DOI: 10.24387/CI.2023.SPECIAL ISSUE.9

Zoltan Csesznik

Creating Trading Algorithms Using Machine Model Building Methods

Summary

Finding trading ideas and systems is no easy task. I know from experience that sooner than later even the most brilliant and diverse mind will run out of ideas. Not to mention the time-consuming task of back testing. The processes of system building not only solve the problem of idea generation, but also increase the speed of back testing by light years. After we have determined which financial instrument, we want to develop the system on, we define the data structure. Then we specify which indicators to consider in different combinations. The genetic algorithms test extremely large number of combinations based on the fitness functions and find the systems that are valuable. Once we have selected those with adequate past performance, we can test them on out-of-sample data. Systems that perform well both learning and testing periods are then subjected to validation procedures. At the end of the process, we receive the most robust systems, which we can also use during live trading. From the success point of view there is no difference between manual or generated strategies. A human built system can fail as much as a generated system. One important aspect of system building is diversification. It is a perfect way to generate multiple systems on different instruments to create an uncorrelated portfolio.

Keywords: System building, trading, algorithms, genetic, indicators

Introduction

Online stock trading has experienced a renaissance in the past five years thanks to technological innovation and legal harmonization. One of the main motivations of the many millions of new entrants is to mutiplicate their money, preferably as much and as quickly as possible. The world has not changed in this respect. People have always looked at the stock market with overflowing expectations, hoping for their financial independence and enrichment (Nassar, 2006). Most of us are easily manipulated by nature and give in to our desires to exchange our usual office life for autonomy and independence. In fact, thinking further, with the help of computer trading algorithms and automation, we don't even have to change our previous lives. It would be enough to have a good idea, a method in which, during automatic execution, the computer continuously runs the code and monitors the price (Kissel, 2020). When all the conditions in the program are met, it sends buy-sell orders to the market and ideally, the money in our account keeps growing. Although automation and implementation hide many pitfalls, at the same time, they can be easily avoided with appropriate expertise and experience. At the same time, the path to a winning strategy is much steeper. Ideas that are waiting to be tested disappear along with the desire to experiment, because back testing trading ideas is a time-consuming task. The question arises as to why the idea generation process could not be automated. My article deals with machine model building: how by specifying a few parameters we get to complex trading systems, which can hold their own during live trading. This can definitely be called a milestone, since the number of possible unique system combinations for each model – depending on its parameters – usually reaches 10^{30} – 10^{40}. It would be impossible to simulate every single combination and put the results in order with the computing capacity available today. In the case of such a large number of combinations, we use efficient search algorithms, with which we do not go through every single case, so we cannot be sure that we have found the global optimum, but at the same time, the results provided by these genetic algorithms are considered adequate, and by choosing a sufficiently large generation and population, we can be sure that results are near the global optimum (Bäck, 1996).

Figure 1 shows the most important steps of system construction. Of course, if the results obtained in the given step are not satisfactory, the step must be repeated after making the appropriate modifications.

Defining the Price Data

The first step in model generation is the definition of the data structure. We have an initial idea that we want to build the

