Neural Strategy

Introduction

In `neural.py` a strategy is implemented for the AI models we've trained. These model can have different architectures and different input data, this difference is worked into the model name as a tag (D#, V#).

This strategy can be used in the same way as classical strategies, however some models (like GRU and LSTM) include a internal state which needs to be restored/saved when loading/storing the strategy. Therefore we use the StatefulStrategy class as a base.

The main strategy (PEPGBase) contains all the models from our research document and some more experimental ones.

Data input

Over the course of our project we have used a few different versions of training/data as indicated by the D# and V# tags. These can be differences in normalization, in actual inputs, or just in training data; the exact details are in `neural.py` iteself.

Evosim

`neural.py` and our AI part in general makes use of Evosim; a submodule created for OBTrader which has various neural network architectures, utlities, tests, and training methods.

In general we only use the architectures as the other components are only used in testing & prototyping.

Testing

The results of our testing is done using the backtesting module, the results can be seen in the research document or by launching a development instance.

All architectures are tested on the realtime module as well to make sure saving/restoring of models goes without issue.

Source code

Code is available at https://git.fhict.nl/I404788/trading-bot/-/blob/develop/tradeEnv/neural.py