Rabatin Investment Technology Ltd
http://www.rabatin.com

Risk Management For Intelligent Trading Portfolios

Arthur Rabatin
First Published 1998 by Applied Derivatives Trading http://www.adtrading.com

Part III

Artificial Intelligence in Trading and Risk Management

Contents

1 – Introduction

Parts I and II of this series gave an overview over several strategies and technqiues traders and trading system designers can use to implement risk management into a portfolio trading environment.

Portfolio trading is not just a matter for large investors - in fact, very large equity fund managers in the US and UK have to compromise their asset allocation strategies due to the

lack of liquidity in many shares compared to the huge sums of investment capital available to some of these funds.

How can a large fund allocate even a small percentage to any one company, if its own action will already influence the price at the moment they want to acquire or dispose of their holdings? The growing size of some investment management firms has influenced the growth of index tracking funds as much as the lack of performance by many traditional fund managers has. Portfolio trading is about managing money for growth, using a range of instruments to capture different trading and investment opportunities and to reduce risk in trading.

Although part I and II have dealt with techniques and concept we understand and can formulate mathematically before we commit our money, any such approach is limited to the actual knowledge that we put in into these formulae. Why is this an issue for trading system or risk system designers? Financial markets and human behaviour in financial markets do not lend themself easily to the same scientific process as engineering, medical research or social sciences.

If we observe a pattern in financial markets today, we may not be able to predict with a high degree of confidence that this pattern will re-appear or will have in future similar consequences. Contrary to the marketing material of some (or many) trading newsletters or "trading experts", we have yet to discover anything about financial markets that we can regard as "market truths" or "market laws".

For the trader or fund manager, the issue of performance in financial markets is even more complex than following a system's or a newsletter's prediction: It is not the market which makes money for the trader, it is the *actions* the trader takes, his *behaviour* that makes or loses money.

For every price pattern or price prediction there is an almost unlimited spectrum of decisions to be made on market selection, position size, portfolio allocation, initial stop loss level, trailing stop, position size variation, averaging in, averaging out, taking profits, or cutting losses. And even if we have established *how* we want to formulate our trading strategy, we still have to determine *what is the right level* for initial stop, trailing stop, position size ... et c.

Here, the capabilities of computers systems become an important new tool in analysing these opportunities and decision scenarious. But traditional computer programming will not be sufficient as a tool to answer these open questions. Traditional programming requires the problem to be solved conceptually before it can be implemented in a programming language, which then will be compiled and executed to produce the information we are looking for.

Even solving numerical problems (such as finding the best trailing stop level for a certain buy/sell strategy) may not always be possible. A complete trading model implemented in our Adaptive Portfolio Trading (APT) Framework (a development environment for portfolio trading models) has up to 10^20 different possible states, including different market timing models, different market selection methods, portfolio allocation and risk management.
Even if each run through the historical database only requires one minute, it would take around 10^14 years to evaluate all possible options over one data set. What is required from computer systems now is that they become more *self-programming*: rather than implementing the program structure for the computer to execute, the computer system should be able to generate it's own rule sets, decision strategies and numerical values without human interference, based on targets defined by a human being and on the input data provided.

In other words, computer systems that may help us in better navigating through financial markets should be able to learn; i.e. become artificially intelligent. The concept of such systems and examples of our implementation are the subject of this final part in the series.

2 - Concepts of Artificial Intelliegence

Artificial Intelligence (AI) systems are systems designed to detect knowledge in data without human intervention. AI systems are "trained" to create models of underlying data. In trading financial markets, as it was mentioned before, an additional component becomes significant: decisions made by a trader influence the performance of the portfolio.

AI systems for portfolio trading must therefore not only be able to create models of the underlying data but also link those models to decisions to buy and sell, but also to exit a position, develop a risk strategy, select individual instruments from a large universe of markets and allocate between instruments.

Such AI systems have to therefore develop successful *behaviour*, similar to successful human traders who develop or use market analysis and combine it with their knowledge and skills in other aspects of portfolio management to make trading and investment decisions.