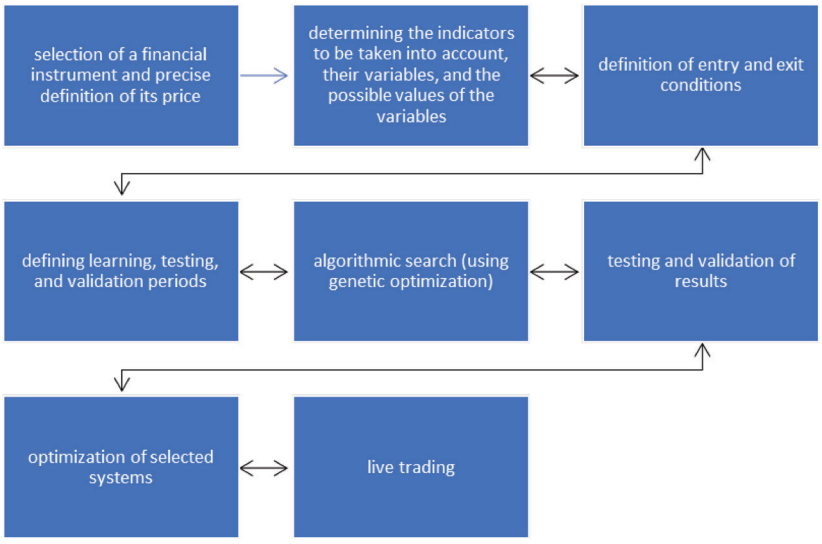

Figure 1: The most important aspects of system construction

Source: own editing

model in a certain market. There can be many reasons for this, from personal preference to the volatility, liquidity, and seasonality of the given asset. It is not enough to specify the sector, but also to select a financial instrument. Because its price, liquidity, market capacity are factors that influence the data structure (Kaufman, 2013). For example, it makes no sense to build high-frequency trading models on a stock with a few thousand shares per day. Thus, when we have selected the instrument and learned about its properties, the definition of the data structure can follow. Here we are really faced with the fact that it can be defined in almost infinite ways. Among the many standardization methods, the simplest and easiest is time-based systematization. To display price movements in time, we define the interval (for example, 5 minutes). Then we select the first and then the last trades of this first interval, the trades that took place at the highest and lowest prices, and then repeat this method for the entire data set. This is called open-high-low-close (OHLC) data structure (Mcallan, 2012). At this step, we decide what time interval models we will build. The larger the interval, the longer-term our system will be, and probably the lower the trading frequency will be.

Indicator Basket, Indicators and Their Variables

The next step is the selection of technical indicators, with the help of which we define the pattern we are looking for. When the value of the indicator reaches the previously observed values, we assume that a similar price movement will occur. In fact, the indicators thus provide entry and exit points (Nassar, 2006). Today, there are thousands of popular indicators that amplify the complexity of system development. Of course, it is not enough to select the indicators, one must be aware of their operation and define the possible values of its parameters. It is advisable to create an indicator basket that includes hundreds of indicators. We determine how many combinations of indicators we want to use for the decision criterion at the same time. For logic reasons, it does not make sense to take hundreds of indicators into account at the same time, since the number of possible combinations can reach 10^{30} even in the case of three indicators. For the sake of efficiency, it is advisable to maximize the number of indicators in the model between three to five (Bacidore, 2020). It is worth studying and understanding the mathematical background of the indicators to be able to correctly interpret how they function. Furthermore, the value range of the indicators' parameters must also be determined.

Entry and Exit Conditions

For the previously selected indicators, we need to develop a mathematical or logical condition that connects them - this equation will be our entry condition. The beauty of system building is that, similar to the definition of the structure of the data set, the entry condition can also be very varied Bacidore (2020). We can create a logical connection between the indicators, or of course we can also use mathematical operators.

Example of a logical connection:

[Indicator1(parameter1) < value1 & Indicator2(parameter2) < value2] = true/false

Example of a mathematical operator:

[Indicator1(parameter1) + Indicator2(parameter2)] = value

We must be aware of the mathematical background of the indicators and how we interpret each value. (For example, low values of oscillators are generally identified as buying zones, while high values are identified as selling zones.).

In the case of several indicators, it is advisable to normalize their values, since in this way all indicators will be included in the entry criteria with the same value range (Kissel, 2020). With normalization, we balance the differences resulting from the mathematical diversity of the indicators. If we want to favor one of them after this, we have the option of weighting them differently.

Then we define the exit conditions, which can also be based on specific values of the indicators (e.g. high values of oscillators) or even on other independent values. This can be a target price or maximum loss, but it can also be a time limit and of course it can be a combination of the former (Kissel, 2020).

Actually, the entry and exit conditions are nothing but our strategy, which we then simulate on the data stream.

Learning, Testing, Validation Periods

It is advisable to divide our available data set into at least two parts. In the first part - learning phase - we train the model. For the given parameters of the indicators, we calculate our entry and exit conditions for each time interval. There will be time slots in which our entry conditions will be met, in which case we will establish a position, similarly, if one of the exit conditions becomes true, we will close the position. We continue this until the last interval, after which we got the list of trades in the learning period. The process is repeated until all indicator parameter values have been taken (Brunton – Kutz, 2019). The unique systems obtained in this way are ranked based on the fitness criterion. We select those that meet the fitness criteria and simulate these systems during the testing period. The data in the testing period is completely unknown to the system, so we can simulate live trading and see how the systems selected in the learning period perform on this unseen data. We also calculate the fitness criteria for the testing period for each system. In the optimal case, the results of the testing phase differ only slightly from the results obtained in the learning phase. If the fitness parameters collapse during the testing period, then our system was not robust enough, so there is no point in investigating these systems further (Bacidore, 2020).

Learning and testing periods can be defined in several ways (Figure 2). It is easiest if the first part of the data set is the learning period, and the second part is the testing period. The disadvantage of this is that over a long period of time the characteristics of the financial instrument change, for example the price increases, volatility and trading volume change. All of these factors can influence the results. If the volatility was low during the learning period and we selected the systems with the highest fitness criteria based on this, it is not certain that we will get similar results during the simulation in a test environment with high volatility (Bacidore, 2020).

