MSDS 6306. Autodidactic trading robot

Ivan Charkin, Volodymyr Orlov

Introduction

Mastering stock and ETF trading might be a challenging and risky pursuit. It takes years of practice and some mistakes to forge a good trading strategy. Once a strategy is developed it should be constantly adjusted to stay current in the always changing environment of financial stock markets. While majority of the trades nowadays are done by computers, there is still a small portion of trades that are made by real-life humans deciding to buy or sell shares in one company or another[1]. If a strategy employed by a human has not been coded using a computer programming language from start it might be hard to automate it later or share it with others when the strategy has matured. In this work we want to see whether machine learning algorithms can be used to automatically learn given strategy from decisions made by a trader without requiring any additional input other than sell/buy signal and stock market data. We demonstrate two methods which can be used to learn to imitate trader's signal. We arrived to a conclusion that while these methods works for simple trading strategies both are not generalizable enough. At last we attempt to explain why this is a hard problem to solve using a single, generic machine learning algorithm.

Problem Statement and Data Description

For this work we've took historical stock and ETF daily quotes from Yahoo Finance's website. A sample of typical 5 data rows are displayed below.

```
##         Date     Open     High      Low    Close  Volume Company
## 1 2005-01-03 49.54198 49.83083 48.84174 48.93802 5251500     AXP
## 2 2005-01-04 48.93802 49.00805 48.01896 48.20277 4642200     AXP
## 3 2005-01-05 48.18527 48.20277 47.58131 47.65133 4681400     AXP
## 4 2005-01-06 47.56380 47.72136 47.09114 47.40625 4947600     AXP
## 5 2005-01-07 47.51128 47.56380 46.86356 47.05613 4426200     AXP
## 6 2005-01-10 46.96860 47.40625 46.93359 47.15241 4945600     AXP
```

Please refer to Table 1 for a detailed description of all variables.

Our goal was to train a robot that learns to imitate a trader by looking only at a signal generated by his/her strategy. Strategies employed by a trader are subject to following constrains:

- A strategy should not get any other input except for daily stock/eft quotes.
- A strategy followed by a trader should not change over time.

Trading Strategies

Although we want to build a robot that can learn any, generic strategy we still have to validate its performance on a set of concrete trading algorithms. For our experiment we selected a number of strategies of over increasing complexity.

Strategies based on Moving Average Crossover

The very basic strategy we've considered is based on an idea of Moving Average Crossover[2]. We have created 2 strategies based on this principle:

[1] https://www.washingtonpost.com/news/wonk/wp/2018/02/07/the-robots-v-robots-trading-that-has-hijacked-the-stock-market
[2] https://en.wikipedia.org/wiki/Moving_average_crossover

1. A simple 5-day moving average is observed and a signal is generated when moving average goes above or drops below the price
2. Two simple (2-day and 8-day moving averages) are observed and a signal is generated when short MA cross long MA.

(Ivan Strategy 1)

TBD

(Ivan Strategy 2)

TBD

Robot Models

One way to think about this problem is to imagine a robot that attempts to learn a general form of a mapping $\phi : X \to Y$ from a d-dimensional vector of the real numbers, $X = (x1, x2, ..., x_L)$, where $x_l \in \mathbb{R}^d$ to a categorical output Y, where $y \in Y = \{y_1, y_2, ..., y_K\}$.

Every day our robot takes a set of features derived from a fixed number of historical quotes preceeding this day and predicts a category which we map to a buy or sell action. Thus, our robot is a function

$$\hat{y} = f(x, \theta).$$

where θ are model parameters, which maximizes a sequence of observed outcomes produced by a human trader.

One important difference of this problem from a typical classification setup is that a set of input features, X are unknown. If we need to build a mapping for a specific trading strategy we could potentially find a best subset of features x that maximazes a likelihood of an observed signal by either going manually through all possible features or employing an automated search over feature space.

$$\underset{y \in Y}{argmax}\{p(Y = y) \prod_{i=1}^{d} p(X_i | Y = y)\}.$$

But to build a generic robot that is capable to imitate any trading strategy we need not only a flexible function f but also a way to automatically search over a set of features X for a best subset that maximizes likelihood of an observed signal.

Evolving Turing-Complete Machine

Our first approach is based on two evolutionary algorithms and a theory of Turing-complete machines[3].

First, lets imagine that there is a machine which is capable to simulate any computable strategy from a fixed set of instructions, supported by this machine. In fact, we don't have to imagine such a machine because all modern computer architectures are Turing-complete[4] and capabale to do exactly that.

Now all we need is to find an ordered set of instructions, A, which, when delivered to this machine, produces desired observed signal Y for any sequence of daily stock/eft quotes.