AI technology is a very wide field of different methods and technologies. We demonstrate here the concept of Evolutionary Systems, in the form of *Genetic Algorithms (GA's)*. Genetic Algorithms are highly effective non-linear search algorithms and are capable of combining different types of learning processes and constraints into one integrated learning process.

We will not explain GA's in technical detail here (for more information see our web site www.rabatin.com) but rather focus on how trading and risk management strategies can be built using AI technology in combination with the knowledge we have about financial markets. Generally GAs operate in such a way that the learning process simultaneously evaluates a large number of potential models ("a population"), within which different models compete against each other for the position of the best model.

The criteria of what constitutes a "good model" is defined by the systems designer - we have developed a measurement for performance consistency for that purpose which is described below. In that competition between solutions, a "survival of the fittest" process is being used to select better solutions and combine them to create a new population of models, which is also referred to as a new "generation". That process repeated many times leads to increasingly fitter individuals created with each generation, as the competition between solutions creates a bias towards such fitter individuals.Once a model with sufficient fitness is created (as defined by the user or systems designer), the model learning process can be stopped and the model used to execute it against new data to create new trading and portfolio decisisons.

Like the terminology of Neural Networks, the expressions used in evolutionary programming are very often taken from nature. The term "genetic" refers to the way how information is encoded in Genetic Algorithms: by converting all numbers, parameters into a long binary string of 0's and 1's, the GA based learning process can process any kind of information or model in the same way and in parallel. The creation of new models based on exisiting, reltively fit models, is done by combining and crossing-over these binary strings, in simulation of how genetic material of living beings is combined in the creation of new life.

3 - Artificial Intelligence for Trading Models

AI systems are not unique to financial market applications; AI systems are successfully applied in commercial and technical environments to develop complex scheduling systems or to detect hidden relationships between behaviour patterns of retail clients, or for credit scoring. What makes financial trading different, is that we need as output from an AI system not only a certain analysis, we also need a decision which we can apply to the data we get in from our real-time datafeed.

Such an automatic learning process evolves around a basic function in the form of

if <condition> then <action>

In portfolio trading, both components of the function usually develop very complex shape. To derive the <condition>, a large number of environment data have to be analysed and interpreted, plus data the system generates as a result of it's own behaviour, i.e. its own trading and performance history. The <action> also represents a large set of possible decisions, that the system takes simultaneously. Most important, this is a decision to buy, sell, or to adjust an existing position, as well as a decision on positions size, portfolio allocation and position risk.

For an adaptive model to learn appropriate behaviour, a payoff value must be available, that will allow the system to interpret it's actions as success or failure. In many robotics applications, such a payoff value may be immediately available after an action is being taken.

For portfolio trading systems no immediate feedback is available. Even though clearly we would wish each decision to be as profitable as possible (or at least avoiding losses), we must expect a real-world trading strategy to result in a stream of both profits and losses. In other words, a certain decision strategy might have resulted in a loss today, but it still was the best decision to take because it had the highest long-term expectation.

What we should expect is stable, profitable performance over a number of trades, where we can measure the distribution of profits and losses. This essentially implies evaluating performance over a longer time frame.

Because each action taken by the trading model includes a number of different decisions, and because the payoff is measured after a stream trading decisions, the performance payoff cannot be directly attributed to a single type of action or decision. This is not necessarily a unique aspect of machine learning based trading models: as every trading decision - systematic or discretionary - is influenced by multiple factors simultaneously.

The challenge of real-time portfolio trading is the nature of the constraints the trader (system) is subject to.

While the individual trading decision (including a decision on price risk, portfolio allocation, portfolio risk) is always made for one single market instrument, the defined constraints - such as overall risk thresholds or exposure limits - are defined or the entire portfolio. It cannot be predicted, if such global thresholds are exceeded, how each individual portfolio component is affected.

More precisely, we do not know, how the performance of each individual portfolio component is affected by the constraints placed on the portfolio as a whole. The only way to simulate the effect of such constraints is to implement them already in the learning process. In other words, the trading model object used during the learning process, must include the risk management parameters placed upon the system during the real-time execution of this strategy.