It can be advantageous if we choose a financial instrument as the basis of trading, with tens of years of data available, and choose a significantly larger learning period than the testing period (e.g. 70/30%). With this, we guarantee that our system encountered as many characteristics as possible during training, so it is more likely to produce similar results during testing as well.

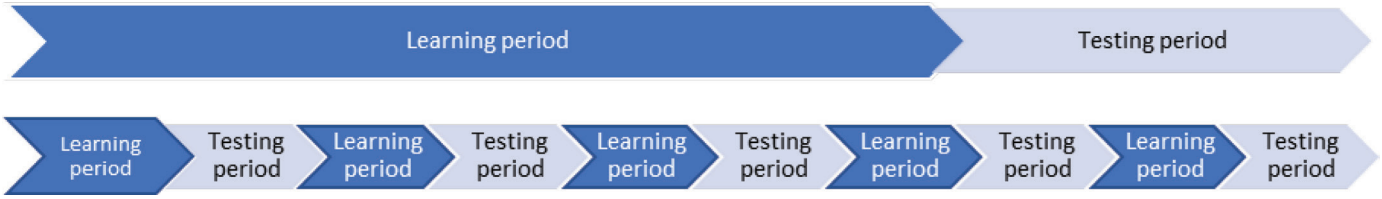

Figure 2: Possible ways of defining learning-testing intervals
Source: own editing

It can also be advantageous to divide the data set into many parts and define testing and learning periods almost identically, but alternately.

The advantage of the method is that we also use the most recent price to train the system, and the characteristics of this data are closest to real-time data, and as well the testing period contains as many price characteristics as possible.

Use and Importance of Genetic Algorithms

This step is the most complicated and perhaps the most critical process of system building, since among the settings previously made, it is necessary to find the systems that provided adequate performance in the past and, more importantly, are able to provide future results. The emphasis is on continuation, as we have to assess and select strategies based on past characteristics and values that are capable of realizing profit in the long term, able to adapt to the unknown and unpredictable, never-repeating market trends and processes.

In the case of simple settings - one or two indicators, with a limited parameter value range - if the number of combinations is below 10,000, we can use linear search algorithms, where every case is evaluated to find the global optimum, and the systems are sorted based on the fitness criterion (Pardo, 2008).

However, we assume that systems with multiple indicators and parameter sets provide better results. With these systems, the number of combinations, as I mentioned countless times, easily reaches 10^{40}. In such a large number of systems, the use of linear algorithms is pointless. Instead, we use advanced search algorithms, the common feature of which is that they will not find the global optimal solution, since they do not look at all possible cases. Instead, we arrive at several local optima, these algorithms are run a finite number of times and each run will produce a different local optimum, but we assume that these results are sufficiently good and close to the global optimum. Advanced search algorithms are simulated cooling, genetic and particle-swarm-based optimization (Pardo, 2008). Among these, I deal with genetic algorithms in the following. These are methods that mimic the biological process of evolution. They were introduced in the early 1970's, but were not widely used until the 1990's. Their important feature is that they turn the chances to their advantage, they are able to form a third even better strategy from two good strategies, which includes all the advantages of the original strategies. They use the process of mutation to reduce the probability of getting stuck at the local optimum. (Pardo, 2008). As a first step, we randomly select a group of parameter sets (population). Each combination (individual) of these parameters covers a possible strategy. The initial group of individuals is scattered in the optimization space. In the second step, the non-exclusive parameter set pairs are copied to the next population, which is the next generation. The chance of each parameter set selection being copied is proportional to its own fitness as measured by the objective function. The effect of this is that parameter sets with higher fitness naturally dominate successive populations. Conversely, lower fitness parameter sets have a higher chance of being left behind. This process is called selection (Pardo, 2008).

The third step is to randomly select pairs of parameter sets and then exchange some parameters from parameter set pair to another. In the process, good values of one set of parameters can be combined with good values of another set of parameters to get a better set of parameters. This is the case even if the good values originally come from two different sets of parameters. The fourth step is to randomize again a new group of parameter sets. In these parameter sets, some parameters are replaced with new and different values. This is called mutation and follows the way that in organic life genetic information cannot always be reproduced with complete accuracy from one generation to the next (Bäck, 1996).

