Trading System

Matteo Terzi

Contents

1 Introduction **1**

2 Framework **1**
2.1 Pattern Recognition 1
2.2 Strategy Builder 2
2.3 Optimisation of execution 3
2.4 Risk management 3
2.5 Double back-test 4

1 Introduction

2 Framework

There are four main parts:

- pattern recognition
- strategy builder
- strategy and transaction optimization
- risk management

2.1 Pattern Recognition

This phase has the role of finding the patterns in the indices, stocks, multi-stocks, which describe the inefficiencies of the markets. The discovered patterns have to be robust. This allows to be more confident that the strategies will be really profitable.

The main core of our idea is that the mainstream ML-based frameworks for trading are wrong. Basically, for two reasons. The first is that it is an ill-posed problem: normally, the strategies are not explicit but they rely on the actions Buy, Sell, Do Nothing. Thus, at each time instant the model has to choose to do something. However, this is conceptually silly as the majority of times the model should not do anything. Thus the problem is imbalanced and carries all the disadvantages of imbalanced problems. Actually, even if we fix the imbalance, the algorithm will be extremely inefficient as the number of transactions will be very high. This is a strong bottleneck in practice. In fact, the literature often considers unrealistic commission costs (unless you are a huge hedge fund). Moreover, live trading has many subtle costs (slippage, ...) which further increase the commissions. This makes all the apparently high performance of ML algorithms, totally unrealistic (in fact, it is not clear why one should publish a paper that shows a 90% annual margin and not to invest secretly their own money....).

The second problem is that strategies are not explainable. A big black-box model is per se unexplainable. And no one, would query this model every time step without knowing how it works.

Our idea is to find patterns that are independent of a prediction for the next time instants. In contrast, we aim at discovering simple or complex ”rules”. For example, at Monday the price of this share rises if Friday the close was under the opening.

Once these rules are found, it is almost straightforward to develop a good strategy. Moreover, a good trading system should be based on operational filters: a strategy should be active only where the market is in the condition experimented in the past. For example, if one strategy is based of patterns that are valid only in high-volatility regimes, it should be active when the market is calm. A black-box model would difficulty handle this case, unless we make it even more complex. However, since high-volatility is rare by nature, building a model which is simultaneously good in all the conditions is again unrealistic. Instead with patterns things are much easier. Why? Because, once a pattern is discovered we learn the market conditions under the form of embeddings or states. Thus, each strategy will be represented by a triplet (state, pattern, strategy) where the state is the market condition at which the pattern is operative and strategy is the practical strategy based on the pattern. An idea is to learn the state by building a classifier that is able to distinguish two different patterns using only the knowledge of market conditions. This phase is a sort of introspective procedure which makes a pattern aware of its state. How the filters are build or tuned is decided by the risk and transaction costs management phase.

2.2 Strategy Builder

This phase has the objective of creating the real strategy. For example, to decide to use limits buy, market buy etc. In other words, this part should tune all the details that are independent of the pattern but are more related with the execution. For example, if the Crude Oil market closes at 5-6 p.m. one should place orders accordingly. The perfect way to conduct this phase is semi-automatically. A basic template will be constructed and then it will be refined with the experience of user. This allows to double check and avoid inconveniences given by details (here details are everything).

2.3 Optimisation of execution

This phase has the role of optimising the executions. For example, to decide if a buy is worth given the current slippage and other transactions cost. Moreover, if more strategies are operative on the same market there may be a margin of optimization. For example, if strategy A wants to buy X at quantity P, and a strategy B wants to sell X at the same quantity, one could simply "buy to cover".

Another example is to decide to change the selling of buying price based on the order book or based on how is necessary to finalize the operation (e.g. we have to close our positions to avoid to go overnight).

This part is very complex but not less important. Actually, with few and profitable strategies optimizing the executions will not lead to a dramatic advantage. However, as soon as we will have to rely on weak strategies (the ones that exploits very small correlations), this part becomes dominant. To give an example, RenTech is known to be the best to handle this phase. The importance arises because, the small margin given by the strategy is covered by the costs.

2.4 Risk management

This part is the crucial part of the trading system. It has the objective of managing the risk of the strategies by deciding how money is allocated. In fact, the strategy choose when to buy but not how much. This depends on many factors: the capital, the number of independent strategies, the leverage, the maximum drawdown etc.

At every instant, this module has to decide how much capital is allocated for an operation. For example, if a pattern is very robust, then we can reserve more capital and higher leverage than a weak strategy. However, this module, should also depend on real-time conditions. For example, if a strategy is operating differently from expectations, then the system should be more cautious and reserve a lesser capital. Moreover, the risk of a strategy is connected to the risk of others. If we rely on 100 strategies where each is correlated 0.5 with the other, then the effective number of strategies is slightly more than 50. This means the actual risk is twice that expected. When the effective number of strategies is very high then, theoretically, one can use infinite leverage (of course this is not feasible in practice). Thus, a good risk management requires to estimate how strategies are correlated to decide how much capital and leverage is to be applied. Regarding risk management, finding patterns makes the system intrinsically more resilient to market correlations etc. This means that once one has a sufficient number of strategies working on a sufficiently large number of conditions and markets, there is no need of hedging because strategies automatically hedge each other.

2.5 Double back-test

Normally, strategies are back-tested to verify their profitability. Here we can do more: back-test patterns and back-test strategies. This allows to decouple the part of algorithmic execution and pattern discovery.

References