3.1 - Performance Measurement / Fitness Measurement

Choosing the appropriate tool for portfolio performance measurement is also an important tool for selecting a fund, trader or trading system for investment purposes. With hindsight, every investor would like see the highest possible return on the account that could have been achieved under given market circumstances. In reality, for the purpose of selecting a trader or system for investing, the investor must define a level of risk he/she is prepared to take. This risk expectation defines the parameters within which the investor would accept the trader/system to perform. It also serves as threshold to define when the trader/system does not perform as expected.

The appropriate notion in this context is that of "risk adjusted return". It means, any rational investor would expect the highest return given his/her level of accepted risk. Risk is typically defined as variance of returns. This concept has become subject to controversy, because it relies on a defined distribution of portfolio value changes as risk measurement. In reality, the distribution of portfolio value changes does not resemble a normal distribution. As a result, a

system relying on estimating risk through variance, will always grossly under-estimate the real risk the system is exposed to.

Within a GA based learning process, performance measurement is especially relevant because it also yields the fitness value through which the "survival of the fittest" process is implemented. The choice of fitness value also determines the success of the trading model during cross-validation. Because the GA process has shown to be a very effective optimisation tool, the success of the learning process must always be interpreted relative to the evaluation on out-of-sample data periods.

In terms of a portfolio trading model, we are therefore interested in the future performance of the portfolio, when applying the parameters and rules the system has learned during the training process. Because such performance can never be perfectly predicted, the investor into a trading strategy therefore seeks consistency of performance.

The performance benchmark must therefore measure this consistency. To create a trading model that adapts without human interference, the performance benchmark must also measure the absolute level of performance, relative to the expected return and the accepted risk. Because the accepted risk is largely a user-defined value (because it is a result of each investor's preference), the trading model must balance return with this risk level.

For evaluating a trading system's performance consistency in an automatic, self-learning process the fitness must take into account the time structure of performance (consistency of performance) as well as the absolute level of performance.

The target function of the learning process applied in our trading models is based on a user-defined Return Path (RP). This return path is a monthly or quarterly range of expected returns within which the system ideally performs. The fitness of the trading model is measured by the error of tracking this target range, the Return Path Error (RPE). Formally, the RPE is defined as

$$RPE = \sqrt{\sum_{n=0}^{N-1} e_n^{\,2}} \Big/ N ,$$

where:

N … number of periods (either calendar quarters or calendar months),
n is the n^{th} calendar period (indexed between 0 and N-1)
e_n is the actual tracking error for the n^{th} period.
e_n is defined as follows:
if ($r_n > RP^+$): $e_n = (r_n - RP^+)W$, if ($r_n < RP^-$): $e_n = (r_n - RP^-)$,
where r_n is the calculated actual percentage return of the portfolio for the n^{th} period, RP^+ is the upper limit of the return path target range, RP^- is the lower limit of the return path target range and W is a weighting applied to smooth the effect of *upside* errors of the portfolio (typically $0.3 \le W \le 1.0$; this model uses an error weighting of 0.4).

The GA process seeks to minimise the RPE, which ideally equals zero, when the system performs completely within the desired return path.

The advantage of using RPE as performance benchmark is that it emphasises and measures the consistency of performance in that it matches the user's expectation on return with any risk-thresholds attached to the portfolio. If the return expected from the model is not compatible with the risk constraints placed upon the portfolio, this discrepancy can then be already detected during the learning process. Either the portfolio constraints or the performance expectations will then have to be adjusted.

3.2 - Learning Process Implementation

The decision making process of the adaptive trading models simulates the human decision making process by learning behaviour patterns that are matched against patterns the system detects in the environment data.

The behaviour of the trading model is therefore a result of events that take place in the environment - both being the market data and it's own accounting database.
It is the purpose of the system's training process to learn to detect what constitutes an *event* and what is the appropriate behaviour in response to it.

Because *behaviour* in the context of a trading model is always a decision to sell, to buy, or to do nothing (keep the current position unchanged) in a particular financial instrument, it also means that risk for the portfolio may either be increased (by establishing a new net position in a market) or decreased (by reducing an open net position in any market). The trading model is designed to retrieve information from both external data (market prices and other data) and from its own trading performance to decide on its trading decisions. These trading decisions (and subsequently decisions on any open positions) are also subject to the risk threshold placed upon the system by the trading manager or system supervisor.

