- **price_list** (*list*) – a list of price, the length should be equal to the size of the DataLoader.

- **crypto_cap** (*int, optional*) – initial balance of crypto. Defaults to 0.
- **fiat_cap** (*int, optional*) – initial balance of fiat. Defaults to 1000.

get_crypto_balance(*idx=None*)
Calculate cryptocurrency balance

Parameters **idx** (*int, optional*) – How many transactions to involve. Defaults to None.

Returns _description_

Return type _type_

get_crypto_value(*idx=None*)
Calculate the value of crypto in balance

Parameters **idx** (*int, optional*) – How many transactions to involve. Defaults to None.

Returns float

get_expenditure(*idx=None*)
Calculate how many fiat was used in the investment.

Parameters **idx** (*int, optional*) – Number of transactions to involve. Defaults to None.

Returns float

get_fiat_balance(*idx=None*)
Calculate fiat balance

Parameters **idx** (*int, optional*) – How many transactions to involve. Defaults to None.

Returns float

get_income(*idx=None*)
Calculate how many value does the agent earn.

Parameters **idx** (*int, optional*) – How many transactions to involve. Defaults to None.

Returns float

get_info_record(*to_dataframe=True*)
Return all history market info

Parameters **to_dataframe** (*bool, optional*) – Whether to convert to dataframe. Defaults to True.

Returns Market information history

Return type (DataFrame, any)

get_roi(*idx=None*)
Calculate the return of investment

Parameters **idx** (*int, optional*) – How many transactions to involve. Defaults to None.

Returns float

get_transaction_record(*idx=None*)
Return all history buy and sell signals generated by the agent

Parameters **idx** (*int, optional*) – Number of records to print. Defaults to None.

Returns A `pandas.DataFrame` containing all history signals.

Return type DataFrame

insert_info(*info*)
Insert market information into the record

Parameters **info** (*array-like*) – an array of current market info

insert_transaction(*transaction: crypto_env.types.Transaction*)
Insert new transaction into the recorder

Parameters **transaction** (Transaction) – Transaction object to insert

reset()
Reset the recorder

2.4 Visualizer

class crypto_env.visualizer.**Visualizer**(*env*, *time_feature_name='time'*)
Bases: object

A handy collection of result visualization schema.

__init__(*env*, *time_feature_name='time'*)

Parameters

- **env** (CryptoEnv) – The main environment
- **time_feature_name** (*str, optional*) – (**deprecated**) Name of the time feature. Defaults to 'time'.

draw_portfolio()
Plot the portfolio

Returns Plotly figure object

draw_return()
Plot the investment return history

Returns Plotly figure object

draw_signal()
Plot the buy and sell signal generated by the agent

Returns Plotly figure object

2.5 Algorithm

class crypto_env.algorithm.**Algorithm**
Bases: abc.ABC

The algorithm wrapper template for the customized agent

abstract take_action(*observation*, *info=None*)
Return an action from the action space.

Parameters

- **observation** (*any*) – The observation from the environment.
- **info** (*any, optional*) – The market information. Defaults to None.

Raises **NotImplementedError** – You have to implement this method

class crypto_env.algorithm.**BuyAndHold**(*buy_amount: float*)
Bases: *crypto_env.algorithm.algorithm.Algorithm*

An example implementation of class *Algorithm*. This algorithm implements the buy and hold strategy. See https://www.investopedia.com/terms/b/buyandhold.asp for more information.

__init__(*buy_amount: float*)

Parameters **buy_amount** – unit in cryptocurrency

take_action(*observation*, *info=None*)
Return an action from the action space.

Parameters

- **observation** (*any*) – The observation from the environment.
- **info** (*any, optional*) – The market information. Defaults to None.

Raises **NotImplementedError** – You have to implement this method

class crypto_env.algorithm.**MovingAverage**(*price_feature_pos*, *short_window=30*, *long_window=180*, *initial_cap=100*)
Bases: *crypto_env.algorithm.algorithm.Algorithm*

An example implementation of class *Algorithm*. This algorithm implements the Dual Moving Average Crossover strategy. See https://faculty.fuqua.duke.edu/~charvey/Teaching/BA453_2002/CCAM/CCAM.htm for more information.

take_action(*observation*, *info=None*)
Return an action from the action space.

Parameters

- **observation** (*any*) – The observation from the environment.
- **info** (*any, optional*) – The market information. Defaults to None.

Raises **NotImplementedError** – You have to implement this method

PYTHON MODULE INDEX

C

INDEX

Symbols

A

B

C

D

E

F

G

H

I

L

M

R

S

T

V