Although the word mutation carries a negative cultural connotation, it can prove to be very useful in the long run if it is not exaggerated. In other words, not all mutations are bad, whether we are talking about life or genetic optimization (Pardo, 2008). Sometimes a mutation results in an individual – or set of parameters – being better adapted to its environment – or market – than those that preceded it. On the other hand, if a parameter set mutation performs poorly, the resulting set is unlikely to carry over to the next generation. Consequently, using mutation to occasionally shake up the optimization process to minimize convergence on a local maximum is better. And since the highest fitness parameter sets are likely to be copied multiple times into the next generation, the best fit is also less likely to be lost to a random mutation (Bäck, 1996). The algorithm continues by repeating the previously mentioned steps 2-4 for successive generations. New parameter set populations are produced in each generation. This continues until a predetermined number of generations has been reached, or a sufficiently good solution has been reached, or a significant improvement in fitness for the population as a whole no longer occurs.

Fitness Criteria

Search methods continuously accept or reject trading models in order to find the best set of parameters in the shortest possible time. That is why it is very important to use the best possible fitness function (Pardo, 2008). The best evaluation method is one that selects the set of parameters for a trading strategy that best predict real-time trading success.

We need to define a fitness criterion based on which we rank the systems (Kaufman, 2013). This can be some kind of per-

formance characteristic (e.g. net profit), but it can also be the maximum loss. The most commonly used fitness criteria are net profit, net profit * average profit per transaction, realized yield on the account, Pearson's correlation of the profit curve. The net profit is easy to interpret, since we want a system that achieves the highest value. At the same time, the net profit alone does not tell us anything about the system, possibly about how much loss we suffered during trading in exchange for this net profit, or whether this profit is the result of an anomaly and a few transactions (Kaufman, 2013).

The realized return on the account tells more, since in this case the net profit is divided by the sum of the maximum loss and the margin. The Pearson correlation of the profit curve shows how our profit curve compares to the optimal 45-degree straight line. 1 is the maximum value, in this case, the profit curve fits the 45-degree line. However, the probability of this happening is minimal. Pearson correlation value of 0.95 for pre-screening and 0.98 for the final curve is considered perfect (Pardo, 2008).

Testing and Validation of Results

When the strategies deemed best according to the fitness criteria have been selected, we must make sure that they will hold their place during live trading as well.

We want to select robust systems, by which we do not necessarily mean the largest profit. The characteristics of robust systems are the even distribution of trades and profit per trade, the relative balance between long and short profits, the acceptable level of risk, and the statistically relevant number of trades (Davey, 2014).

At the same time, in order to make sure that our strategy will also work properly on future data, we must subject it to additional compliance tests:

Testing the strategy on another market, or on another financial instrument: if the strategy produces almost identical results on several different financial instruments, it is definitely a strong indication that it has sufficient flexibility and adaptability (Pardo, 2008). If the results collapse, it is strongly suspected that the strategy is completely tailored to the original data stream and its small future change is more likely to have a negative impact on profitability.

By changing the structure of the data to a small extent, we also make sure that the selected indicators and their parameters can follow the unknown data with sufficient flexibility (Davey, 2014). You can easily change the structure of the data stream using two methods:

- Adding random noise: in this case, the price is slightly distorted, for example, the closing prices are randomly modified by a few ticks.
- Small change of time interval: in which case the originally set time interval will be adjusted, for example instead of 30 minutes it will be 31 minutes.

If the results collapse on the new data structure, there is a strong suspicion that our strategy is not robust enough and will cause disappointment during live trading (Davey, 2014).

Small changes in the parameters of indicators: a robust strategy is expected so that the values of the indicators next to each other do not cause a big change in the performance. For example, if our results change greatly when the moving average indicator parameter is changed from 20 to 21, we can be sure that the optimization space is „rocky and peaked", which also has a negative effect on the future development of the continuity of the results (Pardo, 2008).).

Walk forward analysis: judges the robustness of the system solely on the basis of trading during the test period. If the strategy performs well in this analysis, we are one step closer to running the strategy during live trading. Walk forward analysis provides an answer to how changes in market behavior affect performance, and what real rate of return can be expected (Davey, 2014).

Previous research has shown that strategies with 50-60 percent effectiveness are considered robust. A poorly structured strategy will fail this test.

Figure 3 shows seven datasets are available for training and testing. At the end of the first set, we could have built the M1 model, which we trained on the data from the first set. Then we repeat the process and build the system for the second set and so on. We have seven out-of-sample periods, and we combine these results to create the walk forward out-of-sample results (Brandy, 2015). Models have a unique set of parameter values. If they show stability, it is further confirmation that our system is robust, because the parameters of the M1-M7 systems did not change significantly during the periods.