Every trading decision has four components that simultaneously define the portfolio performance: market timing - price risk - portfolio allocation - portfolio risk. Below, the learning process for each of the components, and the integration into a final trading decision are described.

3.2.1 - Market Timing Decision

The process of mapping environment data patterns to a trading behaviour pattern is implemented through two layers of objects that perform these functions: :a layer of objects performing calculations on data (*Calculation Node Objects*) and a layer of objects interpreting the results of calculations as events and mapping these events to possible decisions.

Calculation Node Objects (CNOs) are an array of objects that retrieve data and perform mathematical, statistical and logical operations on these data, returning a set of numerical values that will be interpreted by another layer as events. It is through the GA process that it is decided for each CNO which data are retrieved and which operator is applied. The calculation that is performed within each CNO can be either very simple (such as a logical comparison { e.g. ">" or "="} or a simple arithmetic operation {e.g. "+", "/" ...}), or it can involve complete statistical calculations, such as the historical volatility of a retrieved time series. Boolean values (true/false) are represented as numerical values (0 / 1).

CNOs are not limited to retrieving input only from an external database. The output of each CNO can also be selected in the learning process as input for calculations. This allows the trading model to detect complex patterns by connecting a large number of CNOs that each perform very simple calculations. In an array of (n > 1) number of CNOs, the n^{th} CNO can use the output of (CNO[0] ... CNO[n-1]) as input.

The actual number of Calculation Node Objects is a matter of trading model design and is largely dependent on available machine time, as increased complexity also dramatically increased the time required to train these trading models. The advantage of using an array of CNOs is the flexibility that otherwise would be restricted by a fixed length Genetic Algorithm.

Event Node Objects (ENOs) are an array of objects that interpret output values of CNOs by learning to map the CNO output to a logical value of true/false in terms of an *event* occurring yes/no.

Because the output of each CNO is a fixed structure of values that is always assigned valid data (independent of the calculation performed with a CNO), ENOs can always interpret the result of a calculation in terms of an event occurring or not.

Event Node Objects also learn to map a particular CNO output set to a particular type of decision - to buy, sell or do nothing in a market.

After both layers of objects have been constructed, the trading model possesses a pattern to interpret data it retrieves from the external database in terms of possible trading decisions. Each Event Node Object that - for the actual data - returns *true* for an *event* to exist, also returns a signal for a decision, to buy, to sell or to remain unchanged in the current position (if any). The trading model then chooses the one decision that occurs most frequently as ENO result, as a final decision to be made in the moment the calculation is performed.

3.2.2 - Risk and Portfolio Allocation Decisions

Risk and allocation decisions are learned differently by the trading model, as these decisions are not "event-driven", but only depend on a position in a market to exist.

Learning market timing behaviour differs from the other components in the way that we have little knowledge on how buy and sell decision should be made. The system can and should therefore have a large degree of freedom to develop these decision making rules by learning to develop appropriate adaptive behaviour patterns. For risk and allocation decisions, we do however have some knowledge about possible rules (e.g. restricting risk exposure, diversifying portfolios), and we especially have certain constraints that we want to enforce on the trader / the system. Consequently, the trading system can have freedom to decide on allocation strategies and risk exposure within a portfolio, but it must do so by observing the global risk and allocation thresholds.

The learning process for risk and allocation decision is a rule-based learning process that uses a "bottom-up" approach by combining simple rules ("sub-rules") using mathematical and logical operators, resulting in a more complex decision rule, onto which certain risk thresholds are applied by the trading model.

For the GA process, each sub-rule and each operator (mathematical or logical) is recognised as a simple integer value indexing the rule within the risk management object of the trading model.

The GA learning process of the trading model creates a rule which is a formula consisting of data parameters (i.e. calculation results of the sub-rules) and the operators. Although the structure of the formula is limited in its flexibility compared to the market timing decision, the trading model can still create a relatively complex structure of rules very different from the simple components it is based on.

