Bipoon User Guide

Bipoon is a toolset for building custom trading bots based on Neural Network, specially designed for this purpose.

The user guide's logic is built in stages *around the use cases of the Service* and simultaneously accompanied by a description of specific menu items.

Note! Bipoon is the keyboard-driven application (arrow keys, Enter and Esc)

Step 1. Create the first algorithm and the first bot

To start trading, you need to create 2 entities: "**algorithm**" and "**bot**".
Algorithm is what generates BUY and SELL signals by analyzing market data.
Bot - a robot that makes trades on an exchange based on the signals generated by the algorithm.

1) To create the algorithm, select **"Create an Algo"** in the main menu. Use latin letters, numbers and symbols without spaces for the name of your algorithm.
2) Then you'll get to the algorithm settings menu ("**Algorithm Settings**"), where you need to choose "**Trade Symbol**" - a trading pair that will be used to generate signals (XBT/USD on Kraken by default); "**Base Algorithm**" - the basic algorithm (only neural network at the moment) and then **Start Learning** (during which the neural network will begin learning process on historical data). Usually it takes 1-2 minutes (you will see the progress in %).
3) Go to the main menu (use "Esc" to go back) and create a bot by selecting "**Create a Bot**", enter the name and get to the "**Bot Settings**" tab. Select the symbol ("**Trade Symbol**"), which we want to trade, the algorithm ("**Algorithms**"), which was created and launched into training. "**Selected** " should appear next to the selected algorithm.

Open Position. Here we choose the direction of entry into the trade (Buy&Sell, Buy or Sell only). Without this bot will not trade!

Size. If you chose the Kraken exchange in the Trade Symbol section for your bot, then specify the value in coins. For example, for the pair XBT/USD, the number "1" means that the bot is trading 1 btc. If you chose Bitmex, then "1" means 1 contract, which is equal to $1 worth of btc (i.e. 0.00013 BTC)

Order Type. The way the bot enters a position (by a market or limit order). This affects the exchange fee and the probability of opening a position (market - with a 100% execution, limit - almost 100%, but you have to check it on an exchange).
Trading Mode. Choose either demo trading or trading on a real account. For demo trading, no API keys required, but for real trading you need to add them in Settings (go to main menu).

Bot Targets. Here we set up bot risk-management parameters: target profit (realized profit) and loss limit (realized loss and trailing stop). After hitting one of the "limits", the bot stops trading. It is important to understand that these limits are applied not to a particular trade, but on the bot entirely! These parameters are not mandatory for trading. Parameters are set in USD per traded coin. For example, realized loss = 300 means that the bot will be turned off when the total loss of all trades is $300 (basis points; 1 point = $1). It could be slightly higher, because the risk-management system waits until the bot exits the position based on a signal from the neural network. **Trailing stop**: Set in USD per 1 traded coin. F. ex., trailing stop = 100. The trailing is activated and will break even when the profit (exactly by price change) reaches +$100. And then it will move in increments of $100, with each increase in profit by $100 (i.e. if the profit is +$200, the trailing will move to +$100).

Clear Bot Statistics.
All the trades are gathered and displayed on the chart. When the target profit (see “realized profit”) is reached, the bot stops trading. To resume the bot, you need to reset the statistics or increase the target (profit, loss) and set “**Open Type"** in the Open Position section (it’ll be resetted when target is hit).

Finally, your first algorithm and bot are ready to trade!

Step 2. How to analyze your algorithm’s learning results (a.k.a. back-test)

After creating the bot, you need to go back to the algorithm and carefully study the results of training. To do this, go back to "**Algo Manager"**. You’ll ses 2 graphs next to the algorithm. The top one is the result of trading on historical data. Our task is to assess the equity curve for the smoothness of its ascent without significant drawdowns (the algorithm should smoothly increase profits with each signal). X axis - the number of trades. Y axis - profit in USD.

If you do not like the equity, then re-train it by hitting "Start Learning" once more. Repeat until you get a decent result.

The second chart (under the historical one) will show a profit on real-time data.

Step 3. Bot management

Go to "**Bot Manager,** select the bot and press “Enter” to go to its settings. You’ll see its statistics of trades and profitability chart.

The statistics table shows the last 10 trades of the bot.
Price change is a profit in basis points (in USD on 1 traded coin)
Profit in USD is the bot’s actual profit in the trade

In "**Dashboard**” (go to main menu) you can see the current state of all of your bots and algorithms.

Step 4. Create a bunch of algorithms and bots on them.

No matter how well the algorithm is trained, it has a lifetime. This means that there is a high probability that it will not be able to work effectively in the future after a certain amount of time, because the market is constantly changing.

In order to get profit in the long run using our neural networks, it is necessary to build the right “portfolio approach” and risk-management system for your algorithms and bots.

1. Portfolio approach. Create a bunch of algorithms (6-8 or more) and put a bot on each of them. If algorithms are trained appropriately, it is highly likely you’ll get the profit in total from all the bots.

2. Risk-management. Limit each bot’s loss by setting realized loss in settings. If the bot starts trading with a loss (despite that the equity curve was pretty good on history), chances are it’ll continue to generate more losses. Your task is to stop it in due time.

3. Lifetime analysis of algorithms. It is also useful to analyze profitable algorithms and see how long they continue to generate profit signals even after your bot has taken profit and stopped trading.

4. Continuous and meticulous algorithm re-learning. If the bot’s equity starts to stagnate and algorithm stops generating profit signals (this will happen sooner or later), then it is necessary to retrain it.

5. Retraining and selection of neural networks. Empirically, we found that those networks that make 150-220 trades on the history have the optimal parameters. If less than 100 trades, this neural net can hold on to a paper loss for quite a long time. If there are more than 220-230 trades, then such neural net can make an excessive number of trades. Therefore, we try to select those neural nets that have 150-200 trades on the history and demonstrate smooth growth. Ideally, the profit per 1 trade should be $25-30 on average.

FAQ:

Question 1: how to run the re-learning process while the bot is active?
Answer: it makes sense to start re-learning process either when the bot hits its target or if the algorithm starts to generate loss. Before you start the learning process, make the following steps:

- close bot's open position by hand
- change "Open Type" to " -- " (inactive), so that the bot does not make a trade accidentally while the algorithm is in process
- reset the bot statistics
- re-train your algorithm
- when completed, turn on the bot again by changing "Open Type" back to the desired mode