Monte Carlo Analysis: replicates or copies the characteristics and behavior of a real system. Thus, the main purpose of Monte Carlo simulation is to try to mimic the behavior of real trades in order to analyze or predict as much as possible how they will evolve. This is an important step because random simulations can show dramatically different profits and maximum losses (Davey, 2014). It is possible that the maximum loss in a given sequence of trades was very small. But since history is unlikely to repeat itself, it is important to see how the losses and the profit curve develop if the transactions are randomly mixed. By repeating the process several times, we get the maximum loss, the yield, the longest drawdown period, and their standard deviation (Davey, 2014).

If a strategy passes all the methods detailed in this chapter then the last validation step is, of course, live trading, where we also have to deal with other problems of the real-time market environment.

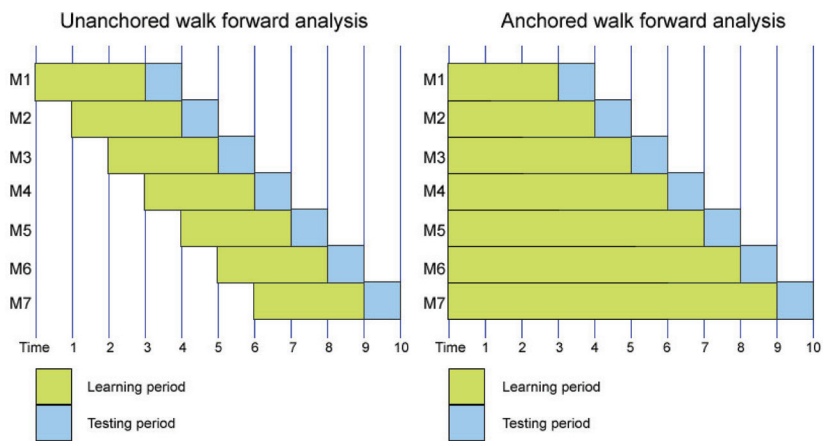

Figure 3: Walk forward analysis
Source: own editing

In this article we went thru the process of system building. Now it is obvious that it speeds up the process of strategy development by using advanced search algorithms and complex validation methods.

Compared to manually evaluating trading ideas system building opens up the world to test almost endless combinations of indicators and their parameters in short time. We can use large basket of indicators and test different instruments. Thru fitness function we can order these systems and select the best ones for live trading. From the success point of view there is no difference between manually or generated strategies. A human built system can fail as much as a generated system. But if we want to be on the safe side, we want to diversify ourselves and want to trade more strategies. Yet system building is a perfect way to generate multiple systems on different instruments to create an uncorrelated portfolio.

References:

JEFFREY, M. BACIDORE (2020): *Algorithmic Trading*, TGB Press, ISBN: 978-0-578-71523-0, pp. 143-144, p. 218

THOMAS, BÄCK (1996): *Evolutionary algorithms in theory and practice*, Oxford University Press, ISBN: 978-0-195-099713, pp. 49-53, pp. 71-73, pp. 155-158.

DR. HOWARD, B. BRANDY (2015): *Quantitative Technical Analysis*, Blue Owl Press, ISBN: 978-097918385-0, pp. 302-306

KEVIN, J. DAVEY (2014): *Building Winning Algorythms and Trading Systems*, John Wiley and Sons, ISBN: 978-1-118-77898-2, pp. 241-245.

STEVEN, L. BRUNTON – J. NATHAN KUTZ (2019): *Data-Driven Science and Engineering*, Cambridge University Press, ISBN: 978-1-108-42209-3, pp. 117-118, pp. 196-199

PERRY, J. KAUFMAN (2013): *Trading Systems and Methods,* John Wiley & Sons Inc., ISBN-9781118043561, pp. 20-23, pp 54-59

ROBERT, KISSEL, Ph.D. (2020): *Algorithmic Trading Methods: Applications using Advanced Statistics, Optimization, and Machine Learning Techniques,* Academic Press, ISBN-13: 978-0128156308, pp. 3-15, pp. 27-28, pp. 175-177, p. 224, pp. 519-523

DAVID, S. NASSAR (2006): *Ordinary people, extraordinary profits*, John Wiley & Sons Inc., ISBN-13 : 978-0471723998, pp. 12-19, p. 34

ROBERT, PARDO (2008): *The Evaluation and Optimization of Trading Strategies*, John Wiley and Sons, ISBN: 978-0-470-12801-5, pp. 69-71, pp. 181-183, pp. 195-196., pp. 208-210.