Although we will focus here on the risk and allocation decision to be made as part of a new trading decision (resulting from a market timing signal), the same calculations are repeated at the end of each trading period (mostly end of trading day), in order to adjust existing open positions. These adjustments are particularly important because they make sure that the portfolio will always be maintained within the desired risk thresholds. Our testing has further found that the constant application of risk thresholds is one of the most significant contributing factors to creating a consistent and predictable performance pattern, largely independent of the actual market timing strategy used.

The trading model uses three levels of risk calculation:
- expected risk (calculating an estimation of the risk a certain trading position carries)
- accepted risk (a risk threshold which the trading model defines for each trading position)
- portfolio risk threshold (the overall constraint that the system supervisor or trading manager puts on the entire portfolio and must be observed by the trading model).

The *expected risk* is used by the trading model to make an initial decision on the size of a trading position, similar to a human trader. Through the learning process described below, the model is trained to improve its risk forecasting ability. The *accepted risk* is a threshold that is associated with a given market position, and is calculated by the trading model as part of the

learning process. This is not necessarily a "worst-case-scenario" threshold. More likely, the trading model uses risk thresholds to actively manage the size of open positions in a market to create a less volatile equity curve.

Portfolio risk thresholds are defined by the supervisor of the system, the trading manager. The trading manager normally has a clear view of the amount of risk he/she is prepared to accept for a given portfolio. This however is a threshold for the aggregated risk of all positions and the trading model learns to translate this global constraint into position limits for each individual component of the portfolio.

The risk calculation for a portfolio is performed in three steps: price risk - portfolio allocation - portfolio risk.

3.2.2.1 - Price Risk Calculation

Price risk is associated with the market in which a position is taken and is independent of the size of a trading position. It is however closely linked to the type of market timing strategy developed by the trading model, since the expected variance of prices is a function of the time horizon of an investment.

The rules which the trading model can learn to use for calculating risk use as input:

- price volatility (in absolute terms over previous n number of days)
- price volatility (in terms of standard deviation)
- previous trading range (price high/low over previous n number of days)
- maximum / minimum price changes (over previous n number of days)
- average/total/minimum/maximum price risk calculations of others components of the portfolio
- price and volatility correlation to other components of the portfolio
-

The "sub-rules" are combined in a rule structure developed by the GA process that result in a risk estimate for a position in that market. Initially this risk estimate is also the maximum accepted price risk for the position. In trading terms, that price risk is the "Stop-Loss" level at which a position is closed out, because the market forecast, implied by the decision taken by the market timing strategy, is accepted to be wrong.

After the initial position is established, trading models re-calculate price risk estimates every time the trading model is updated with new prices. The however the trading model does enforce to cut the position size, if the estimated price risk exceeds the accepted risk. The accepted risk is only allowed to increase by a limited margin when applied to an existing open position.

3.2.2.2 - Portfolio Allocation Decision

Portfolio allocation decides the percentage share of the overall portfolio value that is allocated to one portfolio component. Portfolio allocation within asset investment strategies is a defined process which could easily be implemented through any GA (or other) learning algorithm. The decision for a trading portfolio is more complex, because the system is not invested into all markets at the same time. The trading model must learn how to allocate a share of the portfolio for a new position to be taken, but also taking into account existing positions and any portfolio limits placed onto the system.

For any given portfolio component the allocated share of the portfolio is calculated by the trading model as

$$A_i = F_i \Big/ \sum_{n=1}^{N} F_n$$

where

A_i ..... allocation for the i^{th} market

N ..... number of markets available to the portfolio

F_i ..... the function expressing the rule used to calculate the allocation

Each rule developed by the GA learning process is based on a combination of sub-rules which are based on the input of:

- price data and time series statistics of the given market
- time series statistics in relation to the other markets in the portfolio (cross-correlation and relative strength)
- relative value of each markets performance within the portfolio
-

The result of each calculation is always the re-calculation of allocation for the entire portfolio. This makes sure the trading model not only observes strictly any portfolio constraints, but it also ensures that the trading model can re-balance the entire portfolio after the addition of a new position to the portfolio. Since performance consistency is typically the most important criteria for evaluating trading models, the automatic re-allocation ensures the highest possible level of diversification within the portfolio.