[3] https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf
[4] https://en.wikipedia.org/wiki/Turing_completeness

If we have infinite amount of time and energy we will, in theory[5], arrive to a right set of instructions which match observed signal.

Of course, we don't have infinite amount of time and energy thus we need a better way to search for an optimal set of instructions.

$\mu + \lambda$ Genetic Algorithm

Genetic algorithms provide an effective alternative to a random search. The basic idea of this algirithm is simple and is demonstrated in Figure 1. The set of instructions, θ are translated a binary string, which is called an individual. A population of individuals is then randomly generated ((1) in Figure 1). A fitness function $g(f(X, \theta))$ is then applied to each of the individuals ((2) in Figure 1). The more fit an individual, the more likely it is to be selected to create part of the next generation. Next generation is created from a sample of best individuals by applying crossover and mutation ((3 and 4) in Figure 1). Then a cycle is repeated again. Eventually, an individual is found that encodes an optimal solution.

Covariance Matrix Adaptation Evolution Strategy

When simple Genetic Algorithm proved to be inefficient in search for the best set of instructions we tries a more modern derivative-free mothod for numerical optimization, CMA-ES[6]. We did not change the algorithm. Since in CMA-ES new candidate solutions are sampled according to a multivariate normal distribution in $\mathbb{R}^n$ we had to turn a vector of real numbers to a binary set of instruction using rounding to a nearest integer.

LSTM Network

For our second model we decided to build a Long short-term memory Recurrent neural network with enough cells to learn the strategy.

For general-purpose sequence modeling, LSTM networks has proven stable and powerful tool for modeling long-range dependencies[7]

We used the simplest architecture displayed in Figure 2. We constructed our training and test datasets by taking 60 previous quotes, preceeding each trading day and turning this sequence into an input to our network. Next the network applies a series of non-linear transformations to this data at each network layer and the final layer produces vector of 3 real numbers. We turn it into probabilities by applying a softmax function[8] and use argmax function to choose the one with the highest probability which gives us one of:

- 0 - Sell
- 1 - Buy
- 2 - Do nothing

To optimize our model we've used Adam[9] algorithm with a cross-entropy loss function[10].

Results

Code

All code used to generate models, plots and report related to this work can be found in https://github.com/VolodymyrOrlov/MSDS6306_project2

[5] https://en.wikipedia.org/wiki/Infinite_monkey_theorem
[6] https://arxiv.org/pdf/1604.00772.pdf
[7] https://arxiv.org/pdf/1308.0850.pdf
[8] https://en.wikipedia.org/wiki/Softmax_function
[9] https://arxiv.org/pdf/1412.6980.pdf
[10] https://en.wikipedia.org/wiki/Cross_entropy

Figures and Tables

Variable Name	Description
Date	Date when data was recorded
Open	The first trading price recorded when the market opened on the day
High	Highest price at which a stock has traded that day
Low	Lowest price at which a stock has traded that day
Close	The last trading price recorded when the market closed on the day
Volume	Total number of shares traded for the day, listed in hundreds
Company	A unique alphabetic name which identifies the stock

Table 1: List of variables.

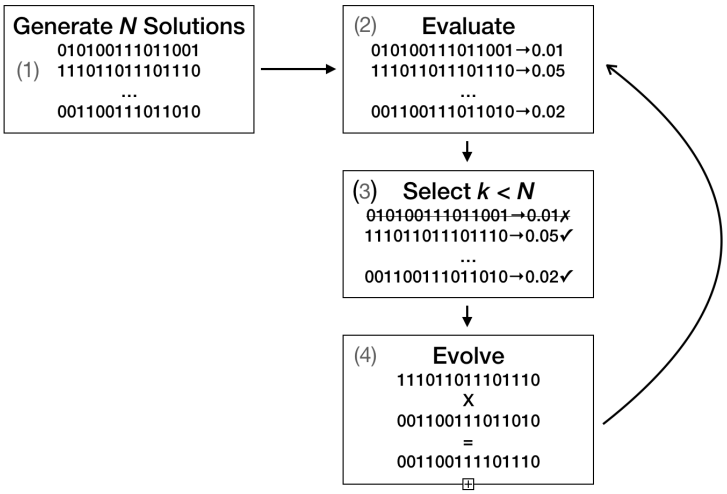

Figure 1: Outline of a simple genetic algorithm.

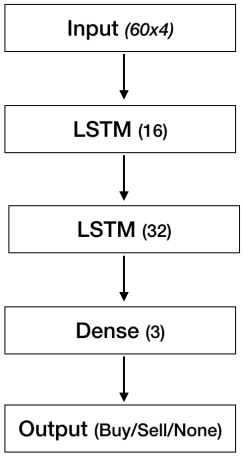

Figure 2: An architecture of the RNN network.