Market Selection is a special case of the portfolio allocation decision, as it is also possible for the system to allocate a share of zero to a market, thus effectively excluding a market from the portfolio. When setting up the adaptive trading model, a number of markets are made available to the system. However, the trading model can in every stage of the calculations decide to reject a certain market either by not creating a buy or sell decision or by allocating a very small share of the portfolio to that market that cannot be traded in the market.

3.2.2.3 - Portfolio Risk Decision

After the trading model estimates the price risk associated with a position and calculates the share of the portfolio allocated to a particular trade, it can calculate the actual quantity to trade, i.e. the size of the trade and of the open position.
In learning to perform this calculation, the trading model uses the calculated price risk, the allocated equity and all input data used by previous calculations to create the decision making rule for the trading quantity.

The decision on the size of each position has shown to be among the most relevant decisions determining the consistency of portfolio performance.
In general terms, the position size is a function of the available (allocated) equity for a given market and the price risk associated with the market (which the trading model both learns to calculate/estimate), i.e. $U_i = f(A_i, P_i, R)$ where U_i is the number of trading units (position size), for the i^{th} market within the portfolio, P_i is the calculated Price Risk, for the i^{th} market, A_i is the calculated allocation for the i^{th} market and R is the portfolio risk accepted by the system expressed as a percentage of total capital accepted to be at risk.

The above function can also be re-written as $R_{actual} = f(A_i, P_i, U_i)$. In other words, the actual percentage of the portfolio at risk (R_{actual}) is a function of a market's price risk, the allocated share of the portfolio and the number of units bought or sold in the market. For the purpose of the learning process, the R_{actual} value should be smaller or equal to the accepted portfolio risk value, R. The learning process of the trading model therefore first finds a percentage portfolio risk value R that is compatible with the global parameters of the portfolio, and then, using both other parameters to the function, A_i, P_i, calculates the actual number of trading units (U_i) that would create the desired exposure.

The calculation of the trading size is the link between the market timing behaviour developed by the system and the required risk management strategy.

For every market timing decision made by the system, it will calculate the associated market risk (per unit price risk) and the associated portfolio allocation.

It is through the variation of the trading size (position size) that the system is performing a trade-off between higher (lower) per unit price risk and smaller (larger) size of the trading position.

Such an approach to managing the portfolio risk is in our view an essential component of a consistently developed trading strategy. It removes the uncertainty that is normally associated with profitable open positions; if profits should be kept "running" or positions liquidated in order to ensure that open profits are protected from any more risk.

In order to develop consistent performance behaviour, the trading model must have the ability to manage constant risk exposure during changing portfolio composition and market events. This process enables the trading model to consistently balance the market price risk and the portfolio risk, by changing the actual position size it will have in any market. Increased risk per unit traded (price risk) can be matched by a decrease in number of units traded and vice versa. As with the calculation of portfolio allocation, portfolio risk management is always performed over the entire portfolio. Therefore, a change in one portfolio component can be matched by shifting (i.e. reducing) exposure in other markets. This allows the trading model to re-balance the portfolio either when a new market position is to be taken or when the daily mark-to-market process is performed.

3.3 - Designing an Adaptive Trading System

Dynamic Parameter Optimisation

Since the trading system is designed to learn behaviour patterns, the optimisation of fixed numerical parameters (such as the number of days over which historical volatility is calculated) should be avoided. Not only is such a parameter unlikely to be usable across different markets, it is also very dependent on the actual distribution of prices within the training period. It must therefore be expected that the optimum parameter changes dramatically when market condition change. Using statically optimised parameters leads to inflexible curve-fitting which - in our previous research - has shown not to hold up in subsequent out-of-sample tests.

The concept used by these trading models are "dynamic parameters"; i.e. parameters for which the value is not directly optimised, but for which the model learns to develop rules for calculation. This is done within the Calculation Node Objects. For any calculations that cannot be developed as rules within the trading model, a hard coded time frame of 200 data periods is used as starting point for calculations.

Example: The system may learn to use a moving average of market prices as part of an estimate of near-term market direction. For this moving average, a parameter is needed for the number of days over which the calculation is performed. Rather than optimising this parameter directly (within a given range of possible values), the system learns to calculate the value based on another calculation, e.g. market volatility (calculated by a different Calculation Node Object). The systems retrieves current market volatility reading, and the min / max value for that calculation (over the time span calculated by the other CNO). The last reading is then expressed as ratio within the historical min/max values. If min = 8 and max = 20, a current volatility value of 15 would result in a ratio value of 0.58. The general formula is: Ratio = (Current - Min) / (Max - Min).

This ratio is the current value expressed in percentage of the range. This percentage value is then applied to the min/max range of possible moving average values used by the system. Assuming we define possible moving average parameters between (10;100) periods, the actual moving average parameter used in this calculation would be ((100-10)*0.58) » 52 period moving average. Note that this resulting value changes very time the underlying calculation data (here: volatility reading) changes.

The benefit of dynamic parameter optimisation is that the learning process focuses on developing calculation *rules* rather than optimising parameters. Such rules can be applied to all markets, since they will adjust automatically to new data by re-calculating the actual parameters used for calculation of events.

Top Fitness / Closest Fitness Behaviour Patterns

After the training process, the trading model selects one single rule system to be applied to new data. Typically this is the rule set which had resulted in the optimum theoretical performance during the training phase. At the beginning of the application phase of each new period, the trading model will adapt its behaviour according to the rules it has learned during the training process. Since every learning process is an optimisation process, the system always carries the risk of overoptimisation during the learning process.

Using overoptimised behaviour patterns on new data is very likely to result in undesirable, negative performance. We have therefore developed a concept of not using the optimised rule set for the actual trading period ("top-fitness" rule set), but to select one rule set, which is likely to be more robust in its real-time performance then a highly optimised behaviour, the "closest-fitness" rule set.

To calculate the "closest-fitness rule set", the trading model uses an internal threshold to find a range of behaviour rules, which resulted in acceptable performance during the training period, including the best strategy, i.e. the top-fitness rule set. Within this group of rule sets, the system then tries to find a smaller group with similar performance results. If such a group is found, the system selects the best rule set of this group to adapt to, for the new period. This rule set is referred to as the closest-fitness rule set. It may be the case that the selected closest-fitness rules set and the top-fitness rule set are identical, but more often this is not the case.

Because this closest-fitness rule sets is not as highly optimised as the top-fitness behaviour, we have found a very significant increase in consistency of performance, when comparing the training results with the application periods.

Continuos Strategy Evaluation

During a real-time application of the trading model, at the end of each actual period (when the system prepares to adapt new behaviour patterns), the trading model already has generated a stream of trading decisions, which have resulted in a profit or loss, and the system may also have open positions in any of the markets of the portfolio. Although the division of the database into several training/application periods is necessary to create an adaptive learning process, it does not correctly reflect how the system would be applied in a real-time environment.

To replicate real-time behaviour, the trading model has the ability to dynamically adapt new behaviour while keeping all existing open positions and existing accounting values. This creates a continuous performance measurement and allows to measure the effect, switches in the behaviour patterns would have on existing market positions.

4 - Outlook: The Trader of the Future

We believe that in the not too distant future, technology in trading and portfolio management will be the most important weapon in an increasingly competitive marketplace. Technology will increasingly be used not only as a static tool to implement concepts developed by human beings, but also to generate new knowledge in a fast changing environment.

Human beings will still be in a very important position in financial markets, but roles will be changing. Like in manufacturing, a software revolution will change the way we operate decision making processes. The need for changes has been recognized by a fund

management industry that has moved large portfolio to index funds, reducing the human input into the decision making process.

The increased use of technology will have an impact on other users of markets as well. Because artificially intelligent computers systems are better able to cope with changes in market environments, these changes will be appear more frequent and the speed of change will be increasingly dominated by the speed of computer systems detecting and acting upon these changes. This process will put other market participants at a disadvantage if they fail to adapt.

For whatever the future path of financial markets is, it will be certain that the human beings participating in financial markets will have a new breed of computer systems to compete with - and since only performance will determine who survives this competition, the financial marketplace will be driven by the same "survival of the fittest" principle as it is now, but only that computers are faster at selecting such fitter individuals and sending them into the competition.